ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Ανάλυση πραγματικών δεδομένων με την χρήση ερωτήσεων κορυφογραμμής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της/του

ΧΡΥΣΟΧΟΟΥ ΓΕΩΡΓΙΟΣ

Επιβλέπουσα : ΒΛΑΧΟΥ ΑΚΡΙΒΗ

Μέλη εξεταστικής επιτροπής: ΓΚΟΥΜΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ, ΚΑΡΥΔΑ ΜΑΡΙΑ

Σάμος, Ιούλιος 2020

Πρόλογος και ευχαριστίες

Η διπλωματική εργασία με τίτλο «Ανάλυση πραγματικών δεδομένων με την χρήση ερωτήσεων κορυφογραμμής» εκπονήθηκε την περίοδο Σεπτεμβρίου 2019 με Ιουλίου 2020. Ως αποτέλεσμα της μεταπτυχιακής μου διατριβής στο τμήμα Μηχανικών πληροφοριακών και επικοινωνιακών συστημάτων του πανεπιστημίου Αιγαίου, χαίρει ιδιαίτερης σημασίας. Η θεματολογία γύρω από τις επερωτήσεις κορυφογραμμής και την εύρεση βέλτιστων λύσεων είναι αναμφισβήτητα επίκαιρη και άκρως πολύτιμη κατά τα ραγδαίως μεταβαλλόμενα τεχνολογικά σημερινά δεδομένα. Για τον λόγο αυτό η παρούσα διπλωματική θίγει πληθώρα ζητημάτων που σχετίζονται με συχνά κωλύματα, η επίλυση των οποίων θα μπορούσε να αξιοποιηθεί από μεγάλο φάσμα ειδικοτήτων και τομέων της επιστήμης της Πληροφορικής.

Αδύνατη θα ήταν η περάτωση της παρούσας διατριβής δίχως την πολύτιμη βοήθεια της επιβλέπουσας καθηγήτριας του Τμήματος Μηχανικών Πληροφοριακών και επικοινωνιακών συστημάτων, κυρίας Ακριβής Βλάχου. Με μια πολυετή εμπειρία σε θεματολογία βελτιστοποίησης και επεξεργασίας επερωτημάτων, η κυρία Βλάχου αποδείχθηκε ακρογωνιαίος λίθος της δομής και της ποιότητας του περιεχομένου της εργασίας αυτής. Συνεπώς, θα ήθελα πρωτίστως να ευχαριστήσω την κυρία Βλάχου για την αμέριστη προσοχή, την πολύωρη επικοινωνία και την πολύτιμη επιτήρηση της για την ολοκλήρωση του εγχειρήματος αυτού. Οι ιδέες και η κριτική της στάθηκαν αρωγός για την έγκαιρη και επιτυχημένη επίτευξη του συνόλου των στόχων που τέθηκαν. Αναμφίβολα, δεδομένων των δύσκολων και απαιτητικών στιγμών που χαρακτήρισαν τις αρχές του έτους 2020, ευχαριστίες πρέπουν σε όλους εκείνους που διασφάλισαν το περιβάλλον και τις συνθήκες για την περάτωση της διπλωματικής εργασίας αυτής. Η ανεκτίμητη βοήθεια των αφανών ηρώων του τομέα Υγείας χρίζει δε ιδιαίτερης μνείας δεδομένης της αυτοθυσίας που επέδειξαν σε καιρούς που βρίθουν προκλήσεων και δοκιμασιών, με δίχως προηγούμενο για την κοινωνία. Τέλος, ένα μεγάλο ευχαριστώ στους γονείς μου για την ενθάρρυνση και την υποστήριξη τους παρουσία πληθώρας δυσκολιών.

ΧΡΥΣΟΧΟΟΥ ΓΕΩΡΓΙΟΣ

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Πίνακας περιεχομένων

Λίστα Σχημάτων

Λίστα Πινάκων

Ακρωνύμια

RAM	Random Access Memory-Μνήμη τυχαίας προσπέλασης
gb	Gigabyte- Μονάδα μέτρησης ποσότητας πληροφορίας
ms	Milisecond- Μονάδα μέτρησης χρόνου
BNL	Block Nested Loop-αλγόριθμος υπολογισμού κορυφογραμμής
k-Skyband	Προσέγγιση εύρεσης βέλτιστων στοιχείων
Constrained Skyline	Προσέγγιση υπολογισμού της υπο περιορισμούς κορυφογραμμής συνόλου δεδομένων
Csv	Comma Separated Values- Τύπος αρχείων κειμένου με τον χαρακτήρα «,» ως διαχωριστικό τιμών
url	Uniform Resource Locator, διεύθυνση πόρου στον παγκόσμιο ιστό
Txt	Text file- τύπος αρχείου κειμένου
Lat	Latitude- γεωγραφικό πλάτος
Long	Longtitude- γεωγραφικό μήκος
VIN	Vehicle Identification Number- Αναγνωριστικός αριθμός οχήματος
IQR	Interquartile Range- Ενδοτεταρτημοριακό εύρος
Q_i	i-th Quartile: Η τιμή κάτω από την οποία βρίσκονται οι τιμές του i-στου τεταρτημόριου
Η.Π.Α.	Ηνωμένες Πολιτείες της Αμερικής
$	Δολάριο – νόμισμα
Min	Minimum – Ελάχιστη τιμή
Max	Maximum – Μέγιστη τιμή
Avg	Average – Μέση τιμή
Buffer	Ενδιάμεση μνήμη προσωρινής αποθήκευσης του υπολογιστικού συστήματος
String	Τύπος τιμών κειμένου στην Java
Int	Τύπος αριθμητικών ακέραιων τιμών στην Java
Float	Τύπος αριθμητικών δεκαδικών τιμών στην Java
Double	Τύπος αριθμητικών δεκαδικών τιμών μεγάλης χωρητικότητας στην Java

Regexp	Regular Expression- Κανονικές παραστάσεις
Id	Identification Number: Αναγνωριστικός αριθμός
True	Μπουλιανή τιμή που δηλώνει αλήθεια στην Java
False	Μπουλιανή τιμή που δηλώνει αναλήθεια στην Java
Null	Τιμή που δηλώνει τη μη ύπαρξη αντικειμένου ή τιμής στην Java
Java	Αντικειμενοστρεφής γλώσσα προγραμματισμού
Freq	Frequency- Συχνότητα
Input	Εισροές
Output	Εκροές
Km	Kilometers- Χιλιόμετρα, μονάδα μέτρησης απόστασης
Keywords	Λέξεις- κλειδιά
Std	Standard Deviation- Τυπική απόκλιση
If..else	Βρόγχος ελέγχου συνθήκης στην java
For	Επαναληπτικός βρόγχος ενεργειών στην java
While	Επαναληπτικός βρόγχος ενεργειών στην java
Try..catch	Βρόγχος δοκιμής ενεργειών και διαχείρισης σφαλμάτων στην java
Byte	Μονάδα μέτρησης ποσότητας πληροφορίας
Craigslist	Αμερικανικός ιστότοπος αγγελιών
Query	Επερώτημα
Outliers	Έκτοπες εγγραφές με ακραίες τιμές

Περίληψη

Η προβληματική του προσδιορισμού της βέλτιστης επιλογής αφορά την πλειονότητα των εκφάνσεων της ραγδαίως τεχνολογικά μεταβαλλόμενης καθημερινότητας. Ιδιαίτερα σε περιπτώσεις ύπαρξης πληθώρας προς εξέτασης διαστάσεων, πολύτιμος αρωγός στην εύρεση του συνόλου των αποδοτικότερων επιλογών διαφαίνεται η χρήση των επερωτημάτων κορυφογραμμής. Στην παρούσα διπλωματική διατριβή γίνεται χρήση πραγματικού συνόλου δεδομένων με σκοπό την εξέταση δυνατότητας άντλησης αξιόπιστων συμπερασμάτων μέσω της εφαρμογής εναλλακτικών μορφών επερωτημάτων κορυφογραμμής σε αυτό. Επιπροσθέτως λαμβάνει χώρα η αναφορά και η υλοποίηση μεθόδων αντιμετώπισης συνηθισμένων προκλήσεων που συνδέονται με πραγματικά δεδομένα όπως η έλλειψη τιμών και η εμφάνιση έκτοπων εγγραφών. Κατόπιν ακολουθεί πρόταση κατάλληλων τροποποιήσεων πάνω σε αλγοριθμικά μοντέλα υπολογισμού κορυφογραμμής με σκοπό την εφαρμογή εξεζητημένων επερωτημάτων πάνω στο επιλεχθέν σύνολο δεδομένων. Τέλος, μέσω της παράλληλης εποπτείας υπολογιστικών πόρων που διατίθενται κατά την διαδικασία προσδιορισμού της κορυφογραμμής, εξετάζεται η επίδραση διαφορετικών παραγόντων πάνω στις επιδόσεις του συστήματος.

Λέξεις Κλειδιά: *βέλτιστη επιλογή, κορυφογραμμή*, *πραγματικό σύνολο δεδομένων*

Abstract

The problematic of determining the most optimal choice pertains to the majority of the aspects of our technologically rapidly developing era. Especially when considering cases consisting of a plethora of dimensions that need to be examined, skyline queries are a valuable tool that can be used to determine the set of the most efficient options. In order to examine the possibility of achieving reliable conclusions through the application of skyline queries, the current thesis utilizes a real-world dataset of entries. In addition, various methods are presented and later implemented aiming to counteract common setbacks regarding real-world datasets such as missing values or the presence of outliers. Several modifications of known algorithmic models that are used to determine the skyline are also proposed in order to apply sophisticated queries on the chosen set of data. Finally, through the parallel monitoring of the resources which are being allocated while determining the skyline, the current thesis examines the effect that different factors may have on the system's performance metrics.

Keywords: *optimal choice, skyline queries, real-world dataset*

1 *ΕΙΣΑΓΩΓΗ*

1.1 *Οι επερωτήσεις στην επιστήμη των υπολογιστών.*

Αν και ο επίσημος ορισμός των επερωτήσεων θα μπορούσε θεωρητικά να ξενίσει την πληθώρα όσων εκτίθενται για πρώτη φορά στην ονομασία τους, αναμφίβολα η παρουσία τους στην καθημερινότητα και οι τρόποι χρήσης τους θα προκαλούσαν έκπληξη στους περισσότερους. Ως επερώτηση ορίζεται κάθε είδους εντολή προς ανάκτηση συγκεκριμένων πληροφοριών δοθέντος ενός συνόλου δεδομένων, καρπός της επεξεργασίας του οποίου θα είναι η ζητούμενη πληροφορία. Ένα παράδειγμα από την φυσική ζωή θα μπορούσε να θεωρηθεί η επίσκεψη σε ένα εστιατόριο και η νύξη προς τον σερβιτόρο για την ημερήσια λίστα των διαθέσιμων γευμάτων, κατά την οποία ο πελάτης αποσκοπεί στην απόκτηση της πληροφορίας (του σημερινού μενού) από ένα σύνολο δεδομένων (όλα τα προς διάθεση πιάτα του εστιατορίου). Σε ό,τι αφορά την καθημερινότητα, οι περισσότεροι υποβάλλουν επερωτήματα ακόμα και χωρίς να το αντιλαμβάνονται. Μια απλή αναζήτηση σε κάποιον ιστότοπο με σκοπό την ανεύρεση ορισμένης πληροφορίας εφαρμόζει αντίστοιχο επερώτημα στην εκάστοτε βάση δεδομένων και επιφέρει, κατ' επέκταση, τα ανάλογα αποτελέσματα.

Το είδος των επερωτημάτων, ωστόσο, διαφέρει ανάλογα με την πολυπλοκότητα τους και τον όγκο της πληροφορίας που θα πρέπει να επεξεργαστεί προκειμένου να σχηματιστεί η ζητούμενη εκροή. Δεν θα ήταν δύσκολο να αναλογιστεί κανείς πως η ανεύρεση ενός τηλεφωνικού αριθμού στον τηλεφωνικό κατάλογο βάσει ενός ονόματος θα απαιτούσε αντικειμενικά ελάχιστους υπολογιστικούς σε χρόνο πόρους εν συγκρίσει του υπολογισμού της οικονομικότερης διαδρομής ανάμεσα σε δυο αεροπορικούς προορισμούς. Η βασική διαφορά των δυο αυτών επερωτημάτων είναι πως στην δεύτερη περίπτωση η διαδικασία που απαιτείται για την παραγωγή των

αποτελεσμάτων βασίζεται στην εξέταση πληθώρας δεδομένων καθώς και στην εφαρμογή συγκεκριμένου αλγοριθμικού υπόβαθρου που θα υποδείξει την ζητούμενη εκροή βέλτιστης διαδρομής, σε αντίθεση με την αναζήτηση εντός του καταλόγου όπου το μόνο που απαιτείται είναι ο εντοπισμός του ανάλογου ονόματος βάσει αλφαβητικής διάταξης. Διαφαίνεται, με αυτόν τον τρόπο, το κόστος σε υπολογιστικούς πόρους που θα μπορούσε να επιφέρει η εφαρμογή κάποιου επερωτήματος. Ιδιαίτερα μάλιστα σε περιπτώσεις όπου οι επερωτήσεις αφορούν την παροχή υπηρεσιών, με πλέον σύνηθες το παράδειγμα των μηχανών αναζήτησης, σημειώνεται ανάγκη διεξαγωγής των επερωτημάτων με τον βέλτιστο δυνατό τρόπο, καθώς η διαδικασία αυτή συνδέεται άμεσα με τα κέρδη της αντίστοιχης επιχείρησης. Ο τρόπος, συμπερασματικά, με τον οποίον θα αντιμετωπιστεί η υποβολή ενός επερωτήματος είναι άρρηκτα συνδεδεμένος με τους υπολογιστικούς πόρους που θα διατεθούν για την περάτωση του.

Μέσω της επιστήμης των υπολογιστών ερευνώνται οι πλέον αποτελεσματικοί τρόποι διεξαγωγής των επερωτημάτων βάσει του είδους τους. Η ανάπτυξη καινοτόμων αλγοριθμικών μοντέλων με σκοπό την βελτιστοποίηση των διαδικασιών που απαιτούνται για την παραγωγή των ζητούμενων εκροών κάθε επερωτήματος οδηγεί στην εξοικονόμηση υπολογιστικών πόρων μέσω της εφαρμογής τους στα δεδομένα της εκάστοτε περίπτωσης. Ως απόρροια της ένταξης τέτοιων αλγοριθμικών μοντέλων στην διαχείριση επερωτημάτων, βελτιώνονται τόσο η εμπειρία του ερωτώντα, ο οποίος θα λαμβάνει την ζητούμενη εκροή σε συντομότερο χρονικό διάστημα, καθώς και η απόδοση του ερωτώμενου συστήματος, το οποίο θα είναι σε θέση να διαθέτει ελάχιστους υπολογιστικούς πόρους για την εξυπηρέτηση του αντίστοιχου επερωτήματος.

1.2 *Αντικείμενο διπλωματικής*

Στην παρούσα διπλωματική θα μελετηθεί η διαδικασία επεξεργασίας πραγματικού συνόλου δεδομένων εγγραφών αγοραπωλησίας μεταχειρισμένων οχημάτων με σκοπό τον προσδιορισμό της κορυφογραμμής του συνόλου στην γλώσσα προγραμματισμού java. Η έννοια της κορυφογραμμής αφορά το υποσύνολο εκείνο των βέλτιστων εγγραφών του αρχικού συνόλου βάσει δοθέντων παραγόντων με τους οποίους οι επιμέρους αυτές εγγραφές θα χαρακτηρίζονται κυρίαρχες έναντι όλων των υπολειπόμενων. Στην περίπτωση που κάθε εγγραφή αποτελείται από ένα χαρακτηριστικό, η έννοια της κορυφογραμμής εκφυλίζεται στην εύρεση των εγγραφών εκείνων με την καλύτερη τιμή σε ό,τι αφορά το χαρακτηριστικό αυτό. Παρόλα αυτά η σημασία της κορυφογραμμής διαφαίνεται υπό την παρουσία πληθώρας χαρακτηριστικών, βάσει των οποίων προσδιορίζεται το υποσύνολο εκείνων των εγγραφών με τα βέλτιστα χαρακτηριστικά. Πρέπει να σημειωθεί, για τον

λόγο αυτό, πως το επιλεχθέν σύνολο δεδομένων αφορά πολυδιάστατες σε χαρακτηριστικά εγγραφές με σκοπό την ουσιαστική μελέτη του συνόλου της κορυφογραμμής.

Θα μπορούσε κανείς να αντιληφθεί την σημασία του υπολογισμού της κορυφογραμμής πολυδιάστατων στοιχείων σε παραδείγματα αγορών αγαθών από την καθημερινότητα. Πολλές φορές οι υποκειμενικές προτιμήσεις των καταναλωτών επηρεάζουν σημαντικά το σύνολο των ιδεατών υποψήφιων προς αγορά προϊόντων. Συχνά, λοιπόν, οι καταναλωτές καλούνται να επιλέξουν ανάμεσα σε κάποιον συνδυασμό χαρακτηριστικών όπως τιμή, ποιότητα και ποσότητα και, κατόπιν, βάσει του συνδυασμού αυτού να καταλήξουν σε ένα προϊόν που τους ικανοποιεί. Δεδομένου πως δεν υπάρχει πάντα ξεκάθαρη απάντηση σε ό,τι αφορά τον βέλτιστο συνδυασμό, η εύρεση της κορυφογραμμής συνεπάγεται τον καθορισμό του συνόλου όλων των βέλτιστων συνδυασμών των προς σύγκριση χαρακτηριστικών που θα μπορούσαν να σχηματιστούν. Κατά αυτόν τον τρόπο η κορυφογραμμή επιλύει το πρόβλημα της εύρεσης των βέλτιστων επιλογών ανάμεσα σε στοιχεία που χαρακτηρίζονται από πληθώρα διαστάσεων.

Αντικείμενο της παρούσας διπλωματικής εργασίας θα αποτελέσει η υλοποίηση λογικής στη γλώσσα προγραμματισμού java που θα αφορά την προσπέλαση του επιλεχθέντος συνόλου πραγματικών δεδομένων και την εφαρμογή σε αυτό επερωτημάτων κορυφογραμμής με σκοπό την εύρεση των βέλτιστων εγγραφών ως προς διαφορετικούς παράγοντες τη φορά. Παράλληλα θα εφαρμόζεται ενδελεχής εποπτεία των υπολογιστικών πόρων που διατίθενται από το σύστημα για την περάτωση των εν λόγω επερωτημάτων. Οι επερωτήσεις κορυφογραμμής πρόκειται να προσεγγιστούν υπό την σκοπιά διαφορετικών αλγοριθμικών μοντέλων το καθένα από τα οποία θα προσαρμόζεται ανάλογα με το σύνολο δεδομένων προς επεξεργασία. Βάση του αλγοριθμικού υπόβαθρου που θα χρησιμοποιηθεί εκτενώς κατά την εφαρμογή των επερωτημάτων κορυφογραμμής αποτελεί μια ιδιαίτερη προσέγγιση των Borzsonyi et al.[1] υπο την ονομασία Block Nested Loop. Επιπροσθέτως, τα ίδια τα επερωτήματα κορυφογραμμής που πρόκειται να εφαρμοστούν θα ποικίλλουν, λαμβάνοντας υπόψιν διαφορετικές διαστάσεις του αρχικού συνόλου των δεδομένων καθώς και εναλλακτικές εισροές από τον χρήστη.

1.3 *Στόχοι προς επίτευξη*

Σκοπός της παρούσας διπλωματικής είναι η μελέτη της εφαρμογής επερωτημάτων κορυφογραμμής πάνω σε σύνολο πραγματικών δεδομένων. Προς επίτευξη της μελέτης αυτής πρέπει να προηγηθεί

[1] (Borzsonyi, 2001)

επεξεργασία του αρχικού συνόλου των δεδομένων η οποία θα συμβάλλει στην προετοιμασία και μορφοποίηση του με σκοπό την αποδοτική του χρήση από την λογική επερωτημάτων κορυφογραμμής. Η υλοποίηση σε java της λογικής της επεξεργασίας αυτής πρέπει να δέχεται ως εισροή το αρχικό σύνολο των δεδομένων και να παράγει την τροποποίηση του, έχοντας διατηρήσει μόνο τα απαραίτητα για τον υπολογισμό της κορυφογραμμής δεδομένα. Ακόλουθο ζητούμενο είναι η ανάπτυξη λογισμικού σε java του κύριου μέρους του αλγοριθμικού μοντέλου υπολογισμού της κορυφογραμμής, το οποίο θα λαμβάνει το μορφοποιημένο σύνολο δεδομένων και θα επιστρέφει το σύνολο εγγραφών κορυφογραμμής. Σε ό,τι αφορά την διεξαγωγή των επερωτημάτων σκοπός της παραπάνω υλοποίησης είναι η δυνατότητα προσαρμογής των παραγόντων ως προς τους οποίους υπολογίζεται το σύνολο της ζητούμενης κορυφογραμμής. Παράλληλα, με στόχο την εξέταση της συμπεριφοράς του συστήματος κατά την διάρκεια της διαδικασίας υπολογισμού της κορυφογραμμής αναγκαία είναι η ανάπτυξη μηχανισμού που εποπτεύει τον όγκο των υπολογιστικών πόρων που έχουν διατεθεί ενόσω δοκιμάζεται η αντίστοιχη υλοποίηση. Τέλος απαραίτητη είναι η τροποποίηση κατά περίπτωση του κύριου μηχανισμού υπολογισμού κορυφογραμμής ούτως ώστε να είναι δυνατή η εφαρμογή εναλλακτικών επερωτημάτων με σκοπό την σε βάθος ανάλυση των επιλεχθέντων αλγοριθμικών μοντέλων καθώς και της απόδοσης τους πάνω στο πραγματικό σύνολο δεδομένων εισροής.

1.4 *Δομή της διπλωματικής*

Στο δεύτερο κεφάλαιο λαμβάνει χώρα η περιγραφή και η στατιστική ανάλυση του συνόλου των δεδομένων των αγοραπωλησιών οχημάτων που θα χρησιμοποιηθεί ως εισροή των επερωτημάτων κορυφογραμμής. Κατόπιν υλοποιείται σε java μηχανισμός που μετασχηματίζει καταλλήλως το εν λόγω σύνολο δεδομένων με σκοπό την αποτελεσματική επεξεργασία του από το κύριο τμήμα υπολογισμού της κορυφογραμμής. Στο κεφάλαιο τρία γίνεται εκτενής αναφορά τόσο στο πρόβλημα της κορυφογραμμής όσο και στο μαθηματικό υπόβαθρο πίσω από αυτήν. Επιπροσθέτως, περιγράφονται δυο αλγοριθμικά μοντέλα που χρησιμοποιούνται για την επίλυση του προβλήματος της κορυφογραμμής και υλοποιείται ο αλγόριθμος BNL που θα αποτελέσει την βάση πάνω στην οποία θα λαμβάνουν χώρα κατάλληλες τροποποιήσεις με σκοπό να εξαχθούν τα ζητούμενα αποτελέσματα. Στο τέταρτο κεφάλαιο μετά από ανάλογη τροποποίηση της λογικής του BNL εφαρμόζεται το πρώτο επερώτημα και μέσω των εξαχθέντων αποτελεσμάτων του οποίου αντλούνται συμπεράσματα σχετιζόμενα με την ύπαρξη έκτοπων εγγραφών στο σύνολο των πραγματικών δεδομένων που χρησιμοποιήθηκε ως εισροή. Το πέμπτο κεφάλαιο προτείνει δυο

αλγοριθμικές προσεγγίσεις με σκοπό την αντιμετώπιση των έκτοπων εγγραφών που εντοπίστηκαν κατά την εφαρμογή του πρώτου επερωτήματος. Ακολουθεί η υλοποίηση και η δοκιμή των δυο προσεγγίσεων αυτών ούτως ώστε να αναλυθεί η δυνατότητα χρήσης τους για την αντιμετώπιση των έκτοπων εγγραφών με ζητούμενο το σύνολο της κορυφογραμμής. Το έκτο κεφάλαιο εισάγει την χρήση ενός νέου είδους εισροών του χρήστη που αφορά σε σημεία συντεταγμένων τα οποία λαμβάνονται υπόψιν για τον υπολογισμό κορυφογραμμής που εξαρτάται πλέον και από την απόσταση της κάθε εγγραφής του αρχικού συνόλου δεδομένων από τα νέα σημεία αυτά των συντεταγμένων. Στο κεφάλαιο αυτό εμπεριέχεται και η αντίστοιχη τροποποίηση του λογισμικού σε java του BNL με στόχο την διαχείριση του νέου αρχείου σημείων συντεταγμένων και την λογική υπολογισμού της κορυφογραμμής βάσει της νέας εισροής αυτής. Στο έβδομο κεφάλαιο εντάσσεται ως παράγοντας υπολογισμού της κορυφογραμμής το πλήθος των λέξεων δοθέντος συνόλου λέξεων κλειδιών που εμπεριέχονται εντός των πεδίων κειμένου της κάθε εγγραφής του συνόλου των δεδομένων εισροής. Ακολουθεί η αντίστοιχη τροποποίηση του κώδικα σε java του BNL και η διεξαγωγή των ζητούμενων επερωτήσεων. Τέλος στο όγδοο κεφάλαιο γίνεται ανάλυση των συμπερασμάτων που μπορούν να εξαχθούν μέσω των πειραμάτων που έλαβαν χώρα στα προηγούμενα κεφάλαια που αφορούν τόσο την διάθεση υπολογιστικών πόρων για τον υπολογισμό της κορυφογραμμής όσο και την δυνατότητα χρήσης του αλγορίθμου BNL για τον ακριβή προσδιορισμό της κορυφογραμμής πραγματικού συνόλου δεδομένων.

2 *ΔΕΔΟΜΕΝΑ ΠΡΟΣ ΕΠΕΞΕΡΓΑΣΙΑ*

2.1 *Περιγραφή συνόλου δεδομένων*

Τα δεδομένα που θα χρησιμοποιηθούν για την υλοποίηση και τον έλεγχο του αλγορίθμου κορυφογραμμής έχουν ληφθεί από τον ιστότοπο **kaggle.com**. Αφορούν αγοραπωλησίες μεταχειρισμένων οχημάτων που έχουν αναρτηθεί στην εφαρμογή Craigslist, η οποία χρησιμοποιείται για την αγοραπωλησία αγαθών ανάμεσα σε ιδιώτες. Πιο συγκεκριμένα, τα δεδομένα που θα χρησιμοποιηθούν συλλέχθηκαν τον Οκτώβριο του 2018 και αφορούν 689.600 οχήματα μέσω της ιστοσελίδας **Craigslist.org**. Οι πληροφορίες που μπορεί κανείς να αντλήσει από τα δεδομένα σχετίζονται με μια πληθώρα μετρήσιμων και μη παραγόντων γύρω από τα οχήματα αυτά όπως διαφαίνεται και από τον παρακάτω πίνακα[2].

ΠΕΔΙΟ	ΠΕΡΙΓΡΑΦΗ
url	Η διεύθυνση της ιστοσελίδας που αφορά το συγκεκριμένο όχημα
City	Η πόλη στην οποία παρέχεται το όχημα
Price	Η τιμή πώλησης του οχήματος
Year	Η χρονιά κατασκευής του οχήματος
Manufacturer	Ο όμιλος κατασκευής του αυτοκινήτου
Make	Το μοντέλο του αυτοκινήτου
Condition	Η κατάσταση στην οποία πωλείται το όχημα
Cylinders	Πλήθος κυλίνδρων του κινητήρα
Fuel	Το είδος της καύσιμης ύλης
Odometer	Τιμή χιλιομετρητή του αυτοκινήτου σε χιλιόμετρα
title_status	Ο τύπος ιδιοκτησίας του οχήματος

[2] Πίνακας 1

Transmission	Το είδος του κιβωτίου ταχυτήτων
VIN	Ο αναγνωριστικός αριθμός του οχήματος
Drive	Τύπος άξονα του αυτοκινήτου
Size	Η κατηγορία μεγέθους του οχήματος
Type	Η κατηγορία του οχήματος
paint_color	Το χρώμα του αυτοκινήτου
image_url	Η διεύθυνση ιστοσελίδας με απεικόνιση του οχήματος
Lat	Γεωγραφικό πλάτος στο οποίο προσφέρεται το όχημα
Long	Γεωγραφικό μήκος στο οποίο προσφέρεται το όχημα

Πίνακας 1 Περιγραφές των διακεκριμένων πεδίων του συνόλου εγγραφών

Με σκοπό την αποτελεσματικότερη επεξεργασία των δεδομένων, στην §2.3 θα προταθεί αλγόριθμος απομόνωσης των σημαντικών πεδίων εκείνων που θα διευκολύνουν την διαδικασία εφαρμογής των προς εξέταση αλγοριθμικών μοντέλων.

Με σκοπό την διατήρηση των σημαντικών πεδίων εκείνων που θα χρησιμοποιηθούν ως καθοριστικοί παράγοντες της κορυφογραμμής του συνόλου των πραγματικών εγγραφών, στην §2.3 θα προταθεί υλοποίηση σε java λογικής που απομονώνει τα πεδία αυτά. Τα σημαντικά χαρακτηριστικά θα μεγιστοποιούνται ή θα ελαχιστοποιούνται βάσει της βέλτιστης τιμής του κάθε πεδίου, ούτως ώστε να προσδιοριστεί το υποσύνολο των βέλτιστων σε αυτά εγγραφών. Πιο συγκεκριμένα, το αρχείο εισροών που εμπεριέχει τις άνωθεν εγγραφές θα δίνεται σε μορφή επέκτασης .csv. Η επέκταση αυτή αποθηκεύει κάθε εγγραφή σε ξεχωριστή σειρά, ενώ οι τιμές των πεδίων της εν λόγω εγγραφής διαχωρίζονται μεταξύ τους με τον χαρακτήρα “,” κόμμα.

Η συγκεκριμένη διαδικασία προ-επεξεργασίας κατά την οποία απομονώνονται τα ζητούμενα πεδία πρόκειται να απομονώσει 17 από τα συνολικά 20 πεδία, καθώς και να προσαρτήσει μοναδικό αναγνωριστικό αριθμό σε κάθε εγγραφή. Τέλος θα προσαρμόσει ανάλογα και την σειρά των πεδίων αυτών.

Αρχικά πεδία	Προσαρμοσμένα πεδία
url	id
city	city
price	price
year	year
manufacturer	odometer
make	lat
condition	long
cylinders	manufacturer
fuel	make
odometer	condition
title_status	cylinders
transmission	fuel
VIN	title_status
drive	transmission
size	drive
type	size
paint_color	type
image_url	paint_color
lat	
long	

Πίνακας 2 Ζητούμενες εκροές του σταδίου προ επεξεργασίας του συνόλου δεδομένων

2.2 *Στατιστική ανάλυση συνόλου δεδομένων*

URL

Το πεδίο url περιλαμβάνει σύνδεσμο που ανακατευθύνει στην ιστοσελίδα που αφορά κάθε επιμέρους εγγραφή αγοραπωλησίας του αντίστοιχου οχήματος. Συνολικά σημειώνονται 689600 υπαρκτοί σύνδεσμοι ενώ δεν υπάρχουν ελλιπείς σε αυτό το πεδίο εγγραφές.

CITY

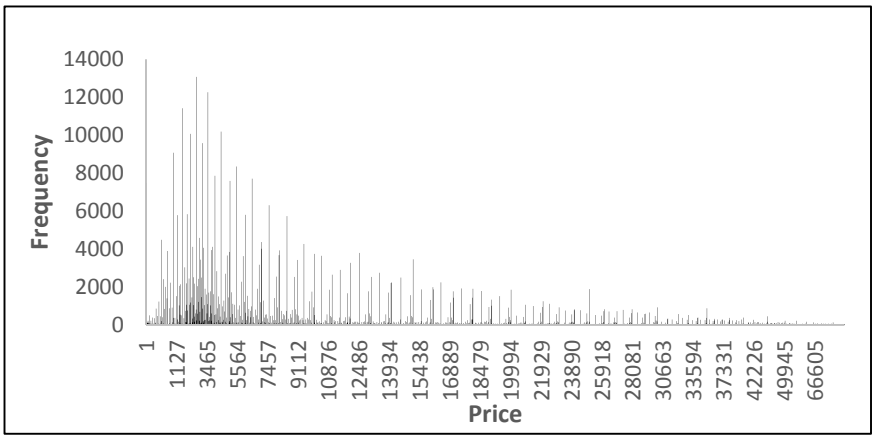
Σε ό,τι αφορά το πεδίο city συνολικά απαντώνται 689600 εγγραφές χωρίς να σημειώνονται ελλιπείς. Ανάμεσα σε αυτές παρατηρούνται 480 διακεκριμένα ονόματα πόλεων. Οι πόλεις με την μεγαλύτερη συχνότητα είναι οι Grandrapids και Cosprings, με συχνότητα 2901 αμφότερες. Η δε πόλη με την μικρότερη συχνότητα εμφάνισης τιμής 1, είναι το Acapulco.

Εικόνα 1 Ραβδόγραμμα συχνοτήτων των διακεκριμένων τιμών του πεδίου city

abbotsford	boston	corvallis	fortwayne	jackson	littlerock	myrtlebeach	peace	salina	stgeorge	washingtondc
abilene	boulder	cosprings	frederick	jacksontn	logan	nacogdoches	pei	saltlakecity	stillwater	waterloo
acapulco	bozeman	csd	fredericksburg	jacksonville	londonon	nanaimo	pennstate	sanangelo	stjoseph	watertown
akroncanton	brainerd	dallas	fresno	janesville	longisland	nashville	pensacola	sanantonio	stlouis	wausau
albany	brantford	danville	ftmcmurray	jerseyshore	losangeles	natchez	peoria	sandiego	stockton	wenatchee
albanyga	brownsville	dayton	gadsden	jonesboro	louisville	nd	peterborough	sandusky	sudbury	westernmass
albuquerque	brunswick	daytona	gainesville	joplin	loz	nesd	philadelphia	sanmarcos	sunshine	westky
allentown	buffalo	decatur	galveston	juarez	lubbock	newbrunswick	phoenix	santabarbara	susanville	westmd
altoona	butte	delaware	glensfalls	juneau	lynchburg	newfoundland	pittsburgh	santafe	swks	westslope
amarillo	calgary	delrio	goldcountry	jxn	macon	newhaven	plattsburgh	santamaria	swmi	wheeling
ames	capecod	denver	grandforks	kalamazoo	madison	newjersey	poconos	sarasota	swv	whistler
anchorage	carbondale	desmoines	grandisland	kalispell	maine	newlondon	porthuron	sarnia	swva	whitehorse
annapolis	cariboo	detroit	grandrapids	kamloops	mankato	neworleans	portland	saskatoon	syracuse	wichita
annarbor	catskills	dothan	greatfalls	kansascity	mansfield	newyork	potsdam	savannah	tallahassee	wichitafalls
appleton	cedarrapids	dubuque	greenbay	kelowna	marshall	nh	prescott	scottsbluff	tampa	williamsport
asheville	cenla	duluth	greensboro	kenai	martinsburg	niagara	princegeorge	scranton	terrehaute	wilmington
ashtabula	centralmich	eastco	greenville	keys	masoncity	nmi	providence	sd	territories	winchester
athensga	cfl	easternshore	guadalajara	killeen	mattoon	norfolk	provo	seattle	texarkana	windsor
athensohio	chambana	eastidaho	guanajuato	kingston	mazatlan	northernwi	pueblo	seks	texoma	winnipeg
atlanta	chambersburg	eastky	guelph	kirksville	mcallen	northmiss	pueblo	semo	thumb	winstonsalem
auburn	charleston	eastnc	gulfport	kitchener	meadville	northplatte	pullman	sfbay	thunderbay	worcester
augusta	charlestonwv	eastoregon	halifax	klamath	medford	nwct	pv	sheboygan	tijuana	wv
austin	charlotte	easttexas	hamilton	knoxville	memphis	nwga	quadcities	sherbrooke	tippecanoe	wyoming
bajasur	charlottesville	esauclaire	hanford	kokomo	mendocino	nwks	quebec	shoals	toledo	yakima
bakersfield	chatham	edmonton	harrisburg	kootenays	merced	oaxaca	quincy	showlow	topeka	york
baltimore	chattanooga	elko	harrisonburg	kpr	meridian	ocala	racine	shreveport	toronto	youngstown
barrie	chautauqua	elmira	hartford	ksu	mexicocity	odessa	raleigh	sierravista	treasure	yubasutter
batonrouge	chicago	elpaso	hat	lacrosse	miami	ogden	rapidcity	siouxcity	tricities	yucatan
battlecreek	chico	enid	hattiesburg	lafayette	milwaukee	okaloosa	reading	siouxfalls	troisrivieres	yuma
beaumont	chihuahua	erie	helena	lakecharles	minneapolis	oklahomacity	reddeer	siskiyou	tucson	zanesville
belleville	chillicothe	eugene	hermosillo	lakecity	missoula	olympic	redding	skagit	tulsa	
bellingham	cincinnati	evansville	hickory	lakeland	mobile	omaha	regina	skeena	tuscaloosa	
bemidji	clarksville	fairbanks	hiltonhead	lancaster	modesto	oneonta	reno	slo	tuscarawas	
bend	cleveland	fargo	holland	lansing	mohave	onslow	richmond	smd	twinfalls	
bgky	clovis	farmington	houma	laredo	monroe	orangecounty	richmondin	soo	twintiers	
bham	cnj	fayar	houston	lasalle	monroemi	oregoncoast	rmn	southbend	up	
bigbend	collegestation	fayetteville	hudsonvalley	lascruces	montana	orlando	roanoke	southcoast	utica	
billings	columbia	fingerlakes	humboldt	lasvegas	monterey	ottawa	rochester	southjersey	valdosta	
binghamton	columbiamo	flagstaff	huntington	lawrence	monterrey	ottumwa	rockford	spacecoast	vancouver	
bismarck	columbus	flint	huntsville	lawton	montgomery	outerbanks	rockies	spokane	ventura	
blacksburg	columbusga	florencesc	imperial	lethbridge	montreal	owensboro	roseburg	springfield	vermont	
bloomington	comoxvalley	fortcollins	indianapolis	lewiston	morgantown	owensound	roswell	springfieldil	victoria	
bn	cookeville	fortdodge	inlandempire	lexington	moseslake	palmsprings	sacramento	statesboro	victoriatx	
boise	cornwall	fortmyers	iowacity	limaohio	muncie	panamacity	saginaw	staugustine	visalia	
boone	corpuschristi	fortsmith	ithaca	lincoln	muskegon	parkersburg	salem	stcloud	waco	

Πίνακας 3 Διακεκριμένες τιμές ονομάτων του πεδίου city

PRICE



Πίνακας 4 Ραβδόγραμμα συχνοτήτων των τιμών του πεδίου price

Οι υπαρκτές τιμές του πεδίου price ανέρχονται σε 689600 ενώ δεν σημειώνονται ελλιπείς εγγραφές. Από αυτές υπάρχει σύνολο 19836 διακεκριμένων τιμών. Συνηθέστερη τιμή είναι αυτή των $2500 και 7968 τιμές παρουσιάζουν την ελάχιστη συχνότητα 1. Η μέγιστη τιμή του πεδίου price είναι $4294967295 και η ελάχιστη $1, διαμορφώνοντας εύρος 4294967294. Το γεγονός αυτό εγείρει υποψίες περί ύπαρξης έκτοπων τιμών των εγγραφών.

Price Descriptive	
Mean	**431846.2739**
Standard Error	**41268.92148**
Median	**6995**
Mode	**2500**
Standard Deviation	**34270602.97**
Sample Variance	**1.17447E+15**
Kurtosis	**10372.37074**
Skewness	**97.34554927**
Range	**4294967294**
Minimum	**1**
Maximum	**4294967295**
Confidence Level(95.0%)	**80885.74175**
Count	**689600**

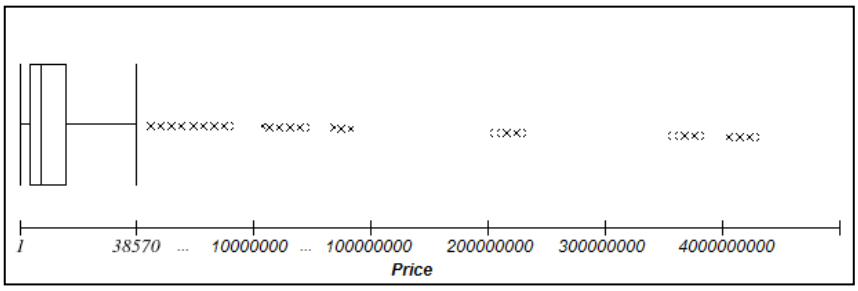
Πίνακας 5 Στατιστικά μεγέθη του πεδίου price

Σε ό,τι αφορά τα μέτρα κεντρικής τάσης, ο μέσος όρος τιμών είναι τα 431846.27 δολάρια. Η διάμεσος είναι 6995 ενώ, όπως ειπώθηκε, η επικρατούσα τιμή είναι τα $2500. Το τυπικό σφάλμα του μέσου ανέρχεται σε 41268.92, γεγονός που ελαττώνει την αξιοπιστία του μέσου όρου. Η τυπική

απόκλιση υπολογίζεται σε 34270602.97, κάτι που υποδηλώνει πως το σύνολο των τιμών του πεδίου price απέχει σημαντικά από την διάμεσο των $6995.

Συμπεράσματα για τον διασκορπισμό των τιμών γύρω από την διάμεσο θα αντληθούν επίσης μέσω των μέτρων διασποράς. Η κύρτωση του συνόλου είναι 10372.37074, διαμορφώνοντας λεπτόκυρτη καμπύλη κατανομής και εντείνοντας τις υποψίες περί ακραίως έκτοπων τιμών. Η λοξότητα έχει τιμή 97.34554927, επιβεβαιώνοντας την θετική ασυμμετρία της κατανομής.

Προκειμένου να αντληθούν περεταίρω πληροφορίες περί της εμφάνισης ακραίων τιμών θα ακολουθηθεί η μέθοδος φραγμάτων Tukey[3] για τον χαρακτηρισμό υποσυνόλου των τιμών ως έκτοπες. Πιο συγκεκριμένα μετά από υπολογισμό, οι τιμές των τεταρτημορίων ανέρχονται σε Q1=3200, Q2=6995, Q3=14990, Q4=4294967295. Ως αποτέλεσμα, το ενδοτεταρτημοριακό εύρος IQR=11790 θα οδηγήσει στον καθορισμό των ορίων εκείνων που θα χαρακτηρίσουν τις κατά Tukey άτοπες τιμές. Το διάστημα εκτός του οποίου θα παρουσιάζονται οι έκτοπες τιμές θα είναι το [Q1-1.5IQR, Q3+1,5IQR]=[0, 20885]. Συμπερασματικά 103.360 είναι έκτοπες κατά Tukey. Επιπροσθέτως, ακραίως έκτοπες κατά Tukey θα θεωρούνται οι τιμές εκείνες εκτός του διευρυμένου διαστήματος [Q1-3(IQR), Q3+3IQR]=[0, 38570], αποκλείοντας έτσι εν συνόλω 20562 εγγραφές.

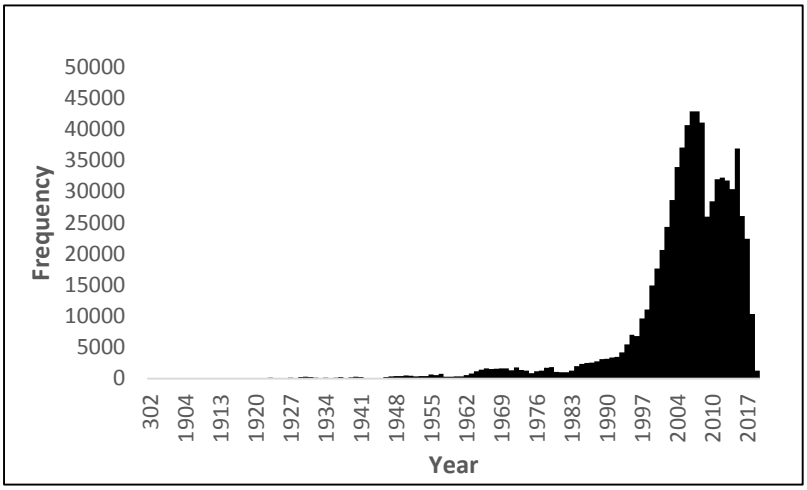


Εικόνα 2 Θηκόγραμμα του πεδίου price κατόπιν της εφαρμογής φραγμάτων Tukey

Ως απόρροια των παραπάνω, το σύνολο τιμών του πεδίου price αποτελείται από πληθώρα διακεκριμένων τιμών οι 3180 εκ των οποίων θεωρούνται ακραία άτοπες κατά Tukey.

[3] (Neil C. Schwertman, 2007)

YEAR

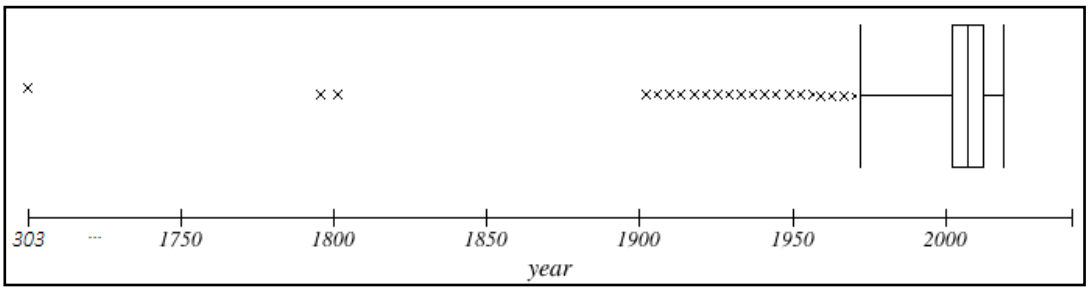


Εικόνα 3 Ραβδόγραμμα συχνοτήτων των διακεκριμένων τιμών του πεδίου year

Σε ό,τι αφορά το πεδίο του έτους κυκλοφορίας Year, σε αυτό απαντώνται 688993 υπαρκτές τιμές ενώ οι υπόλοιπες 307 είναι ελλιπείς στο πεδίο αυτό. Το πλήθος των διακεκριμένων τιμών του πεδίου έτους κυκλοφορίας ανέρχεται σε 123. Ως προς τα μέτρα κεντρικής τάσης, η μέση τιμή του έτους είναι τα 2004.56 έτη, ενώ η διάμεσος και η επικρατούσα τιμή είναι τα 2007 έτη. Υποδεικνύεται με αυτόν τον τρόπο αρνητική ασυμμετρία των τιμών του έτους. Η μέγιστη τιμή είναι τα 2019 έτη και η ελάχιστη τα 302 διαμορφώνοντας κατ' αυτόν τον τρόπο εύρος ύψους 1717. Συμπερασματικά, ενδέχεται στο σύνολο δεδομένων να απαντώνται εγγραφές με έκτοπες τιμές. Περισσότερες πληροφορίες θα δοθούν από τις τιμές των μέτρων διασποράς.

Η τυπική απόκλιση υπολογίζεται σε 12.40. Οι έκτοπες τιμές που διαφέρουν στατιστικά σημαντικά θα εξαχθούν μέσω της μεθόδου Tukey[4]. Πιο συγκεκριμένα, οι τιμές των αντίστοιχων τεταρτημόριων είναι Q1=2002, Q2=2007, Q3=2012, Q4=2019. Το ενδοτεταρτημοριακό εύρος IQR=Q3-Q1=10. Ως έκτοπες τιμές θα θεωρούνται αυτές εκτός του πεδίου [Q1-1.5(IQR), Q3+1, 5IQR]=[1987, 2027]. Τέλος άτοπες τιμές θα θεωρούνται όλες οι τιμές εκτός του πεδίου [Q1-3(IQR), Q3+3IQR]=[1972, 2042]. Συμπερασματικά, υπάρχει σύνολο 25841 εγγραφών που σημειώνουν έτος κυκλοφορίας μικρότερο της τιμής 1972 που είναι άτοπες κατά Tukey.

[4] (Neil C. Schwertman, 2007)



Εικόνα 4 Θηκόγραμμα του πεδίου year κατόπιν της εφαρμογής φραγμάτων Tukey

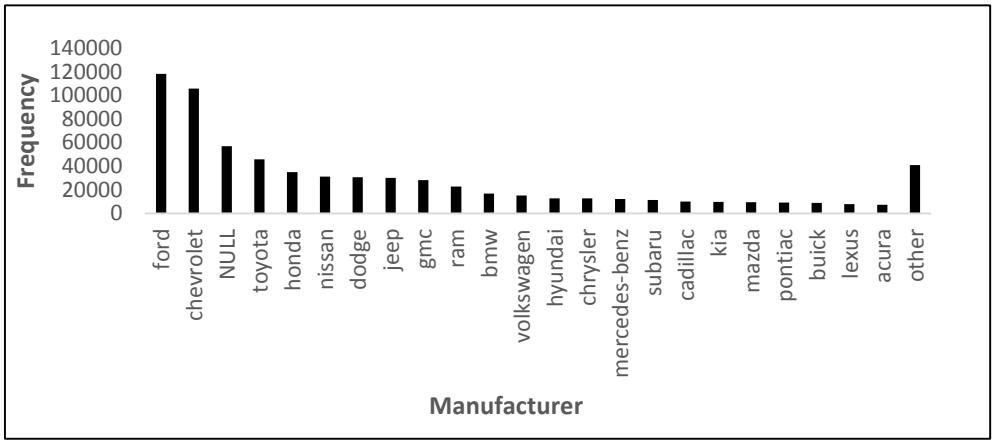
Η λοξότητα ανέρχεται σε -6.29 και δεδομένης της αρνητικής αυτής τιμής συμπεραίνεται πως το σύνολο των τιμών συσσωρεύονται στην δεξιά πλευρά του αντίστοιχου θηκογράμματος και επιβεβαιώνεται η αρνητική ασυμμετρία. Η κύρτωση υπολογίζεται σε 524.18, κάτι που προσδίδει λεπτόκυρτο χαρακτήρα στην καμπύλη κατανομής.

Εν συνόλω, σε ό,τι αφορά τις τιμές του πεδίου year των εγγραφών, παρατηρούνται 307 ελλιπείς εγγραφές και 25841 εγγραφές με άτοπες τιμές έτους κυκλοφορίας. Οι υπόλοιπες κυμαίνονται στο διάστημα [1972, 2019] με συνηθέστερη τιμή το έτος 2007.

YEAR descriptive	
Mean	**2004.562961**
Standard Error	**0.014940493**
Median	**2007**
Mode	**2007**
Standard Deviation	**12.40414621**
Sample Variance	**153.8628431**
Kurtosis	**524.1854044**
Skewness	**-6.294830899**
Range	**1717**
Minimum	**302**
Maximum	**2019**
Sum	**1381731217**
Count	**689293**
Confidence Level(95.0%)	**0.029282881**

Πίνακας 6 Στατιστικά μεγέθη του πεδίου year

MANUFACTURER



Εικόνα 5 Ραβδόγραμμα συχνοτήτων των τιμών του πεδίου manufacturer

Το πεδίο manufacturer που αφορά τον κατασκευαστή του οχήματος κάθε εγγραφής, απαντώνται 632725 υπαρκτές τιμές, ενώ 56875 είναι ελλιπείς. Από αυτές, σημειώνονται 44 διακριτές τιμές κατασκευαστών με συνηθέστερη αυτήν της Ford συχνότητας 118249. Εν τέλει, ελάχιστη συχνότητα τιμής 1 παρουσιάζουν αμφότερες οι Hennessey και Noble.

MANUFACTURER	FREQUENCY	MANUFACTURER	FREQUENCY	MANUFACTURER	FREQUENCY
hennessey	1	mercury	4133	chrysler	12686
noble	1	volvo	4450	hyundai	12899
morgan	4	mitsubishi	4506	volkswagen	15101
land rover	32	infiniti	4899	bmw	16804
aston-martin	37	lincoln	5343	ram	22722
porche	49	audi	5733	gmc	28213
ferrari	76	acura	7388	jeep	30170
alfa-romeo	91	lexus	7787	dodge	30630
datsun	274	buick	9102	nissan	31322
harley-davidson	335	pontiac	9296	honda	35008
fiat	787	mazda	9424	toyota	45783
jaguar	1725	kia	9733	Null	56875
rover	2101	cadillac	10080	chevrolet	105815
mini	2493	subaru	11406	ford	118249
saturn	3764	mercedes-benz	12273		

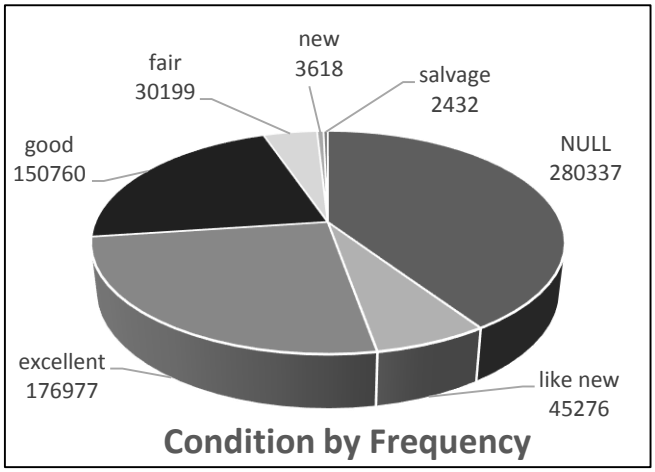
Πίνακας 7 Συχνότητες των διακεκριμένω τιμών του πεδιου manufacturer

MAKE

Σε ό,τι αφορά το πεδίο make που περιγράφει το μοντέλο του οχήματος κάθε εγγραφής, σε αυτό σημειώνονται 54438 διακεκριμένες εγγραφές. Συνολικά οι 29099 εγγραφές είναι ελλιπείς στο πεδίο αυτό, ενώ οι υπόλοιπες 660502 είναι υπαρκτές και περιέχουν σειρές αλφαριθμητικών. Ως προς την συχνότητα, η πιο σύνηθης εγγραφή είναι αυτή των "1500" συχνότητας 9501, ενώ 35175 άλλες εγγραφές παρουσιάζουν την ελάχιστη συχνότητα 1.

CONDITION

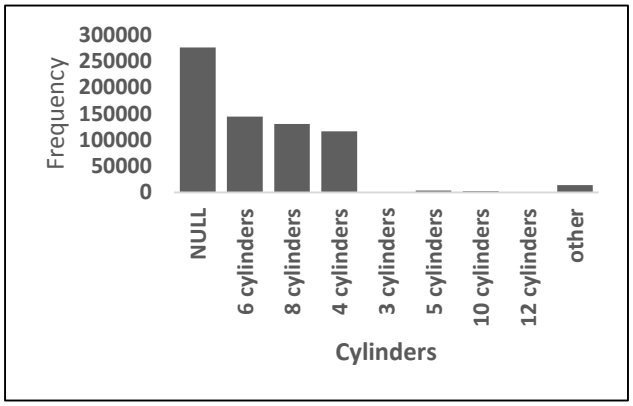
Στο πεδίο condition, που αφορά την κατάσταση στην οποία βρίσκεται το όχημα κατά την αγοραπωλησία, απαντώνται συνολικά 409263 υπαρκτές εγγραφές, ενώ οι υπόλοιπες 280337 είναι ελλιπείς. Το πλήθος των διακεκριμένων τιμών των εγγραφών του πεδίου condition ανέρχεται σε 7. Η συνηθέστερη τιμή συχνότητας 176977 είναι η "excellent" ενώ σπανιότερη είναι η τιμή "salvage" συχνότητας 2432.



Εικόνα 6 Γράφημα πίτας συχνότητας των διακεκριμένων τιμών του πεδίου condition

CYLINDERS

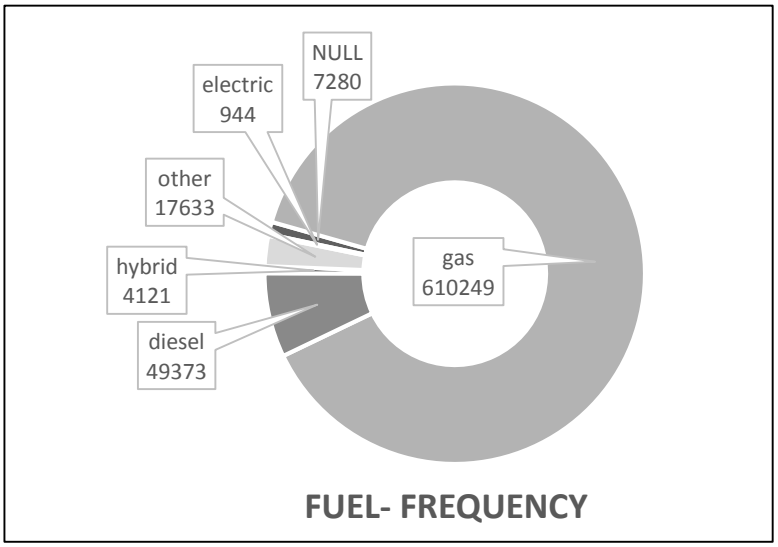
Το πεδίο Cylinders, που περιέχει το πλήθος των κυλίνδρων του κινητήρα του οχήματος της κάθε εγγραφής, απαντώνται 413323 τιμές, ενώ οι υπόλοιπες 276278 είναι ελλιπείς. Σε αυτές σημειώνονται 9 διακεκριμένες τιμές με την πλέον συνηθέστερη αυτήν των 6 κυλίνδρων συχνότητας 144299. Τέλος, η σπανιότερη τιμή είναι αυτή των 12 κυλίνδρων συχνότητας 326.



Εικόνα 7 Ραβδόγραμμα συχνοτήτων των τιμών του πεδίου cylinders

FUEL

Το πεδίο Fuel περιέχει τον τύπο καυσίμου του οχήματος της κάθε εγγραφής. Συνολικά σημειώνονται 682320 υπαρκτές εγγραφές ενώ οι υπόλοιπες 7280 είναι ελλιπείς. Οι διακεκριμένοι τύποι καυσίμων είναι 6 στο πλήθος. Συνηθέστερο καύσιμο συχνότητας 610249 είναι η βενζίνη καταλαμβάνοντας την συντριπτική πλειοψηφία του 88% των εγγραφών. Σπανιότερος τύπος καυσίμου εν τέλει είναι ο ηλεκτρικός συχνότητας 944.



Εικόνα 8 Διάγραμμα δακτυλίου συχνοτήτων των τιμών του πεδίου fuel

ODOMETER

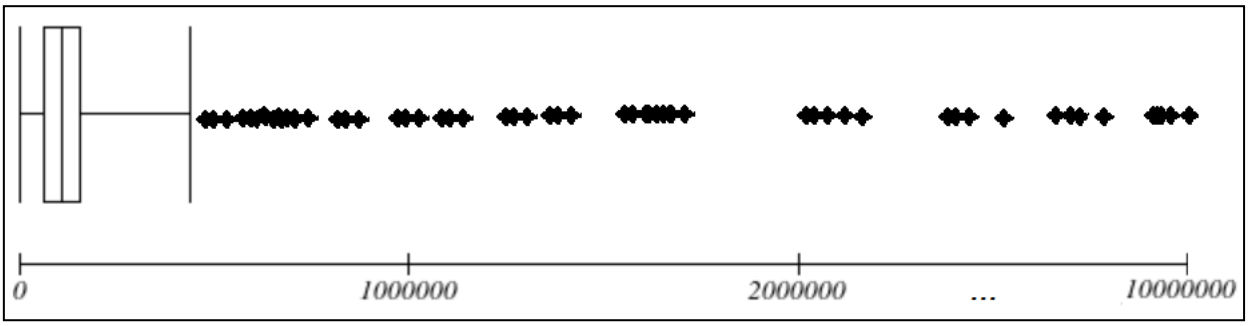
Σε ό,τι αφορά το πεδίο odometer, αυτό περιλαμβάνει την απόσταση σε χιλιόμετρα την οποία διάνυσε το όχημα της αντίστοιχης εγγραφής πριν από την εν λόγω αγοραπωλησία. Συνολικά σημειώνονται 454602 υπαρκτές τιμές στο πεδίο αυτό ενώ οι υπόλοιπες 234998 τιμές είναι ελλιπείς. Σε αντίθεση, σημειώνονται 58201 τιμές που εμφανίζονται μια φορά. Το πλήθος των διακεκριμένων τιμών του πεδίου αυτού ανέρχεται σε 111326. Η συνηθέστερη τιμή είναι η τιμή των 150000 χιλιομέτρων, συχνότητας 2787. Σε αντίθεση, σημειώνονται 58201 τιμές που εμφανίζονται μοναδική φορά Η ελάχιστη τιμή που εμφανίζεται είναι τα 0 χιλιόμετρα, υποδηλώνοντας νέο όχημα. Η μέγιστη τιμή που εμφανίζεται είναι τα 10000000 χιλιόμετρα, διαμορφώνοντας ανάλογο εύρος.

ODOMETER	
Mean	**115042.0308**
Standard Error	**210.5939362**
Median	**108000**
Mode	**150000**
Standard Deviation	**141991.2339**
Sample Variance	**20161510493**
Kurtosis	**2192.51419**
Skewness	**36.01890517**
Range	**10000000**
Minimum	**0**
Maximum	**10000000**
Sum	**52298337266**
Count	**454602**
Confidence Level(95.0%)	**412.7576293**

Πίνακας 8 Στατιστικά μεγέθη του πεδίου odometer

Όσον αφορά τα μέτρα κεντρικής τάσης, ο μέσος όρος του συνόλου των τιμών υπολογίζεται σε 115042 χιλιόμετρα, ενώ η διάμεσος σε 108000 χιλιόμετρα. Η τυπική απόκλιση ανέρχεται σε 141991.23 υποδηλώνοντας μεγάλη απόσταση των σημείων από την διάμεσο, με σαφείς προεκτάσεις και στην ολική τους διασπορά. Η κυρτότητα του συνόλου είναι 2192, υποδεικνύοντας μια λεπτόκυρτη κατανομή ιδιαίτερα μακριά από την κανονική. Τέλος η λοξότητα υπολογίζεται σε 36, διαμορφώνοντας θετική ασυμμετρία. Ο έλεγχος για έκτοπες τιμές θα διεξαχθεί μέσω της μεθόδου Tukey[5]. Οι τιμές των τεσσάρων τεταρτημόριων υπολογίζονται σε 59335, 108000, 153741.75 και 10000000 αντίστοιχα. Το ενδοτεταρτημοριακό εύρος συμπερασματικά ανέρχεται σε 94406.75. Συνεπώς, οι έκτοπες κατά Tukey τιμές βρίσκονται εκτός του διαστήματος [Q1-1.5*IQR, Q1+1.5*IQR]= [0, 295351.875], χαρακτηρίζοντας συνολικά 5251 τιμές έκτοπες κατά Tukey. Ομοίως, συνολικά 1861 τιμές είναι ιδιαιτέρως έκτος κατά Tukey εφόσον βρίσκονται εκτός του διαστήματος [Q1-3*IQR, Q1+3*IQR]= [0, 436962]

[5] (Neil C. Schwertman, 2007)



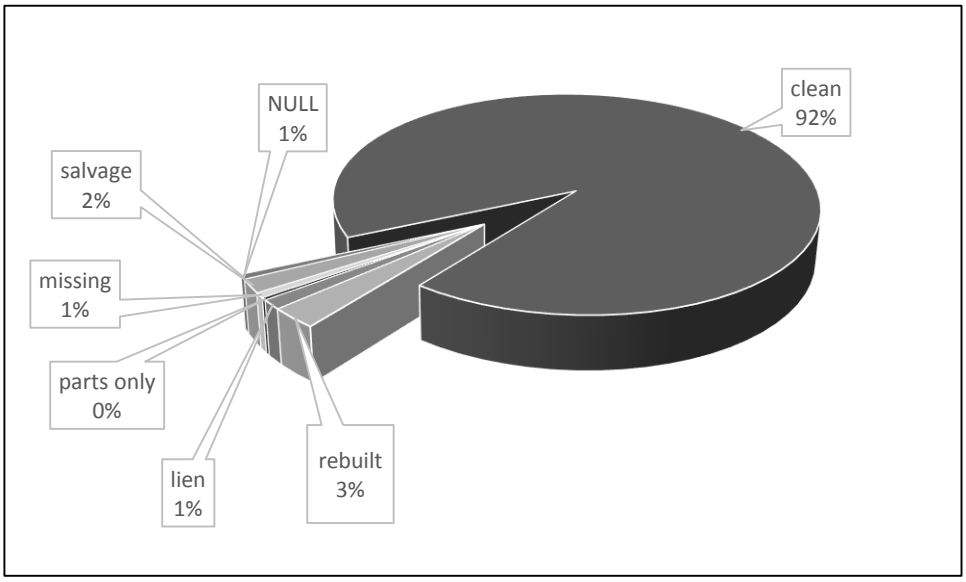
Εικόνα 9 Θηκόγραμμα του πεδίου odometer κατόπιν της εφαρμογής φραγμάτων Tukey

TITLE_STATUS

Το πεδίο title αφορά την ιδιοκτησία του οχήματος καθώς και την κατάσταση του. Πιο συγκεκριμένα, εμπεριέχει πληροφορίες για τον πρώην ιδιοκτήτη ο οποίος διαθέτει το εν λόγω όχημα προς πώληση, καθώς και για την κατάσταση του ίδιου του οχήματος. Η τιμή "clean" υποδεικνύει πως το όχημα είναι ασφαλές προς χρήση και η ασφάλιση του αντικατοπτρίζει την αρχική τιμή αγοράς του οχήματος. Η τιμή "salvage" αφορά σε οχήματα τα οποία δεν δύνανται πλέον να χρησιμοποιηθούν στους δρόμους διότι πιθανότατα να είχαν εμπλακεί σε ατύχημα καθιστώντας τα ακατάλληλα προς ασφάλιση. Η τιμή "rebuilt" αφορά οχήματα τα οποία είχαν εμπλακεί σε ατύχημα και έχουν επισκευαστεί. Η τιμή "lien'' αφορά σε οχήματα τα οποία έχουν οριστεί ως υποθήκη για την αποπληρωμή κάποιου χρέους που εκκρεμεί. Η τιμή "parts only" αφορά σε τμήματα οχήματος τα οποία διατίθενται σε πώληση. Τέλος η τιμή "missing" υποδηλώνει απώλεια του τίτλου της κατάστασης του οχήματος. Συνολικά, 685070 υπαρκτές εγγραφές σημειώνονται στο πεδίο title_status ενώ οι υπόλοιπες 4531 είναι ελλιπείς. Το πλήθος τον διακεκριμένων τιμών του πεδίου αυτού ανέρχεται σε 7. Συνηθέστερη τιμή αποτελεί η "clean" συχνότητας 636580 καταλαμβάνοντας έτσι το 92% των συνολικών εγγραφών. Σπανιότερη τιμή σημειώνεται η "parts only" συχνότητας 2152 καταλαμβάνοντας το 0.3% του συνόλου των εγγραφών.

title_status	Frequency
clean	636580
rebuilt	19544
lien	9074
parts only	2152
missing	4796
salvage	12923
null	4531

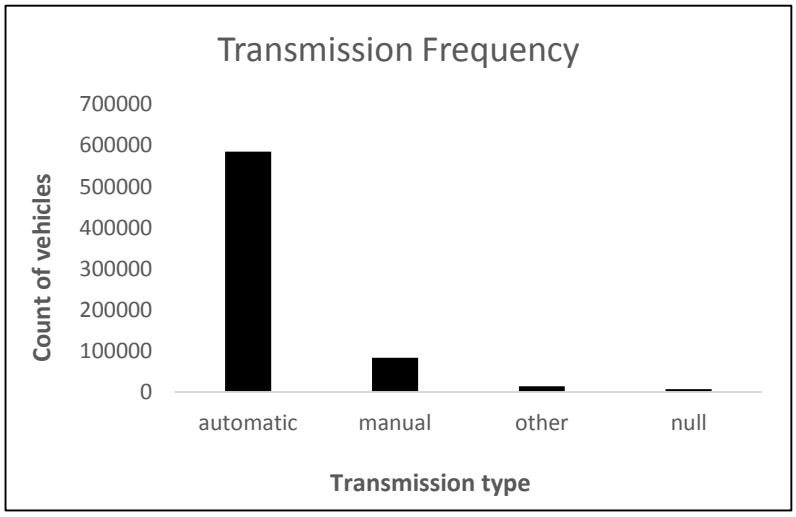
Πίνακας 9 Συχνότητες διακεκριμένων τιμών του πεδίου title_status



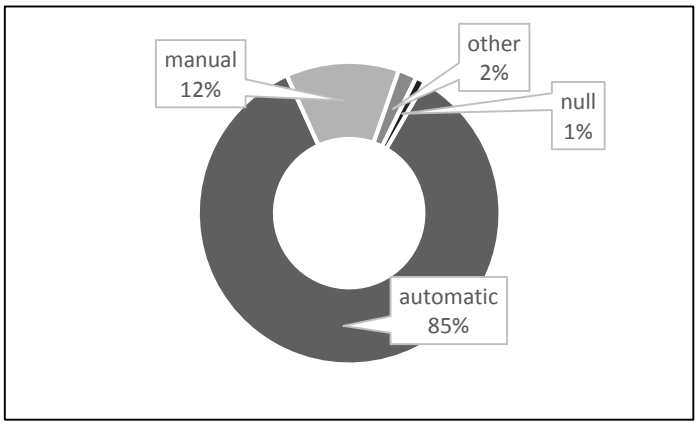
Εικόνα 10 Διάγραμμα πίτας ποσοστιαίων συχνοτήτων των τιμών του πεδίου title_status

TRANSMISSION

Το πεδίο transmission αφορά τον τύπο του κιβωτίου ταχυτήτων. Συνολικά σημειώνονται 682537 υπαρκτές εγγραφές στο πεδίο αυτό ενώ οι υπόλοιπες 7063 είναι ελλιπείς. Το πλήθος των διακεκριμένων τιμών των εγγραφών ανέρχεται σε 4. Η συνηθέστερη τιμή του πεδίου transmission είναι η "automatic" υποδηλώνοντας το αυτόματο κιβώτιο ταχυτήτων συχνότητας 585492, καταλαμβάνοντας έτσι το 85% των εγγραφών. Η σπανιότερη τιμή είναι η "other" συχνότητας 13655 καλύπτοντας το 2% των εγγραφών.



Εικόνα 11 Ραβδόγραμμα συχνοτήτων των τιμών του πεδίου transmission



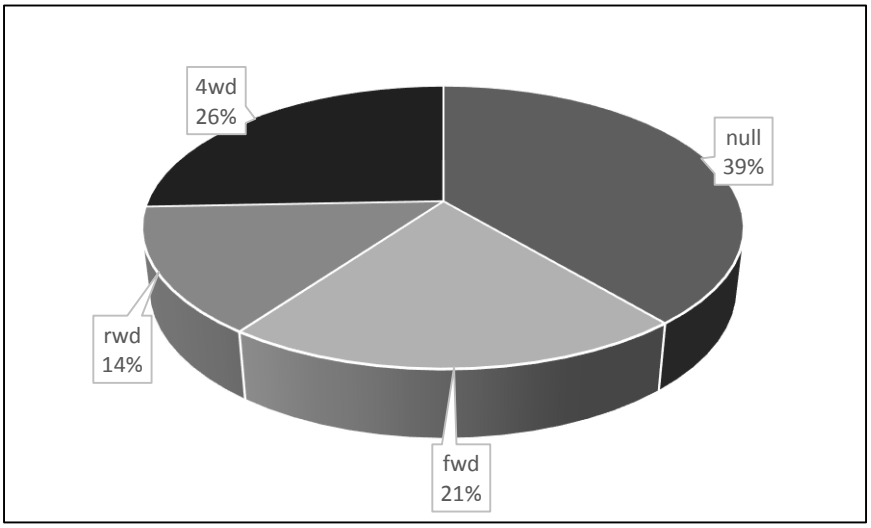
Εικόνα 12 Διάγραμμα δακτυλίου ποσοστιαίων συχνοτήτων του πεδίου transmission

VIN

Σε ό,τι αφορά το πεδίο VIN, αυτό εμπεριέχει μια σειρά αλφαριθμητικών που μοναδικά περιγράφει το όχημα της κάθε εγγραφής. Συνολικά απαντώνται 232825 υπαρκτές τιμές ενώ οι υπόλοιπες 456775 λείπουν. Συμπερασματικά το 66% των αναγνωριστικών αριθμών των οχημάτων δεν έχει καταχωρηθεί στις εν λόγω εγγραφές.

DRIVE

Το πεδίο drive αφορά στον τύπο μεταφοράς ενέργειας από τον κινητήρα στους τροχούς του οχήματος της κάθε εγγραφής. Συνολικά σημειώνονται 421328 υπαρκτές τιμές ενώ οι υπόλοιπες 268272 είναι ελλιπείς. Το πλήθος των διακεκριμένων τιμών του πεδίου Drive ανέρχεται σε 4. Συνηθέστερη τιμή αποτελεί η "4wd" συχνότητας 176715, υποδηλώνοντας ισχύ σε 4 τροχούς καταλαμβάνοντας έτσι το 26% των εγγραφών. Σπανιότερη τιμή καθίσταται η "rwd" συχνότητας 99520, υποδηλώνοντας ισχύ στους οπίσθιους τροχούς και καταλαμβάνοντας το 14% του συνόλου των εγγραφών.



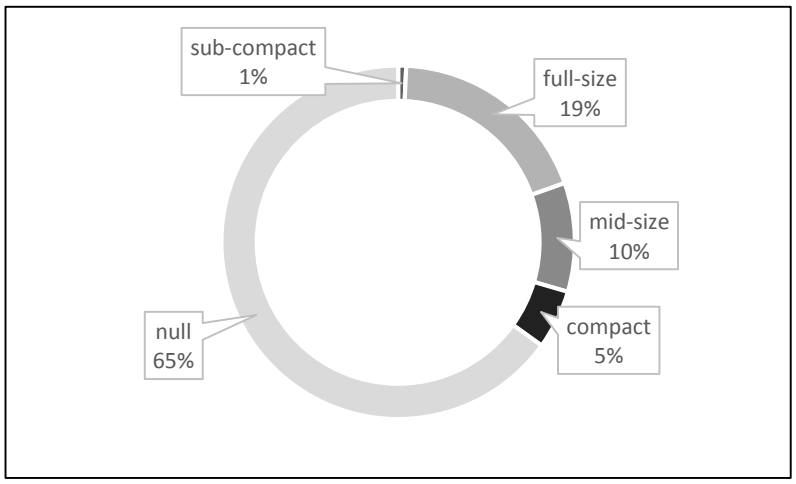
Εικόνα 13 Διάγραμμα πίτας ποσοστιαίων συχνοτήτων των διακεκριμένων τιμών του πεδίου drive

DRIVE	FREQUENCY
4wd	176715
fwd	145093
rwd	99520
null	268272
Total	689600

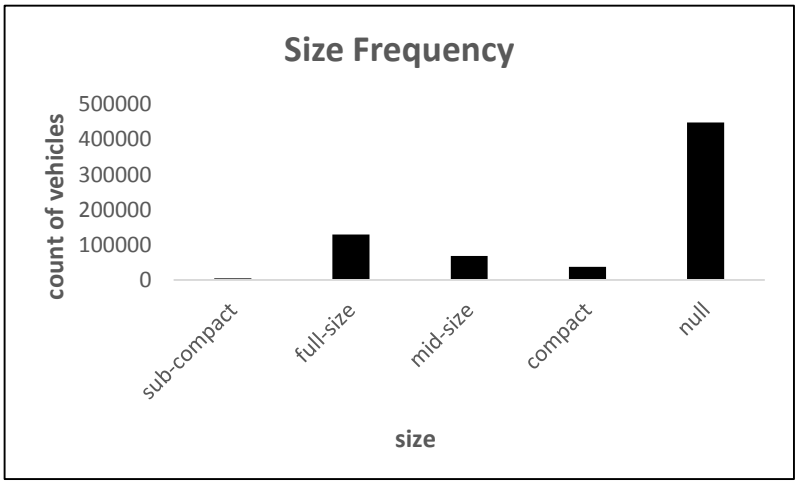
Πίνακας 10 Συχνότητες διακεκριμένων τιμών του πεδίου drive

SIZE

Το πεδίο size αφορά το μέγεθος του οχήματος της κάθε εγγραφής. Συνολικά σημειώνονται 240881 υπαρκτές τιμές, ενώ οι υπόλοιπες 448719 είναι ελλιπείς. Το πλήθος των διακεκριμένων τιμών του πεδίου αυτού ανέρχεται σε 5. Η συνηθέστερη τιμή του πεδίου size είναι η "full-size" συχνότητας 129833, καταλαμβάνοντας το 19% των εγγραφών. Σπανιότερη τιμή είναι η "sub-compact" συχνότητας 5032 που καταλαμβάνει το 1% του συνόλου των εγγραφών.



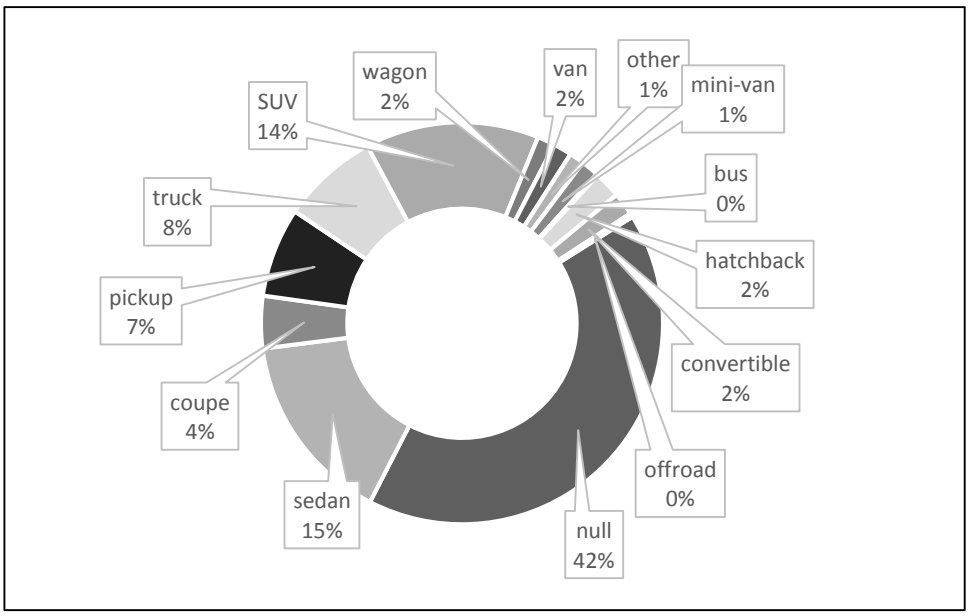
Εικόνα 14 Διάγραμμα δακτυλίου διακεκριμένων συχνοτήτων των τιμών του πεδίου size



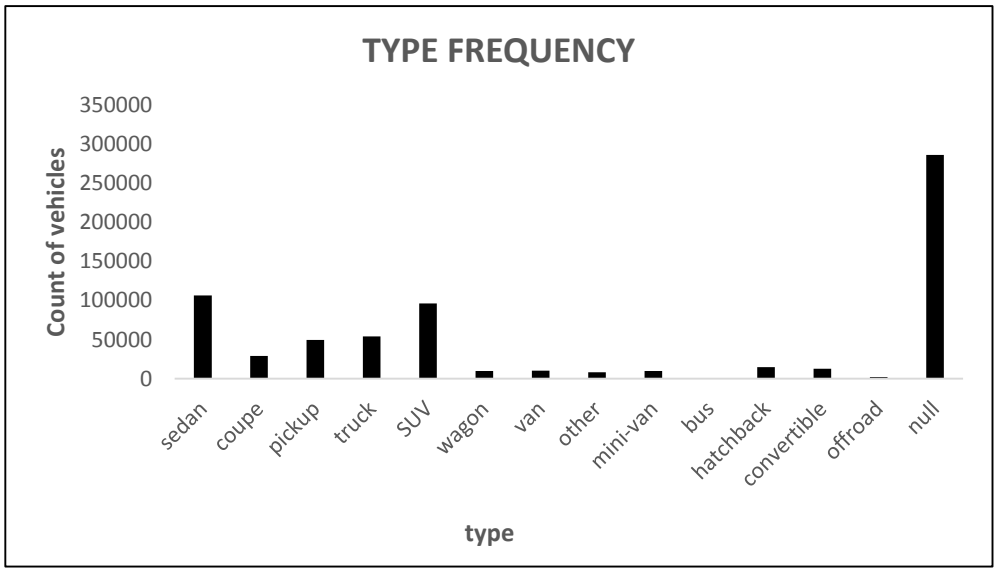
Εικόνα 15 Ραβδόγραμμα συχνοτήτων τιμών του πεδίου size

TYPE

Το πεδίο type αντανακλά τον τύπο του οχήματος. Συνολικά σημειώνονται 403725 υπαρκτές τιμές ενώ οι υπόλοιπες 285875 είναι ελλιπείς. Το πλήθος των διακεκριμένων τιμών του πεδίου type ανέρχεται σε 14. Συνηθέστερη τιμή αποτελεί η "sedan" συχνότητας 106322 καταλαμβάνοντας το 15% του συνόλου των εγγραφών. Σπανιότερη τιμή υπολογίζεται η "bus" συχνότητας 895 που καταλαμβάνει το 0.12% των εγγραφών.



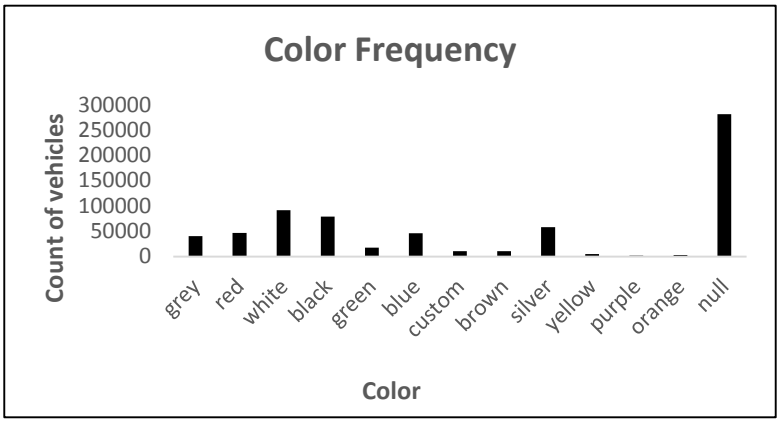
Εικόνα 16 Διάγραμμα δακτυλίου ποσοστιαίων συχνοτήτων των διακεκριμένων τιμών του πεδίου type



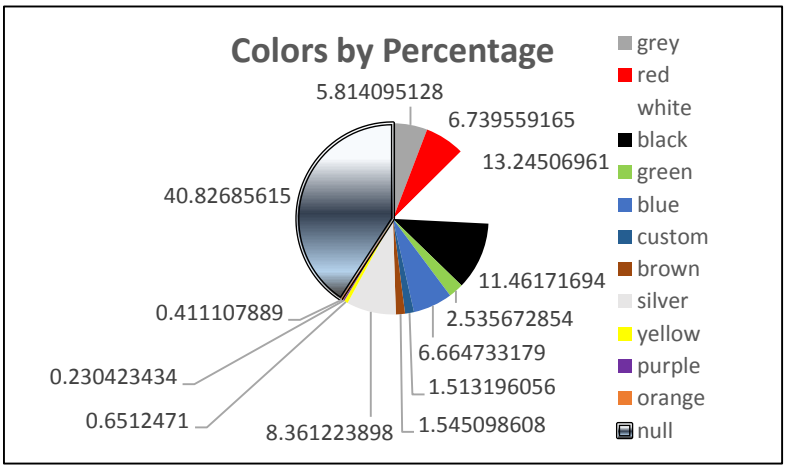
Εικόνα 17 Ραβδόγραμμα συχνοτήτων των τιμών του πεδίου type

PAINT COLOR

Το πεδίο paint color εμπεριέχει το είδος χρωματισμού του οχήματος της κάθε εγγραφής. Οι υπαρκτές τιμές του πεδίου αυτού ανέρχονται σε 408058 ενώ οι υπόλοιπες 281542 είναι ελλιπείς. Το πλήθος των διακεκριμένων τιμών του πεδίου αυτού υπολογίζεται σε 12. Το συνηθέστερο χρώμα είναι το άσπρος, συχνότητας 91338, καταλαμβάνοντας το 13.2% των εγγραφών. Το σπανιότερο χρώμα είναι το μωβ, συχνότητας 1589, που καταλαμβάνει το 0.4% του συνόλου των εγγραφών.



Εικόνα 18 Ραβδόγραμμα συχνοτήτων των τιμών του πεδίου color



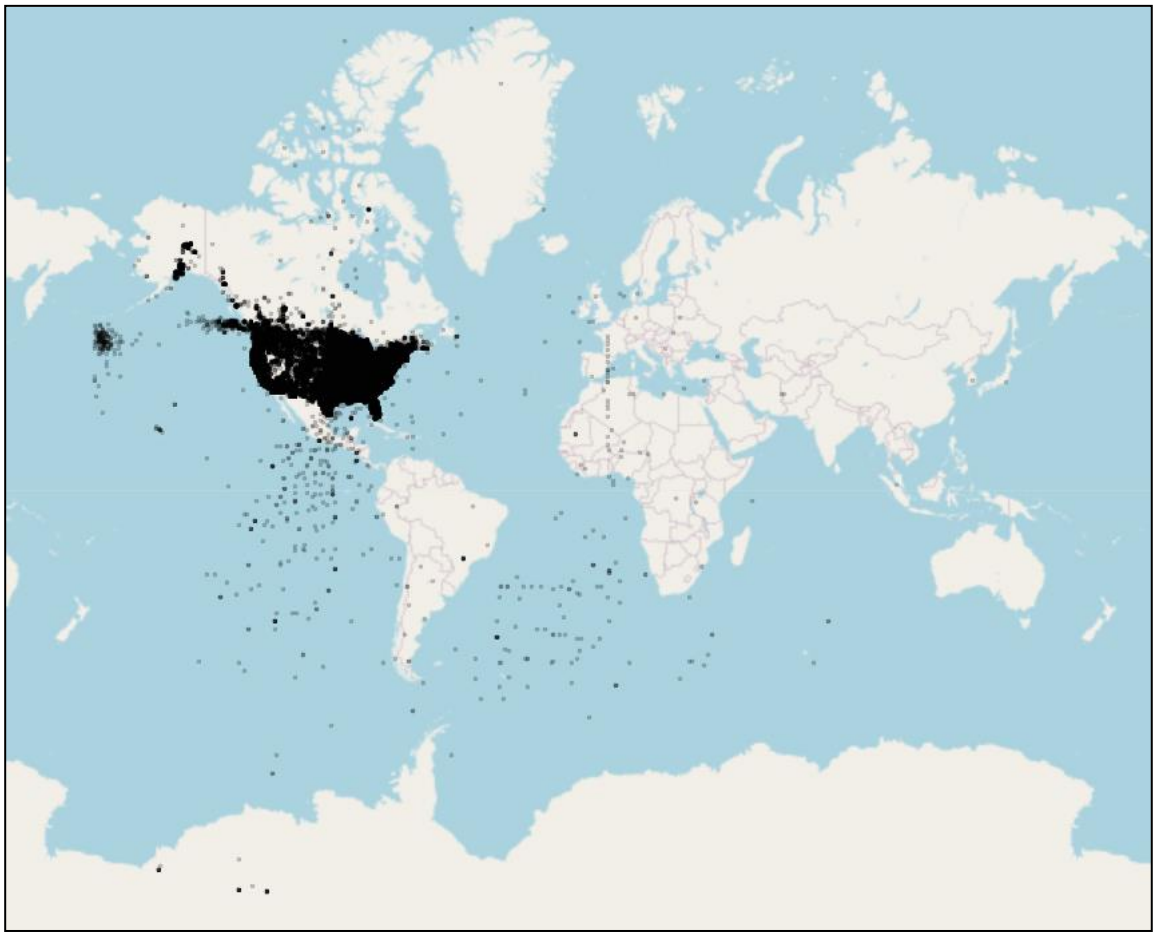
Εικόνα 19 Διάγραμμα πίτας των ποσοστιαίων συχνοτήτων των τιμών του πεδίου colors

IMAGE_URL

Το πεδίο image_url περιέχει ηλεκτρονικό σύνδεσμο που ανακατευθύνει προς την εικόνα του οχήματος της κάθε εγγραφής. Συνολικά σημειώνονται 689576 υπαρκτοί σύνδεσμοι ενώ οι υπόλοιποι 24 είναι ελλιπείς.

LAT LONG

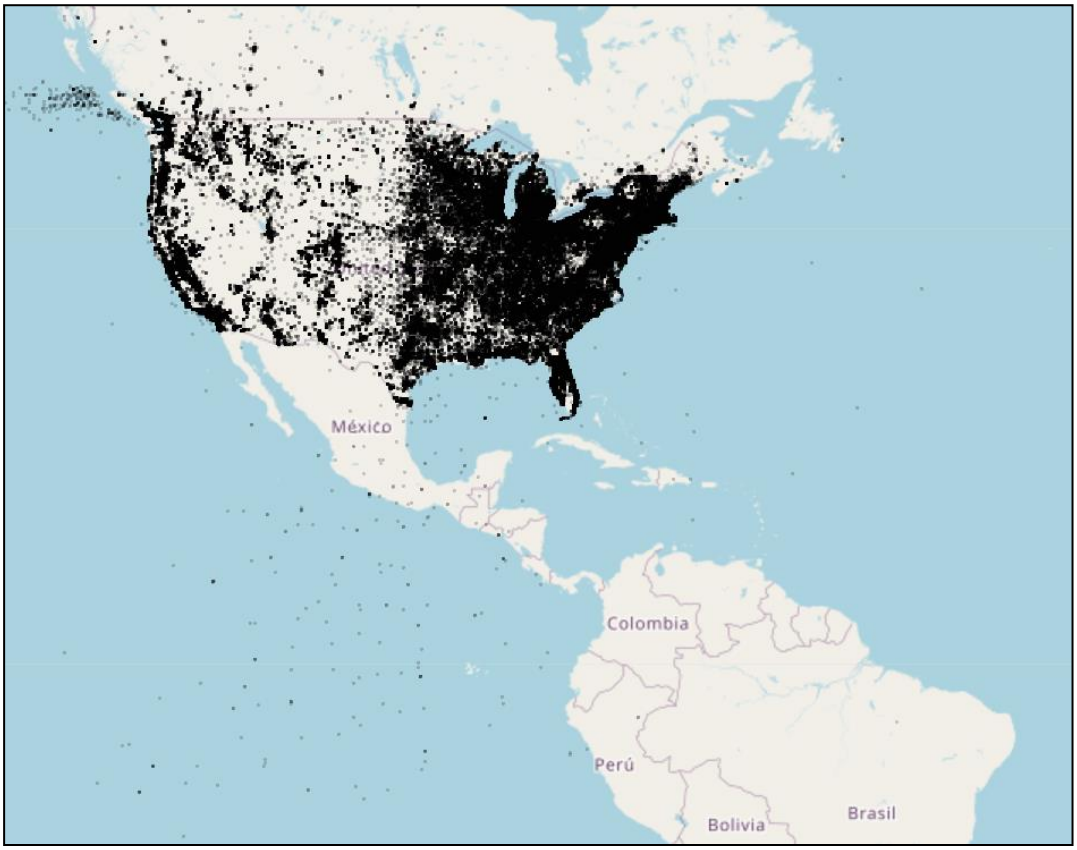
Τα πεδία lat και long, αφορούν αντίστοιχα το γεωγραφικό πλάτος και μήκος της κάθε εγγραφής. Το πλήθος των εγγραφών με συμπληρωμένα τα πεδία αυτά ανέρχεται σε 682426, ενώ οι υπόλοιπες 7174 είναι ελλιπείς και στα δύο αυτά πεδία.



Εικόνα 20 Παγκόσμιος χάρτης απεικόνισης κάθε εγγραφής βάσει των πεδίων lat,και long

Όπως παρατηρείται από την αναπαράσταση κάθε εγγραφής με μια κουκκίδα στον χάρτη, η πλειοψηφία των σημείων είναι συγκεντρωμένη στο βορειοδυτικό τμήμα. Ειδικότερα, το

μεγαλύτερο σύμπλεγμα εγγραφών σημειώνεται στις Η.Π.Α. Ακολουθούν μικρότερα συμπλέγματα εγγραφών δυτικά του Καναδά, και στην Αλάσκα. Το σύνολο των εγγραφών που υπολείπονται είναι διασκορπισμένο σε όλον τον υπόλοιπο χάρτη.



Εικόνα 21 Μεγεθυμένος χάρτης της διάταξης των εγγραφών στην Αμερική

Συμπερασματικά, η πλειοψηφία των εγγραφών σημειώνονται στην Αμερική και μάλιστα στο ανατολικό τμήμα της. Ενώ το μεγαλύτερο μέρος των υπολειπόμενων εντός των Η.Π.Α. περιοχών συγκεντρώνεται στο παραθαλάσσιο δυτικό τμήμα τους.

Σε ό,τι αφορά το σύνολο των πραγματικών δεδομένων, παρατηρείται πλήθος πεδίων στα οποία περιλαμβάνονται έκτοπες τιμές. Η παρουσία έκτοπων τιμών θα μπορούσε εν δυνάμει να επηρεάσει την εύρεση βέλτιστων εγγραφών, ιδιαίτερα αναφορικά με του παράγοντες αυτούς. Πιο συγκεκριμένα στα πεδία Year, Price και Odometer σημειώνεται πληθώρα έκτροπων τιμών κατά Tukey. Τα υπόλοιπα πεδία, αποτελούμενα κατά βάση από κείμενο, δεν παρουσιάζουν ανησυχητικές ενδείξεις για τα βέλτιστα στοιχεία. Πρέπει ωστόσο να σημειωθεί πως παρατηρούνται ελλιπείς τιμές σε πολλά από τα πεδία, τόσο ποσοτικά όσο και ποιοτικά. Το γεγονός αυτό οφείλεται πιθανώς στο ότι το σύνολο δεδομένων είναι πραγματικό και βασίζεται σε εισροές χρηστών, καθιστώντας έτσι όχι αδύνατη την αμέλεια συμπλήρωσης στοιχείων. Παρόλα αυτά θα επιχειρηθεί αξιοποίηση του

εγγραφών αυτών στο σύνολο τους ούτως ώστε να παραχθούν αντικειμενικά συμπεράσματα ακόμα και σε ιδιάζουσες τέτοιες περιπτώσεις που δεν είναι σπάνιες κατά την χρήση πραγματικού συνόλου δεδομένων.

2.3 *Ανάλυση του μετασχηματισμού προ-επεξεργασίας των δεδομένων*

2.3.1 Η κύρια κλάση

Για την ανάλυση των δεδομένων και την απομόνωση των σημαντικών στοιχείων κάθε εγγραφής θα χρησιμοποιηθεί ειδικός μετασχηματισμός. Σκοπός του θα είναι η εξαγωγή ενός υποσυνόλου του αρχικού συνόλου των δεδομένων ούτως ώστε να καταστεί ευκολότερη η εξέταση, επεξεργασία και η ανάλυση τους. Δεδομένου του ότι το προς επεξεργασία σύνολο δίνεται σε τύπο .csv (Comma Separated Values), που αποτελεί κείμενο διαχωρισμένο με κόμμα για το κάθε στοιχείο της εκάστοτε εγγραφής, θα γίνει χρήση προσαρμοσμένου αλγορίθμου για να την ορθή εξαγωγή των απαιτούμενων δεδομένων.

```
TestMain.java > ...
1   public class TestMain {
        Run | Debug
2       public static void main(String[] args) {
3
4           String filename="ProcSmall.csv";
5           String OutputName="result.csv";
6           CSVDataParser parser= new CSVDataParser(filename,OutputName);
7           parser.StartProcess();
8
9       }
10  }
```

Οι σειρές 1-10 απεικονίζουν την κύρια κλάση *TestMain.java* που θα χρησιμοποιηθεί ως κεντρικός άξονας για την αρχικοποίηση της διαδικασίας της επεξεργασίας. Οι γραμμές 4 και 5 αποτελούν τους τίτλους των, αντίστοιχα, εισακτέων και προς εξαγωγή αρχείων τύπου .csv που θα καθορίσουν τις εισροές και εκροές του προγράμματος προ-επεξεργασίας του αρχικού συνόλου εγγραφών. Πρέπει να σημειωθεί πως ουσιαστικά οι γραμμές αυτές αντιπροσωπεύουν την θέση στο υπολογιστικό σύστημα από την οποία θα διαβάσει και στην οποία, αντίστοιχα, θα εγγραφεί το εξαγώγιμο αρχείο. Στη σειρά 6 μέσω της κλάσης *CSVDataParser*, στην οποία θα γίνει εκτενέστερη αναφορά μετέπειτα, υλοποιείται ένα αντικείμενο αναφοράς της με όνομα parser που θα χρησιμεύσει στην ανάγνωση, επεξεργασία και μορφοποίηση των δεδομένων εισαγωγής καθώς και την τελική

δημιουργία του εξαγώγιμου αρχείου *result.csv*. Στο αντικείμενο αυτό παρατίθενται ως μεταβλητές οι προαναφερθείσες τοποθεσίες του συστήματος των δοθέντων εισροών και εκροών. Τέλος στην σειρά 7 μέσω της εγγενούς μεθόδου της κλάσης *CSVDataParser, StartProcess(),* γίνεται η εκκίνηση του αλγορίθμου, μέσω της λογικής του οποίου θα παραχθούν οι απαραίτητες εκροές των εγγραφών.

2.3.2 Η κλάση επεξεργασίας CSVDataParser

Η επόμενη κλάση, αρχικοποίηση του αντικειμένου parser της οποίας έγινε στην παράγραφο 2.3.1 αποτελεί η κλάση CSVDataParser(). Ως μοχλός της κεντρικής λογικής του αλγορίθμου, η κλάση αυτή θα υλοποιήσει την πληθώρα των απαιτούμενων στόχων ούτως ώστε να ολοκληρωθεί επιτυχώς η διαδικασία της προ-επεξεργασίας του αρχικού συνόλου δεδομένων. Πιο αναλυτικά, σε πρώτη φάση θα γίνει ανάγνωση κάθε γραμμής του αρχικού εγγράφου δεδομένων μέσω ενός buffer, που αποτελεί μνήμη που δεσμεύεται με στόχο την προσωρινή αποθήκευση δεδομένων. Στην συνέχεια θα γίνει εξαγωγή των στοιχείων της κάθε σειράς που έχει αποθηκευτεί στον αντίστοιχο buffer μέσω του διαχωρισμού κειμένου που προσφέρει η λογική κανονικής παράστασης που θα χρησιμοποιηθεί. Παράλληλα της ανάγνωσης θα εφαρμοστεί μηχανισμός εξασφάλισης της ορθής λειτουργίας του διαχωρισμού της κανονικής παράστασης κάθε σειράς για να προκύψει η ομαλή επεξεργασία των απαραίτητων εγγραφών και το αναγκαίο πλήθος των στοιχείων των εκροών. Έπειτα θα γίνει απομόνωση των σημαντικών ανά σειρά στηλών του αρχείου εισροών. Τέλος θα λάβει χώρα η εγγραφή των μορφοποιημένων και απομονωμένων στοιχείων της κάθε εγγραφής στο αρχείο εκροών.

Σε ό,τι αφορά τις εγγενείς βιβλιοθήκες της java που θα χρησιμοποιηθούν στην κλάση CSVDataParser, αυτές αφορούν τον τύπο των δομών δεδομένων που θα χρησιμοποιηθούν για την αποτελεσματική επεξεργασία των εισροών, τον διαχωρισμό που θα συμβεί μέσω κανονικών παραστάσεων και την παραγωγή της κάθε σειράς προς εγγραφή στο αρχείο των εκροών. Πιο συγκεκριμένα οι σειρές 1 εώς 4 επιτρέπουν την χρήση των δομών δεδομένων ArrayList,Arrays,HashSet και Set, που αποτελούν δομές αποθήκευσης δεδομένων με ποικίλες δυνατότητες η καθεμία. Τα Arrays αποτελούν ένα σύνολο αριθμημένων ομοειδών μεταβλητών με συγκεκριμένο αναλλοίωτο μέγεθος. Οι ArrayList αποτελούν ένα σύνολο αριθμημένων μεταβλήτών με δυναμικά μεταβαλλόμενο μέγεθος στο οποίο αποθηκεύονται δυναμικώς αναφορές αντικειμένων.

Τα Set είναι μια κλάση μη αριθμημένων συνόλων αντικειμένων με δυναμικά μεταβαλλόμενο μέγεθος. Τέλος τα HashSets αποτελούν την υλοποίηση της κλάσης Set.

```
1    import java.util.ArrayList;
2    import java.util.Arrays;
3    import java.util.HashSet;
4    import java.util.Set;
5    import java.util.regex.Pattern;
6    import java.util.regex.Matcher;
```

Όπως προαναφέρθηκε, η κύρια λειτουργία της κλάσης CSVDataParser έχει ως στόχο την επεξεργασία του αρχικού συνόλου εγγραφών ούτως ώστε να παραχθεί το εξαγώγιμο αρχείο. Η δομή του αρχείου του πηγαίου κώδικα της κλάσης έχει διαμορφωθεί ανάλογα με τα διαφορετικά στάδια-στόχους που πρέπει εκάστοτε να επιτευχθούν από την κλάση αυτή.

```
8   public class CSVDataParser {
9
10      private String filename,OutputName;
11      public XXFile file_dataset = new XXFile();
12      public int max=-1,min=100;
13      public boolean verified=true;
14
15      public CSVDataParser(String filename, String OutputName)
16      {
17          this.filename = filename;
18          this.OutputName=OutputName;
19      }
20
21      public void StartProcess()···
41
42      public String[] RegexSeparation(String temp){···
57
58      public String ColumnIsolation(String[] input, int id){···
84
85      public void RegexVerification(int count){···
95
96  }
```

Όσον αφορά την αρχικοποίηση των μεταβλητών που θα χρησιμοποιηθούν κατά την ροή της κλάσης CSVDataParser, τα αλφαριθμητικά filename και OutputName αποτελούν τις μεταβλητές στις οποίες μέσω της αρχικοποίησης της κλάσης στις σειρές 15-19 και κλήσης από το κεφάλαιο 2.3.1 δίνονται οι τοποθεσίες των αρχείων εισροών και εκροών αντίστοιχα. Η μεταβλητή file_dataset αποθηκεύει ένα αντικείμενο της κλάσης XXFile η οποία θα αναλυθεί εκτενέστερα μετέπειτα και αποσκοπεί στην ορθή διαχείριση των πόρων μνήμης του συστήματος. Εν τέλει τόσο οι ακέραιοι max και min, όσο και η μεταβλητή λογικής τιμής verified, θα χρησιμεύσουν στο προαναφερθέν στάδιο της επαλήθευσης της λειτουργίας του μηχανισμού κανονικής παράστασης που θα εφαρμοστεί κατά την πάροδο του αλγορίθμου. Ακολουθούν από την σειρά 21 μέχρι και 95, τέσσερις μέθοδοι που αφορούν τα διάφορα στάδια μέχρι την παραγωγή του εξαγώγιμου αρχείου.

```
21      public void StartProcess()
22      {
23          file_dataset.openRFile(filename);
24          file_dataset.openWFile(OutputName);
25
26          String processedLine="",line=file_dataset.readLine();
27          int id=0;
28
29          while(line != null){
30              String[] separated=RegexSeparation(line);
31              processedLine=ColumnIsolation(separated,id);
32              if(!verified)break;
33              file_dataset.writeLine(processedLine);
34              line=file_dataset.readLine();
35              id++;
36          }
37
38          file_dataset.closeRFile();
39          file_dataset.closeWFile();
40      }
```

Η μέθοδος StartProcess, της οποίας έγινε κλήση κατά την παράγραφο 2.3.1, αποτελεί τον κορμό της κλάσης CSVDataParser και εκκινεί την διαδικασία επεξεργασίας των εισροών. Οι σειρές 23,24,33,34,38 και 39 αφορούν μεθόδους της κλάσης XXFile και θα αναλυθούν περεταίρω σε επόμενη παράγραφο. Στη σειρά 26 αρχικοποιούνται οι δυο μεταβλητές αλφαριθμητικών line και processedLine που στις οποίες θα αποθηκεύονται αντίστοιχα η σειρά που διαβάζει το σύστημα και αντιστοιχεί σε μια εγγραφή του αρχικού συνόλου και η σειρά μετά την επεξεργασία της που προορίζεται να εγγραφεί στο αρχείο των εκροών. Ο ακέραιος id, θα αποτελεί τον αύξοντα αριθμό

της κάθε εγγραφής του τελικού αρχείου. Οι σειρές 29-36 εκτελούν έναν επαναληπτικό βρόγχο while μέσα στον οποίο εκτελείται η πληθώρα των επιθυμητών στόχων της κλάσης CSVDataParser.Αρχικά στη σειρά 30, κάθε διαβασμένη γραμμή διαχωρίζεται μέσω της λογικής κανονικής παράστασης που χρησιμοποιεί η μέθοδος RegexSeparation() που θα αναλυθεί στη συνέχεια. Στη σειρά 31 μέσω της μεθόδου ColumnIsolation, η διαχωρισμένη σε τμήματα σειρά επεξεργάζεται και απομονώνονται οι σημαντικές στήλες. Η σειρά 32 αποτελεί τον έλεγχο του ορθού διαχωρισμού του κειμένου μέσω της λειτουργίας κανονικής παράστασης που εφαρμόστηκε στην σειρά 30. Στη σειρά 33 και 34 αντίστοιχα, η επεξεργασμένη εγγραφή μεταφέρεται στο αρχείο εκροών και στην μεταβλητή line αποθηκεύεται η επόμενη σειρά, με αντίστοιχη ενημέρωση και του αύξοντα αναγνωριστικού αριθμού id της επόμενης εγγραφής.

```
42      public String[] RegexSeparation(String temp){
43          String regExp ="(?:^|,)(\"(?:(?:\"\")*[^\"]*)*\"|[^\",]*)" ;
44          Pattern p = Pattern.compile(regExp);
45          String[] tokens= new String[20];
46          int counter=0, resultCounter=0;
47          Matcher matcher = p.matcher(temp);
48          while (matcher.find()) {
49              if(matcher.groupCount()>1)verified=false;
50              tokens[counter++]=matcher.group(1);
51              resultCounter++;
52          }
53          RegexVerification(resultCounter);
54          return tokens;
55      }
```

Η μέθοδος RegexSeparation στις σειρές 42-55, της οποίας όπως προαναφέρθηκε γίνεται χρήση στην σειρά 30 στοχεύει στον διαχωρισμό κάθε διαβασμένης σειράς στις στήλες-στοιχεία που αντιπροσωπεύει. Αυτό θα γίνει με το μοτίβο κανονικής παράστασης της σειράς 43. Πιο συγκεκριμένα το κομμάτι (?:^|,) προτρέπει την έναρξη της ταυτοποίησης στοιχείων από την αρχή της σειράς ή αμέσως μετά από κάθε χαρακτήρα ‘,’ κόμμα. Οι ομάδες στοιχείων ταυτοποίησης που θα αναγνωρίσει το μοτίβο αμέσως μετά θα είναι δύο. Η πρώτη **\"(?:(?:\"\")*[^\"]*)*\"** αποτελείται αλφαριθμητικά που αρχίζουν και τελειώνουν με διαλυτικά που εμπεριέχουν καμία ή περισσότερες ομάδες αλφαριθμητικών ή ζευγαριών διαλυτικών. Αυτό συμβαίνει για τι στον τύπο αρχείων .csv τα ζεύγη διαλυτικών εμπεριέχουν κείμενο. Η δεύτερη ομάδα **[^\",]*** ταιριάζει μηδέν ή περισσότερους χαρακτήρες που δεν είναι διαλυτικά, που αποτελεί την πληθώρα των περιπτώσεων αριθμών, απόντων στοιχείων, και λέξεων. Η σειρά 44 δημιουργεί το μοτίβο που ορίστηκε στην σειρά 43

ούτως ώστε να είναι προσβάσιμο παρακάτω. Στις επόμενες δύο σειρές αρχικοποιούνται τρείς μεταβλητές. Η tokens, θα αποθηκεύσει τα ευρήματα του διαχωρισμού μέσω της κανονικής παράστασης ενώ η counter θα την διατρέξει την σειρά tokens για να αποθηκεύσει στις 20 αναμενόμενες θέσεις τα ευρήματα της αναζήτησης. Ο ακέραιος resultCounter θα χρησιμοποιηθεί για τη μέτρηση του συνολικού πλήθους των ευρημάτων και κατ' επέκταση την επαλήθευση της λειτουργίας της λογικής κανονικής παράστασης που χρησιμοποιήθηκε. Η σειρά 47 αρχικοποιεί αντικείμενο της κλάσης Matcher, μιας κλάσης που δοθέντος ενός μοτίβου πραγματοποιεί λειτουργίες ταυτοποίησης πάνω σε μια σειρά χαρακτήρων. Οι σειρές 48-52 αποτελούν τον κύκλο αναζήτησης των ταυτοποιήσεων της προηγούμενης σειράς δοθέντος του μοτίβου που δημιουργήθηκε στην 44 και της μεταβλητής εισροών temp, που αποτελεί την ανεπεξέργαστη σειρά που έχει διαβάσει το πρόγραμμα. Πιο συγκεκριμένα, όσο υπάρχει επιτυχής ταυτοποίηση του αρχικοποιημένου matcher, τα απαιτούμενα δεδομένα αποθηκεύονται στην αντίστοιχη θέση της σειράς tokens και κατόπιν προσαυξάνονται οι δείκτες counter και resultCounter. Πρέπει να σημειωθεί πως τα αποτελέσματα ταυτοποίησης θα δίνονται πάντα ως μια σειρά δυο στοιχείων, αυτών που χρησιμοποιήθηκαν για τον ορισμό της αρχής ταυτοποίησης από την πρώτη παρένθεση του μοτίβου. Το πρώτο θα είναι ο χαρακτήρας νέας σειράς ^, που σημαίνει πως τη ταυτοποίηση συνέβη με αρχή την νέα σειρά ή το κοινό κόμμα που σηματοδοτεί την ταυτοποίηση μετά από μια στήλη εγγραφής, δηλαδή ένα νέο κελί. Το δεύτερο θα είναι το κομμάτι που αποτελεί το εσωτερικό του κελιού, δηλαδή η ουσιαστική στήλη της εγγραφής που θα ταυτοποιηθεί βάσει της δεύτερης παρένθεσης του μοτίβου κανονικής παράστασης. Με αυτόν τον τρόπο αιτιολογείται η λογική της σειράς 50, και αποθηκεύεται στην tokens η δεύτερη και σημαντική ταυτοποίηση με δείκτη 1. Ως αποτέλεσμα θα μπορούσε κανείς να συνάγει πως εαν υπάρξει τρίτο κομμάτι ταυτοποίησης υπάρχει σφάλμα και για αυτόν τον λόγο στην σειρά 49 αντιμετωπίζεται η περίπτωση εσφαλμένης εφαρμογής της λογικής regExp. Στην σειρά 53 καλείται η μέθοδος επαλήθευσης λειτουργίας της κανονικής παράστασης στην οποία δίνεται ως εισροή το πλήθος των ταυτοποιηθέντων στοιχείων για την οποία

θα γίνει λόγος παρακάτω. Τέλος επιστρέφεται στη σειρά 54 η σειρά με τις 20 πρώτες ταυτοποιήσεις των εσωτερικών των κελιών, tokens.

```
57    public String ColumnIsolation(String[] input, int id){
58        String[] result=new String[17];
59        ArrayList<String> MainSet=new ArrayList<>(),Subset=new ArrayList<>();
60        Set<Integer> MainSet_Indexes=new HashSet<Integer>(), Subset_Indexes=new HashSet<Integer>();
61        MainSet_Indexes.addAll(Arrays.asList(new Integer[] {2,3,9,18,19}));
62        Subset_Indexes.addAll(Arrays.asList(new Integer[] {4,5,6,7,8,10,11,13,14,15,16}));
63        for (int i = 0; i < input.length; i++) {
64            if(MainSet_Indexes.contains(i))MainSet.add(input[i]);
65            else if(Subset_Indexes.contains(i))Subset.add(input[i]);
66        }
67        if(id==0)result[0]="id";
68        else result[0]=String.valueOf(id);
69        MainSet.addAll(Subset);
70        for (int i = 1; i < result.length; i++) {
71            result[i]=MainSet.get(i-1);
72        }
73        return String.join(",",result);
74    }
```

Οι σειρές 57-74 αποτελούν την μέθοδο ColumnIsolation, κλήση της οποίας γίνεται στη σειρά 31. Εισροές λαμβάνει τις μεταβλητές input και id, που αποθηκεύουν τη σειρά κελιών της προηγούμενης συνάρτησης RegexSeparation και τον άυξοντα αριθμό της εγγραφής αντίστοιχα. Η μέθοδος αυτή έχει ως στόχο την απομόνωση των σημαντικών στηλών που ενδιαφέρουν τον αλγόριθμο, την κατάταξη τους στην επιθυμητή σειρά και την παραγωγή του τελικού εγγράψιμου στο αρχείο εκροών κειμένου. Στην σειρά 58 αρχικοποιείται η σειρά αποθήκευσης 17 στοιχείων result αλφαριθμητικών που θα συντελέσει στην παραγωγή του τελικού μετασχηματισμένου κειμένου εγγραφής. Στην επόμενη σειρά αρχικοποιούνται δυο σειρές στοιχείων αλφαριθμητικών δυναμικώς μεταβαλλόμενου μεγέθους MainSet και SubSet. Σε αυτές θα απομονωθούν οι δυο ομάδες σημαντικών στηλών της σειράς ταυτοποιηθέντων στοιχείων της μεθόδου RegexSeparation. Στις επόμενες τρείς σειρές αρχικοποιούνται δυο αποθήκες στοιχείων ακεραίων αριθμών MainSet_Indexes και Subset_Indexes, και αποθηκεύονται σε αυτές οι δείκτες των αντίστοιχων σημαντικών στηλών που πρέπει να απομονωθούν στις δυο κύριες ομάδες. Στις σειρές 63-66 διατρέχεται στον επαναληπτικό βρόγχο for η σειρά ταυτοποιηθέντων στοιχείων εκροής input και τα στοιχεία που αντιστοιχούν στους επιθυμητούς δείκτες αποθηκεύονται στις μεταβλητές MainSet και Subset. Στις επόμενες δυο σειρές αποθηκεύεται ως πρώτο στοιχείο της μεταβλητής result ο αύξων αριθμός της κάθε εγγραφής ή το κείμενο “id” ως τίτλος της πρώτης στήλης. Κατόπιν στην σειρά 69 προστίθενται στο MainSet όλα τα στοιχεία του Subset με σκοπό την παραγωγή του της σειράς αποτελεσμάτος result. Στον

επαναληπτικό βρόγχο των επόμενων τριών σειρών τοποθετούνται τα στοιχεία της σειράς MainSet στην σειρά αποτελεσμάτων result με διαφορά ενός δείκτη λόγω της μεταβλητής αύξοντα αριθμού id. Τέλος στην σειρά 73 επιστρέφεται το κείμενο που προκύπτει από την ένωση όλων των στοιχείων της σειράς result με διαχωριστικό τον χαρακτήρα κόμμα, για την σωστή λειτουργία του τύπου αρχείων .csv .

```
76      public void RegexVerification(int count){
77          max=Math.max(count,max);
78          min=Math.min(count,min);
79          if(min!=max){
80              System.out.println("REGEXP ERROR");
81              verified= false;
82              return;
83          }
84          verified= true;
85      }
```

Οι σειρές 78-85 αποτελούν την μέθοδο RegexVerification, της οποίας γίνεται κλήση στην σειρά 53 της μεθόδου RegexSeparation και αποσκοπεί στην επαλήθευση της ορθής λειτουργίας της λογικής κανονικής παράστασης που χρησιμοποιήθηκε στην μέθοδο αυτή. Πιο συγκεκριμένα οι σειρές 77 και 78 ενημερώνουν το ελάχιστο και μέγιστο πλήθος ταυτοποιηθέντων στοιχείων που μέχρι τώρα έχει διαχωρίσει η λογική κανονικής παράστασης που έχει εφαρμοστεί στη μέθοδο RegexSeparation. Σε περίπτωση που τα δυο νούμερα αυτά δεν είναι ίσα, οι σειρές 79-83 ενημερώνουν την τιμή της μεταβλητής λογικής τιμής verified σε false. Πρέπει να σημειωθεί πως σε κάθε επεξεργασία διαβασμένης γραμμής γίνεται τόσο ο έλεγχος μέσω της μεθόδου RegexVerification αλλά και ο έλεγχος της σειράς 49 για εσφαλμένο πλήθος ταυτοποιηθέντων ομάδων στοιχείων και αντικατοπτρίζουν στην μεταβλητή verified το αποτέλεσμα των ελέγχων αυτών. Με τον τρόπο αυτόν, σε περίπτωση σφάλματος η σειρά 32 τερματίζει την επεξεργασία σειρών και κατ' επέκταση τον ίδιο τον αλγόριθμο.

2.3.3 Η κλάση διαχείρισης αποθηκευτικών πόρων XXFile

Η τελευταία κλάση, χρήση της οποίας έγινε κατά την ανάγνωση και εγγραφή από και προς τα αρχεία εισροών και εκροών τύπου csv αντίστοιχα είναι η κλάση XXFile. Στόχος της συγκεκριμένης κλάσης είναι η αποδοτική διαχείριση των αποθηκευτικών πόρων του συστήματος με στόχο την ομαλή λειτουργία του αλγορίθμου κατά την αλληλεπίδραση του με το αρχικό και το παραγόμενο επιθυμητό αρχείο.

```
1   import java.io.BufferedReader;
2   import java.io.BufferedWriter;
3   import java.io.FileReader;
4   import java.io.FileWriter;
5   import java.io.IOException;
```

Οι εγγενείς κλάσεις της γλώσσας java που θα χρησιμοποιηθούν κατά το πέρας του μετασχηματισμού αφορούν την διαχείριση προσωρινής ενδιάμεσης μνήμης κατά την οποία τα δεδομένα προς ανάγνωση και εγγραφή αποθηκεύονται προσωρινά ούτως ώστε να βελτιστοποιηθεί η απόδοση του αλγορίθμου. Στις σειρές 1 και 2 εισάγονται οι εγγενείς κλάσεις BufferedReader και BufferedWriter, που χρησιμοποιούνται για τη δημιουργία ενός προσωρινού αποθηκευτικού χώρου, στον οποίον θα συσσωρευτούν τα δεδομένα που θα προκύψουν από την ανάγνωση του αρχείου εισροών, καθώς και αυτά που θα προορίζονται για την εγγραφή τους στο αρχείο εκροών. Η κλάσεις των σειρών 3-4 FileReader και FileWriter διαβάζουν και αντίστοιχα εγγράφουν σειρές χαρακτήρων χωρίς όμως να τις αποθηκεύουν σε κάποια περιοχή. Πιο συγκεκριμένα η FileReader διαβάζει 1 byte τη φορά μεταφράζοντας το από τον εγγενή τύπο χαρακτήρων προς τον επιθυμητό και κατόπιν το διαβάζει. Ομοίως η FileWriter εγγράφει 1 byte τη φορά μεταφράζοντάς το αντίστοιχα προς τον επιθυμητό τύπο. Τέλος η κλάση IOException της πέμπτης σειράς θα ειδοποιήσει για τυχούσες εξαιρέσεις που σχετίζονται με τις εισροές ή εκροές του μετασχηματισμού.

```
 7  public class XXFile {
 8      private BufferedReader m_Reader;
 9      private BufferedWriter m_Writer;
10      public void openRFile(String sFile) {···
18      public void openWFile(String sFile) {···
26      public String readLine() {···
37      public void writeLine(String sLine) {···
47      public void closeRFile() {···
55      public void closeWFile() {···
63
64  }
```

Στις σειρές 8 και 9 αρχικοποιούνται δυο μεταβλητές των κλάσεων BufferedReader και BufferedWriter. Αυτές θα χρησιμοποιηθούν για την υλοποίηση των αντίστοιχων αντικειμένων τους, με σκοπό την περιοδική ανάγνωση και εγγραφή δεδομένων μέσω της εκμετάλλευσης της προσωρινής αποθηκευτικής μνήμης που οι κλάσεις αυτές προσφέρουν. Ακολουθούν 6 μέθοδοι που αφορούν το άνοιγμα και κλείσιμο των αρχείων εισροών και εκροών, την ανάγνωση κάθε γραμμής από το αρχικό σύνολο δεδομένων και την εγγραφή κάθε επεξεργασμένης γραμμής προς το παραγόμενο επιθυμητό επεξεργασμένο αρχείο.

```
10      public void openRFile(String sFile) {
11          try {
12              m_Reader = new BufferedReader(new FileReader(sFile));
13          }
14          catch (IOException ioex) {
15              ioex.printStackTrace();
16          }
17      }
18      public void openWFile(String sFile) {
19          try {
20              m_Writer = new BufferedWriter(new FileWriter(sFile));
21          }
22          catch (IOException ioex) {
23              ioex.printStackTrace();
24          }
25      }
```

Οι σειρές 10-17 και 18-25 αποτελούν τις μεθόδους openRFile και openWFile, που στόχο έχουν το άνοιγμα του αρχικού προς ανάγνωση και αντίστοιχα τελικού προς εγγραφή αρχείου. Πιο συγκεκριμένα δοθέντων δυο σειρών χαρακτήρων αλφαριθμητικών που αποτελούν τις τοποθεσίες τους στο σύστημα, κατανέμουν συγκεκριμένο μέγεθος προσωρινής αποθηκευτικής μνήμης που θα διατεθεί για την πρόσκαιρη τοποθέτηση των αναγνωσμένων και προς εγγραφή από τα αρχεία δεδομένων. Σε όλες τις μεθόδους της κλάσης XXFile χρησιμοποιούνται οι βρόγχοι τύπου try catch. Αυτοί έχουν ως στόχο την προσπάθεια εκτέλεσης όσων σειρών κώδικα βρίσκονται μέσα στον βρόγχο try και, σε περίπτωση κάποιας εξαίρεσης (σφάλματος), των σειρών εκείνων μέσα στον βρόγχο catch. Ο τύπος του σφάλματος που αναμένεται να συμβεί ορίζεται να σχετίζεται με τις εισροές και εκροές εφόσον αυτή είναι και η λειτουργία της κλάσης αυτής. Στις σειρές 12 και 20 αρχικοποιούνται αντικείμενα των κλάσεων BufferedReader και BufferedWriter που κατανέμουν τους ανάλογους προσωρινούς αποθηκευτικούς χώρους στους οποίους θα τοποθετηθούν τα δεδομένα που διαβάζει το αντικείμενο της κλάσης FileReader και πρόκειται να εγγράψει η κλάση FileWriter από και προς τις αντίστοιχες τοποθεσίες αρχείων. Σε περίπτωση σφάλματος που σχετίζεται με εισροές και εκροές δεδομένων, οι σειρές 15 και 23 ενδιάμεσα των βρόγχων catch θα εκτελεστούν, αποτυπώνοντας πληροφορίες σχετικές με την προέλευση του σφάλματος που συνέβη.

```
26      public String readLine() {
27          String sLine = null;
28          try {
29              if ((sLine = m_Reader.readLine()) != null)return sLine;
30              else return null;
31          }
32          catch (IOException ioex) {
33              ioex.printStackTrace();
34              return null;
35          }
36      }
```

Οι σειρές 26-36 αποτελούν την μέθοδο readLine. Στόχο έχει την σηματοδότηση της ανάγνωσης σειράς από το αρχείο εισροών και την προσωρινή αποθήκευση μέσω της κλάσης BufferedReader της διαβασμένης και κατόπιν μεταφρασμένης στην ανάλογη κωδικοποίηση σειράς bit. Το αποτέλεσμα της ανάγνωσης αυτής θα είναι μια σειρά αλφαριθμητικών που θα αποθηκευτεί στην

αρχικοποιημένη μεταβλητή της σειράς 27 με όνομα sLine και θα επιστραφεί σε περίπτωση μη κενής σειράς. Σε περίπτωση κενής σειράς θα επιστραφεί η τιμή null, που αποτελεί την τιμή που παίρνουν όλες οι αρχικοποιημένες μεταβλητές στη γλώσσα java προτού τους δοθεί μια συγκεκριμένη τιμή. Όπως και στις προηγούμενες μεθόδους την περίπτωση σφάλματος σχετιζόμενου με τα αρχεία εισροών και εκροών διαχειρίζονται οι σειρές 32-35 στον βρόγχο catch. Σύμφωνα με αυτόν αποτυπώνονται στη σειρά 33 πληροφορίες για την προέλευση αυτού του σφάλματος και στην επόμενη σειρά επιστρέφεται η τιμή null, δεδομένης της απαραίτητης επιστροφής κάποιας τιμής λόγω του τρόπου ορισμού της μεθόδου αυτής.

```
37      public void writeLine(String sLine) {
38          try {
39              m_Writer.write(sLine);
40              m_Writer.newLine();
41              m_Writer.flush();
42          }
43          catch (IOException ioex) {
44              ioex.printStackTrace();
45          }
46      }
```

Η επόμενη μέθοδος που απεικονίζεται στις σειρές 37-46 είναι η writeLine. Στόχο έχει, δοθείσης μιας ακολουθίας αλφαριθμητικών, την εγγραφή της σειράς αυτής στο αρχείο εκροών. Στη σειρά 39 χρησιμοποιείται η μέθοδος της κλάσης BufferedWriter .write(), η οποία αποθηκεύει το περιεχόμενο της σειράς αλφαριθμητικών στην καθορισμένη προσωρινή αποθηκευτική μνήμη του αντίστοιχου αρχικοποιημένου αντικειμένου της κλάσης αυτής m_Writer. Η επόμενη σειρά θα προσθέσει στην ήδη εγγεγραμμένη στην προσωρινή μνήμη σειρά τον χαρακτήρα διαχωρισμού γραμμών, ανάλογα βέβαια με το σύστημα που χρησιμοποείται στην εκάστοτε εφαρμογή του αλγορίθμου. Τέλος στη σειρά 41 η εγγενής μέθοδος της κλάσης BufferedWriter flush() θα διαγράψει ό,τι είναι γραμμένο στην καθορισμένη προσωρινή αποθηκευτική μνήμη και κατόπιν θα το μεταφέρει στο αρχείο εκροών. Οι σειρές 43- 45 χρησιμοποιούνται για ακόμη μια φορά για την διαχείριση σφαλμάτων σχετιζόμενων με τα αρχεία εισροών και εκροών.

```
47      public void closeRFile() {
48          try {
49              m_Reader.close();
50          }
51          catch (IOException ioex) {
52              ioex.printStackTrace();
53          }
54      }
55      public void closeWFile() {
56          try {
57              m_Writer.close();
58          }
59          catch (IOException ioex) {
60              ioex.printStackTrace();
61          }
62      }
```

Οι τελευταίες δυο μέθοδοι της κλάσης XXFile βρίσκονται στις σειρές 47-54 και 55-62 με ονόματα closeRFile και closeWFile. Χρήση τους γίνεται στην μέθοδο StartProcess της κλάσης CSVDataParser. Στόχο έχουν την τελμάτωση της διαδικασίας ανάγνωσης και γραφής των κλάσεων BufferedReader και BufferedWriter. Οι σειρές 49 και 57 χρησιμοποιώντας τις εγγενείς μεθόδους close() διαγράφουν όλα τα δεδομένα που υπάρχουν στις καθορισμένες από τις κλάσεις αυτές προσωρινές μνήμες αποθήκευσης ελευθερώνοντας όλους τους δεσμευμένους αποθηκευτικούς πόρους. Ως αποτέλεσμα καθίσταται αδύνατη η περεταίρω ανάγνωση και εγγραφή στα αρχεία εισροών και εκροών και το τελικό επιθυμητό αρχείο έχει παραχθεί.

2.4 *Εξαγώγιμα αποτελέσματα*

Η δοκιμή της προσέγγισης που αναπτύχθηκε στην παράγραφο 2.3 πάνω στο αρχικό σύνολο δεδομένων των 689600 εγγραφών απαραίτητο είναι να μορφοποιεί κατάλληλα το αρχείο των εγγραφών ούτως ώστε η επεξεργασία του να επιτελείται με τον βέλτιστο δυνατό τρόπο. Για τον λόγο αυτό η προηγούμενη παράγραφος επιχείρησε να απομονώσει τα σημαντικά πεδία που πρόκειται να απασχολήσουν την παρούσα διπλωματική εργασία καθώς και να αφαιρέσει τον

θόρυβο των περιττών πεδίων. Επιπροσθέτως διενεργήθηκαν μεταβολές στην σειρά με την οποίαν εμφανίζονται τα πεδία ούτως ώστε να επιτευχθεί καλύτερη συνοχή ανάμεσα στο είδος των περιεχομένων του κάθε πεδίου και προστέθηκε μοναδικός αναγνωριστικός αριθμός για κάθε εγγραφή.

url	city	price	year	manufacture	make	condition	cylinders	fuel	odometer	title_status	transmissior	VIN	drive	size	type	paint_color	image_url	lat	long
https://tricitie	tricities	5200	2007	audi	a4			gas		clean	automatic						https://image	35.8762	-84.1746
https://tricitie	tricities	5000	1978	ford	bronco			gas		clean	automatic						https://image	37.13284	-95.78558
https://tricitie	tricities	11300	2011	honda	cr-v			gas		clean	automatic						https://image	36.513501	-82.530221
https://tricitie	tricities	5000	2008	buick	lucerne cxl \	like new	6 cylinders	gas	51000	clean	automatic		fwd	full-size	sedan	grey	https://image	35.777999	-83.612533
https://tricitie	tricities	13500	2006		Pont GTO	excellent	8 cylinders	gas	93000	clean	automatic		rwd	mid-size	coupe	red	https://image	36.3339	-82.3408
https://tricitie	tricities	12500	2008	chevrolet	1500 silverac	excellent	8 cylinders	gas	125234	clean	automatic		4wd	full-size	pickup	red	https://image	36.3339	-82.3408
https://tricitie	tricities	6200	2006	mercedes-benz				gas		rebuilt	automatic						https://image	36.000092	-84.018302
https://tricitie	tricities	1	1999	ford	f150	excellent	8 cylinders	gas		clean	automatic		4wd	full-size	pickup	white	https://image	36.2954	-82.4902
https://tricitie	tricities	37900	2016	ford	f350	excellent	8 cylinders	diesel	70500	clean	automatic		4wd		truck	white	https://image	36.272932	-82.537537
https://tricitie	tricities	4999	2007	chevrolet	trailblazer	excellent	6 cylinders	gas	160000	clean	automatic		4wd				https://image	36.3107	-82.381
https://tricitie	tricities	7900	2000		Wrangler T.	good	4 cylinders	gas		clean	manual		4wd		SUV	red	https://image	36.3996	-82.4523
https://tricitie	tricities	1200	1992	mercedes-b	benz			gas	194000	clean	automatic						https://image	36.4162	-83.0108
https://tricitie	tricities	5000	1964	chevrolet	c-10		8 cylinders	gas		clean	automatic						https://image	36.2832	-83.0374
https://tricitie	tricities	5000	2003	chevrolet	silverado	like new	8 cylinders	gas		clean	automatic						https://image	36.3445	-82.2015
https://tricitie	tricities	9950	2010	nissan	armada	excellent	8 cylinders	gas	155244	clean	automatic	5N1AA0NC7	4wd	full-size	SUV	black	https://image	37.286501	-80.056719
https://tricitie	tricities	5950	2003	ford	expedition	excellent	8 cylinders	gas	141602	clean	automatic	1FMFU18L73	4wd	full-size	SUV	green	https://image	37.286501	-80.056719
https://tricitie	tricities	8950	2007	chevrolet	avalanche	excellent	8 cylinders	gas	205932	clean	automatic	3GNEC120X	rwd	full-size	SUV	black	https://image	37.286501	-80.056719
https://tricitie	tricities	7950	2000	ram	2500	excellent	8 cylinders	gas	199237	clean	automatic	1B7KF23Z6Y	4wd	full-size	truck	blue	https://image	37.286501	-80.056719

Πίνακας 11 Αρχικό σύνολο δεδομένων εγγραφών πριν την χρήση του αλγορίθμου της παραγράφου 2.3

id	city	price	year	odometer	lat	long	manufacti	make	condition	cylinders	fuel	title_statu	transmissi	drive	size	type	paint_color
1	tricities	5200	2007		35.8762	-84.175	audi	a4			gas	clean	automatic				
2	tricities	5000	1978		37.1328	-95.786	ford	bronco			gas	clean	automatic				
3	tricities	11300	2011		36.5135	-82.53	honda	cr-v			gas	clean	automatic				
4	tricities	5000	2008	51000	35.778	-83.613	buick	lucerne cx	like new	6 cylinder	gas	clean	automatic	fwd	full-size	sedan	grey
5	tricities	13500	2006	93000	36.3339	-82.341		Pont GTO	excellent	8 cylinder	gas	clean	automatic	rwd	mid-size	coupe	red
6	tricities	12500	2008	125234	36.3339	-82.341	chevrolet	1500 silve	excellent	8 cylinder	gas	clean	automatic	4wd	full-size	pickup	red
7	tricities	6200	2006		36.0001	-84.018	mercedes-	benz			gas	rebuilt	automatic				
8	tricities	1	1999		36.2954	-82.49	ford	f150	excellent	8 cylinder	gas	clean	automatic	4wd	full-size	pickup	white
9	tricities	37900	2016	70500	36.2729	-82.538	ford	f350	excellent	8 cylinder	diesel	clean	automatic	4wd		truck	white
10	tricities	4999	2007	160000	36.3107	-82.381	chevrolet	trailblazer	excellent	6 cylinder	gas	clean	automatic	4wd			
11	tricities	7900	2000		36.3996	-82.452		Wrangler T	good	4 cylinder	gas	clean	manual	4wd		SUV	red
12	tricities	1200	1992	194000	36.4162	-83.011	mercedes-	benz			gas	clean	automatic				
13	tricities	5000	1964		36.2832	-83.037	chevrolet	c-10		8 cylinder	gas	clean	automatic				
14	tricities	5000	2003		36.3445	-82.202	chevrolet	silverado	like new	8 cylinder	gas	clean	automatic				
15	tricities	9950	2010	155244	37.2865	-80.057	nissan	armada	excellent	8 cylinder	gas	clean	automatic	4wd	full-size	SUV	black
16	tricities	5950	2003	141602	37.2865	-80.057	ford	expedition	excellent	8 cylinder	gas	clean	automatic	4wd	full-size	SUV	green
17	tricities	8950	2007	205932	37.2865	-80.057	chevrolet	avalanche	excellent	8 cylinder	gas	clean	automatic	rwd	full-size	SUV	black
18	tricities	7950	2000	199237	37.2865	-80.057	ram	2500	excellent	8 cylinder	gas	clean	automatic	4wd	full-size	truck	blue

Πίνακας 12 Σύνολο δεδομένων εγγραφών κατόπιν της χρήσης του αλγορίθμου μορφοποίησης της παραγράφου 2.3

3 *ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΚΟΡΥΦΟΓΡΑΜΜΗΣ*

3.1 *Περιγραφή*

Η τεχνολογική πρόοδος που χαρακτηρίζει τις τελευταίες δεκαετίες οδήγησε αναμφισβήτητα στην κυριαρχία της επιστήμης πάνω σε μυριάδα προβλημάτων βελτιστοποίησης με την εφαρμογή κατάλληλων αλγοριθμικών σεναρίων τροποποιημένων έτσι ώστε οι παραγόμενες λύσεις να είναι όντως οι βέλτιστες και η παραγωγή τους να γίνεται με τον αποδοτικότερο δυνατό τρόπο. Το πρόβλημα της κορυφογραμμής σχετίζεται με την επιλογή ενός βέλτιστου υποσυνόλου στοιχείων με n διαστάσεις, τέτοιο ώστε κάθε σημείο του υποσυνόλου αυτού να μην κυριαρχείται από κάποιο άλλο. Η έννοια της κυριαρχίας επί στοιχείο αφορά σε κάθε ξεχωριστή διάσταση που ενδιαφέρει την εκάστοτε εφαρμογή. Ένα στοιχείο α είναι κυρίαρχο έναντι κάποιου άλλου στοιχείου β αν είναι ισοδύναμο ή καλύτερο σε όλες τις διαστάσεις τους β και τουλάχιστον καλύτερο σε μια από αυτές. Η κορυφογραμμή ενός συνόλου στοιχείων αποτελείται από όλα τα στοιχεία εκείνα που δεν κυριαρχούνται από κανένα στοιχείο πλην του εαυτού τους. Η κορυφογραμμή ενός συνόλου στοιχείων μπορεί να υποδείξει με αυτόν τον τρόπο την βέλτιστη επιλογή από ένα πλήθος σεναρίων και να κατευθύνει σε πλέον αποδοτικά αποτελέσματα[6].

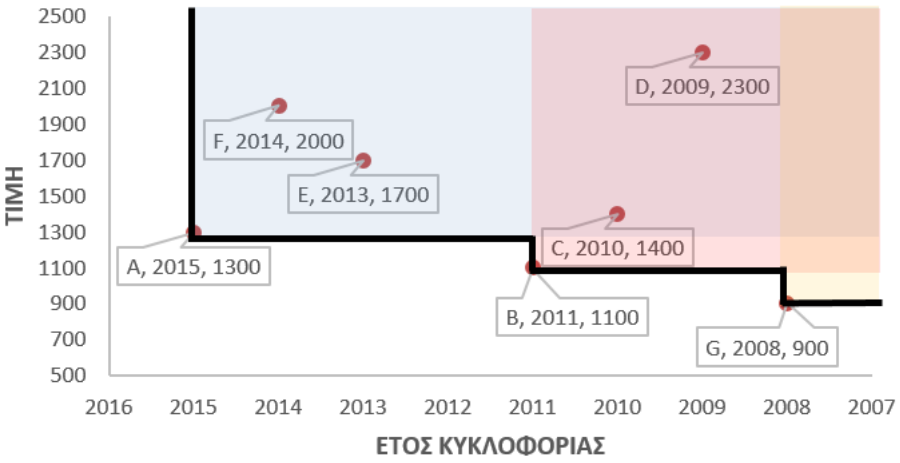
Ένα απλό παράδειγμα επερωτήματος κορυφογραμμής θα μπορούσε να αποτελέσει η απόφαση αγοράς ενός αυτοκινήτου βάσει της τιμής του και του έτους κυκλοφορίας του. Δοθέντος ενός συνόλου δεδομένων υποψήφιων αυτοκινήτων τριών διαστάσεων: όνομα, τιμή σε δολάρια και έτος κυκλοφορίας, η κορυφογραμμή του συνόλου αυτού θα είναι το υποσύνολο του αρχικού συνόλου τέτοιο ώστε κάθε σημείο που ανήκει σε αυτό να μην κυριαρχείται βάσει της τιμής και του έτους κυκλοφορίας. Μια εγγραφή θα κυριαρχεί κάποια άλλη όταν έχει βέλτιστα τα πεδία της τιμής και του έτους κυκλοφορίας. Ως βέλτιστη τιμή θα θεωρηθεί η χαμηλότερη ενώ ως βέλτιστο έτος κυκλοφορίας θα θεωρηθεί το πίο πρόσφατο. Ακολουθεί πίνακας στον οποίο διαφαίνοντα τα στοιχεία του αρχικού συνόλου δεδομένων, καθώς και η αναπαράσταση τους σε δισδιάστατο χώρο

[6] (Kalyvas Christos, 2017)

με οριζόντιο άξονα το έτος κυκλοφορίας σε φθίνουσα κατάταξη και κάθετο άξονα την τιμή του αυτοκινήτου σε αύξουσα.

ΑΥΤΟΚΙΝΗΤΟ	ΤΙΜΗ ($)	ΕΤΟΣ ΚΥΚΛΟΦΟΡΙΑΣ
A	1300	2015
B	1100	2011
C	1400	2010
D	2300	2009
E	1700	2013
F	2000	2014
G	900	2008

Πίνακας 11 Παράδειγμα αυτοκινήτων με χαρακτηριστικά την τιμή και το έτος κυκλοφορίας



Εικόνα 22 Αναπαράσταση κορυφογραμμής των αυτοκινήτων του παραδείγματος

Στο γράφημα η έντονη μαύρη γραμμή αποτελεί την κορυφογραμμή του αρχικού συνόλου των εγγραφών αυτοκινήτων. Επάνω σε αυτήν βρίσκονται τρία αυτοκίνητα, το A,B και C με κοινό χαρακτηριστικό το ότι δεν κυριαρχούνται από καμία άλλη εγγραφή αυτοκινήτου ως προς την χαμηλότερη τιμή και το πιο πρόσφατο έτος κυκλοφορίας. Πιο συγκεκριμένα παρατηρεί κανείς πως το αυτοκίνητο A κυριαρχεί επι των στοιχείων εντός του γαλάζιου πλαισίου, δηλαδή όλων των στοιχείων εκτός των A,B,G. Αντίστοιχα το αμάξι B κυριαρχεί επι των στοιχείων εντός του κόκκινου πλαισίου και το G επι αυτών εντός του κίτρινου. Ως αποτέλεσμα, η εύρεση της κορυφογραμμής υποδεικνύει τρείς βέλτιστες επιλογές για την αγορά του αντίστοιχου αυτοκινήτου.

Η πορεία της κορυφογραμμής σε κάθε περίπτωση εξαρτάται από μια συνάρτηση που προσαρμόζεται ανάλογα με πεδία που πρεπει να βελτιστοποιηθούν. Αυτή η συνάρτηση ονομάζεται συνάρτηση κατάταξης (scoring function) που είναι μονότονη σε όλες τις διαστάσεις[7]. Στην παραπάνω περίπτωση για παράδειγμα το αυτοκίνητο Α κυριαρχεί επι του αυτοκινήτου F καθώς είναι καλύτερο και στις 2 διαστάσεις (έτος κυκλοφορίας,τιμή) που αφορούν την επιλεχθείσα συνάρτης κατάταξης (ελαχιστοποίηση της τιμής και μεγιστοποίηση του έτους κυκλοφορίας). Συμπερασματικά, στη συγκεκριμένη περίπτωση η κορυφογραμμή παρέχει πληροφορίες που αφορούν τους πιθανούς συμβιβασμούς που κάποιος θα μπορούσε να κάνει ανάμεσα σε τιμή και έτος κυκλοφορίας του αυτοκινήτου για να επιλέξει τη βέλτιστη εκάστοτε επιλογή.

Ο υπλογισμός της κορυφογραμμής αποκτά ιδιαίτερη σημασία δεδομένης της ραγδαίας ανάπτυξης της ανάγκης επεξεργασίας όλο και μεγαλύτερων συνόλων δεδομένων για την εύρεση βέλτιστων τιμών. Το κόστος και ο χρόνος που σχετίζονται με τη μεταφορά ή επεξεργασία μεγάλου όγκου δεδομένων επηρρεάζονται σημαντικά απο την ανάγκη βελτιστοποίησης. Συνεπώς δεν αποτελεί έκπληξη η εφαρμογή που βρίσκει ο προσδιορισμός της κορυφογραμμής στον τομέα της έρευνας βάσεων δεδομένων. Παρόλα αυτά η ιδέα είναι γνώστη και σε άλλους κλάδους όπως τον Οικονομικό ως το πρόβλημα βελτιστοποίησης με πολλαπλά κριτήρια Pareto[8] αλλά και στον τομέα της Μαθηματικής βελτιστοποίησης ως το πρόβλημα της εύρεσης του μεγίστου διανύσματος[9].

3.2 *Αλγόριθμοι εύρεσης κορυφογραμμής*

Το 2001 ήταν η χρονιά που προτάθηκε η ιδέα χρήσης του τελεστή κορυφογραμμής σε context σχεσιακών συστημάτων μοντέλων βάσεων δεδομένων[10]. Οι δύο βασικοί αλγόριθμοι που προτείνονται για τον υπολογισμό της κορυφογραμμής συνόλου εγγραφών είναι ο BNL (Block Nested Loops) και αλγόριθμος που βασίζεται στην μέθοδο επίλυσης υπολογιστικών προβλημάτων Διαίρει και Βασίλευε (Divide and Conquer).

[7] (Papadias D., 2003)

[8] (Kacem Imed, 2002)

[9] (Godfrey Parke, 2006)

[10] (Borzsonyi, 2001)

3.2.1 Μαθηματικό υπόβαθρο κορυφογραμμής

Αρχικά, ωστόσο, θα γίνει μια προσπάθεια επεξήγησης της μαθηματικής βάσης πίσω από το πρόβλημα της κορυφογραμμής. Ας δοθεί λοιπόν ο επίσημος ορισμός του προβλήματος κορυφογραμμής.

Ορισμός 1: Δοθέντος συνόλου X n-διάστατων σημείων $x_i = (d_1, d_2, \dots, d_n)$, ορίζουμε μια νέα σχέση κυριαρχίας $\neg$ ανάμεσα σε δυο σημεία $x_i = (d_1, d_2, \dots, d_n)$, $x_j = (\kappa_1, \kappa_2, \dots, \kappa_n)$, $i \neq j$ ως εξής. Αν $x_i \neg x_j$ θα λέμε πως το σημείο x_i κυριαρχεί επί του σημείου x_j όταν δοθείσης συνάρτησης κατάταξης φ, $\forall d_q, k_q : \varphi(d_q) \geq \varphi(k_q)$ $q = 1,2, \cdots, n$ και τουλάχιστον για κάποιο $q, \varphi(d_q) > \varphi(k_q)$. Η φ θα λαμβάνει υπόψιν κάθε διάσταση ξεχωριστά και θα είναι μονότονη. Η κορυφογραμμή του συνόλου X ορίζεται ως το υποσύνολο S του X που περιέχει στοιχεία $x_i \epsilon X$ τα οποία έχουν την εξής ιδιότητα., $x_i \epsilon S \Leftrightarrow \nexists x_j \epsilon X : x_j \neg x_i$, $i \neq j \Leftrightarrow \forall\, x_j \epsilon X : x_j ! \neg x_i$ (x_j δεν κυριαρχεί επι του x_i). Δηλαδή η κορυφογραμμή αποτελείται από στοιχεία που δεν κυριαρχούνται από κάποιο άλλο στοιχείο του X.

Λήμμα 1: Αν $x_i, x_j x_z \epsilon X, i \neq j \neq z$ $με$ $x_i \neg x_j$ $και$ $x_j \neg x_z \Rightarrow x_i \neg\, x_z$

Απόδειξη:

Έστω τα σημεία $x_i = (d_1, d_2, \dots, d_n)$, $x_j = (\kappa_1, \kappa_2, \dots, \kappa_n)$, $x_z = (t_1, t_2, \dots, t_n)$

Αφού $x_i \neg x_j \Rightarrow \forall d_q, k_q : \varphi(d_q) \geq \varphi(k_q)$ (1) $q = 1,2, \cdots, n$ και τουλάχιστον για κάποιο $q_1 : \varphi(d_{q_1}) > \varphi(k_{q_1})$ (3).

Ομοίως $x_j \neg x_z \Rightarrow \forall k_q, t_q : \varphi(k_q) \geq \varphi(t_q)$ (2) $q = 1,2, \cdots, n$ και $\exists q_2 : \varphi(k_{q_2}) > \varphi(t_{q_2})$ (4).

Από τις (1), (2) προκύπτει ότι $\varphi(d_q) \geq \varphi(k_q)$ && $\varphi(k_q) \geq \varphi(t_q)$ $\forall q \epsilon [1, n] \setminus \{q_1, q_2\}$.

Εφόσον η φ αποτελεί μονότονη συνάρτηση συνεπάγεται πως $\varphi(d_q) \geq \varphi(t_q)$ $\forall q \epsilon [1, n] \setminus \{q_1, q_2\}$.

Τέλος τα παραπάνω σε συνδυασμό με τις (3),(4)

$$\varphi(d_{q_1}) > \varphi(k_{q_1}) \geq \varphi(t_{q_1}) \;\&\&\; \varphi(d_{q_2}) \geq \varphi(k_{q_2}) > \varphi(t_{q_2}) \Rightarrow \varphi(d_{q_1}) > \varphi(t_{q_1}) \; και \; \varphi(d_{q_2}) > \varphi(t_{q_2}).$$

Συνοψίζοντας αποδείχθηκε πως

$$\forall q\epsilon[1,n], \varphi(d_q) \geq \varphi(t_q)\ \kappa\alpha\iota\ \exists q_1\colon\ \varphi(d_{q_1}) > \varphi(t_{q_1}) \Rightarrow x_i \neg x_z$$

<u>Ορισμός 2</u>: Αν $x_i, x_j \epsilon X, i \neq j\ \theta\alpha\ \lambda\acute{\epsilon}\mu\epsilon\ \pi\omega\varsigma\ \tau o\ x_i$ δεν είναι συγκρίσιμο κατά $\neg\ \mu\epsilon\ \tau o\ x_j$,όταν ταυτόχρονα $x_i!\neg x_j\ \kappa\alpha\iota\ x_j!\neg x_i$. Ο συμβολισμός που θα χρησιμοποιείται όταν x_i δεν είναι συγκρίσιμο κατά ¬ με x_j είναι $x_i \nless x_j$. Είναι προφανές πως αν $x_i \nless x_j \Rightarrow\ x_j \nless x_i$.

<u>Θεώρημα 1:</u> Αν $x_i, x_j \epsilon X, i \neq j$ με $x_i = (d_1, d_2, \dots, d_n)$, $x_j = (\kappa_1, \kappa_2, \dots, \kappa_n)$. Τότε

$$x_i \nless x_j\ \alpha\nu - \nu\ \exists n_1, n_2 \epsilon [1,n], n_1 \neq n_2 : \varphi(d_{n_1}) > \varphi(k_{n_1})\ \kappa\alpha\iota\ \varphi(k_{n_2}) >\ \varphi(d_{n_2})$$

<u>Απόδειξη :</u>

I. (→)

Από τον Ορισμό 2 έχουμε $x_i \nless x_j \Rightarrow x_i!\neg x_j$ && $x_j!\neg x_i$. Μέσω της άρνησης του Ορισμού 1 $x_i!\neg x_j\ \Rightarrow \exists q\epsilon[1,n]\ : \varphi(d_q) < \varphi(k_q)$. Ομοίως $x_j!\neg x_i \Rightarrow \exists p\epsilon[1,n]:\ \varphi(k_q)\ < \varphi(d_p)$. Λόγω του ορισμού της συνάρτησης κατάταξης φ ως μονότονη ισχύει $p \neq q$, αποδεικνύοντας το ζητούμενο.

II. (←)

Έστω πως $x_i \neg x_j$ ή $x_j \neg x_i$

$$\exists n_1, n_2 \epsilon [1,n], n_1 \neq n_2 : \varphi(d_{n_1}) > \varphi(k_{n_1})\ \kappa\alpha\iota\ \varphi(k_{n_2}) >\ \varphi(d_{n_2})$$

Όμως εφόσον $x_i \neg x_j \Rightarrow \forall n_k \epsilon [1,n] : \varphi(d_{n_k}) \geq \varphi(k_{n_k})$, άτοπο γιατί $\varphi(k_{n_2}) >\ \varphi(d_{n_2})$.

Ομοίως αν $x_i \neg x_j \Rightarrow \forall n_k \epsilon [1,n] : \varphi(k_{n_k}) \geq \varphi(d_{n_k})$, άτοπο γιατί $\varphi(d_{n_1}) >\ \varphi(k_{n_1})$.

Συνολικά συμπεραίνουμε πως $!\left(x_i \neg x_j\ \text{ή}\ x_j \neg x_i\right) \Rightarrow x_i!\neg x_j\ \ \kappa\alpha\iota\ x_j!\neg x_i\ \Rightarrow x_i \nless x_j$ που είναι το ζητούμενο.

<u>Θεώρημα 2:</u> Δοθέντων στοιχείου $x_i \epsilon X$, και συνόλου στοιχείων κορυφογραμμής $S \subseteq X$ ισχύει

$$\forall x_j \epsilon X, i \neq j : x_j \nless x_i\ \text{ή}\ x_i \neg x_j\ \ \alpha\nu - \nu\ \ x_i \epsilon S$$

<u>Απόδειξη :</u>

Έστω $x_i = (d_1, d_2, \dots, d_n)$, $x_j = (\kappa_1, \kappa_2, \dots, \kappa_n)$

I. (←)

$$x_i \epsilon S \Rightarrow\ \nexists x_j \epsilon X : x_j \neg x_i \Rightarrow \forall x_j \epsilon X\ , x_j!\neg x_i =>$$

$$(\ \exists m \epsilon [1,n]\colon\ \varphi(\kappa_m) < \varphi(d_m)\ )\ ||\ \ \left(\ \forall q \epsilon [1,n]\colon\ \varphi(\kappa_q) \leq \varphi(d_q)\right)\quad (1)$$

Το πρώτο σκέλος της (1) θα ονομαστεί Α, και το δεύτερο Β. Η (1) είναι αληθής σε τρείς περιπτώσεις:

- Α αληθής Β ψευδής

  $(1) \Rightarrow \exists m\epsilon[1,n]:\ \varphi(\kappa_m) < \varphi(d_m)\ και\ \exists q\epsilon[1,n]:\ \varphi(\kappa_q) > \varphi(d_q)$

  $\Rightarrow\ x_j \not\prec x_i\ μέσω\ του\ θεωρήματος\ 1.$

- Α αληθής Β αληθής

  $(1) \Rightarrow \exists m\epsilon[1,n]:\ \varphi(\kappa_m) < \varphi(d_m)\ \ και\ \ \forall q\epsilon[1,n]:\ \varphi(\kappa_q) \leq \varphi(d_q)$

  $\Rightarrow x_i \neg x_j\ μέσω\ του\ Ορισμού\ 1$

- Α ψευδής Β αληθής

  $(1) \Rightarrow\ \forall m\epsilon[1,n]:\ \varphi(\kappa_m) \geq \varphi(d_m)\ και\ \ \forall q\epsilon[1,n]:\ \varphi(\kappa_q) \leq \varphi(d_q)\ \Rightarrow$

  $Αν\ m = q \Rightarrow οι\ διαστάσεις\ των\ σημείων\ ταυτίζονται$ *, οπότε εφόσον* $x_i\epsilon S \Rightarrow x_i \neg x_j$ *. Εναλλακτικά* $x_j \not\prec x_i\ μέσω\ του\ θεωρήματος\ 1$

Συνολικά συμπεραίνουμε πως $\forall x_j\epsilon X, i \neq j : x_j \not\prec x_i$ ή $x_i \neg x_j$, που είναι το ζητούμενο.

II. $(\rightarrow)$

$$\forall x_j\epsilon X, i \neq j : x_j \not\prec x_i \text{ ή } x_i \neg x_j \quad (2)$$

Το πρώτο σκέλος της (2) θα ονομαστεί Γ, και το δεύτερο Δ.

- Αν Γ αληθής $\Rightarrow \forall x_j\epsilon X, i \neq j : x_j \not\prec x_i \Rightarrow \forall x_j\epsilon X, i \neq j :$

  $x_i !\neg x_j\ \ και\ x_j !\neg x_i\ μέσω\ του\ ορισμού\ 2 \Rightarrow \forall x_j\epsilon X, i \neq j\quad x_j !\neg x_i\ \Rightarrow\ x_i\epsilon S$

- Αν Δ αληθής$\Rightarrow \forall x_j\epsilon X, i \neq j : x_i \neg x_j \Rightarrow \nexists x_j\epsilon X, i \neq j\ \ : x_j \neg x_i\ \Rightarrow\ x_i\epsilon S$

Συνολικά συμπεραίνουμε πως $x_i\epsilon S$ που είναι το ζητούμενο.

<u>Πρόταση 1:</u> Έστω σύνολο στοιχείων $W \subseteq X$ με σημεία μη συγκρίσιμα κατά $\neg$ ανά δυο μεταξύ τους, δηλαδή $\forall w_i, w_j\epsilon W\ i \neq j : w_i \not\prec w_j$. Τότε $\nexists x\epsilon\ X \setminus W : w_i \neg x\ \ και\ x \neg w_j$

<u>Απόδειξη :</u>

Ας υποτεθεί πως $\exists x\epsilon\ X \setminus W : w_i \neg x\ \ και\ x \neg w_j$.Έστω $w_i = (\alpha_1, \alpha_2, \dots, \alpha_n), w_j = (b_1, b_2, \dots, b_n), x = (c_1, c_2, \dots, c_n).$

$w_i \not\prec w_j \Rightarrow \exists n_1, n_2\epsilon[1,n], n_1 \neq n_2 : \varphi(a_{n_1}) > \varphi(b_{n_1})\ και\ \varphi(b_{n_2}) >\ \varphi(b_{n_2})$μέσω του Θεωρήματος 1.

$$w_i \neg x \Rightarrow \forall q \epsilon [1,n]: \varphi(\alpha_q) \geq \varphi(c_q) \; και \; \exists \kappa \epsilon [1,n]: \varphi(\alpha_q) > \varphi(c_q) \quad (1)$$

$$x \neg w_j \Rightarrow \forall q \epsilon [1,n]: \varphi(c_q) \geq \varphi(b_q) \; και \; \exists \lambda \epsilon [1,n]: \varphi(c_\lambda) > \varphi(b_\lambda) \quad (2)$$

Από τις (1),(2)$\Rightarrow \forall q \epsilon [1,n]: \varphi(\alpha_q) \geq \varphi(c_q) \; και \; \varphi(c_q) \geq \varphi(b_q) \Rightarrow \forall q \epsilon [1,n] : \varphi(\alpha_q) \geq \varphi(b_q)$, το οποίο είναι άτοπο καθώς $\exists n_2 \, \epsilon [1,n] \; \varphi(b_{n_2}) > \varphi(b_{n_2})$.

Εναλλακτικά, εφόσον υπέθεσα πως $w_i \neg x \; και \; x \neg w_j \Rightarrow w_i \neg w_j$ μέσω του Λήμματος 1, το οποίο είναι άτοπο καθώς $w_i \neq w_j$.

Πρόταση 2: Έστω σύνολο στοιχείων $W \subseteq X$ με σημεία μη συγκρίσιμα κατά $\neg$ ανά δυο μεταξύ τους, δηλαδή $\forall w_i, w_j \epsilon W \; i \neq j : w_i \neq w_j$ Τότε $\forall x \epsilon X$

$$x \neq w \text{ ή } w \neg x \quad \alpha \nu - \nu \; W \subseteq S$$

Χάριν ευκολίας, ένα τέτοιο υποσύνολο W θα το ονομάζουμε *βέλτιστο υποσύνολο* του X.

Απόδειξη :

I. ($\leftarrow$)

Αν $W \subseteq S \Rightarrow w \epsilon S \Rightarrow \forall x \epsilon X, x \neq w : x \neq w$ ή $w \neg x$ μέσω του Θεωρήματος 2.

II. ($\rightarrow$)

$\forall w \epsilon W$: $\forall x \epsilon X \; x \neq w$ ή $w \neg x \Rightarrow w \epsilon S \; μέσω \; του \; θεωρήματος \; 2 \Rightarrow W \subseteq S$

3.2.2 Αφελής προσέγγιση κορυφογραμμής

Σε αυτό το σημείο μπορεί να προταθεί η αφελής προσέγγιση επίλυσης του προβλήματος κορυφογραμμής. Δεδομένου πως σκοπός είναι η εύρεση του συνόλου εκείνου των στοιχείων που δεν κυριαρχούνται από κάποιο άλλο στοιχείο του X, αν αποφανθούμε για τη σχέση κυριαρχίας ανάμεσα σε κάθε πιθανή δυάδα στοιχείων, ευκόλως θα προκύψει το σύνολο των στοιχείων κορυφογραμμής.

Naive Skyline

Input: set $\boldsymbol{X}$ of points

Output: set $\boldsymbol{S}$ of skyline points

```
1.  S initialization as the empty set
2.  For x_i of X
3.          isDominated: = false;
4.          For x_j of X
5.                  if (i == j)
6.                          continue;
7.                  if (x_j ¬ x_i)
8.                          isDominated: = true;
9.                          break;
10.         end For
11.         if (isDominated == false)
12.                 Add x_i to S;
13. end For
14. return S
```

Εικόνα 23: Ο ψευδοκώδικας της αφελούς προσέγγισης του προβλήματος κορυφογραμμής

Πιο αναλυτικά ο παραπάνω ψευδοκώδικας διατρέχει το σύνολο X μέσω των βρόγχων For στις σειρές 2 και 4 και δοκιμάζει κάθε στοιχείο x_i με όλα τα υπόλοιπα στοιχεία x_j. Σε περίπτωση που κάποιο άλλο στοιχείο κυριαρχεί το x_i, αυτό απορρίπτεται ως πιθανό σημείο της κορυφογραμμής και δεν επεξεργάζονται στοιχεία πέραν του x_j όπως υποδεικνύουν οι σειρές 7-9. Η Μπουλιανή μεταβλητή $isDominated$ θα λάβει τιμή true σε περίπτωση κυριαρχίας επί του εξεταζόμενου στοιχείου x_i. Εναλλακτικά αν δεν υπάρχει στοιχείο x_j τέτοιο ώστε να κυριαρχεί το x_i, τότε το x_i αποτελεί σημείο της κορυφογραμμής και στην γραμμή 12 εισάγεται στο σύνολο Κορυφογραμμής S. Ακολουθεί η απόδειξη ορθότητας του παραπάνω αλγορίθμου.

<u>Απόδειξη</u>:

Υπόθεση: Έστω το μέγεθος του συνόλου X είναι m. Τότε σύνολο S περιέχει την κορυφογραμμή των m πρώτων σημείων του X με το πέρας των βρόγχων For στις σειρές 2 και 4, δηλαδή με το πέρας της επεξεργασίας του m-οστού στοιχείου.

Η παραπάνω υπόθεση θα αποδειχθεί μέσω της μαθηματικής επαγωγής. Ως σταθερά κατάσταση του βρόγχου θα θεωρηθεί πως : Με το πέρας της επεξεργασίας του ν-οστού στοιχείου, το S περιέχει την κορυφογραμμή των ν πρώτων στοιχείων του X.

Για $\nu = 0$, δηλαδή για την περίπτωση $i = 0, j = m$ που αποτελεί την εισαγωγή στον βρόγχο το S είναι το κενό σύνολο, που αποτελεί την κορυφογραμμή των πρώτων 0 στοιχείων, δίχως να άρει την σταθερά του βρόγχου.

Για $\nu = 1$, δηλαδή για την περίπτωση $i = 1, j = m$ που είναι το πέρας της επεξεργασίας του πρώτου στοιχείου, το $S = \{x_1\}$. Η κορυφογραμμή του πρώτου στοιχείου είναι το ίδιο το στοιχείο.

Έστω ότι η πρόταση ισχύει για ν=κ. Δηλαδή το S περιέχει την κορυφογραμμή των κ πρώτων στοιχείων.

Για $\nu = \kappa + 1$, δηλαδή για την περίπτωση που $i = k + 1\ , j = m$, διακρίνονται δυο περιπτώσεις.

Πρώτη περίπτωση είναι η ύπαρξη τουλάχιστον ενός σημείου $x_r \epsilon X, r \leq m, r \neq\ k + 1\ : x_r \neg x_{k+1}$. Σε αυτήν την περίπτωση το σημείο x_{k+1} κυριαρχείται από κάποιο σημείο και κατ' επέκταση δεν ανήκει στην κορυφογραμμή. Οι γραμμές 7-9 θα οδηγήσουν στο πέρας της επεξεργασίας του $\kappa +$ 1- οστού σημείου και οι γραμμές 11-12 θα το απορρίψουν ως σημείο της κορυφογραμμής. Συνεπώς με το πέρας της επεξεργασίας του $\kappa + 1$-οστού σημείου το S θα περιέχει την κορυφογραμμή των πρώτων $\kappa + 1$ στοιχείων. Οπότε διατηρείται το αναλλοίωτο της συνθήκης βρόγχου.

Η δεύτερη περίπτωση είναι η μη ύπαρξη σημείου $x_r \epsilon X, r \leq m, r \neq\ k + 1\ : x_r \neg x_{k+1}$. Σε αυτήν την περίπτωση δεδομένου πως το σημείο x_{k+1} δεν κυριαρχείται από κανένα άλλο σημείο $x_r \epsilon X$, ανήκει στην κορυφογραμμή. Οι γραμμές 11-12 θα εισάγουν το x_{k+1} επιτυχώς στο σύνολο S της κορυφογραμμής. Συνεπώς με το πέρας της επεξεργασίας του $\kappa + 1$-οστού σημείου το S θα περιέχει την κορυφογραμμή των πρώτων $\kappa + 1$ στοιχείων, διατηρώντας έτσι και πάλι το αναλλοίωτο της συνθήκης βρόγχου.

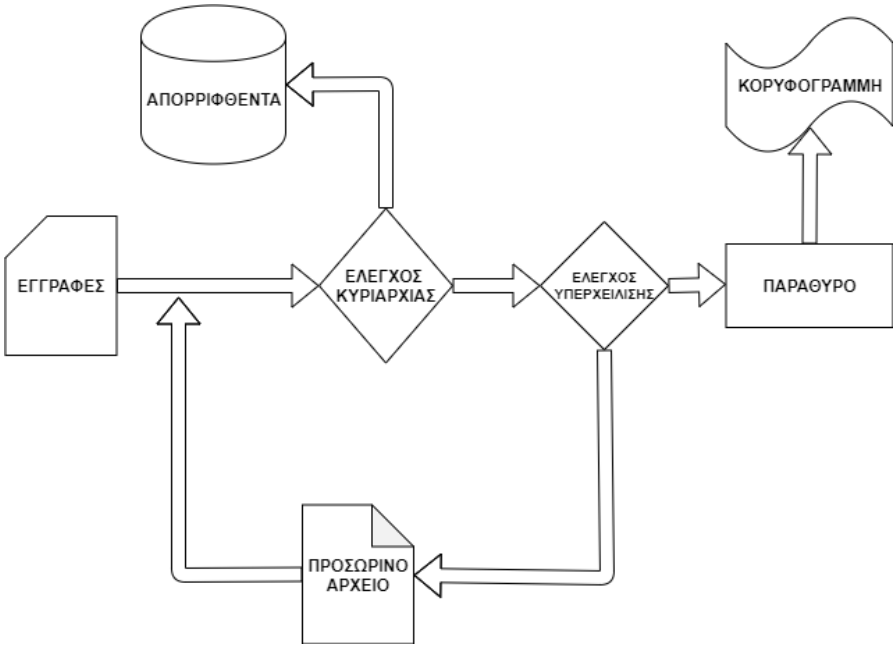
Αποδεικνύεται λοιπόν πως η αρχική υπόθεση ισχύει μέσω της μαθηματικής επαγωγής.

3.2.3 Ο αλγόριθμος BNL

Οι Borzsonyi et al.[11] πατώντας πάνω αφελή προσέγγιση επίλυσης του προβλήματος κορυφογραμμής της §3.2.2, προτείνουν το 2001 τον αλγόριθμο BNL (block nested loops) σε context σχεσιακών βάσεων δεδομένων. Παρατηρείται πως η αφελής προσέγγιση, η οποία χρησιμοποιούνταν μέχρι τότε για την εύρεση της κορυφογραμμές σε βάσεις δεδομένων είναι μη αποδοτική και προτείνεται γρηγορότερος αλγόριθμος που βασίζεται σε ένα δυναμικά

[11] (Borzsonyi, 2001)

μεταβαλλόμενο παράθυρο δεσμευμένο από την κύρια μνήμη στο οποίο θα αποθηκεύονται σημεία που δεν κυριαρχούνται ανά δυο μεταξύ τους. Σε περίπτωση που ο αποθηκευτικός χώρος από την κύρια μνήμη είναι εξαντληθεί, γίνεται χρήση ενός προσωρινού αρχείου. Πιο συγκεκριμένα ένας αλγόριθμος χαρακτηρίζεται online όταν έχει την δυνατότητα να επεξεργάζεται σειριακά τα εισαγόμενα δεδομένα κομμάτι-κομμάτι. Ο BNL αποτελεί έναν online αλγόριθμο καθώς επεξεργάζεται εν μέρει τις εγγραφές μέχρι να καταλήξει στο επιθυμητό εξαγώγιμο, την κορυφογραμμή του συνόλου των εγγραφών.



Εικόνα 24 Το σχεσιακό διάγραμμα του BNL

Αναλυτικότερα, κάθε εγγραφή e των εισακτέων δεδομένων ακολουθεί την εξής διαδρομή σε σχέση με το παράθυρο που χρησιμοποιείται από τον BNL για την αποθήκευση μη συγκρίσιμων εγγραφών. Τίθενται δυο περιπτώσεις. Αν το παράθυρο είναι άδειο, το e εισάγεται κατευθείαν στο παράθυρο.Αν το παράθυρο έχει στοιχεία, τότε ελέγχεται η σχέση κυριαρχίας ¬ του e με όλα τα στοιχεία του παραθύρου. Τίθενται τρεις περιπτώσεις.

- Υπάρχουν κ εγγραφές μέσα στο παράθυρο που κυριαρχούν επι του e. Σε αυτήν την περίπτωση, με το πρώτο στοιχείο εντός του παραθύρου που κυριαρχεί επι το e, το e απορρίπτεται κατευθείαν ως πιθανό σημείο της κορυφογραμμής εφόσον κυριαρχείται από κάποιο άλλο στοιχείο και δεν επεξεργάζεται μελλοντικά.

- Το e κυριαρχεί επι κ εγγραφών του παραθύρου. Σε αυτήν την περίπτωση, κάθε κυριαρχημένη εγγραφή αφαιρείται από το παράθυρο και απορρίπτεται ως πιθανό σημείο της κορυφογραμμής δίχως να επεξεργάζεται μελλοντικά. Το e προτείνεται για εισαγωγή εντός του παραθύρου. Θα ακολουθήσει ο έλεγχος υπερχείλισης για να αποφασιστεί ο προορισμός του e.
- Το e είναι μη συγκρίσιμο κατά ¬ με κάθε εγγραφή του παραθύρου. Το e προτείνεται για εισαγωγή εντός του παραθύρου. Θα ακολουθήσει ο έλεγχος υπερχείλισης για να αποφασιστεί ο προορισμός του e.

Ο έλεγχος υπερχείλισης αφορά την χωρητικότητα που έχει αφιερώσει η κύρια μνήμη στο παράθυρο. Σε περίπτωση που η χωρητικότητα αυτή δεν έχει εξαντληθεί, το στοιχείο e εισάγεται κανονικά μέσα στο παράθυρο. Διαφορετικά το στοιχείο e αποθηκεύεται προσωρινά σε ένα προσωρινό αρχείο. Η διαδικασία αυτή θα συνεχιστεί μέχρι να εξαντληθούν όλες οι εγγραφές του εισαγόμενου αρχείου. Το γεγονός αυτό θα σηματοδοτήσει το πέρας της πρώτης επανάληψης του αλγορίθμου. Συνεπώς, με την επεξεργασία της τελευταίας εγγραφής από τα εισαγόμενα δεδομένα, τρείς είναι πλέον οι πιθανοί προορισμοί των εγγραφών. Πρώτος είναι το σύνολο των απορριφθέντων στοιχείων τα οποία δεν θα απασχολήσουν πλέον τον αλγόριθμο εφόσον δεν είναι υποψήφια σημεία της κορυφογραμμής, Δεύτερος προορισμός είναι το παράθυρο που περιέχει όλες τις εγγραφές που είναι μη συγκρίσιμες κατά ¬ μεταξύ τους και κυριαρχούν επι όλων των απορριφθέντων σημείων. Τρίτος προορισμός είναι το προσωρινό αρχείο στο οποίο έχουν περάσει όλες οι εγγραφές που δεν χωρούσαν στο κύριο παράθυρο.

Με το πέρας της κάθε επανάληψης, όλα τα σημεία εντός του παραθύρου μεταφέρονται στην κορυφογραμμή αδειάζοντας το παράθυρο. Αυτό συμβαίνει καθώς βρέθηκαν να είναι κυρίαρχα επι όλων των απορριφθέντων σημείων και μη συγκρίσιμα κατά ¬ με όλα τα στοιχεία που έχουν αποθηκευτεί στο προσωρινό αρχείο, συνεπώς σύμφωνα με την Πρόταση 2 της §3.2.1 όντως ανήκουν στην κορυφογραμμή. Σε περίπτωση που το προσωρινό αρχείο είναι άδειο, ο αλγόριθμος φτάνει στο τέλος και αναφέρει την κορυφογραμμή δίχως περεταίρω επαναλήψεις. Σε περίπτωση που το προσωρινό αρχείο έχει στοιχεία, οι επαναλήψεις συνεχίζονται με τα νέα εισακτέα δεδομένα να είναι όσες εγγραφές έχουν αποθηκευτεί στο προσωρινό αρχείο της προηγούμενης επανάληψης. Η διαδικασία θα συνεχιστεί μέχρι όλα τα σημεία να έχουν απορριφθεί ή να ανήκουν στην κορυφογραμμή, δηλαδή μέχρι το προσωρινό αρχείο και το αρχικό να αδειάσουν.

BNL

Input: set $\boldsymbol{X}$ of points

Output: set $\boldsymbol{S}$ of skyline points

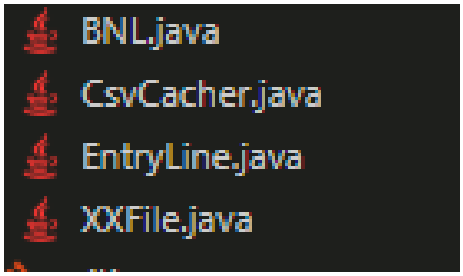
```
1.  S initialization as the empty set
2.  Input: = X;
3.  while(true)
4.      if (Input.isEmpty)
5.          ∀w_i ϵ Window ⇒ Move w_i to S
6.          if (Temp.isEmpty)
7.              break
8.          else
9.              Input: = Temp
10.             Temp.empty( )
11.     For x_i of Input
12.         isDominated: = false;
13.         if (Window.isEmpty)
14.             Move x_i to Window
15.             continue
16.         For x_j of Window
17.             if (x_j ¬ x_i)
18.                 isDominated: = true;
19.                 Remove x_j from Input
20.                 break;
21.             if (x_i ¬ x_j)
22.                 Remove x_j from Window
23.         end For
24.         if (isDominated == false)
25.             if (Window.isFull)
26.                 Move x_i to Temp
27.             else
28.                 Move x_i to Window
29.     end For
30. end while
```

Εικόνα 25 Ψευδοκώδικας του αλγορίθμου Block Nested Loop

Στον παραπάνω ψευδοκώδικα του BNL ως εισροή εισάγεται σύνολο n-διάστατων εγγραφών X. Εκροή θα αποτελεί το υποσύνολο $S \subseteq X$ που περιέχει τα στοιχεία της κορυφογραμμής. Ως $Input$ θα θεωρούμε αρχικά το σύνολο εισροών X, ενώ αργότερα το προσωρινό αρχείο αποθήκευσης εγγραφών $Temp$. Ο βρόγχος while των σειρών 3-30 αποτελεί τις διαφορετικές επαναλήψεις του αλγορίθμου. Ενώ υπάρχουν αρχεία στο $Input$ ή στο $Temp$, οι εντολές εντός του while θα συνεχίσουν να εκτελούνται. Οι σειρές 4-10 θα διαχειριστούν την περίπτωση στην οποία το $Input$

έχει αδειάσει. Οι γραμμές 4-10 διαχειρίζονται την περίπτωση που το $Input$ είναι άδειο. Αυτό μπορεί να συμβαίνει μόνο σε δυο στιγμές του αλγορίθμου. Πρώτη είναι όταν έχει ολοκληρωθεί η πρώτη επανάληψη του βρόγχου while και τα στοιχεία του X είναι πλέον είτε στο παράθυρο $Window$ είτε στο προσωρινό αρχείο $Temp$ είτε έχουν απορριφθεί. Δεύτερη στιγμή στην οποία το σύνολο Input είναι άδειο, είναι το τέλος κάθε μελλοντικής επανάληψης του βρόγχου while όπου όλα τα στοιχεία του έχουν καταλήξει στο παράθυρο, το προσωρινό αρχείο, ή έχουν απορριφθεί. Εν πάση περιπτώσει, η σειρά 5 μεταφέρει όλα τα σημεία του παραθύρου $Window$ στο σύνολο κορυφογραμμής. Ο βρόγχος if 6-10 αφορά το αν στο προσωρινό αρχείο $Temp$ έχουν αποθηκευτεί εγγραφές. Αν το $Temp$ δεν έχει εγγραφές, ο αλγόριθμος έχει φτάσει στο τέλος του και μέσω της σειράς 7 βγαίνει από τον βρόγχο while και απλά επιστρέφει την κορυφογραμμή στη σειρά 31. Αν στο $Temp$ υπάρχουν εγγραφές, η διαδικασία πρέπει να επαναληφθεί με νέο $Input$ τις εγγραφές εκείνες που κατά την τελευταία επανάληψη αποθηκεύτηκαν στο $Temp$. Οι σειρές 9-10 θέτουν ως νέο $Input\ το\ Temp$, και αδειάζοντας το $σύνολο\ Temp$ για επόμενες πιθανές εγγραφές που θα χρειαστεί να αποθηκεύσει. Οι δυο βρόγχοι For των σειρών 11 και 16 που ακολουθούν συγκρίνουν κάθε σημείο του εκάστοτε $Input$ με όλα τα σημεία του παραθύρου $Window$. Η Μπουλιανή μεταβλητή $isDominated$ θα διατηρήσει την απόφαση του αλγορίθμου αν το στοιχείο x_i του $Input$ προτείνεται για να εισαχθεί στο παράθυρο $Window$. Αυτό επιτυγχάνεται με τους δυο βρόχους if των σειρών 17 και 21. Σε περίπτωση που βρεθεί σημείο εντός του παραθύρου $Window$ κυρίαρχο επι του σημείου x_i του $Input$, η μεταβλητή μεταβάλλεται επιτόπου σε αληθής, και ο βρόχος For της σειράς 17 διακόπτεται, καθώς δεν υπάρχει λόγος να συνεχιστεί η σύγκριση με τα υπόλοιπα στοιχεία του παραθύρου, εφόσον το x_i δεν ανήκει στην κορυφογραμμή όντας κυριαρχημένο. Το στοιχείο απορρίπτεται και αφαιρείται από το $Input$. Στην αντίθετη περίπτωση ο βρόγχος if της σειράς 21 αφαιρεί από το παράθυρο όλα τα στοιχεία εκείνα που κυριαρχούνται από σημείο x_iτου $Input$. Στην περίπτωση που η Μπουλιανή μεταβλητή $isDominated$ έχει τιμή $false$, αυτό συνεπάγεται πως το σημείο x_i έχει προταθεί για εισαγωγή στο παράθυρο. Μένει μόνο να γίνει ο έλεγχος υπερχείλισης της σειράς 25. Αν η χωρητικότητα του παραθύρου δεν έχει εξαντληθεί, το στοιχείο εισάγεται κανονικά στο $Window$. Εναλλακτικά η σειρά 26 μεταφέρει το στοιχείο στο προσωρινό αρχείο αποθήκευσης $Temp$, εξασφαλίζοντας άλλη μια τουλάχιστον επανάληψη του βρόγχου $while$ της σειράς 3.

3.3 *Υλοποίηση σε Java του BNL*



Συνολικά, η υλοποίηση του BNL σε java θα γίνει με τέσσερα αρχεία .java. Το κύριο αρχείο, BNL θα έχει στόχο την αρχικοποίηση τιμών και την κύρια λειτουργία του αλγορίθμου. Το αρχείο CsvCacher θα είναι υπεύθυνο για την σωστή επεξεργασία του εξαγώγιμου αρχείου από το στάδιο της προ-επεξεργασίας του κεφαλαίου 2. Το αρχείο EntryLine αποτελεί μια κλάση για την αποθήκευση στην κύρια μνήμη αντικειμένου που αντιστοιχεί σε κάθε εγγραφή. Το αρχείο XXFile είναι το ίδιο με αυτό του κεφαλαίου 2, και απευθύνεται στην διαχείριση αποθηκευτικών πόρων του συστήματος.

Η διαδικασία που θα ακολουθηθεί για την υλοποίηση του BNL δεν διαφέρει πολύ από τον ψευδοκώδικα που παρουσιάστηκε στην §2.3.3. Πρέπει να σημειωθεί εδώ πως την κύρια διαφορά ανάμεσα στις 2 προσεγγίσεις αποτελεί η μορφή των εισροών. Όπως παρατηρείται από την § 2.3, αυτή παρέχει ως εκροές ένα αρχείο τύπου .csv που αποτελεί το εξαγόμενο της διαδικασίας προ-επεξεργασίας του αρχικού συνόλου δεδομένων των εγγραφών αγοραπωλησίας αυτοκινήτων. Για να γίνεται αποτελεσματικότερη χρήση πόρων του συστήματος, αφού ο αλγόριθμος διαβάσει τις εγγραφές αυτές, θα διαχωρίζονται με παρόμοια λογική regexp της §2.2 μέσω της CsvCacher.java και θα υλοποιούνται αντικείμενα της κλάσης EntryLine που αντιστοιχούν σε κάθε εγγραφή, με τα στοιχεία της εγγραφής αυτής. Αυτό θα επιτρέψει την ομαλότερη ροή του αλγορίθμου και την αποτελεσματικότερη σύγκριση των εγγραφών μεταξύ τους. Πιο συγκεκριμένα, η πρώτη επανάληψη του αλγορίθμου κατά την οποία, όπως ειπώθηκε στην §3.2.3 όλες οι αρχικές εγγραφές καταλήγουν στο προσωρινό αρχείο ή στο παράθυρο στοιχείων ή απορρίπτονται, θα αφορά την επεξεργασία του αρχείου .csv που εξάγεται από την διαδικασία προ-επεξεργασίας του κεφαλαίου 2. Μετά από αυτήν θα ακολουθηθεί συγκεκριμένη πορεία για την εφαρμογή της λογικής του BNL. Ανά πάσα στιγμή από την δεύτερη επανάληψη και έπειτα του αλγορίθμου, θα υφίστανται 2 αρχεία τύπου .txt τα οποία θα δρουν ως εισροή του BNL και ως προσωρινό αρχείο αποθήκευσης εγγραφών αντίστοιχα. Με αυτόν τον τρόπο, σε κάθε επόμενη επανάληψη, το νέο αρχείο εισροής θα αποτελεί το παλιό αρχείο

εκροής που αντιστοιχεί στο προσωρινό αρχείο αποθήκευσης εγγραφών. Το νέο προσωρινό αρχείο αποθήκευσης εγγραφών θα αποτελεί το προηγούμενο αρχείο εισροών, αφού βέβαια αρχικοποιηθεί καταλλήλως ως κενό. Η διαδικασία εναλλαγής αυτή θα συνεχιστεί μέχρις ότου ο αλγόριθμος περατωθεί και επιστραφεί η κορυφογραμμή.

3.3.1 Η κύρια κλάση BNL

```
1   import java.util.ArrayList;
2   import java.util.Iterator;
3   import java.io.File;
4   import java.io.FileInputStream;
5   import java.io.FileNotFoundException;
6   import java.io.FileOutputStream;
7   import java.io.IOException;
8   import java.io.ObjectInputStream;
9   import java.io.ObjectOutputStream;
10
```

Οι εγγενείς βιβλιοθήκες-εργαλεία της java που θα χρησιμοποιηθούν στην κύρια Κλάση BNL.java στόχο έχουν την βελτιστοποίηση της διαδικασίας επεξεργασίας των δεδομένων ούτως ώστε να μην γίνεται άσκοπη δαπάνη πόρων για να επιτευχθεί το τελικό αποτέλεσμα. Παρακάτω θα αναφερθούν όσες δεν αναλύθηκαν στο κεφάλαιο 2. Το εργαλείο Iterator αφορά όλες της ομάδες αντικειμένων της java, όπως τα ArrayLists. Επιτρέπει την ανάγνωση και διαγραφή αντικειμένων την ίδια στιγμή καθώς διατρέχει την εν λόγω ομάδα αντικειμένων. Τα υπόλοιπα στοιχεία της εγγενούς βιβλιοθήκης αφορούν την διαχείριση εισροών και εκροών της κλάσης και είναι μέρη του πακέτου java.io. Πιο συγκεκριμένα, τα FileInputStream και FileOutputStream επιτρέπουν αντίστοιχα την ανάγνωση και γραφή σειρών bytes από και προς αρχείο που ορίζεται μέσω του java.io.File. Τα ObjectInputStream και ObjectOutputStream χρησιμοποιούνται αντίστοιχα για να αποσειριοποιήσουν και να σειριοποιήσουν αντικείμενα από και προς αρχείο. Το FileNotFoundException διαχειρίζεται την περίπτωση ο προορισμός αρχείου δεν υφίσταται. Τέλος το IOException διαχειρίζεται σφάλματα που προκύπτουν κατά την επεξεργασία εισροών και εκροών.

```
11  public class BNL {
        Run | Debug
12      public static void main(String[] args) {···
119
120     public static Boolean isDominatedBy(EntryLine A,EntryLine B){···
123
124     public static ObjectOutputStream newOutputFrom(String filePath){···
142
143     public static ObjectInputStream newInputFrom(String filePath){···
160
161 }
```

Σε ό,τι αφορά την κύρια κλάση BNL αυτή αποτελείται από 4 μεθόδους. Η main θα οδηγήσει το σύνολο του αλγορίθμου και θα επιστρέψει το σύνολο στοιχείων κορυφογραμμής καθώς και τον χρόνο που διένυσε μέχρι τις εκροές σε ms και το σύνολο των πόρων μνήμης που χρησιμοποιήθηκε σε GB. Η στατική μέθοδος isDominatedBy είναι η αντίστοιχη συνάρτηση κατάταξης φ που χρησιμοποιήθηκε στην §3.2.1. Στόχο έχει να λαμβάνει υπόψιν συγκεκριμένες διαστάσεις των εγγραφών και να καθορίζει αν μια εγγραφή κυριαρχεί επι κάποιας άλλης. Οι επόμενες δυο μέθοδοι, newOutputFrom και newInputFrom θα χρησιμοποιηθούν για τις διαδικασίες ανάγνωσης και εγγραφής στοιχείων στο προσωρινό αποθηκευτικό αρχείο.

```
12  public static void main(String[] args) {
13      long start = System.currentTimeMillis();
14      String filename="CSVfilepath", newCache="Temptxt1 path", TempPath="Temptxt2 path";
15      CsvCacher cache=new CsvCacher(filename);
16      int TempFileEntries=0,CacheEntriesCount=0,maxException=1500000;
17      Boolean mainFinished=false,switcher=true,isDominated=false, writeTempOnly=false;
18      ArrayList<Integer> resultSkyLine=new ArrayList<>();
19      ArrayList<EntryLine> window= new ArrayList<EntryLine>();
20      EntryLine newEntry;
21      try{
22          ObjectInputStream input=null;
23          ObjectOutputStream output = newOutputFrom(TempPath);
24          while( true ){···
100     }
101     catch (FileNotFoundException e) {
102         System.out.println("Does not exist");
103     }
104     catch (IOException e) {
105         System.out.println("I/O Issue");
106     }
107     catch (ClassNotFoundException e) {
108         e.printStackTrace();
109     }
110
111     System.out.println(resultSkyLine);
112
113     long elapsedTimeMillis = System.currentTimeMillis()-start;
114     System.out.printf("\nElapsed Time is %d ms,  Space used is %.3f GB",elapsedTimeMillis,
115     (Runtime.getRuntime().totalMemory()-Runtime.getRuntime().freeMemory())/(1024.0 * 1024.0* 1024.0));
116 }
```

Όπως ειπώθηκε ήδη, η μέθοδος main είναι η βάση και ο οδηγός του αλγορίθμου BNL. Στη σειρά 13 αποθηκεύεται σε μεταβλητή με όνομα start ένας αριθμός που αντιστοιχεί στον χρόνο σε ms που έχει παρέλθει από την 1η Ιανουαρίου 1970, μέσω της μεθόδου του συστήματος currentTimeMillis. Η μεταβλητή start θα φανεί χρήσιμη για να υπολογιστεί ο εκ των πραγμάτων χρόνος σε ms που δαπανήθηκε κατά την ροή του αλγορίθμου BNL. Όπως φαίνεται στη σειρά 115, μια δεύτερη κλήση της ίδιας μεθόδου θα επιστρέψει τον ακριβή χρόνο που αντίστοιχα έχει παρέλθει από την 1η Ιανουαρίου 1970 μέχρι το πέρας του αλγορίθμου. Μια απλή αφαίρεση των δυο θα αποδώσει τον συνολικό χρόνο που διένυσε το πρόγραμμα σε ms, κάτι που γίνεται στην σειρά 114, στην οποία

επιστρέφεται ο χρόνος αυτός, καθώς και η συνολική δεσμευμένη μνήμη με παρόμοια λογική. Η σειρά 14 αποθηκεύει αλφαριθμητικά που αντιστοιχούν στα μονοπάτια των τριών αρχείων που θα χρησιμοποιηθούν για την ανάγνωση και εγγραφή στοιχείων. Το filename θα αποθηκεύει το μονοπάτι προς το αρχείο επέκτασης .csv που είναι αυτό της εκροής του σταδίου προ-επεξεργασίας του κεφαλαίου 2. Τα newCache και TempPath αποτελούν τα μονοπάτια για τα δυο εναλλασσόμενα αρχεία εισροών και εκροών που περιεγράφηκαν παραπάνω μορφής .txt . Στην επόμενη σειρά αρχικοποιείται αντικείμενο της κλάσης CsvCacher, υπεύθηνης για την επεξεργασία αποκλειστικά του αρχείου filename επέκτασης .csv. Στην επόμενη σειρά αρχικοποιούνται 2 μετρητές και 1 όριο. Πιο συγκεκριμένα η μεταβλητή TempFileEntries θα περιέχει το σύνολο των εγγραφών που έχουν αποθηκευτεί αν πάσα στιγμή στο προσωρινό αρχείο ενώ η CacheEntriesCount το σύνολο των εγγραφών του αρχείου εισροής από το οποίο διαβάζονται στην εκάστοτε επανάληψη τα στοιχεία προς επεξεργασία. Πρέπει ωστόσο να σημειωθεί πως η μεταβλητή CacheEntriesCount δεν αφορά το αρχείο επέκτασης .csv αλλά όλα τα αρχεία εισροής μετά από αυτό. Το όριο maxException θα βοηθήσει στην προσομοίωση ανεπάρκειας αποθηκευτικών πόρων του παραθύρου και θα το φράξει σε ένα συγκεκριμένο πλήθος εγγραφών. Η σειρά 17 αποθηκεύει 4 Μπουλιανές τιμές που θα βοηθήσουν στην ομαλότερη λειτουργία του αλγορίθμου καθώς τα στάδια του εναλλάσσονται μεταξύ τους. Η μεταβλητή mainFinished θα καθορίσει την περάτωση της πρώτης επανάληψης του αλγορίθμου κατά την οποία όλες οι εγγραφές του αρχείου προ-επεξεργασίας επέκτασης .csv έχουν επεξεργαστεί και καταλήξει είτε στο παράθυρο είτε στο προσωρινό αποθηκευτικό αρχείο ή έχουν απορριφθεί. Με την εναλλαγή της τιμής της σε αληθής, θα μεταβληθεί και η λειτουργία του αλγορίθμου στο να διαβάζει και να εγγράφει αντίστοιχα στα εναλασσόμενα μεταξύ τους αρχεία μορφής .txt newCache και TempPath. Η μεταβλητή switcher θα είναι υπεύθυνη για την εν λόγω εναλλαγή του αρχείου εκροών και εισροών σε κάθε επανάληψη του αλγορίθμου. Η μεταβλητή isDominated θα αποθηκεύει το αποτέλεσμα της σύγκρισης κυριαρχίας εγγραφής έναντι μιας άλλης. Τέλος η μεταβλητή writeTempOnly υποδεικνύει στο πρόγραμμα να αποθηκεύει, από τη στιγμή που εξαντλήθηκαν οι αποθηκευτικοί πόροι, όλες τις εγγραφές στο προσωρινό αποθηκευτικό αρχείο μόνο.

Στη σειρά 18 αρχικοποιείται σειρά ακεραίων με το όνομα resultSkyLine όπου θα αποθηκεύονται τα id όλων των εγγραφών εκείνων που ανήκουν στην κορυφογραμμή. Προφανώς, με το πέρας του αλγορίθμου η μεταβλητή αυτή θα περιέχει το ζητούμενο σύνολο στοιχείων κορυφογραμμής που επιστρέφει το πρόγραμμα. Στη γραμμή 19 αρχικοποιείται σειρά αντικειμένων window της κλάσης EntryLine που όπως ειπώθηκε είναι αντικείμενα που αντιστοιχούν στις εγγραφές του αρχείου .csv

που προέκυψε από την προ-επεξεργασία του κεφαλαίου 2. Η μεταβλητή window θα αποτελεί το αντίστοιχο παράθυρο του BNL στο οποίο θα αποθηκεύονται εγγραφές μη συγκρίσιμες ανα δυο μεταξύ τους, οι οποίες σε κάθε τέλος επανάληψης θα αποτελούν σημεία κορυφογραμμής, όπως αποδείχθηκε από την Πρόταση 2 της § 3.2.1. Τέλος, η μεταβλητή newEntry θα αποτελεί αντικείμενο της κλάσης EntryLine που αντιστοιχεί στην τρέχουσα εγγραφή που έχει αναγνωστεί από το αρχείο εκροών και πρόκειται να τοποθετηθεί στο παράθυρο window, ή στο προσωρινό αρχείο ή να απορριφθεί.

Ο κώδικας των σειρών 21-109 που ακολουθεί διαχειρίζεται το σύνολο της λογικής του αλγορίθμου BNL. Αρχικά, γίνεται χρήση του βρόγχου try- catch εφόσον θα ακολουθήσει κώδικας που αλληλοεπιδρά με τα εξωτερικά αρχεία εισροών και εκροών. Τα 3 catch στις σειρές 101,104 και 107 διαχειρίζονται τις περιπτώσεις στις οποίες προκύπτουν σφάλματα σχετιζόμενα με ελλιπή αρχείο, προβλήματα εισροών ή εκροών και ελλιπή κλάση αντίστοιχα. Οι σειρές 22 και 23 αρχικοποιούν αντικείμενα των κλάσεων ObjectInputStream και ObjectOutputStream, input και output αντίστοιχα που θα αποτελούν τα αρχεία εισροών και εκροών κάθε επανάληψης. Το αρχικό αρχείο εισροής αρχικοποιείται με τιμή null, εφόσον έχει ανατεθεί η επεξεργασία του στην κλάση CsvCacher, δεδομένου ότι είναι επέκτασης .csv. Το αρχείο output κάνει κλήση της μεθόδου newOutputFrom, η οποία αρχικοποιεί κατάλληλα αρχείο επέκτασης .txt που θα επεξηγηθεί αναλυτικότερα παρακάτω.

```
        while( true ){
            if(mainFinished){
                for (EntryLine entryLine : window) resultSkyLine.add(entryLine.id);
                if(TempFileEntries==0){
                    break;
                }
                else{
                    writeTempOnly=false;
                    CacheEntriesCount=TempFileEntries;
                    TempFileEntries=0;
                    input= newInputFrom(switcher?TempPath:newCache);
                    output = newOutputFrom(switcher?newCache:TempPath);
                    window=new ArrayList<EntryLine>();
                }
            }
            else cache.startProcess();

            for (int i = 0; true; i++) {
                isDominated=false;
                if(i>=CacheEntriesCount&&mainFinished)break;
                if(mainFinished)newEntry=(EntryLine) input.readObject();
                else newEntry=cache.nextEntry();

                if(newEntry==null){
                    cache.endProcess();
                    output.flush();
                    output.close();
                    mainFinished=true;
                    break;
                }

                if(window.size()==0){
                    if(writeTempOnly==false){
                        window.add(newEntry);
                        continue;
                    }
                }

                for( Iterator<EntryLine> iterator=window.iterator(); iterator.hasNext(); ){
                    EntryLine eleEntry=iterator.next();
                    if( isDominatedBy(newEntry, eleEntry)){
                        isDominated=true;
                        break;
                    }
                    if( isDominatedBy(eleEntry, newEntry))iterator.remove();
                }

                if(isDominated==false){
                    if(writeTempOnly==true){
                        output.writeObject(newEntry);
                        TempFileEntries++;
                    }
                    else {
                        try{
                            if( window.size()>maxException){
                                throw new OutOfMemoryError("LOW MEMORY");
                            }
                            else{
                                window.add(newEntry);
                            }
                        }
                        catch( OutOfMemoryError exception){
                            output.writeObject(newEntry);
                            TempFileEntries++;
                            writeTempOnly=true;
                        }
                    }
                }
            }
            if(CacheEntriesCount!=0){
                switcher=!switcher;
                input.close();
                output.flush();
                output.close();
            }
        }
```

Ο επαναληπτικός βρόγχος while των σειρών 24-99 διαχειρίζεται το σύνολο των επαναλήψεων του αλγορίθμου BNL. Ο βρόγχος if των σειρών 25-39 εξαρτάται από το αν έχει τελειώσει η επεξεργασία του αρχικού αρχείου .csv. Αν έχει όντως τελειώσει, η σειρά 26 μεταφέρει όλα τα id των αποθηκευμένων στο παράθυρο window εγγραφών στο τελικό αποτέλεσμα κορυφογραμμής. Σε περίπτωση που δεν υπάρχουν περεταίρω αρχεία εγγεγραμμένα στο προσωρινό αρχείο, ο αλγόριθμος έχει φτάσει στο πέρας του, οπότε ο βρόγχος if των σειρών 27-29 τερματίζει τον επαναληπτικό βρόγχο while. Εναλλακτικά οι σειρές 31-36 αποτελούν το μεταβατικό στάδιο κατά το οποίο συμβαίνει η κατάσταση του νέου αρχείου εισροών ως το προηγούμενο προσωρινό αποθηκευτικό αρχείο. Πιο συγκεκριμένα, η σειρά 31 θέτει την τιμή της λογικής μεταβλητής writeTempOnly σε ψευδή καθώς κατά την διάρκεια της νέας επανάληψης υπάρχει η δυνατότητα εγγραφής στοιχείων στο παράθυρο. Η επόμενη σειρά θέτει το νέο πλήθος εγγραφών των εκροών ως το παλιό πλήθος εγγραφών που είχαν αποθηκευτεί στο προσωρινό αρχείο σύμφωνα με την εναλλαγή που πρόκειται να συμβεί. Η σειρά 33 θέτει τον μετρητή του πλήθους εγγραφών του νέου προσωρινού αποθηκευτικού αρχείου σε 0. Οι σειρές 34 και 35 αρχικοποιούν, μέσω των μεθόδων newInputFrom και newOutputFrom που θα αναλυθούν παρακάτω, τα νέα για την επανάληψη input και output που είναι αντίστοιχα το αρχείο εισροών και το προσωρινό αποθηκευτικό αρχείο. Πρέπει να σημειωθεί πως αυτό γίνεται βάσει της λογικής Ternary Operator της java, μια συντομογραφία ενός βρόγχου if. Αναλυτικότερα, αν η τιμή της Μπουλιανής μεταβλητής switcher είναι αληθής, το ζευγάρι (input,outpout) είναι τα αρχεία των προορισμών των (TempPath,newCache), εναλλακτικά είναι τα αρχεία των προορισμών (newCache,TempPath) αντίστοιχα. Στην περίπτωση που το οι εγγραφές του αρχικού αρχείου επέκτασης .csv δεν έχουν ακόμα επεξεργαστεί, δηλαδή κατά την πρώτη επανάληψη του αλγορίθμου, καλείται η μέθοδος της κλάσης CsvCacher startProcess η οποία ουσιαστικά ξεκινάει να διαβάζει τις εγγραφές και θα αναλυθεί εκτενέστερα σε επόμενη παράγραφο. Η λειτουργία των δυο επαναληπτικών βρόγχων for των σειρών 41,62 που ακολουθούν είναι πανομοιότυπη με αυτήν του ψευδοκώδικα της §3.2.3. Συνοπτικά, στόχο έχει την σύγκριση κάθε στοιχείου του εκάστοτε αρχείου εισροών με όλα τα στοιχεία του παραθύρου, και την τοποθέτηση του είτε στο προσωρινό αρχείο, είτε στο παράθυρο ή τελικά την απόρριψη του ως πιθανό σημείο κορυφογραμμής. Η σειρά 42 αρχικοποιεί την Μπουλιανή μεταβλητή isDominated ως ψευδή, υποθέτοντας πως το στοιχείο που πρόκειται να ληφθεί από το αρχείο εισροών δεν κυριαρχείται από κάποιο στοιχείο του παραθύρου $window$. Η σειρά 43 τερματίζει τον βρόγχο for εφόσον δεν υπάρχουν άλλα στοιχεία του αρχείου εισροών προς επεξεργασία και συγχρόνως το .csv αρχικό αρχείο εγγραφών έχει ήδη επεξεργαστεί, δηλαδή βρισκόμαστε σε επανάληψη πέραν της πρώτης.

Οι επόμενες 2 σειρές αποθηκεύουν στη μεταβλητή newEntry την εγγραφή που θα απασχολήσει την συγκεκριμένη επανάληψη του βρόγχου for, με σκοπό να τη συγκρίνει με όλα τα στοιχεία του παραθύρου $window$. Σε περίπτωση που η επεξεργασία του αρχικού αρχείου εισροών επέκτασης .csv έχει περατωθεί, η εγγραφή λαμβάνεται από το αρχείο input, ειδάλλως αυτή λαμβάνεται από το αρχείο .csv. Σε περίπτωση που το αντικείμενο που αποθηκεύτηκε στην μεταβλητή newEntry είναι το κενό αντικείμενο null, αυτό μπορεί να σημαίνει μόνο πως ενώ επιχειρήθηκε η ανάγνωση από το αρχικό αρχείο .csv, δεν υπήρχαν περεταίρω εγγραφές. Κατ' επέκταση, οι σειρές 47-53 θα εκτελεστούν μόνο μια φορά, στο τέλος της πρώτης επανάληψης του εξωτερικού επαναληπτικού βρόγχου $while$. Πιο συγκεκριμένα, η σειρά 48 καλεί την μέθοδο της κλάσης CsvCacher endProcess η οποία κλείνει το αρχείο επέκτασης .csv. Οι σειρές 49 και 50 εγγράφουν ολοκληρωτικά στο αρχείο output όσες εγγραφές εκκρεμούν, και το κλείνουν. Τέλος η τιμή της Μπουλιανής μεταβλητής mainFinished γίνεται αληθής, σηματοδοτώντας την λήξη της πρώτης επανάληψης του εξωτερικού βρόγχου while. Οι σειρές 55-60 είναι πανομοιότυπες με τις σειρές 13-15 του ψευδοκώδικα στην §3.2.3. Σε περίπτωση που το παράθυρο δεν περιέχει στοιχεία και υπάρχει χώρος στο παράθυρο, το προς εξέταση στοιχείο newEntry εισάγεται κατευθείαν στο παράθυρο και περνάμε στην επόμενη επανάληψη του επαναληπτικού βρόγχου for της σειράς 52. Οι σειρές 62-69 διατρέχουν μέσω της χρήσης iterator όλα τα αντικείμενα EntryLine του παραθύρου $window$ μέχρι να εξαντληθούν. Κάθε τέτοιο αντικείμενο το αναθέτει στη μεταβλητή eleEntry, την οποία συγκρίνει με την εγγραφή newEntry ως προς την κυριαρχία μέσω της μεθόδου isDominatedBy. Σε περίπτωση που η newEntry κυριαρχείται από την eleEntry, η Μπουλιανή μεταβλητή isDominated γίνεται αληθής και ο επαναληπτικός βρόγχος for τερματίζεται εφόσον η εγγραφή newEntry δεν αποτελεί υποψήφιο σημείο της κορυφογραμμής. Εναλλακτικά, αν η εγγραφή newEntry κυριαρχεί επι μιας ή περισσοτέρων εγγραφών του παραθύρου $window$, αυτές αφαιρούνται επι τόπου από το παράθυρο.

Οι σειρές 71-91 διαχειρίζονται την περίπτωση που η τιμή της Μπουλιανής μεταβλητής isDominated είναι ψευδής, δηλαδή η εγγραφή newEntry έχει προταθεί για να εισαχθεί μέσα στο παράθυρο $window$ καθώς δεν κυριαρχείται από καμία εγγραφή εντός του παραθύρου. Κατόπιν ελέγχεται εάν ο αλγόριθμος βρίσκεται στην κατάσταση όπου οι εγγραφές πρέπει να μεταφέρονται μόνο στο προσωρινό αποθηκευτικό αρχείο. Σε αυτήν την περίπτωση εγγράφεται στο output το αντικείμενο της εγγραφής newEntry και ο μετρητής εγγεγραμμένων στο προσωρινό αυτό αρχείο αντικειμένων αυξάνεται κατά μονάδα. Σε διαφορετική περίπτωση επιχειρείται ο έλεγχος πλασματικής υπερχείλισης συγκρίνοντας το πλήθος εγγραφών εντός του παραθύρου με το αρχικό όριο maxException. Αν η χωρητικότητα του παραθύρου $window$ έχει εξαντληθεί, προωθείται

αντίστοιχο σφάλμα ανεπάρκειας μνήμης, αλλιώς το στοιχείο newEntry εισάγεται κανονικά στο παράθυρο. Την περίπτωση του σφάλματος ανεπάρκειας μνήμης διαχειρίζεται ο βρόγχος catch 85-89. Πιο συγκεκριμένα η εγγραφή newEntry περνάει πλέον στο προσωρινό αρχείο, αυξάνοντας τον αντίστοιχο μετρητή και διαμορφώνοντας την κατάσταση του αλγορίθμου ούτως ώστε να αποθηκεύει πλέον όλα τα σημεία που προορίζονται για το παράθυρο στο προσωρινό αρχείο μέσω της Μπουλιανής μεταβλητής writeTempOnly. Ο βρόγχος if των σειρών 93-98 εκτελείται μόνο μετά την δεύτερη επανάληψη του εξωτερικού επαναληπτικού βρόγχου while και στόχο έχει να αλλάξει την λογική τιμή της Μπουλιανής μεταβλητής switcher ούτως ώστε στην επόμενη επανάληψη το προηγούμενο input να είναι το νέο output, και το νέο output να είναι το νέο input. Κατ' επέκταση τελειοποιούνται όποιες εγγραφές προορίζονταν στο προσωρινό αρχείο output και κλείνουν και τα 2 αρχεία.

```
118     public static Boolean isDominatedBy(EntryLine A,EntryLine B){
119         return B.price<=A.price && B.odometer<=A.odometer && B.year>=A.year && (B.price<A.price||B.odometer<A.odometer||B.year>A.year);
120     }
```

Η μέθοδος isDominatedBy αποτελεί την συνάρτηση κατάταξης $\varphi(x)$ που περιεγράφηκε και χρησιμοποιήθηκε εκτενώς στην §3.2.1. Πιο συγκεκριμένα ανάλογα με τα κριτήρια της κορυφογραμμής, δηλαδή τις ενδιαφέρουσες διαστάσεις των εγγραφών, η συνάρτηση αυτή συγκρίνει δυο εγγραφές A και B που υλοποιούνται μέσω αντικειμένων της κλάσης EntryLine. Παραδείγματος χάριν, η παραπάνω μορφή της συνάρτησης σχηματίζει κορυφογραμμή ελάχιστης τιμής, ελάχιστων χιλιομέτρων και μέγιστου έτους κυκλοφορίας που αφορά το σύνολο δεδομένων αγοραπωλησιών αυτοκινήτων του κεφαλαίου 2.

```
124     public static ObjectOutputStream newOutputFrom(String filePath){
125         try{
126             File out=new File(filePath);
127             out.delete();
128             FileOutputStream f = new FileOutputStream(out);
129             return new ObjectOutputStream(f);
130         }
131         catch (FileNotFoundException e) {
132             System.out.println("DOESNT EXIST");
133         }
134         catch (IOException e) {
135             System.out.println("CANT BE WRITTEN");
136             e.getCause();
137             e.printStackTrace();
138         }
139         return null;
140 
141     }
```

Η μέθοδος newOutputFrom λαμβάνει ως είσοδο το μονοπάτι κάποιου αρχείου στο σύστημα και επιστρέφει ένα αντικείμενο τύπου ObjectOuputSteram το οποίο χρησιμοποιείται για να σειριοποιήσει κάποιο αντικείμενο σε σειρά bytes, που μέσω ενός FileOutputStream θα εγγραφεί στο αρχείο αυτό. Πιο συγκεκριμένα στις σειρές 126 και 127 αρχικοποιείται μέσω της κλάσης File αρχείο που αντιστοιχεί στο δοθέν μονοπάτι και διαγράφεται το εσωτερικό του, αν αυτό υπάρχει. Κατόπιν αρχικοποιείται αντικείμενο της κλάσης FileOutputStream που εγγράφει σειρά bytes σε αρχείο και επιστρέφεται το εν λόγω αντικείμενο της κλάσης ObjectOutputStream που κάνει χρήση του αντικειμένου αυτού.

```
143     public static ObjectInputStream newInputFrom(String filePath){
144         try{
145             File newinputFile=new File(filePath);
146             FileInputStream fi = new FileInputStream(newinputFile);
147             return new ObjectInputStream(fi);
148         }
149         catch (FileNotFoundException e) {
150             System.out.println("DOESNT EXIST");
151         }
152         catch (IOException e) {
153             System.out.println("CANT BE WRITTEN");
154             e.getCause();
155             e.printStackTrace();
156         }
157         return null;
158 
159     }
```

Η μέθοδος newInputFrom είναι αντίστοιχη με την newOutputFrom σε ό,τι αφορά τις εισροές. Ειδικότερα, δοθέντος μονοπατιού συστήματος ενός αρχείου επέκτασης .txt, δημιουργείται αντικείμενο της κλάσης File newInputFile και αναγνώστης fi σειρών byte σε αυτό στις σειρές 145 και 146 αντίστοιχα. Τέλος επιστρέφεται αντικείμενο της κλάσης ObjectInputStream που αποσειριοποιεί τις σειρές bytes αυτές σε αντικείμενα. Οι σειρές 149-156 διαχειρίζονται αντίστοιχα περιπτώσεις σφάλματος ελλιπούς αρχείου ή σφάλματος σχετιζόμενου με εισροές και εκροές.

3.3.2 Η κλάση CsvCacher

Η κλάση CsvCacher όπως προ-είπαμε έχει τον ρόλο της επεξεργασίας του αρχικού συνόλου δεδομένων που δίνονται ως εισροές με τη μορφή αρχείου επέκτασης .csv, δηλαδή του εξαγώγιμου

αρχείου της διαδικασίας προ-επεξεργασίας που λαμβάνει χώρα στο κεφάλαιο 2. Αναλυτικότερα, είναι υπεύθυνη για την ανάγνωση σειρών του αρχείου .csv, τον διαχωρισμό τους μέσω λογικής regexP παρόμοιας με αυτή του κεφαλαίου 2 και δημιουργίας αντίστοιχου αντικειμένου της κλάσης EntryLine για κάθε εγγραφή.

```
1  import java.util.regex.Matcher;
2  import java.util.regex.Pattern;
```

Η λειτουργία των εργαλείων regex.Matcher και regex.Pattern είναι πανομοιότυπη με την §2.3.2. Σκοπό έχει τον σωστό διαχωρισμό σειράς χαρακτήρων δοθέντος συγκεκριμένου μοτίβου.

```
4   public class CsvCacher{
5       private String inputFile;
6       public XXFile file_dataset = new XXFile();
7       private Integer[] defaultDominated={0 ,0,Integer.MAX_VALUE,0,Integer.MAX_VALUE};
8
9   >   public CsvCacher(String inputFile){ ···
12
13  >   public void startProcess(){ ···
17
18  >   public  EntryLine nextEntry(){ ···
29
30  >   public void endProcess(){ ···
33
34  >   public String[] RegexSeparation(String temp){ ···
45
46  >   public EntryLine createLine(String[] separated){ ···
77
78  >   public Boolean AbsoluteLine(String[] separated){ ···
83
84      private Integer[] constraints={0,0,500,1900,1000};
85  >   public Boolean ConstrainedLine(String[] separated){ ···
94
95  }
```

```
9       public CsvCacher(String inputFile){
10          this.inputFile=inputFile;
11      }
```

Κατά την αρχικοποίηση αντικειμένου της κλάσης CsvCacher, δίνεται ως εισροή αλφαριθμητικό inputFile που αποτελεί το μονοπάτι συστήματος του αρχείου .csv που προκύπτει από την προ-επεξεργασία του αρχικού συνόλου εγγραφών του κεφαλαίου 2. Επίσης αρχικοποιούνται τρείς εγγενείς για κάθε αντικείμενο μεταβλητές ονόματι inputFile, file_dataset και defaultDominated. Στο inputFile θα αποθηκευτεί η εισροή αλφαριθμητικού inptuFile μέσω της κατασκευαστικής συνάρτησης CsvCacher της κλάσης. Η υλοποίηση αντικειμένου της κλάσης XXFile λειτουργεί πανομοιότυπα με αυτήν του κεφαλαίου 2, όπου έχει ήδη αναλυθεί εκτενώς. Τέλος η σειρά ακεραίων

defaultDominated θα χρησιμοποιηθεί για ελλιπή πεδία εγγραφών τα οποία θα θεωρούνται απευθείας κατακτημένα στην διάσταση που λείπει.

```
14      public void startProcess(){
15          file_dataset.openRFile(inputFile);
16          file_dataset.readLine();
17      }
```

Η μέθοδος startProcess καλεί τις μεθόδους openRFile και readLine της κλάσης XXFile. Η λειτουργία τους έχει αναφερθεί ήδη στην §2.3.3. Πρέπει να σημειωθεί πως στη σειρά 16 γίνεται κλήση μιας φοράς της μεθόδου readLine, αυτή είναι η ανάγνωση της πρώτης σειράς του αρχείου .csv που αποτελεί τους τίτλους της κάθε διάστασης. Συμπερασματικά η μέθοδος ουσιαστικά απλά απορρίπτει την πρώτη σειρά του αρχείου καθώς δεν περιέχει ουσιαστική εγγραφή.

```
19      public  EntryLine nextEntry(){
20          String line=file_dataset.readLine();
21          while(true){
22              if(line==null)break;
23              String[] separated=RegexSeparation(line);
24              EntryLine myLine=createLine(separated);
25              if(myLine!=null)return myLine;
26              line=file_dataset.readLine();
27          }
28          return null;
29      }
```

Η μέθοδος nextEntry επιστρέφει αντικείμενο της κλάσης EntryLine που είναι το αντίστοιχο σε μια εγγραφή αντικείμενο με όλες τις απαραίτητες πληροφορίες των πεδίων της. Πιο συγκεκριμένα, αρχικοποιείται αλφαριθμητικό ονόματι line στο οποίο αποθηκεύεται η σειρά του αρχείου επέκτασης .csv που αντιστοιχεί σε εγγραφή. Ο επαναληπτικός βρόγχος while των σειρών 21-27 σκοπό έχει να επιστρέψει αντικείμενο της κλάσης EntryLine που αντιστοιχεί στην πρώτη έγκυρη εγγραφή ή null σε περίπτωση που δεν υπάρχει εναπομείνουσα έγκυρη εγγραφή στο αρχείο. Αναλυτικότερα, αν η πρώτη εγγραφή που διαβάζεται από την κλήση της μεθόδου readLine της σειράς 20 επιστρέψει null, αυτό σημαίνει πως δεν υπάρχουν περεταίρω εγγραφές στο αρχείο .csv, οπότε η σειρά 22 σπάει τον βρόγχο while και επιστρέφει null. Εναλλακτικά αρχικοποιείται σειρά αλφαριθμητικών separated στην οποία αποθηκεύεται η εκροή της μεθόδου RegexSeparation που χωρίζει δοθέν αλφαριθμητικό, δηλαδή το κείμενο εγγραφής, στα αντίστοιχα πεδία διαστάσεων του. Η σειρά 24 αρχικοποιεί αντικείμενο της κλάσης EntryLine και αποθηκεύει σε αυτό την εκροή της μεθόδου createLine δοθείσης της παραπάνω σειράς αλφαριθμητικών separated. Αν το αντικείμενο εγγραφής myLine δεν είναι null, επιστρέφεται κατευθείαν στη σειρα 25. Αν από την μέθοδο createLine επιστραφεί η τιμή null, αυτό σημαίνει πως υπήρξε κάποιο πρόβλημα με την δημιουργία του εν λόγω αντικειμένου

εγγραφής, περίπτωση η οποία θα αναλυθεί παρακάτω, οπότε η σειρά 26 διαβάζει την επόμενη γραμμή και επαναλαμβάνει την διαδικασία χωρισμού και ελέγχου εγκυρότητας.

```
31      public void endProcess(){
32          file_dataset.closeRFile();
33      }
34
```

Η μέθοδος endProcess κάνει κλήση της μεθόδου closeRfile της κλάσης XXFile, η οποία κλείνει το αρχείο επέκτασης csv.

```
35      public String[] RegexSeparation(String temp){
36          String regExp ="(?:^|,)(\"(?:(?:\"\")*[^\"]*)*\"|[^\",]*)" ;
37          Pattern p = Pattern.compile(regExp);
38          String[] tokens= new String[18];
39          int counter=0;
40          Matcher matcher = p.matcher(temp);
41          while (matcher.find()) {
42              tokens[counter++]=matcher.group(1);
43          }
44          return tokens;
45      }
```

Η μέθοδος RegexSeparation έχει πανομοιότυπη λειτουργία με αυτήν της §2.3.2 και σκοπό έχει, μέσω λογικής διαχωρισμού κανονικών παραστάσεων να διαχωρίσει δοθέν αλφαριθμητικό βάσει χαρακτήρα κόμματος και να επιστρέψει σειρά 18 πεδίων με τα αποτελέσματα του διαχωρισμού αυτού. Πρέπει να σημειωθεί πως το πλήθος των πεδίων που αναμένονται από τον διαχωρισμό του αρχείου .csv είναι 18 καθώς η διαδικασία προ-επεξεργασίας έχει ως αποτέλεσμα 18 εναπομείναντα πεδία εγγραφής. Για αυτόν τον λόγο στη σειρά 38 αρχικοποιείται σειρά 18 αλφαριθμητικών tokens, στην οποία, με τρόπο παρόμοιο της παραγράφου §2.3.2 αποθηκεύονται οι τιμές των διαχωρισμένων αλφαριθμητικών.

```
47      public EntryLine createLine(String[] separated){
48          if(separated.length!=18)System.out.println("FALSE SEPARATION");
49          int id=Integer.parseInt(separated[0]);
50          String city=separated[1];
51          Long price=separated[2].length()==0?defaultDominated[2]: Long.parseLong(separated[2]);
52          int year=separated[3].length()==0?defaultDominated[3]: Integer.parseInt(separated[3]);
53          Long odometer=separated[4].length()==0?defaultDominated[4]: Long.parseLong(separated[4]);
54          float lat=separated[5].length()==0?0:Float.parseFloat(separated[5]);
55          float dlong=separated[6].length()==0?0:Float.parseFloat(separated[6]);
56          String manufacturer=separated[7];
57          String make=separated[8];
58          String condition=separated[9];
59          int cylinders=0;
60          if(separated[10].length()!=0&&!separated[10].equals("other")){
61              Pattern p = Pattern.compile("\\d+");
62              Matcher matcher = p.matcher(separated[10]);
63              if(!matcher.hitEnd()){
64              matcher.find();
65              cylinders=Integer.parseInt(matcher.group(0));}
66          }
67          String fuel=separated[11];
68          String title_status=separated[12];
69          String transmission=separated[13];
70          String drive=separated[14];
71          String size=separated[15];
72          String type=separated[16];
73          String paint_color=separated[17];
74
75          EntryLine entrylineobject= new EntryLine(id,city,price,year,odometer,lat,dlong,manufacturer,make,condition,cylinders,fuel,title_status,transmission,drive,
            size,type,paint_color);
76          return entrylineobject;
77      }
```

Η μέθοδος createLine λαμβάνει σειρά αλφαριθμητικών πεδίων εγγραφής και επιστρέφει αντικείμενο της κλάσης EntryLine που αντιστοιχεί στην εν λόγω εγγραφή. Πρέπει να σημειωθεί πως η διαδικασία που ακολουθείται για να δημιουργηθούν τα πεδία διαστάσεων της κάθε μεταβλητής είναι η εξής. Σε περίπτωση που η εγγραφή δεν υπάρχει, ως τιμή του πεδίου ανατίθεται η αντίστοιχη τιμή της σειράς defaultDominated, μια τιμή που είναι εκ των προτέρων κυριαρχημένη. Η διαδικασία αυτή ακολουθείται για κάθε ένα από τα 18 πεδία πλήν του πεδίου cylinders. Σε αυτό, λόγω της μετρήσιμης μορφής των κυλίνδρων γίνεται προσπάθεια μέσω λογικής διαχωρισμού regexp παρόμοιας της μεθόδου RegexSeparation, να εξαχθεί η εν λόγω τιμή. Ως προεπιλεγμένη τιμή της περίπτωσης "other" τίθεται η τιμή 0. Η σειρά 75 δημιουργεί αντικείμενο της κλάσης EntryLine δοθέντων όλων των πεδίων στους αντίστοιχους τύπους τους, και το επιστρέφει στην επόμενη σειρά.

```
79      public Boolean AbsoluteLine(String[] separated){
80          if(separated.length!=18)System.out.println("FALSE SEPARATION");
81          if(separated[3].length()==0||separated[2].length()==0||separated[4].length()==0)return false;
82          return true;
83      }
```

Οι σειρές 79-83 αποτελούν την μέθοδο AbsoluteLine, η οποία δοθείσης σειράς αλφαριθμητικών επιστρέφει false αν κάποιο από τα σημαντικά πεδία που ενδιαφέρουν την εκάστοτε συνάρτησης κατάταξης είναι κενό και true αν όλα τα ενδιαφέροντα πεδία έχουν τιμές. Η μέθοδος-εργαλείο αυτή μπορεί να χρησιμοποιηθεί σε συνδυασμό με την μέθοδο createLine για να απορριφθούν όλες οι εγγραφές εκείνες οι οποίες έχουν ελλιπή σημαντικά πεδία.

```
85      private Integer[] constraints={0,0,500,1900,1000};
86      public Boolean ConstrainedLine(String[] separated){
87          int year=separated[3].length()==0?0: Integer.parseInt(separated[3]);
88          long price=separated[2].length()==0?0: Long.parseLong(separated[2]);
89          long odometer=separated[4].length()==0?0: Long.parseLong(separated[4]);
90
91          //constraints
92          if(year>=constraints[3]||price<=constraints[2]||odometer<=constraints[4])return false;
93          return true;
94      }
```

Η σειρά 85 αρχικοποιεί σειρά ακεραίων constrained η οποία αποθηκεύει τιμές-όρια, που αποτελούν τις αποδεκτές τιμές που μπορεί να έχει πεδίο εγγραφής. Χρήση της σειράς αυτής, κάνει η παρακάτω μέθοδος constrained Line, που αποτελεί παρόμοιο εργαλείο της μεθόδου AbsoluteLine. Η μόνη διαφορά είναι πως η μέθοδος ConstrainedLine επιστρέφει false αν μια τουλάχιστον τιμή σημαντικού πεδίου είναι εκτός του αντίστοιχου ορίου. Εναλλακτικά αν όλες οι τιμές σημαντικών πεδίων βρίσκονται εντός των δοθέντων ορίων της σειράς 85 επιστρέφεται true.

3.3.3 Η κλάση EntryLine

```
2    import java.io.Serializable;
```

Η κλάση EntryLine έχει στόχο την δημιουργία αντικειμένου που αντιστοιχεί σε κάθε εγγραφή και περιέχει στον σωστό τύπο την τιμή κάθε πεδίου της εγγραφής αυτής. Για να είναι δυνατή η ιδιότητα του προγράμματος να σειριοποιεί σε bytes το εκάστοτε αντικείμενο εγγραφής, γίνεται χρήση του interface Serializable[12], το οποίο και υλοποιεί η κλάση αυτή.

```
3   public class EntryLine implements Serializable{
4       private static final long serialVersionUID = 1L;
5       public int id,year,cylinders;
6       public float dlong,lat;
7       public long price,odometer;
8       public String city, manufacturer, make,condition,fuel,title_status,transmission,drive,size,type,paint_color;
9       EntryLine(Integer  id,String city,  Long Price, Integer  Year, Long Odometer ,
10      float dlong, float lat, String manufacturer, String make, String condition, Integer cylinders, String fuel, String title_status, String transmission, String
        drive, String size, String type,String paint_color){
11          this.id=id;
12          this.year=Year;
13          this.price=Price;
14          this.odometer=Odometer;
15          this.dlong=dlong;
16          this.lat= lat;
17          this.manufacturer=manufacturer;
18          this.make=make;
19          this.condition=condition;
20          this.cylinders=cylinders;
21          this.fuel=fuel;
22          this.title_status=title_status;
23          this.transmission=transmission;
24          this.drive=drive;
25          this.size=size;
26          this.type=type;
27          this.paint_color=paint_color;
28      }
29
30  }
31
```

Για κάθε πεδίο η κλάση αρχικοποιεί μεταβλητή αντίστοιχου τύπου και ονόματος. Πιο συγκεκριμένα οι ακέραιες μεταβλητές είναι το id της εγγραφής, το έτος κυκλοφορίας του αυτοκινήτου year, και το πλήθος των τροχών cylinders. Ως δεκαδικές μεταβλητές dlong, και lat αποτελούν αντίστοιχα το γεωγραφικό μήκος και πλάτος που αφορούν την εγγραφή. Ως ενδεχομένως μεγάλες τιμές ακεραίων τοποθετούνται η τιμή και τα χιλιόμετρα του οχήματος. Τέλος ως αλφαριθμητικές μεταβλητές αρχικοποιούνται τα υπόλοιπα 11 πεδία πόλη, κατασκευαστής, μοντέλο, κατάσταση, τύπος καυσίμων, τίτλος ιδιοκτησίας, εξάτμιση, σύστημα τροχών, μέγεθος και χρώμα.

[12] https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html

4 *ΠΕΙΡΑΜΑΤΙΚΗ ΑΝΑΛΥΣΗ: ΒΑΣΙΚΗ ΠΕΡΙΠΤΩΣΗ*

4.1 *Επερώτημα BNL*

Αρχικά θα επιχειρηθεί η δοκιμή του αλγορίθμου BNL πάνω στο αρχείο-εκροή επέκτασης .csv του σταδίου της προ-επεξεργασίας του αρχικού συνόλου εγγραφών του κεφαλαίου δύο. Πιο συγκεκριμένα θα αναζητηθεί η κορυφογραμμή του συνόλου των 689.600 εγγραφών βάσει προεπιλεγμένων κριτηρίων που θα αντανακλά η συνάρτηση κατάταξης της §3.3.1. Τα κριτήρια αυτά θα είναι η φθίνουσα τιμή, το αύξον έτος κυκλοφορίας και ο φθίνων αριθμός χιλιομέτρων του οχήματος. Συμπερασματικά, αναλόγως θα προσαρμοστεί και η συνάρτηση κατάταξης που θα καθορίζει την σχέση κυριαρχίας ανάμεσα σε δύο αντικείμενα εγγραφής της §3.3.3.

```
118     public static Boolean isDominatedBy(EntryLine A,EntryLine B){
119         return B.price<=A.price && B.odometer<=A.odometer && B.year>=A.year && (B.price<A.price||B.odometer<A.odometer||B.year>A.year);
120     }
```

Στον παραπάνω κώδικα Java διαφαίνεται η συνάρτηση της §3.3.1 isDominatedBy, η οποία έχει προσαρμοστεί στις διαστάσεις της επερώτησης που πρόκειται να δοκιμάσει την παραπάνω υλοποίηση του BNL. Η μέθοδος αυτή δοθέντων δυο αντικειμένων εγγραφών θα συγκρίνει ταυτόχρονα τις τρεις σημαντικές διαστάσεις τιμής, έτους κυκλοφορίας και χιλιομέτρων και θα συμπεραίνει κυριαρχία εγγραφής της B επι της A εφόσον οι σημαντικές διαστάσεις της B είναι εξίσου καλές σε σύγκριση με αυτών της A και υπάρχει τουλάχιστον μια τουλάχιστον ανώτερη σημαντική διάσταση της B.

4.2 *Εξαγώγιμα αποτελέσματα*

Μετά την εφαρμογή του παραπάνω επερωτήματος το πρόγραμμα επιστρέφει τρία id εγγραφών. Αυτά είναι τα [215337, 359425, 383892]. Οι σημαντικές διαστάσεις των εγγραφών αυτών

διαφαίνονται στον παρακάτω πίνακα. Πρέπει επίσης να σημειωθεί πως ο χρόνος που δαπανήθηκε για την περάτωση του επερωτήματος BNL κυμαίνεται στα 3032ms ενώ ο χώρος στα 0.067gb. Τέλος όπως συμπεραίνεται και από τις εκροές, το μήκος της κορυφογραμμής είναι 3 στοιχεία.

Id	Price	Year	Odometer
215337	10	2019	0
359425	1	2019	12
383892	1	2018	0

Πίνακας 12 Τα στοιχεία της κορυφογραμμής της πρώτης επερώτησης

4.3 *Συμπεράσματα*

Τα αποτελέσματα του ερωτήματος είναι αντικειμενικά η κορυφογραμμή όλων των εγγραφών του αρχικού συνόλου δεδομένων. Το γεγονός αυτό είναι προφανές αν αναλογιστεί κανείς την ποιότητα των διαστάσεων των παραπάνω εγγραφών που προκύπτει από την αντίστοιχη συνάρτηση κατάταξης isDominated που χρησιμοποιήθηκε στην §4.1. Αρχικά παρατηρείται πως οι εγγραφές, όπως ήταν αναμενόμενο, είναι ξένες κατά την εν λόγω σχέση κυριαρχίας ανά δυο μεταξύ τους. Καμία εγγραφή από τις παραπάνω 3 δεν κυριαρχεί επι άλλης. Σε ό,τι αφορά τις ίδιες τις τιμές των σημαντικών διαστάσεων δε, δεν προκαλεί εντύπωση η επιλογή των συγκεκριμένων εγγραφών από τον αλγόριθμο για την κορυφογραμμή, δεδομένων των ιδανικών τιμών των διαστάσεων τους που δύσκολα κυριαρχούνται από άλλες εγγραφές. Χαρακτηριστικά, όπως παρατηρείται από τον πίνακα οι επιλεχθείσες κυρίαρχες τιμές του πεδίου Price είναι αντίστοιχα $10, $1, $1 υποδηλώνοντας την αγοραπωλησία αμαξιού στις τιμές αυτές. Επίσης οι τιμές του πεδίου Year, έτους κυκλοφορίας του οχήματος είναι 2019, 2019 και 2018. Τέλος τα χιλιόμετρα που τα αυτοκίνητα αυτά διένυσαν πριν την αγοραπωλησία ήταν 0km, 12km και 0km. Συμπερασματικά, ο βέλτιστος χαρακτήρας των τιμών αυτών δίκαια οδήγησε στην επιλογή τους από την υλοποίηση του BNL της §3.3.

Πρέπει ωστόσο να σημειωθεί πως ο βέλτιστος χαρακτήρας των τιμών των επιλεχθέντων σημείων αποτελεί γεγονός που θα μπορούσε να αμαυρώσει την ουσιαστική αυθεντικότητα των εγγραφών αυτών. Δεδομένου πως η πλειοψηφία των εγγραφών προέρχεται από χρήστες και δεν υφίσταται στάδιο ελέγχου εγκυρότητας ή αυθεντικότητας από την ιστοσελίδα προέλευσης του συνόλου δεδομένων, εύκολα θα μπορούσε κανείς να εισάγει έκτοπες εγγραφές με αντικειμενικό αντίκτυπο

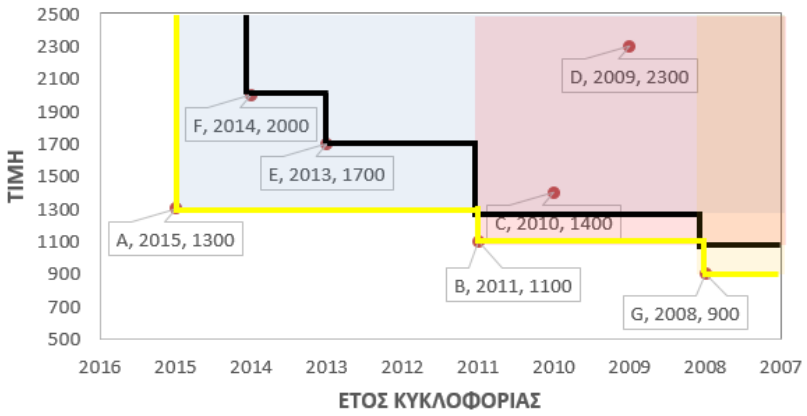
στην κορυφογραμμή του συνόλου των εγγραφών. Διακρίνεται λοιπόν ευαισθησία του αλγορίθμου BNL στις αναξιόπιστες εγγραφές εκείνες που, ενδεχομένως, οδηγούν στην απόρριψη των ζητούμενων αληθινών, βέλτιστων εγγραφών. Με σκοπό παραγωγής αντικειμενικών και ρεαλιστικών εκροών κορυφογραμμής, συνεπώς, κρίνεται αναγκαία η ύπαρξη μηχανισμού που θα διαχειρίζεται τέτοιου είδους έκτοπες εγγραφές. Τέλος, η ανάπτυξη στρατηγικής διαχείρισης αφύσικων εγγραφών θα οδηγήσει σε αποτελέσματα πιο συνεπή με την πραγματικότητα και θα αποτελούσε χρήσιμο εργαλείο για επερωτήματα πάνω σε σύνολα πραγματικών δεδομένων.

5 *ΣΤΡΑΤΗΓΙΚΕΣ ΑΝΤΙΜΕΤΩΠΙΣΗΣ ΕΚΤΟΠΩΝ ΕΓΓΡΑΦΩΝ*

5.1 Επερωτήματα K-Skyband

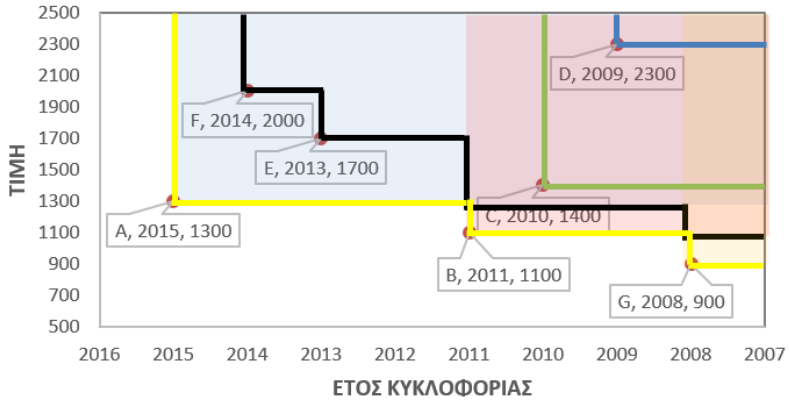
5.1.1 Ορισμός

Ένας ενδιαφέρων τρόπος αντιμετώπισης της ύπαρξης έκτοπων εγγραφών είναι τα επερωτήματα k-Skyband. Παρόμοια με τα επερωτήματα κορυφογραμμής, τα k-Skyband queries επιστρέφουν εγγραφές κυρίαρχες επι της πλειοψηφίας των εγγραφών του συνόλου δεδομένων. Η μόνη διαφορά με τα skyline queries είναι πως αποτέλεσμα των επερωτημάτων k-Skyband είναι τα σημεία εκείνα που κυριαρχούνται από το πολύ k εγγραφές. Είναι σαφές πως για k=0 οι εκροές k-Skyband και κορυφογραμμής είναι ίδιες. Όσο μεγαλώνει το k, τόσο χαλαρώνει ο περιορισμός κυριαρχίας και, κατ' επέκταση, επιτρέπει περισσότερα κυρίαρχα σημεία επι εγγραφή. Ακολουθεί εφαρμογή k-Skyband για k=1 στο σύνολο δεδομένων του παραδείγματος της §3.1



Εικόνα 26 Το σχήμα της 1-Skyband στο παράδειγμα αυτοκινήτων του κεφαλαίου 3

Η κίτρινη γραμμή διατρέχει όλα τα σημεία που κυριαρχούνται από ακριβώς 0 σημεία και αποτελεί την γνωστή κορυφογραμμή. Η μαύρη γραμμή διατρέχει όλα τα σημεία που κυριαρχούνται από ακριβώς 1 άλλο σημείο. Συνεπώς η 1-Skyband θα είχε ως αποτέλεσμα το σύνολο των σημείων πάνω στην κίτρινη και μαύρη γραμμή. Αναλυτικότερα, η εγγραφή $A\neg F, A\neg E$ άρα η F και η E είναι οι μόνες εγγραφές που κυριαρχούνται από ακριβώς 1 άλλη εγγραφή, ενώ οι A, B, G κυριαρχούνται από ακριβώς 0 εγγραφές. Άρα το αποτέλεσμα του Skyband στην περίπτωση αυτή για k=1 είναι {A, F, E, B, G}.



Εικόνα 27 Το σχήμα k-Skyband για διάφορες τιμές του k

Με παρόμοιο τρόπο, η παραπάνω εικόνα αναπαριστά την πράσινη γραμμή να διατρέχει όλα τα σημεία που κυριαρχούνται από ακριβώς 2 σημεία, και την μπλε να διατρέχει το σημείο που έχει ακριβώς 5 κυρίαρχα επι αυτό σημεία. Αντίστοιχα λοιπόν, αν εφαρμόζονταν k-Skyband για k=2,3,4 το αποτέλεσμα θα ήταν το σύνολο {A, B, C, E, F, G }, και για k=5 το αποτέλεσμα θα ήταν { A, B, C, D, E, F, G }.

Σε ό,τι αφορά το πρόβλημα που καλείται να αντιμετωπιστεί μέσω της χρήσης k-Skyband, οι εκροές των επερωτημάτων αυτών θα παρέχουν σημεία τα οποία κυριαρχούνται από k το πολύ άλλα σημεία. Αυτό θα επεκτείνει το εκάστοτε αποτέλεσμα σε εγγραφές πέραν των έκτοπων, βελτιώνοντας τις πιθανότητες ενός αντιπροσωπευτικού και ρεαλιστικού συμπεράσματος.

Οι [13] πρότειναν πως είναι δυνατή η τροποποίηση του αλγορίθμου BNL ούτως ώστε να επιστρέφεται αντί της κορυφογραμμής, το k-Skyband, δοθέντος ακεραίου k. Η τροποποίηση που πρόκειται να

[13] (Papadias D., 2003)

συμβεί θα είναι η συνθήκη εισαγωγής και εξαγωγής εγγραφής προς και από το παράθυρο που χρησιμοποιεί ο BNL για να αποθηκεύσει εγγραφές υποψήφιες για το αποτέλεσμα. Πιο συγκεκριμένα, εγγραφή θα εισέρχεται στο παράθυρο εφόσον κυριαρχείται από το πολύ k εγγραφές εντός του παραθύρου. Επίσης θα εξέρχεται από αυτό και θα απορρίπτεται ως πιθανό σημείο του k-Skyband, αν σε οποιοδήποτε σημείο εκτέλεσης του BNL αποδειχθεί πως κυριαρχείται από πλήθους μεγαλύτερου του k στοιχεία.

BNL k-Skyband

Input: set $\boldsymbol{X}$ of points, integer k

Output: set $\boldsymbol{S}$ of k-Skyband points

```
1.  S initialization as the empty set
2.  Input: = X;
3.  while(true)
4.      if (Input.isEmpty)
5.          ∀w_i ϵ Window ⇒ Move w_i to S
6.          if (Temp.isEmpty)
7.              break
8.          else
9.              Input: = Temp
10.             Temp.empty( )
11.     For x_i of Input
12.         if( x_i.Dominators == null)
13.             Dominators: = 0;
14.         else
15.             Dominators: = x_i.Dominators
16.         if (Window.isEmpty)
17.             x_i.Dominators:= Dominators
18.             Move x_i to Window
19.             continue
20.         For x_j of Window
21.             if (x_j¬x_i)
22.                 Dominators + +
23.             if (x_i¬x_j)
24.                 x_j.Dominators + +
25.                 if (x_j.Dominators > k)
26.                     Remove x_j from Window
27.         end For
28.         if (Dominators ≤ k)
29.             x_i.Dominators:=Dominators
30.             if (Window.isFull)
31.                 Move x_i to Temp
32.             else
33.                 Move x_i to Window
34.     end For
35. end while
36. return S
```

Εικόνα 28 Ψευδοκώδικας του αλγορίθμου k-Skyband μέσω BNL

Ο παραπάνω ψευδοκώδικας προτείνει τροποποίηση του BNL ούτως ώστε να επιστρέφει το k-Skyband εγγραφών δοθέντος ακεραίου k. Ακολουθείται διαδικασία πολύ παρόμοια με την κλασσική προσέγγιση του BNL για την εύρεση της κορυφογραμμής με μερικές διαφορές. Αρχικά πρέπει να σημειωθεί πως σε αντίθεση με την κλασική προσέγγιση, όπως ήδη αναφέρθηκε οι συνθήκες εισαγωγής και εξαγωγής εντός και εκτός του παραθύρου μεταβάλλονται. Στον βρόγχο if της σειράς 12 αρχικοποιείται μεταβλητή Dominators με τιμή που εξαρτάται από την ύπαρξη ή όχι χαρακτηριστικού ***Dominators*** μιας εγγραφής x_i. Το νούμερο αυτό θα αποτελεί το πλήθος των εγγραφών που κυριαρχούν επι την εξεταζόμενη εγγραφή x_i που λαμβάνεται από το σύνολο εισροών $Input$. Σε περίπτωση που το χαρακτηριστικό $x_i.\boldsymbol{Dominators}$ προϋπάρχει, η νέα μεταβλητή Dominators λαμβάνει την τιμή της, εναλλακτικά αρχικοποιείται με 0. Αν το παράθυρο είναι άδειο, στο x_i ανατίθενται τα κυρίαρχα επι αυτό σημεία και εισάγεται κατευθείαν στο παράθυρο. Ο επαναληπτικός βρόγχος for των σειρών 20-27 διαφέρει σε σύγκριση με αυτόν του BNL της §3.2.3. Σε περίπτωση που κατά την διάρκεια που διατρέχεται το παράθυρο, η τυχαία του εγγραφή x_j κυριαρχεί επι του εξεταζόμενου στοιχείου x_i, η μεταβλητή Dominators αυξάνεται κατά μονάδα, καθώς βρέθηκε νέο κυρίαρχο σημείο επι αυτό, το x_j. Σε αντίθετη περίπτωση στη σειρά 23, αν η εγγραφή x_j κυριαρχείται από το εξεταζόμενο στοιχείο x_i, το χαρακτηριστικό $\boldsymbol{x_j.Dominators}$ αυξάνεται κατά μονάδα εφόσον $x_i \neg x_j$. Επίσης, γίνεται ο έλεγχος περίπτωσης που τα κυρίαρχα στοιχεία επι του x_j έχουν ξεπεράσει σε πλήθος το k, στην οποία το x_j αφαιρείται από το παράθυρο εφόσον δεν είναι σημείο του k-Skyband καθώς έχει περισσότερα του k κυρίαρχα επι αυτό στοιχεία. Ο βρόγχος if της σειράς 28 ελέγχει αν το πλήθος των κυρίαρχων μέχρι στιγμής στοιχείων επι του εξεταζόμενου στοιχείου x_i είναι μικρότερο ή ίσο του k, ούτως ώστε να προταθεί κατ' επέκταση για εισαγωγή εντός του παραθύρου. Σε περίπτωση που οι αποθηκευτικοί πόροι για το παράθυρο

έχουν εξαντληθεί, το x_j περνάει στο προσωρινό αρχείο, εναλλακτικά στο παράθυρο. Η κύρια διαφορά ως προς την υλοποίηση του BNL για k-Skyband, είναι πως πρέπει να διατηρείται το πλήθος των κυρίαρχων στοιχείων των υποψήφιων εγγραφών σε αντίθεση με τον BNL για κορυφογραμμή, όπου αρκούσε μια προσωρινή Μπουλιανή μεταβλητή για να αποφανθεί ο αλγόριθμος την υποψηφιότητα του εκάστοτε εξεταζόμενου στοιχείου ως σημείο της κορυφογραμμής.

5.1.2 Υλοποίηση σε Java

Ως προς την εκ των πραγμάτων υλοποίηση σε Java της k-Skyband μέσω του BNL οι τροποποιήσεις στον κώδικα της §3.3 είναι ελάχιστες. Αρχικά πρέπει να σημειωθεί πως οι κλάσεις CsvCacher και XXFile θα ληφθούν αυτούσιες από την §3.3 εφόσον δεν θα χρειαστεί περεταίρω τροποποίηση σε αυτές. Οι κύριες αλλαγές θα συμβούν στις υπόλοιπες δυο κλάσεις EntryLine και Skyband (πρώην κλάση BNL).

```
2   import java.io.Serializable;
3
4   public class EntryLine implements Serializable{
5       private static final long serialVersionUID = 1L;
6       public int id,year,cylinders,Dominators;
7       public float dlong,lat;
8       public long price,odometer;
9       public String city, manufacturer, make,condition,fuel,title_status,transmission,
        drive,size,type,paint_color;
10      EntryLine(Integer  id,String city,  long Price, Integer  Year, long Odometer ,
11      float dLong, float lat, String manufacturer, String make, String condition,
        Integer cylinders, String fuel, String title_status, String transmission, String
        drive, String size, String type,String paint_color){
12          this.id=id;
13          this.year=Year;
14          this.price=Price;
15          this.odometer=Odometer;
16          this.dlong=dlong;
17          this.lat= lat;
18          this.manufacturer=manufacturer;
19          this.make=make;
20          this.condition=condition;
21          this.cylinders=cylinders;
22          this.fuel=fuel;
23          this.title_status=title_status;
24          this.transmission=transmission;
25          this.drive=drive;
26          this.size=size;
27          this.type=type;
28          this.paint_color=paint_color;
29          this.Dominators=0;
30      }
31
32  }
```

Σε ό,τι αφορά την κλάση EntryLine, η μόνη τροποποίηση που συνέβη για να επιστραφεί το k-Skyband, ήταν η προσθήκη επιπλέον μεταβλητής ακεραίων τιμών Dominators στη σειρά 6. Αυτή θα αποθηκεύει το πλήθος των εγγραφών κυρίαρχων επι της εγγραφής στην οποία αντιστοιχεί το

αντικείμενο της κλάσης αυτής. Τέλος όπως φαίνεται και από τη σειρά 29. Κατά την αρχικοποίηση αντικειμένου της κλάσης EntryLine, η τιμή της μεταβλητής Dominators τίθεται κατευθείαν σε 0.

```
2  import java.util.ArrayList;
3  import java.util.Iterator;
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.io.FileNotFoundException;
7  import java.io.FileOutputStream;
8  import java.io.IOException;
9  import java.io.ObjectInputStream;
10 import java.io.ObjectOutputStream;
11 public class Skyband {
       Run | Debug
12     public static void main(String[] args) {···
117
118    public static Boolean isDominatedBy(EntryLine A,EntryLine B){···
121
122    public static ObjectOutputStream newOutputFrom(String filePath){···
140
141    public static ObjectInputStream newInputFrom(String filePath){···
158 }
```

Η κλάση Skyband, αποτελώντας τροποποίησης της κλάσης BNL της §3.3.1, χρησιμοποιεί όλα τα εγγενή εργαλεία της Java που χρησιμοποιούνται σε αυτήν. Επίσης, διατηρεί αυτούσιες τις μεθόδους isDominatedBy, newOutputFrom και newInputFrom με την BNL της §3.3.1. Η μόνη μέθοδος που τροποποιείται είναι η main.

```
12     public static void main(String[] args) {
13         Long start = System.currentTimeMillis();
14         String filename="CSVfilepath", newCache="Temptxt1 path", TempPath="Temptxt2 path";
15         CsvCacher cache=new CsvCacher(filename);
16         int TempFileEntries=0,CacheEntriesCount=0,maxException=1500000,k;
17         Boolean mainFinished=false,switcher=true,isDominated=false, writeTempOnly=false;
18         ArrayList<Integer> resultSkyLine=new ArrayList<>();
19         ArrayList<EntryLine> window= new ArrayList<EntryLine>();
20         EntryLine newEntry;
21         try{···
101        catch (FileNotFoundException e) {
102            System.out.println("Does not exist");
103        }
104        catch (IOException e) {
105            System.out.println("I/O Issue");
106        }
107        catch (ClassNotFoundException e) {
108            e.printStackTrace();
109        }
110
111        System.out.println(resultSkyLine);
112
113        Long elapsedTimeMillis = System.currentTimeMillis()-start;
114        System.out.printf("\nElapsed Time is %d ms,  Space used is %.3f GB",elapsedTimeMillis,
115        (Runtime.getRuntime().totalMemory()-Runtime.getRuntime().freeMemory())/(1024.0 * 1024.
           0* 1024.0));
116    }
```

Σε ό,τι αφορά την ίδια την κλάση main, στην αρχικοποίηση των μεταβλητών πραγματικών τιμών τις σειράς 16 προστίθεται η μεταβλητή k που από την οποία θα εξαρτώνται οι εκροές του αλγορίθμου k-Skyband. Εκτός του βρόγχου try, οι υπόλοιπες σειρές ακολουθούν ταυτόσημη πορεία με αυτές του BNL.

```
21      try{
22          ObjectInputStream input=null;
23          ObjectOutputStream output = newOutputFrom(TempPath);
24          while( true ){
25              if(mainFinished){···
39              else cache.startProcess();
40
41              for (int i = 0; true; i++) {
42                  if(i>=CacheEntriesCount&&mainFinished)break;
43                  if(mainFinished)newEntry=(EntryLine) input.readObject();
44                  else newEntry=cache.nextEntry();
45
46                  if(newEntry==null){···
53
54                  if(window.size()==0){···
60
61                  for( Iterator<EntryLine> iterator=window.iterator(); iterator.hasNext(); ){
62                      EntryLine eleEntry=iterator.next();
63                      if( isDominatedBy(newEntry, eleEntry)){
64                          newEntry.Dominators++;
65                      }
66                      if( isDominatedBy(eleEntry, newEntry)){
67                          eleEntry.Dominators++;
68                          if(eleEntry.Dominators>k)iterator.remove();
69                      }
70                  }
71
72                  if(newEntry.Dominators<=k){
73                      if(writeTempOnly==true){···
77                      else {
78                          try{···
86                          catch( OutOfMemoryError exception){
87                              output.writeObject(newEntry);
88                              TempFileEntries++;
89                              writeTempOnly=true;
90                          }
91                      }
92                  }
93              }
94              if(CacheEntriesCount!=0){
95                  switcher=!switcher;
96                  input.close();
97                  output.flush();
98                  output.close();
99              }
100         }
101     }
```

Το σημείο τροποποίησης εντός του βρόγχου *try* των σειρών 21-100 που χρίζει ιδιαίτερης σημασίας είναι η διαδικασία σύγκρισης του εξεταζόμενου στοιχείου newEntry με το τυχαίο αντικείμενο εγγραφής eleEntry του παραθύρου window. Ακολουθώντας λογική ίδια με τον ψευδοκώδικα της προηγούμενης παραγράφου, σε περίπτωση που το newEntry κυριαρχείται από το eleEntry, αυξάνεται το πεδίο Dominators του αντικειμένου αυτού. Εναλλακτικά αυξάνεται το πεδίο Dominators του eleEntry. Η σειρά 68 διαχειρίζεται την περίπτωση κατά την οποία το πλήθος κυρίαρχων εγγραφών επι του eleEntry ξεπεράσει το k, όπου το στοιχείο αφαιρείται κατευθείαν από το παράθυρο window, και απορρίπτεται ως σημείο του k-Skyband. Τέλος σημείο τροποποίησης

αποτέλεσε και η συνθήκη του βρόγχου if της σειράς 72. Η νέα συνθήκη πρότασης εισαγωγής εγγραφής εντός του παραθύρου window είναι το πλήθος κυρίαρχων μέχρι τώρα στοιχείων επι την εξεταζόμενη εγγραφή να είναι μικρότερο ή ίσο του ορίου k.

5.1.3 Πειραματικά συμπεράσματα

Εφαρμόζοντας k-Skyband για $k\epsilon[0,3]$ τα id των σημείων εκροής διαφαίνονται στο παρακάτω σχεδιάγραμμα.

k	k-Skyband new
0	[215337, 359425, 383892]
1	[260004, 305141, 530096, 531228, 531229, 602106]
2	[49999, 90117, 153383, 169332, 249786, 259666, 411194, 428945, 531040]
3	[89417, 147628]

Πίνακας 13 Τα στοιχεία των k-Skyband για k=0,1,2,3

Όπως φαίνεται, η εκροή για k=0 είναι πανομοιότυπη με αυτήν του BNL για κορυφογραμμή όπως ήταν αναμενόμενο. Τα σημεία παρόλα αυτά εξακολουθούν να αποτελούν έκτοπες εγγραφές.

Για κ=1, πέραν των τριών πρώτων έκτοπων εγγραφών, επιστρέφεται επιπλέον σύνολο 6 νέων εγγραφών που έχουν την κοινή ιδιότητα να κυριαρχούνται μόνο από 1 σημείο, και πιο συγκεκριμένα, κάποιο από αυτά της προηγούμενης σειράς. Μια αναφορά στις τιμές των αντίστοιχων σημαντικών πεδίων θα αναδείξει την ποιότητα των επιπλέον σημείων του 1-Skyband.

Id	Price	Year	Odometer
260004	1	2019	1111
305141	1	2016	0
530096	1	2018	1
531228	1	2018	1
531229	1	2018	1
602106	35	2019	0

Πίνακας 14 Οι εγγραφές που ανήκουν στο 1-Skyband

Ο παραπάνω πίνακας παρουσιάζει τις τιμές των σημαντικών πεδίων Price, Year και Odometer των επιπλέον εγγραφών που ο 1-Skyband προσέθεσε στην εκροή του 0-Skyband. Όπως διαφαίνεται οι 5 από τις 6 εγγραφές παρουσιάζουν τιμή $1 και χιλιόμετρα το πολύ 1. Το γεγονός αυτό σε συνδυασμό με το πρόσφατο του έτους κυκλοφορίας εγείρει υποψίες για την εγκυρότητα τους. Δεδομένου μάλιστα πως μεταξύ των ετών 2016-2019 το μέσο κόστος αγοράς οχήματος στις Η.Π.Α. ήταν περί των $30.000[14], η αυθεντικότητα των παραπάνω εγγραφών μπορεί ευκόλως να αμφισβητηθεί.

Id	Price	Year	Odometer
49999	1	2019	3079
90117	20	2018	0
153383	60	2018	1
169332	1	2019	3079
249786	10	2017	0
259666	30	2019	0
411194	1	2019	3079
428945	125	2019	0
531040	1	2014	0

Πίνακας 15 Οι εγγραφές που ανήκουν στο 2-Skyband

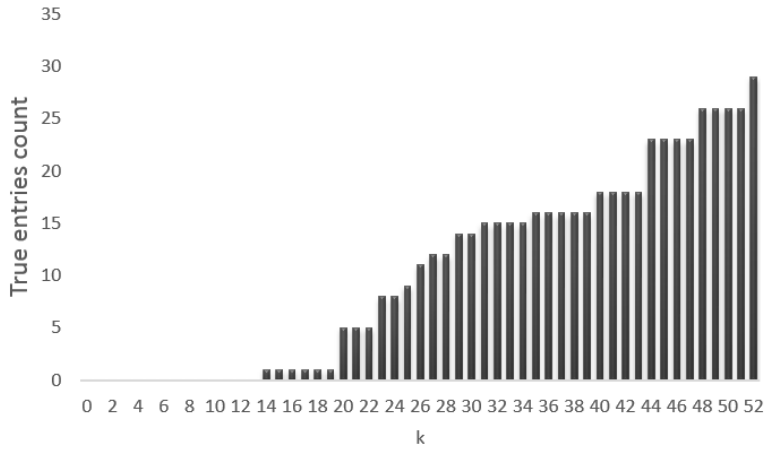
Id	Price	Year	Odometer
89417	1300	2019	0
147628	1	2012	0

Πίνακας 16 Οι εγγραφές που ανήκουν στο 3-Skyband

Ομοίως για k=2,3 παρατηρείται πως η πλειοψηφία των εγγραφών έχουν τιμές που πιθανότατα δεν ανταποκρίνονται στην πραγματικότητα και η αυθεντικότητα τους αμφισβητείται.

[14] (Wagner, 2020)

Με σκοπό την ανάλυση της συμπεριφοράς του k-Skyband ως προς τις έκτοπες εγγραφές, θα οριστεί ως κάτω όριο τιμής και χιλιομέτρων τα $5000 και 1000km αντίστοιχα. Με αυτόν τον τρόπο θα είναι εύκολη η απόδειξη πιθανής εξάρτησης του πλήθους πραγματικών εγγραφών από το όριο k.



Εικόνα 29 Πλήθος αληθινών εγγραφών ως προς τις διάφορες τιμές του k σε context k-Skyband

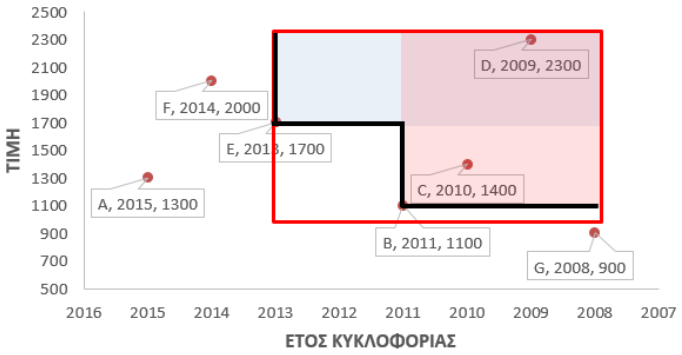
Εφαρμόζοντας τον k-Skyband για τις διάφορες τιμές του k, παρατηρείται πως το πλήθος των επιστρεφόμενων πραγματικών εγγραφών αυξάνεται καθώς αυξάνεται το k. Η εμφάνιση όλο και περισσότερων πραγματικών εγγραφών είναι έκδηλη στο σχήμα για $k \geq 14$. Αυτό αποτελεί κάτι το αναμενόμενο καθώς χαλαρώνοντας τον περιορισμό του πλήθους επιτρεπόμενων κυριάρχων εγγραφών επι στοιχείο, επιτρέπονται όλο και περισσότερα στοιχεία στην εκροή του k-Skyband. Συμπερασματικά ένας τρόπος αντιμετώπισης των έκτοπων εγγραφών είναι η σταδιακή αύξηση του k μέχρις ότου στις επιστρεφόμενες εγγραφές να εμφανιστούν πραγματικές.

Παρόλο που εν τέλει η αύξηση του k θα επιφέρει και ανάλογο πλήθος πραγματικών εγγραφών θα πρέπει να σημειωθεί πως η προσέγγιση αυτή δεν έρχεται δίχως περιορισμούς. Η αύξηση του k συνεπάγεται περισσότερους ελέγχους ανάμεσα σε εγγραφές καθώς εντός του παραθύρου επιτρέπεται μεγαλύτερο πλήθος στοιχείων, παρουσιάζοντας έτσι την αύξηση του χρόνου που απαιτείται μέχρι να παραχθεί η ανάλογη εκροή. Στην περίπτωση που ο BNL Skyband, μάλιστα, έχει χωρικούς περιορισμούς σε ό,τι αφορά τη χωρητικότητα που αφιερώνει η κύρια μνήμη στο παράθυρο, οι υπολογιστικοί πόροι που θα πρέπει να καταναλωθούν θα αυξηθούν ραγδαία ενόσω μεγαλώνει η τιμή του k.

5.2 *Επερωτήματα Constrained Skyline*

5.2.1 Ορισμός

Μια άλλη προσέγγιση για την διαχείριση έκτοπων εγγραφών είναι οι επερωτήσεις περιορισμένης κορυφογραμμής. Σε αυτήν την περίπτωση, δοθέντων περιορισμών $c = \{c_1, c_2, c_3, \dots, c_n\}$ για κάθε μια από τις διαστάσεις εγγραφής $x_j = \{d_1, d_2, d_3, \dots, d_n\}$ με $c_i = \{d_{i_{min}}, d_{i_{max}}\}$. Οι επερωτήσεις περιορισμένης εγγραφής επιστρέφουν την κορυφογραμμή εκείνη των στοιχείων x_j του αρχικού συνόλου για τα οποία $\forall i \epsilon [1, n] : d_{i_{min}} \leq d_i \leq d_{i_{max}}$. Δηλαδή επιστρέφεται η κορυφογραμμή ενός υποσυνόλου των αρχικών εγγραφών τα στοιχεία του οποίου βρίσκονται εντός των δοσμένων διαστημάτων c_i.



Εικόνα 30 Κορυφογραμμή υπο περιορισμούς συνόλου c={[2008,2013],[950,2350]}

Το παραπάνω σχήμα περιέχει την περιορισμένη κορυφογραμμή δοθέντων περιορισμών $c = \{[2008,2013], [950,2350]\}$ στα πεδία Έτος κυκλοφορίας και τιμή αντίστοιχα. Όπως παρατηρείται, το κόκκινο πλαίσιο περικλείει την επιφάνεια που καταλαμβάνουν οι περιορισμοί αυτοί καθώς και όλα τα σημεία που τους ικανοποιούν. Πιο συγκεκριμένα, τα οχήματα A, F, G απορρίπτονται ως πιθανά σημεία της περιορισμένης κορυφογραμμής, καθώς οι διαστάσεις τους, τιμή και έτος κυκλοφορίας, δεν ικανοποιούν τους δοθέντες περιορισμούς. Η μαύρη γραμμή που διατρέχει τα σημεία E, B αποτελεί την ζητούμενη κορυφογραμμή καθώς τα σημεία αυτά δεν έχουν σημεία που τα κυριαρχούν εντός του κόκκινου πλαισίου περιορισμών. Κατ' επέκταση τα στοιχεία αυτά είναι και η εκροή του αντίστοιχου επερωτήματος.

Σε ό,τι αφορά τον αλγόριθμο BNL, εύκολα μπορεί να προσαρμοστεί ούτως ώστε να επιστρέφει την περιορισμένη κορυφογραμμή. Η κύρια ιδέα είναι η κρίση ενός σημείου του αρχικού συνόλου

δεδομένων ως υποψήφιο για επεξεργασία ή όχι. Προφανώς σε περίπτωση που οι τιμές των διαστάσεων του σημείου δεν ικανοποιούν τους δοθέντες περιορισμούς, το σημείο δεν επεξεργάζεται καν.

CONSTRAINED BNL

Input: set $\boldsymbol{X}$ of points, set c of constraints

Output: set $\boldsymbol{S}$ of constrained skyline points

1. $\boldsymbol{S}$ initialization as the empty set
2. $\boldsymbol{Input} := \boldsymbol{X}$;
3. $\boldsymbol{while}(\boldsymbol{true})$
4. $\quad \boldsymbol{if}\ (\boldsymbol{Input}.isEmpty)$
5. $\quad\quad \forall w_i \epsilon\ \boldsymbol{Window} \Rightarrow \boldsymbol{Move}\ w_i\ to\ \boldsymbol{S}$
6. $\quad\quad \boldsymbol{if}\ (\boldsymbol{Temp}.isEmpty)$
7. $\quad\quad\quad \boldsymbol{break}$
8. $\quad\quad \boldsymbol{else}$
9. $\quad\quad\quad \boldsymbol{Input} := \boldsymbol{Temp}$
10. $\quad\quad\quad \boldsymbol{Temp}.\boldsymbol{empty}()$
11. $\quad \boldsymbol{For}\ x_i\ \boldsymbol{of}\ \boldsymbol{Input}$
12. $\quad\quad isDominated := \boldsymbol{false;}$
13. $\quad\quad \boldsymbol{if}\ (!\ \boldsymbol{x_i}\ \boldsymbol{satisfies}\ \boldsymbol{c})$
14. $\quad\quad\quad \boldsymbol{Remove}\ \boldsymbol{x_j}\ from\ \boldsymbol{Input};$
15. $\quad\quad\quad \boldsymbol{continue;}$
16. $\quad\quad \boldsymbol{if}\ (\boldsymbol{Window}.isEmpty)$
17. $\quad\quad\quad \boldsymbol{Move}\ x_i\ to\ \boldsymbol{Window}$
18. $\quad\quad\quad \boldsymbol{continue}$
19. $\quad\quad \boldsymbol{For}\ x_j\ \boldsymbol{of}\ \boldsymbol{Window}$
20. $\quad\quad\quad \boldsymbol{if}\ (\boldsymbol{x_j} \neg \boldsymbol{x_i})$
21. $\quad\quad\quad\quad isDominated := \boldsymbol{true};$
22. $\quad\quad\quad\quad \boldsymbol{Remove}\ \boldsymbol{x_j}\ from\ \boldsymbol{Input}$
23. $\quad\quad\quad\quad \boldsymbol{break};$
24. $\quad\quad\quad \boldsymbol{if}\ (\boldsymbol{x_i} \neg \boldsymbol{x_j})$
25. $\quad\quad\quad\quad \boldsymbol{Remove}\ \boldsymbol{x_j}\ from\ \boldsymbol{Window}$
26. $\quad\quad \boldsymbol{end\ For}$
27. $\quad\quad \boldsymbol{if}\ (isDominated == \boldsymbol{false})$
28. $\quad\quad\quad \boldsymbol{if}\ (\boldsymbol{Window}.\boldsymbol{isFull})$
29. $\quad\quad\quad\quad \boldsymbol{Move}\ x_i\ to\ \boldsymbol{Temp}$
30. $\quad\quad\quad \boldsymbol{else}$
31. $\quad\quad\quad\quad \boldsymbol{Move}\ x_i\ to\ \boldsymbol{Window}$
32. $\quad \boldsymbol{end\ For}$
33. $\boldsymbol{end\ while}$
34. $\boldsymbol{return\ S}$

Εικόνα 31 Ψευδοκώδικας του αλγορίθμου περιορισμένης κορυφογραμμής μέσω BNL

Ο παραπάνω ψευδοκώδικας αποτελεί τροποποίηση του BNL της §3.2.3 ούτως ώστε η εκροή να αποτελεί την περιορισμένη κορυφογραμμή δοθέντων συνόλου σημείων X και συνόλου περιορισμών c. Η μόνη τροποποίηση που έγινε στο σύνολο του κώδικα της §3.2.3 αποτελεί η προσθήκη των σειρών 13- 15. Ο βρόγχος if της σειράς 13 ελέγχει αν το εκάστοτε εξεταζόμενο στοιχείο του συνόλου εκροών $Input$ ικανοποιεί το σύνολο δοθέντων περιορισμών c. Σε περίπτωση που οι περιορισμοί δεν ικανοποιούνται, το σημείο αφαιρείται από το σύνολο εκροών καθώς έχει απορριφθεί, και ο έλεγχος συνεχίζεται για τα υπόλοιπα σημεία του συνόλου. Εναλλακτικά ο έλεγχος για το x_i συμβαίνει κανονικά με διαδικασία ταυτόσημη με αυτή της §3.2.3.

5.2.2 Υλοποίηση σε Java

Σε ό,τι αφορά την υλοποίηση στη γλώσσα Java, η τροποποίηση του BNL για επιστροφή της περιορισμένης κορυφογραμμής θα γίνει πάνω στον κώδικα της §3.3. Δεδομένου πως η βέλτιστη προσέγγιση απόρριψης στοιχείων που δεν ικανοποιούν δοθέντες περιορισμούς περιλαμβάνει, απαραίτητα, τον έλεγχο κάθε στοιχείου, αυτός θα γίνεται πάνω στις εγγραφές του αρχείου επέκτασης csv που έχει ως εκροή το στάδιο της προ επεξεργασίας του κεφαλαίου 2. Πιο συγκεκριμένα, κατά την αρχική ανάγνωση εγγραφής από το αρχείο αυτό θα ελέγχεται η ικανοποίηση των συνθηκών της εν λόγω εγγραφής και σε περίπτωση μη τήρησης των δοθέντων περιορισμών, η εγγραφή θα απορρίπτεται. Οι κλάσεις που θα χρησιμοποιηθούν για την περιορισμένη κορυφογραμμή είναι ταυτόσημες με αυτές της §3.3. Η μόνη τροποποίηση που θα συμβεί θα είναι πάνω στην κλάση CsvCacher.java και θα αφορά τον έλεγχο περιορισμών των αρχικών εγγραφών του αρχείου .csv, όπως περιεγράφηκε παραπάνω.

```
2   import java.util.regex.Matcher;
3   import java.util.regex.Pattern;
4   public class CsvCacher{
5       private String inputFile;
6       public XXFile file_dataset = new XXFile();
7       private Integer[] defaultDominated={0 ,0,Integer.MAX_VALUE,0,Integer.MAX_VALUE};
8
9       public CsvCacher(String inputFile){···
12
13      public void startProcess(){···
17
18      public  EntryLine nextEntry(){···
29
30      public void endProcess(){···
33
34      public String[] RegexSeparation(String temp){···
45
46      public EntryLine createLine(String[] separated){···
78
79      public Boolean AbsoluteLine(String[] separated){···
84
85      private long[][] constraints={null,null,{500,Long.MAX_VALUE},{1910,2019},{1000,Long.MAX_VALUE} };
86      public Boolean ConstrainedLine(String[] separated){···
98
99  }
```

Οι τροποποιήσεις πάνω στην κλάση CsvCacher θα συμβούν μόνο στις μεθόδους των σειρών 46 και 86, createLine και ConstrainedLine αντίστοιχα. Όλες οι άλλες μέθοδοι καθώς και η λειτουργία τους μεταφέρεται αυτούσια από την §3.3. Τέλος στη σειρά 85 τροποποιείται σειρά σειρών αριθμών constraints που θα περιέχει τους περιορισμούς στους οποίους θα δοκιμαστούν οι τιμές των σημαντικών πεδίων των εγγραφών.

```
85      private long[][] constraints={null,null,{500,Long.MAX_VALUE},{1910,2019},{1000,Long.MAX_VALUE} };
86      public Boolean ConstrainedLine(String[] separated){
87          long price=     separated[2].length()==0?    0: Long.parseLong(separated[2]);
88          int year=       separated[3].length()==0?    0: Integer.parseInt(separated[3]);
89          long odometer=  separated[4].length()==0?    0: Long.parseLong(separated[4]);
90
91          if(
92              (price<constraints[2][0]     ||  price>constraints[2][1])  ||
93              (year<constraints[3][0]      ||  year>constraints[3][1])   ||
94              (odometer<constraints[4][0] ||  odometer>constraints[4][1])
95              )return false;
96          return true;
97      }
```

Όσον αφορά την μέθοδο της σειράς 86 ConstrainedLine αυτή θα επιστρέφει Μπουλιανή τιμή true, αν δοθείσας διαχωρισμένης μέσω της μεθόδου RegexSeparation εγγραφής, οι τιμές των σημαντικών πεδίων της βρίσκονται εντός των ορίων που ορίζει η σειρά σειρών αριθμών constraints. Εναλλακτικά, αν τουλάχιστον μια τιμή σημαντικού πεδίου βρίσκεται εκτός του διαστήματος περιορισμών που ορίζει η μεταβλητή constraints, επιστρέφεται false.

```
46      public EntryLine createLine(String[] separated){
47          if(separated.length!=18)System.out.println("FALSE SEPARATION");
48          if(ConstrainedLine(separated)==false)return null;
49          int id=Integer.parseInt(separated[0]);
50          String city=separated[1];
51          Long price=separated[2].length()==0?defaultDominated[2]: Long.parseLong(separated[2]);
52          int year=separated[3].length()==0?defaultDominated[3]: Integer.parseInt(separated[3]);
53          Long odometer=separated[4].length()==0?defaultDominated[4]: Long.parseLong(separated[4]);
54          float lat=separated[5].length()==0?0:Float.parseFloat(separated[5]);
55          float dlong=separated[6].length()==0?0:Float.parseFloat(separated[6]);
56          String manufacturer=separated[7];
57          String make=separated[8];
58          String condition=separated[9];
59          int cylinders=0;
60          if(separated[10].length()!=0&&!separated[10].equals("other")){
61              Pattern p = Pattern.compile("\\d+");
62              Matcher matcher = p.matcher(separated[10]);
63              if(!matcher.hitEnd()){
64              matcher.find();
65              cylinders=Integer.parseInt(matcher.group(0));}
66          }
67          String fuel=separated[11];
68          String title_status=separated[12];
69          String transmission=separated[13];
70          String drive=separated[14];
71          String size=separated[15];
72          String type=separated[16];
73          String paint_color=separated[17];
74
75          EntryLine entrylineobject= new EntryLine(id,city,price,year,odometer,lat,dlong,manufacturer,make,condition,
            cylinders,fuel,title_status,transmission,drive,size,type,paint_color);
76          return entrylineobject;
77      }
```

Υπενθυμίζεται πως η λειτουργία της μεθόδου createLine περιλαμβάνει, δοθείσας σειράς αλφαριθμητικών που αντιστοιχούν στην διαχωρισμένη στα πεδία της εγγραφή, την δημιουργία, μετά από συγκεκριμένους ελέγχους διαχείρισης ελλιπών τιμών, αντικειμένου της κλάσης EntryLine που αντιστοιχεί στην εγγραφή αυτή. Η τροποποίηση με σκοπό την επιστροφή της περιορισμένης κορυφογραμμής αποτελεί η προσθήκη του βρόγχου if της σειράς 48. Αυτός ελέγχει αν η τιμή της μεθόδου ConstrainedLine που περιεγράφηκε παραπάνω, δοθείσας της εισροής σειράς αλφαριθμητικών που δέχεται η συνάρτηση αυτή, επιστρέφει τιμή false. Δηλαδή ελέγχεται αν οι τιμές των σημαντικών πεδίων ικανοποιούν τους δοθέντες περιορισμούς. Σε περίπτωση που δεν τους ικανοποιούν, επιστρέφεται το κενό αντικείμενο null. Σε διαφορετική περίπτωση ακολουθείται η ίδια διαδικασία της §3.3.2 που δημιουργεί και επιστρέφει το αντίστοιχο στην εγγραφή αντικείμενο της κλάσης EntryLine.

Συμπερασματικά, ο έλεγχος ικανοποίησης συνθηκών της κάθε εγγραφής συμβαίνει κατά την διάρκεια της πρώτης φοράς που θα αναγνωστεί από το πρόγραμμα, δηλαδή το στάδιο επεξεργασίας του αρχικού αρχείου επέκτασης .csv. Αυτό θα οδηγήσει στο φιλτράρισμα του συνόλου των σημείων από το πρώτο στάδιο του αλγορίθμου, και δεν θα χρειάζονται περεταίρω έλεγχοι ικανοποίησης συνθηκών κατά την διάρκεια του σταδίου ανάγνωσης και εγγραφής από και προς το προσωρινό αρχείο που χρησιμοποιεί ο BNL.

5.2.3 Πειραματικά Συμπεράσματα

Η εφαρμογή του αναπτυχθέντος προγράμματος υπολογισμού της περιορισμένης κορυφογραμμής της προηγούμενης παραγράφου θα βοηθήσει στην εξαγωγή συμπερασμάτων για την αποδοτικότητα του σε σχέση με την προσέγγιση του k-Skyband.

Id	Price	Year	Odometer
19187	800	2019	233600
99750	2000	2019	140688
117150	575	2004	1000
143661	1600	2019	212000
151970	800	2018	1000
184812	16800	2019	1000
184817	16800	2019	1000
217699	12900	2019	1568
232379	500	2018	10300
232551	7995	2019	72000
232605	7995	2019	72000
232728	7995	2019	72000
232728	500	2014	1245
369011	10500	2019	25835
372134	675	2015	1000
512183	516	2018	4900
638186	500	2015	2015
638188	500	2015	2015
638190	500	2015	2015
638196	500	2015	2015
638197	500	2015	2015
638205	500	2015	2015
661202	595	2017	1231
687700	589	2018	1347

Πίνακας 17 Οι εγγραφές που ανήκουν στο Constrained Skyline

Όταν στο πρόγραμμα δίνεται η αναμενόμενη εκροή του συνόλου των εγγραφών του αρχείου επέκτασης .csv που προέρχεται από την διαδικασία προ-επεξεργασίας του κεφαλαίου 2, επιστρέφεται σύνολο 24 εγγραφών. Ο παραπάνω πίνακας απεικονίζει τις τιμές των σημαντικών πεδίων των εγγραφών αυτών. Ευκόλως παρατηρείται πως κάθε εγγραφή ικανοποιεί τους δοθέντες περιορισμούς της §4.4.5, συμπεραίνοντας πως το επιστρεφόμενο σύνολο αποτελεί την ζητούμενη περιορισμένη κορυφογραμμή. Ως αποτέλεσμα, μια προσέγγιση καθορισμού των βέλτιστων αξιόπιστων εγγραφών, είναι η τροποποίηση του BNL με τέτοιο τρόπο ώστε όλα τα σημεία που δεν ικανοποιούν τους περιορισμούς αυτούς να απορρίπτονται κατά την πρώτη επανάληψη του αλγορίθμου, δίνοντας ως εκροή την περιορισμένη κορυφογραμμή.

5.3 *Συμπεράσματα αντιμετώπισης έκτοπων εγγραφών*

Σε ό,τι αφορά την σύγκριση των δυο προσεγγίσεων λήψης των βέλτιστων αξιόπιστων εγγραφών μέσω του αλγορίθμου BNL, παρατηρείται πως και στις 2 περιπτώσεις οι εκροές περιέχουν τα ζητούμενα βέλτιστα σημεία που, σύμφωνα με τους εκάστοτε περιορισμούς, δεν μπορεί να αμφισβητηθεί η εγκυρότητα τους. Παρόλα αυτά σε ό,τι αφορά την τροποποίηση του BNL για k-Skyband, είναι προφανής η ανάγκη σταδιακής αύξησης του k μέχρι οι εκροές να περιέχουν έγκυρες εγγραφές. Ως αποτέλεσμα δεν είναι δυνατή η εκ των προτέρων γνώση του κατάλληλου k που θα οδηγήσει στα ζητούμενα σημεία. Η ανάγκη συνεπώς για εκτέλεση του προγράμματος k-Skyband πολλαπλές φορές θα οδηγήσει στην αντικειμενικά αυξημένη ανάγκη ύπαρξης των αντίστοιχων υπολογιστικών πόρων προς διάθεση, ούτως ώστε να προσδιοριστούν οι αναγκαίες πραγματικές εγγραφές. Σε αντίθεση, σε ό,τι αφορά την τροποποίηση του BNL για τον υπολογισμό της περιορισμένης κορυφογραμμής, η παραπάνω προσέγγιση οδηγεί στο σύνολο βέλτιστων πραγματικών εγγραφών μέσω μοναδικής εκτέλεση του προγράμματος. Αυτό οφείλεται στο ότι τα αναξιόπιστα στοιχεία απορρίπτονται σε πρώιμο στάδιο του αλγορίθμου, με αποτέλεσμα κατά την διάρκεια της επιστροφής των εκροών τα μόνα υποψήφια για την κορυφογραμμή στοιχεία είναι όλα έγκυρα, σύμφωνα πάντα με τους περιορισμούς που έχουν τεθεί. Πρέπει σημειωθεί πως κάθε μια επιστρεφόμενη εγγραφή των στοιχείων περιορισμένης κορυφογραμμής είναι έγκυρη καθώς εμπίπτει στους περιορισμούς, σε αντίθεση με το επιστρεφόμενο σύνολο εγγραφών σε k-Skyband που αν και περιέχει έγκυρες εγγραφές, αναγκαστικά προηγούνται, για μικρότερα k, οι μη ρεαλιστικές εγγραφές εκείνες που πρέπει να απορριφθούν. Παρόλα αυτά πρέπει να τονιστεί η υποκειμενική φύση των εκροών της περιορισμένης κορυφογραμμής δεδομένων των περιορισμών που τίθενται. Μια λανθασμένη επιλογή των περιορισμών αυτών θα μπορούσε να οδηγήσει σε πιθανό αποκλεισμό έγκυρων εγγραφών με σαφείς επιπτώσεις στην διαμόρφωση του τελικού αποτελέσματος της κορυφογραμμής. Επίσης η επιλογή άτοπων περιορισμών θα μπορούσε να αγνοήσει εγγραφές που αναγκαίο είναι να αποκλειστούν για την αντικειμενική παραγωγή του αποτελέσματος της κορυφογραμμής. Συνεπώς κρίνεται απαραίτητη η επιλογή των σωστών περιορισμών εκείνων που θα

επιφέρουν τον αποκλεισμό αμφισβητήσιμων στοιχείων και θα οδηγήσουν στις επιθυμητές εκροές κορυφογραμμής της τροποποίησης πάνω σε Constrained Skyline.

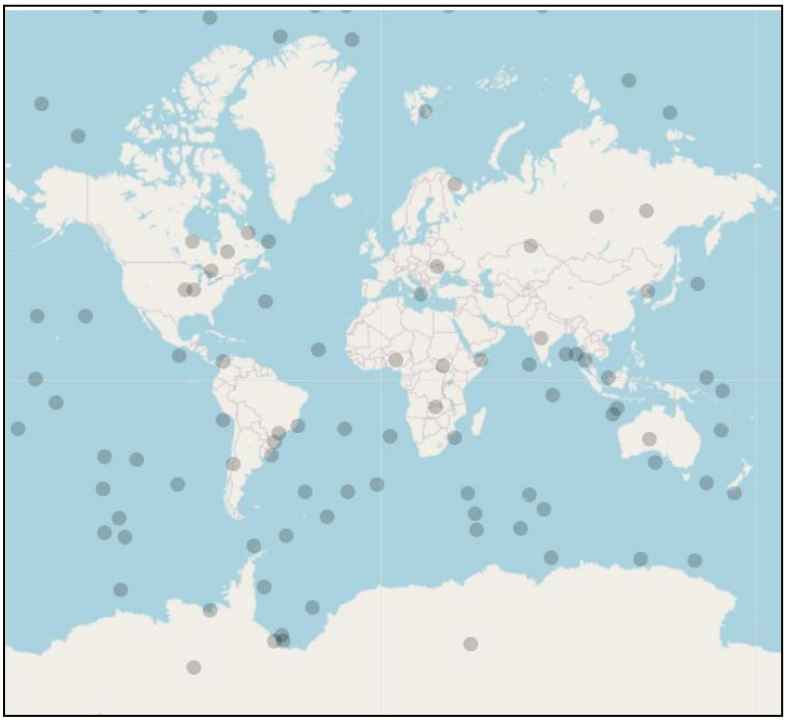
6 *ΠΕΙΡΑΜΑΤΙΚΗ ΑΝΑΛΥΣΗ: ΠΕΡΙΠΤΩΣΗ ΜΕ ΑΠΟΣΤΑΣΗ*

6.1 *Εισαγωγή συνιστώσας απόστασης*

Η δεύτερη επερώτηση θα διενεργηθεί με τον εξής τρόπο. Τα σημαντικά πεδία θα είναι η φθίνουσα τιμή, το αύξον έτος, ο φθίνων αριθμός χιλιομέτρων που το όχημα έχει διανύσει καθώς και ένα νέο πεδίο που θα αποτελεί την απόσταση σε χιλιόμετρα της κάθε εγγραφής από σημείο που δίνει ο χρήστης. Σκοπός είναι η μελέτη της συμπεριφοράς του αλγορίθμου BNL σε ό,τι αφορά τους υπολογιστικούς πόρους που θα δαπανηθούν σε περίπτωση μεγάλου πλήθους επερωτήσεων. Για τον λόγο αυτό, αρχικά θα δημιουργηθεί ένα αρχείο επέκτασης .csv που θα περιέχει τις συντεταγμένες γεωγραφικού μήκους και πλάτους εκατό σημείων. Τα σημεία αυτά θα αποτελούν την εισροή του χρήστη. Κατόπιν, για κάθε σημείο θα εκτελεστεί επερώτημα κορυφογραμμής BNL λαμβάνοντας υπόψιν το κόστος σε χρόνο και χώρο για κάθε ένα από αυτά τα επιμέρους επερωτήματα καθώς και το μήκος της κορυφογραμμής κάθε επερωτήματος.

#	lat	long	#	lat	long	#	lat	long	#	lat	long
1	-18.73178389	-75.72662508	26	-59.79160364	-123.0617655	51	-62.05354145	-61.1461993	76	87.98876085	-31.91092222
2	9.503745388	7.171379786	27	-77.15992072	-47.398393	52	12.45025388	88.5483234	77	-57.64037716	66.5306671
3	68.0325989	35.32057231	28	-26.35126402	34.86951583	53	80.72845886	118.7695711	78	-73.42524456	-82.00165315
4	39.61553527	128.1664171	29	-64.47777489	81.611725	54	7.756619706	71.34542636	79	41.93938305	151.8665923
5	-6.900377867	82.24777615	30	11.73900014	-96.75032322	55	-69.84506296	-56.02967426	80	-34.18039537	-132.9686656
6	86.8274878	77.66119316	31	54.33501867	71.39895637	56	9.773707581	47.74617236	81	-72.99903355	-33.01317273
7	-84.51180779	-149.364253	32	-54.46662496	-26.29818281	57	-28.38666594	-51.18703313	82	46.37062367	-81.76892604
8	35.43296013	-55.87264539	33	-58.249479	45.56226895	58	-76.55562855	-47.65568354	83	-22.92134593	163.0167208
9	48.02012833	26.62326585	34	29.30051874	-142.1246306	59	-47.13893412	-36.52898229	84	-10.25084782	-156.4161914
10	87.89106854	-136.5677644	35	9.132261849	-76.27224508	60	14.50355829	-30.02372101	85	52.39050869	-73.74656141
11	-47.01977055	-16.49908983	36	7.003876447	29.70017907	61	89.79164615	32.64540231	86	77.98564418	138.1791806
12	-53.82733275	44.95918094	37	-37.23626354	-71.27588188	62	12.30389306	93.33446929	87	-13.12116517	113.2095354
13	-26.02380117	3.963201167	38	-52.56232308	78.11566023	63	78.75164446	-162.8477564	88	75.39673103	-145.6666886
14	-64.75144422	124.6981062	39	-47.61447259	169.6488812	64	-12.25622379	26.14805911	89	-77.09162495	-51.66654175
15	83.42335341	-14.25826441	40	-77.39426496	42.74087043	65	-36.78206536	131.1635298	90	87.43603795	-17.0756794
16	1.578083145	156.4391798	41	-44.09039195	155.8278978	66	38.36080585	19.00535629	91	-54.96546342	-125.5921672
17	-47.75578058	70.73437416	42	-58.94788354	-132.9314852	67	-65.06616149	150.1274557	92	-26.74852666	128.8825741
18	-79.68814008	-90.33098407	43	39.98456331	-90.16017511	68	-46.30222391	-133.4335058	93	-47.39008973	41.55378968
19	-21.00774065	-40.10480784	44	63.02833635	127.2086435	69	61.53027787	103.4425555	94	55.31235201	-54.40150562
20	55.19172497	-90.95372608	45	-22.60223809	-17.78343352	70	84.51451563	-81.99133358	95	-83.46316438	-135.0378379
21	-24.39021335	-48.98952855	46	-33.61360458	-53.04124232	71	-16.07849485	111.0464094	96	83.56499169	-48.17454648
22	-35.67805121	-117.4764103	47	-44.6741805	-1.88461048	72	10.01796835	97.98266263	97	57.53971268	-63.97639232
23	1.166372052	108.9009209	48	39.69026741	-94.34093096	73	19.82905288	76.54916971	98	87.03855095	-114.3409596
24	78.03214717	21.35311637	49	-70.45050239	-125.3377387	74	-59.60070122	-45.68245006	99	-5.018312681	164.0478359
25	-22.2717117	-174.210466	50	-44.64029452	-97.45751254	75	0.842341501	-165.6748225	100	29.38868839	-165.0021454

Πίνακας 18 Τα 100 σημεία συντεταγμένων που θα χρησιμοποιηθούν ως εισροή στην δεύτερη επερώτηση



Εικόνα 32 Αναπαράσταση στον χάρτη των 100 σημείων συντεταγμένων που θα χρησιμοποιηθούν ως εισροής στην δεύτερη επερώτηση

Για την αποτελεσματική εκτέλεση της παραπάνω διαδικασίας, θα υπάρξουν ορισμένες προσθήκες στο πρόγραμμα σε Java του BNL της §3.3. Αρχικά σε ό,τι αφορά την κύρια κλάση BNL η οδηγός μέθοδος main θα υποστεί τροποποιήσεις που θα διαχειρίζονται την ανάγνωση και την προσπέλαση του αρχείου επέκτασης .csv του συνόλου των 100 σημείων εισροής. Επίσης το κομμάτι του κώδικα που αφορά την εκτέλεση του αλγορίθμου BNL θα μεταφερθεί σε ξεχωριστή μέθοδο ούτως ώστε να είναι δυνατή η εφαρμογή πληθώρας επερωτημάτων για κάθε σημείο ξεχωριστά.

6.2 *Υλοποίηση σε Java*

```
12  public static void main(String[] args) {
13      XXFile file_dataset = new XXFile();
14      file_dataset.openWFile("C:\\Users\\George\\Desktop\\THESIS\\src\\0Dataset\\timesQ2.csv");
15      CoordsCsvSplitter LLsplit=new CoordsCsvSplitter("C:\\Users\\George\\Desktop\\THESIS\\src\\0Dataset\\LATLONGINPUT.csv");
16      LLsplit.startProcess();
17      ArrayList<Float[]> LatLongInput= LLsplit.CoordsSplit();
18      for (Float[] line : LatLongInput) {
19          double startTime,startSpace,elapsedTimeMillis,usedSpace;
20          ArrayList<Integer> result;
21          System.gc();
22          startTime = System.currentTimeMillis();
23          startSpace=Runtime.getRuntime().totalMemory()-Runtime.getRuntime().freeMemory();
24          result=RunBNL(line[0],line[1]);
25          elapsedTimeMillis = System.currentTimeMillis()-startTime;
26          usedSpace=(Runtime.getRuntime().totalMemory()-Runtime.getRuntime().freeMemory()-startSpace)/(1024.0 * 1024.0* 1024.0);
27          file_dataset.writeLine(line[0]+","+line[1]+","+elapsedTimeMillis+","+usedSpace + ","+result.size() );
28      }
29      file_dataset.closeWFile();
30      LLsplit.endProcess();
31  }
```

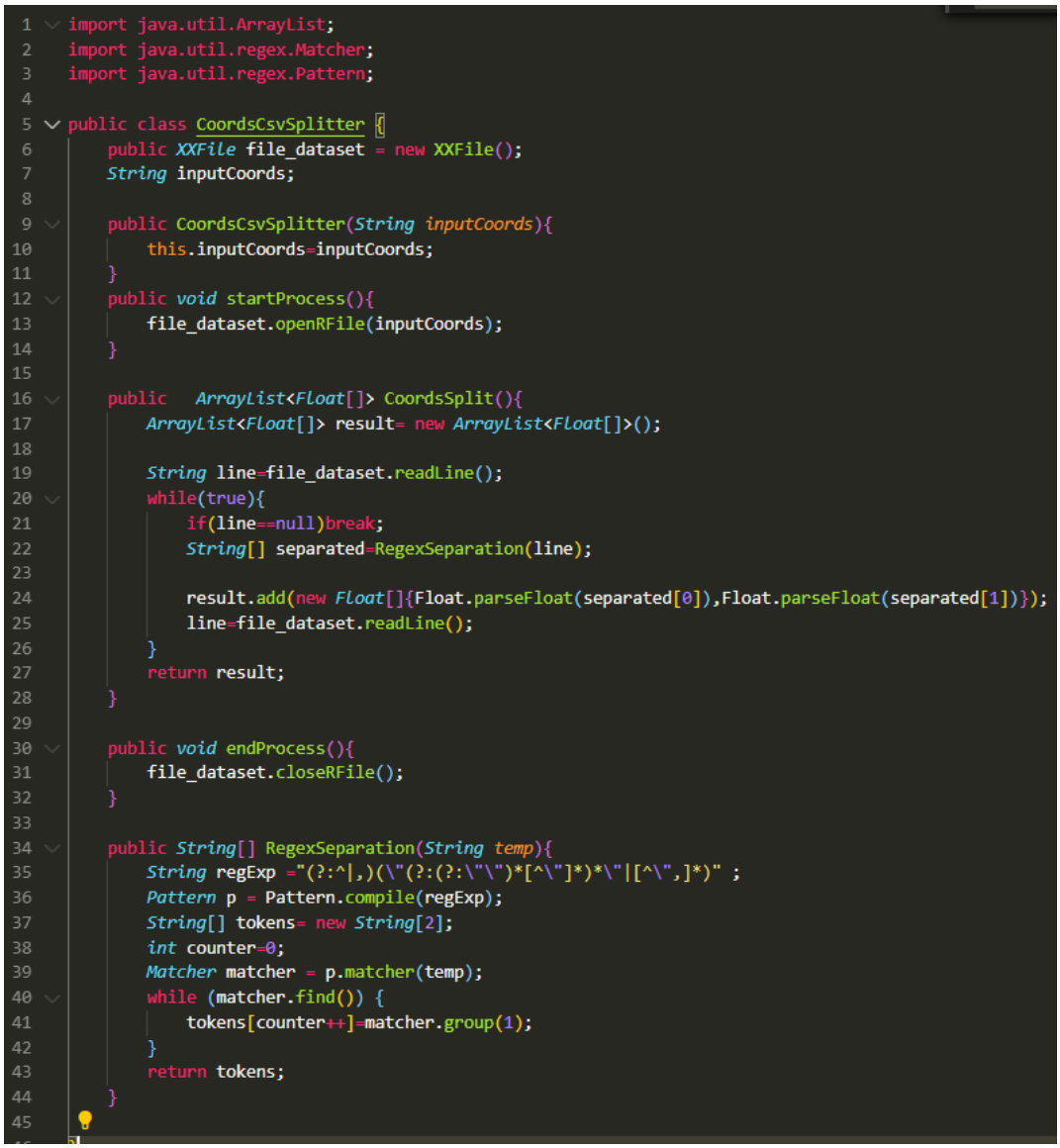
Η τροποποιημένη οδηγός μέθοδος main της κύριας κλάσης BNL στη σειρά 13 δημιουργείται αντικείμενο της κλάσης XXFile της §2.3.3 με σκοπό να οριστεί η τοποθεσία εκροών δοθείσης διεύθυνσης ενός εγγράψιμου αρχείου στην επόμενη σειρά. Στην σειρά 15 υλοποιεί αντικείμενο της νέας κλάσης CoordsCsvSplitter η οποία δοθέντος αλφαριθμητικού που αντιστοιχεί στο μονοπάτι του συστήματος όπου εντοπίζεται το αρχείο εισροών συντεταγμένων επέκτασης .csv, το διαβάζει και το προσπελαύνει. Κατόπιν στη σειρά 17, μέσω μεθόδου της κλάσης CoordsCsvSplitter, αποθηκεύονται στην σειρά σειρών δεκαδικών αριθμών LatLongInput όλα τα σημεία εισροής, καθένα από τα οποία καθορίζεται από 2 συντεταγμένες. Εντός του βρόγχου for των σειρών 18-28, για κάθε σημείο της μεταβλητής LatLongInput, καλείται η νεοεισαχθείσα μέθοδος RunBnl, στην οποία δίνεται ως εισροή το σύνολο των δυο συντεταγμένων του εκάστοτε σημείου εισροής. Η μέθοδος αυτή θα εκτελέσει τον BNL και θα επιστρέψει το σύνολο των αναγνωριστικών αριθμών id των σημείων της κορυφογραμμής. Οι σειρές 19 και 20 αρχικοποιούν τις μεταβλητές που πρόκειται να χρησιμοποιηθούν στον εντός του βρόγχου. Οι μεταβλητές startTime,startSpace,elapsedTimeMillis και usedSpace δεκαδικών αριθμών θα αποθηκεύσουν την λογική υπολογισμού των δαπανηθέντων υπολογιστικών πόρων. Η σειρά 21 καλεί την διαδικασία garbage collection που σκοπό έχει την αντικειμενική μέτρηση των υπολογιστικών πόρων που θα χρησιμοποιηθούν κατά την διαδικασία προσδιορισμού της κορυφογραμμής του αντίστοιχου σημείου συντεταγμένων. Οι σειρές 22, 23, 26,27 είναι πανομοιότυπες με λογική που έχει αναλυθεί στην §3.3.1 και σκοπό έχουν τον προσδιορισμό των υπολογιστικών πόρων χωροχρόνου που δαπανήθηκαν κατά την εφαρμογή του κάθε επερωτήματος BNL. Τέλος στη σειρά 27 τα αποτελέσματα χρόνου, χώρου και πλήθους σημείου κορυφογραμμής εγγράφονται στο αρχείο εκροών.

```
32    public static Boolean isDominatedBy(EntryLine A,EntryLine B){
33        return B.price<=A.price && B.odometer<=A.odometer && B.year>=A.year && B.DistanceFromInput<=A.DistanceFromInput
34        && (B.price<A.price||B.odometer<A.odometer||B.year>A.year ||B.DistanceFromInput<A.DistanceFromInput);
35    }
```

Σε ό,τι αφορά την μέθοδο isDominatedBy, πλέον για κάθε εγγραφή λαμβάνεται υπόψιν και μια νέα ιδιότητα κάθε αντικειμένου εγγραφής EntryLine, η DistanceFromInput. Αυτή αποτελεί την απόσταση του σημείου στο οποίο συγκλίνουν οι συντεταγμένες κάθε εγγραφής από καθένα από τα σημεία συντεταγμένων εισροής του νέου αρχείου επέκτασης .csv.

```
70    public static ArrayList<Integer> RunBNL(Float pointX, Float pointY ){
71        String path= "C:\\Users\\George\\Desktop\\THESIS\\src\\0Dataset\\", filename= path+  "PreProcessingResult.csv";//
72        CsvCacher cache=new CsvCacher(filename,pointX,pointY);
73        String newCache=path+"newCache.txt", TempPath=path+"temp.txt";
74        int TempFileEntries=0,CacheEntriesCount=0,maxException=40000;
75        Boolean mainFinished=false,switcher=true,isDominated=false, writeTempOnly=false;
76        ArrayList<Integer> resultSkyLine=new ArrayList<>();
77        ArrayList<EntryLine> window= new ArrayList<EntryLine>();
78        EntryLine newEntry;
79  >         try{ ···
182           catch (FileNotFoundException e) {
183               System.out.println("Does not exist");
184           }
185           catch (IOException e) {
186               System.out.println("I/O Issdsdsue");
187               e.printStackTrace();
188           }
189           catch (ClassNotFoundException e) {
190               e.printStackTrace();
191           }
192
193           return resultSkyLine;
194   }
195
196   }
```

Η μέθοδος RunBnl, είναι η αφηρημένη εκδοχή της πρότερης διαδικασίας υπολογισμού κορυφογραμμής μέσα στην συνάρτηση main, στην §3.3.1. Οι μόνες διαφορές αφορούν τις συντεταγμένες pointX και pointY που δίνονται ως εισροή και λαμβάνονται υπόψιν σε κάθε αρχικοποίηση των εγγραφών στην σειρά 72. Οι υπόλοιπες σειρές πλήν της 193 είναι ταυτόσημες με αυτές της §3.3.1. Στην σειρά 193, η μέθοδος επιστρέφει το αποτέλεσμα των αναγνωριστικών αριθμών των εγγραφών που ανήκουν στην κορυφογραμμή.



```java
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class CoordsCsvSplitter {
    public XXFile file_dataset = new XXFile();
    String inputCoords;

    public CoordsCsvSplitter(String inputCoords){
        this.inputCoords=inputCoords;
    }
    public void startProcess(){
        file_dataset.openRFile(inputCoords);
    }

    public   ArrayList<Float[]> CoordsSplit(){
        ArrayList<Float[]> result= new ArrayList<Float[]>();

        String line=file_dataset.readLine();
        while(true){
            if(line==null)break;
            String[] separated=RegexSeparation(line);

            result.add(new Float[]{Float.parseFloat(separated[0]),Float.parseFloat(separated[1])});
            line=file_dataset.readLine();
        }
        return result;
    }

    public void endProcess(){
        file_dataset.closeRFile();
    }

    public String[] RegexSeparation(String temp){
        String regExp ="(?:^|,)(\"(?:(?:\"\")*[^\"]*)*\"|[^\",]*)" ;
        Pattern p = Pattern.compile(regExp);
        String[] tokens= new String[2];
        int counter=0;
        Matcher matcher = p.matcher(temp);
        while (matcher.find()) {
            tokens[counter++]=matcher.group(1);
        }
        return tokens;
    }

}
```

Η νεοεισαχθείσα κλάση CoordsCsvSplitter σκοπό έχει την ανάγνωση και προσπέλαση του αρχείο εισροών συντεταγμένων 100 σημείων που προαναφέρθηκαν. Η κλάση δανείζεται την πληθώρα της λογικής της από την κλάση CsvCacher της §3.3.2, εφόσον 4 από τις 5 μεθόδους τους είναι ταυτόσημες. Σε ό,τι αφορά την μέθοδο CoordsSplit, αυτή θα επιστρέψει το ζητούμενο της κλάσης, μια σειρά με δυάδες δεκαδικών αριθμών, κάθεμια από τις οποίες περιλαμβάνει το γεωγραφικό μήκος και πλάτος της κάθε εγγραφής. Έτσι στη σειρά 17 αρχικοποιείται η σειρά δυάδων result που θα αποτελέσει την εκροή της μεθόδου CoordsSplit και στον βρόγχο while των σειρών 20-26 διαβάζει σειρά-σειρά το αρχείο επέκτασης .csv και αποθηκεύει στην μνήμη την δυάδα του εκάστοτε σημείου. Εν τέλει στην σειρά 27 επιστρέφεται η σειρά των 100 δυάδων.

Σε ό,τι αφορά την κλάση CsvCacher, αυτή υπέστη μερική τροποποίηση ούτως ώστε κατά την προσπέλαση κάθε επιμέρους αντικειμένου εγγραφής EntryLine να δημιουργείται και μια επιπλέον ιδιότητα, αυτή της απόστασης του σημείου που υποδηλώνουν τα πεδία lat και long από το καθένα από τα 100 σημεία εισροής συντεταγμένων.

```
15      public CsvCacher(String inputFile, double pX,double pY){
16          this.inputFile=inputFile;
17          this.pX=pX;
18          this.pY=pY;
19      }
```

Αρχικά η κατασκευαστική μέθοδος της κλάσης δέχεται επιπλέον δυο δεκαδικούς αριθμούς pX και pY που θα αφορούν το γεωγραφικό πλάτος και μήκος καθενός από τα 100 σημεία του νέου αρχείου εισροής συντεταγμένων επέκτασης .csv και θα τα αποθηκεύει στις εγγενείς της κλάσης μεταβλητές με το ίδιο όνομα. Οι μέθοδοι startProcess, nextEntry, endProcess και RegexSeparation είναι ταυτόσημες και δεν έχουν υποστεί αλλαγή.

```
108         double distanceFromInput=Double.MAX_VALUE;
109         if(lat!=defaultDominated[5]&&dlong!=defaultDominated[6]){
110             distanceFromInput=getDistance(lat, dlong, pX, pY);
111         }
```

Σε ό,τι αφορά την μέθοδο createLine, πριν την επιστροφή του ζητούμενου αντικειμένου εγγραφής EntryLine, υπολογίζεται η τιμή της νέας μεταβλητής DistanceFromInput που περιγράφει την απόσταση σε χιλιόμετρα της εγγραφής από το καθένα από τα 100 σημεία του αρχείου εισροών συντεταγμένων. Στη σειρά 108, η μεταβλητή αρχικοποιείται με τη μέγιστη τιμή, την οποία θα διατηρήσει εφόσον είναι κενό κάποια από τα πεδία lat και long, μέσω του βρόγχου if των σειρών 109-111. Σε περίπτωση που οι τιμές των πεδίων lat και long είναι υπαρκτές, η σειρά 110 καλώντας την νεοεισαχθείσα μέθοδο getDistance, αποθηκεύει στην μεταβλητή distanceFromInput την ζητούμενη απόσταση.

```
return  new EntryLine(id,city,price,year,odometer,lat,dlong,manufacturer,make,condition,cylinders,fuel,title_status,transmission,drive,size,
type,paint_color,distanceFromInput);
```

Τέλος, μέσω της κλάσης EntryLine, η μέθοδος createLine επιστρέφει το αντίστοιχο αντικείμενο εγγραφής περιλαμβάνοντας ως εισροή και την τιμή της νέας μεταβλητής distanceFromInput.

Επιπροσθέτως, δυο νέες μέθοδοι έχουν δημιουργηθεί εντός της κλάσης CsvCacher για τον υπολογισμό της απόστασης σε χιλιόμετρα ανάμεσα στο σημείο που υποδεικνύουν οι τιμές των πεδίων lat και long της κάθε εγγραφής και καθενός από τα 100 σημεία του νέου αρχείου εισροών συντεταγμένων.

```
142     public double DegreesToGrad(double deg){
143         return deg*(Math.PI)/180;
144     }
```

Η πρώτη μέθοδος είναι αποτελεί ένα μαθηματικό εργαλείο που δοθέντος δεκαδικού θα επιστρέψει την τιμή του σε ακτίνια. Αυτό θα γίνει μέσω του πολλαπλασιασμού του αριθμού με το π και την διαίρεση του αποτελέσματος με 180 όπως δείχνει και η σειρά 143.

```
146     public double getDistance(double pX,double pY,double eX,double eY){
147         int R= 6371003;
148         double f1 = DegreesToGrad(eX);
149         double f2 = DegreesToGrad(pX);
150         double DF = DegreesToGrad(pX-eX);
151         double DL = DegreesToGrad(pY-eY);
152         double a = Math.sin(DF/2) * Math.sin(DF/2) +
153                 Math.cos(f1) * Math.cos(f2) *
154                 Math.sin(DL/2) * Math.sin(DL/2);
155         double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
156         return R*c*(Math.pow(10, -3));
157     }
```

Η δεύτερη μέθοδος είναι η getDistance που δοθέντων των δύο ζευγών γεωγραφικού μήκους και πλάτους σημείων θα επιστρέψει την απόσταση σε χιλιόμετρα ανάμεσα τους. Για να επιτευχθεί χρησιμοποιείται ο μαθηματικός τύπος Haversine που υπολογίζει, μέσω του νόμου των συνημιτόνων, την απόσταση σημείων επι σφαιρικής επιφάνειας. Παράμετροι που χρησιμοποιούνται στον τύπο Haversine είναι τα δυο ζεύγη γεωγραφικού πλάτους και μήκους καθώς και η ακτίνα της γής σε μέτρα.

$$D = 2 * R * \arcsin\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1) * \cos(\varphi_2) * sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Τύπος 1 Η φόρμουλα Haversine για τον υπολογισμό απόστασης σημείων επι σφαιρικής επιφάνειας μέσω των συντεταγμένων τους και της ακτίνας της Γης

```
12      public double DistanceFromInput;
13      public double dlong,lat;
14
15      EntryLine(Integer  id,String city,  Long Price, Integer  Year, long Odometer ,
16      double dlong, double lat, String manufacturer, String make, String condition, Integer cylinders, String fuel, String title_status, String
        transmission, String drive, String size, String type,String paint_color,double DistanceFromInput){
17          this.id=id;
18          this.year=Year;
19          this.price=Price;
20          this.odometer=Odometer;
21          this.dlong=dlong;
22          this.lat= lat;
23          this.manufacturer=manufacturer;
24          this.make=make;
25          this.condition=condition;
26          this.cylinders=cylinders;
27          this.fuel=fuel;
28          this.title_status=title_status;
29          this.transmission=transmission;
30          this.drive=drive;
31          this.size=size;
32          this.type=type;
33          this.paint_color=paint_color;
34          this.DistanceFromInput=DistanceFromInput;
35      }
```

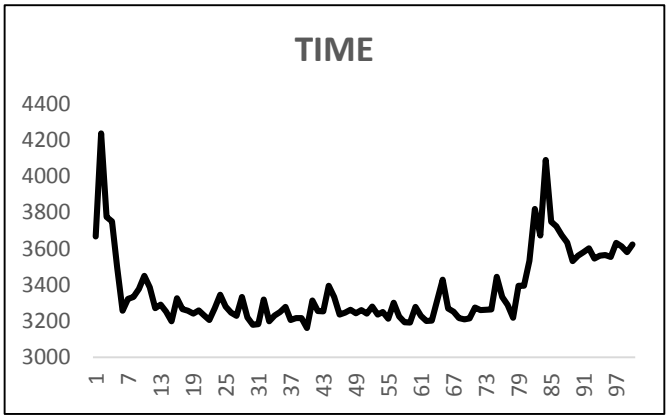
Τέλος σε ό,τι αφορά την κλάση EntryLine αντικειμένου εγγραφής, η κατασκευαστική μέθοδος πλέον λαμβάνει επιπλέον μεταβλητή δεκαδικής τιμής DistanceFromInput ως εισροή την οποία και αποθηκεύει σε εγγενή μεταβλητή ίδιας ονομασίας. Η τιμή της μεταβλητής αυτής αποτελεί την απόσταση ανάμεσα στο σημείο των συντεταγμένων της εγγραφής και το εκάστοτε σημείο του αρχείου επέκτασης .csv εισροών συντεταγμένων.

6.3 *Πειραματικά δεδομένα*

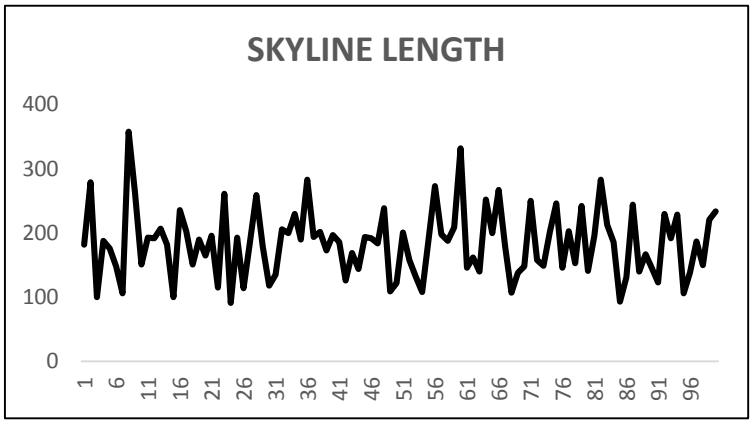
Η εκτέλεση του κώδικα της υλοποιημένης διαδικασίας των εκατό σημείων δημιουργεί αρχείο εκροών επέκτασης .csv των σημείων αυτών δίπλα στο καθένα από τα οποία εμφανίζεται ο χρόνος και ο δεσμευμένος από την RAM χώρος που δαπανήθηκαν για τον υπολογισμό της κορυφογραμμής με παράγοντα και την απόσταση από το αντίστοιχο σημείο συντεταγμένων, καθώς και το μήκος της κορυφογραμμής. Μετά από την χρήση του αρχείου 100 σημείων τυχαίων συντεταγμένων που περιεγράφηκε στην §5.1 ως εισροή, το αρχείο εκροών αποτελείται από εκατό σειρές , αντίστοιχα, με τα ζητούμενα πειραματικά δεδομένα για κάθε επερώτηση κορυφογραμμής BNL πάνω στο κάθε ένα σημείο συντεταγμένων. Πιο συγκεκριμένα, σε ό,τι αφορά τον δαπανώμενο χρόνο, η μέση τιμή των 100 αντίστοιχων χρόνων διαμορφώνεται σε 3369.77ms ενώ η διάμεσος στα 3276ms. Μέγιστη τιμή σημειώνονται τα 4239ms ενώ ελάχιστη τιμή τα 3162ms, διαμορφώνοντας εύρος της τάξης των 1077ms. Το μήκος των σημείων κορυφογραμμής παρουσιάζει μέση τιμή τα 183.85 σημεία και διάμεσο τα 186.5 σημεία . Η μεγαλύτερη σε μήκος κορυφογραμμή αποτελείται από 358 σημεία ενώ η μικρότερη από 91, σχηματίζοντας έτσι εύρος 267 σημείων. Τέλος σε ό,τι αφορά τον δαπανηθέντα χώρο, μέση τιμή του χώρου των 100 σημείων είναι τα 0.068354141GB και η διάμεσος τα 0.068086248. Ο μέγιστος χώρος σημειώθηκε σε 0.132457055GB, ενώ ο ελάχιστος ανέρχεται σε 0.003577873GB. Πρέπει ωστόσο να σημειωθεί πως οι παραπάνω μετρήσεις των σημαντικών μεγεθών του χώρου και χρόνου αφορούν την μέτρηση αμέσως πριν και μετά τον υπολογισμό της κορυφογραμμής για καθένα από τα εκατό σημεία συντεταγμένων. Επιπροσθέτως, στα παραπάνω μεγέθη συμπεριλαμβάνονται και οι πόροι που δαπανήθηκαν για την προσπέλαση του βασικού αρχείου εισροών των 689600 εγγραφών αγοραπωλησιών οχημάτων μέσω της κλάσης διαχωρισμού CsvCacher της 2.3.2.

	TIME(ms)	SPACE(gb)	LENGTH(pts)
MAX	4239	0.132457	358
MIN	3162	0.003578	91
MEAN	3369.77	0.068354	183.85
MEDIAN	3276	0.068086	186.5

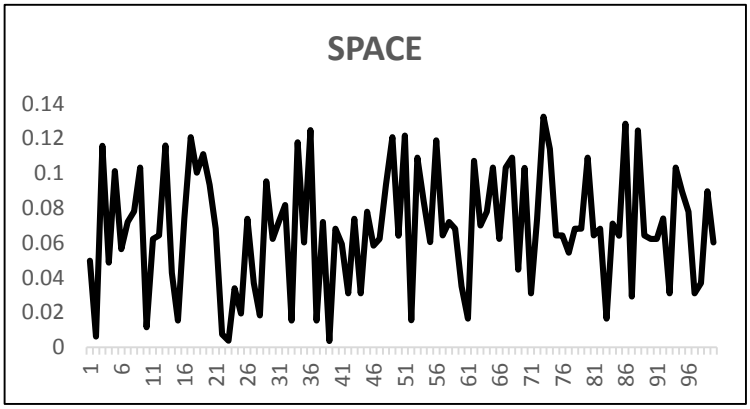
Πίνακας 19 Στατιστικά μετρικών των 100 επερωτημάτων BNL



Εικόνα 33 Διάγραμμα χρόνου περάτωσης των 100 επερωτημάτων BNL



Εικόνα 34 Διάγραμμα μήκους της κορυφογραμμής των 100 επερωτημάτων BNL



Εικόνα 34 Διάγραμμα δεσμευμένου χώρου κατά την εφαρμογή των 100 επερωτημάτων BNL

6.4 *Συμπεράσματα*

Σε ό,τι αφορά τον χώρο και τον χρόνο που δαπανώνται για την περάτωση κάθε επερωτήματος BNL, δεν φαίνεται να επηρεάζονται σημαντικά από την εισαγωγή του νέου παράγοντα απόστασης της κάθε εγγραφής από το εκάστοτε τυχαίο σημείο συντεταγμένων. Παρόλα αυτά σε σύγκριση με το πρώτο επερώτημα BNL, ο μέσος χρόνος αυξήθηκε περί των 200ms και εντός της κορυφογραμμής απαντώνται κατά πολύ περισσότερα σημεία, δεδομένου του μήκους της κορυφογραμμής των τριών έκτοπων εγγραφών κατά το πρώτο επερώτημα BNL. Αυτό συνεπάγεται πως οι έκτοπες εγγραφές που κατά το πρώτο επερώτημα BNL επηρέασαν την εγκυρότητα των αποτελεσμάτων, τώρα έχουν περιορισμένο έλεγχο επάνω στις υπόλοιπες εγγραφές λόγω του αδιαμφισβήτητου παράγοντα της

απόστασης. Ως αποτέλεσμα, αυξάνεται η αξιοπιστία των εκροών μαζί με το μέγεθος της κορυφογραμμής δεδομένης της προσθήκης του αντικειμενικού παράγοντα απόστασης.

7 ΠΕΙΡΑΜΑΤΙΚΗ ΑΝΑΛΥΣΗ: ΠΕΡΙΠΤΩΣΗ ΜΕ ΚΕΙΜΕΝΟ

7.1 *Εισαγωγή συνιστώσας ομοιότητας κειμένου*

Η τρίτη επερώτηση θα αφορά την εφαρμογή του αλγορίθμου BNL για την εύρεσης της κορυφογραμμής του συνόλου δεδομένων βάσει 5 παραγόντων. Οι παράγοντες αυτοί θα είναι η αύξουσα τιμή, το αύξον έτος κυκλοφορίας, ο φθίνων αριθμός χιλιομέτρων, η μικρότερη απόσταση από δοθέν σημείο συντεταγμένων, και ο αύξων αριθμός ενός νέου μεγέθους κατάταξης, βάσει της εμφάνισης ή όχι συγκεκριμένου κειμένου στα πεδία της κάθε εγγραφής. Πιο συγκεκριμένα, δοθέντων λέξεων- κλειδιών που παρέχει ο χρήστης, θα δημιουργηθεί ένα καινούργιο μέγεθος που θα αντιπροσωπεύει το πλήθος των λέξεων που περιλαμβάνονται σε κάποιο από τα πεδία αλφαριθμητικών Manufacturer, Make, Condition, Cylinders, Fuel, Title_status, Transmission, Drive, Size, Type, Paint_color. Το μέγεθος αυτό θα ονομαστεί kwFreq και θα ορίζεται ως εξής:

$$kwFreq = \frac{M}{N}\ , N > 0$$

Τύπος 2 Μέγεθος που χαρακτηρίζει ομοιότητα ανάμεσα σε εγγραφές και σε δοθέν σύνολο λέξεων κλειδιών

Όπου M είναι το πλήθος των λέξεων-κλειδιών που εμφανίζονται τουλάχιστον μια φορά στα δοθέντα πεδία αλφαριθμητικών της συγκεκριμένης εγγραφής, ενώ το N αποτελεί το πλήθος των λέξεων-κλειδιών που δίνει ο χρήστης. Συνεπώς το kwFreq κυμαίνεται στο διάστημα [0,1] με την τιμή 1 να αντικατοπτρίζει την ύπαρξη όλων των λέξεων κλειδιών στα σημαντικά πεδία κειμένου της αντίστοιχης εγγραφής ενώ την τιμή 0, την έλλειψη ίδιων λέξεων ανάμεσα στα σημαντικά πεδία του κειμένου της εγγραφής και του συνόλου των λέξεων- κλειδιών που δίνεται από τον χρήστη.

7.2 *Υλοποίηση σε Java*

Δεδομένου του ό,τι ο τρόπος με τον οποίον θα διενεργηθούν τα επερωτήματα BNL περιλαμβάνει την χρήση των 100 σημείων εισροής συντεταγμένων που πρωτοπαρουσιάστηκε στην §6.1, η υλοποίηση σε java θα γίνει μέσω τροποποιήσεων στο σύνολο του κώδικα της παραγράφου αυτής. Σκοπός είναι η

αποτελεσματική διαχείριση της λογικής της νέας εισροής των λέξεων κλειδιών του χρήστη και η συμπερίληψη τους στην διαμόρφωση του τελικού αποτελέσματος της κορυφογραμμής.

```
13  public static void main(String[] args) {
14      XXFile file_dataset = new XXFile();
15      file_dataset.openWFile("C:\\Users\\George\\Desktop\\THESIS\\src\\0Dataset\\NtimesQ320KW.csv");
16
17      String[] keywords=new String[]{};
18
19
20      CoordsCsvSplitter LLsplit=new CoordsCsvSplitter("C:\\Users\\George\\Desktop\\THESIS\\src\\0Dataset\\LATLONGINPUT.csv");
21      LLsplit.startProcess();
22      double maxTime=-1,minTime=500000,maxSpace=-1,minSpace=689000,avgTime=0,avgSpace=0,minLen=689000,maxLen=-1,avgLen=0;
23      ArrayList<Float[]> LatLongInput= LLsplit.CoordsSplit();
24      file_dataset.writeLine("LAT,LONG,TIME,SPACE,LENGTH" );
25
26      for (Float[] line : LatLongInput) {
27          double startTime,startSpace,elapsedTimeMillis,usedSpace;
28          ArrayList<Integer> result;
29           System.gc();
30           startTime = System.currentTimeMillis();
31           startSpace=Runtime.getRuntime().totalMemory()-Runtime.getRuntime().freeMemory();
32           result=RunBNL(line[0],line[1],keywords);
33           elapsedTimeMillis = System.currentTimeMillis()-startTime;
34           usedSpace=(Runtime.getRuntime().totalMemory()-Runtime.getRuntime().freeMemory()-startSpace)/(1024.0 * 1024.0* 1024.0);
35          file_dataset.writeLine(line[0]+","+line[1]+","+elapsedTimeMillis+","+usedSpace + ","+result.size() );
36      }
37      file_dataset.closeWFile();
38      LLsplit.endProcess();
39  }
```

Σε ό,τι αφορά την κλάση BNL, η οδηγός μέθοδος main παρουσιάζει εισαγωγή μιας νέας σειράς στοιχείων κειμένου keywords στη σειρά 17 που αποτελούν την εισροή λέξεων κλειδιών του χρήστη. Την μεταβλητή αυτή θα χρησιμοποιήσει επίσης η λογική υπολογισμού κορυφογραμμής της σειράς 32 με τη μέθοδο RunBNL η οποία τροποποιείται κατάλληλά ούτως ώστε να δέχεται ως εισροή την σειρά αυτή. Τα υπόλοιπα στοιχεία της μεθόδου main παραμένουν αναλλοίωτα και στόχο έχουν τη δημιουργία αρχείου εκροών τέτοιο ώστε να παρουσιάζει για καθένα από τα 100 σημεία τον χρόνο και τον χρόνο που δαπανήθηκαν για τον υπολογισμό της αντίστοιχης κορυφογραμμής καθώς και το μέγεθος της.

```
41  public static Boolean isDominatedBy(EntryLine A,EntryLine B){
42      return B.price<=A.price && B.odometer<=A.odometer && B.year>=A.year && B.DistanceFromInput<=A.DistanceFromInput
43      &&B.kwFreq>=A.kwFreq &&(B.price<A.price||B.odometer<A.odometer||B.year>A.year ||B.DistanceFromInput<A.DistanceFromInput||B.kwFreq>A.kwFreq);
44  }
```

Η μέθοδος σύγκρισης αντικειμένων κορυφογραμμής isDominatedBy λαμβάνει υπόψιν το καινούργιο χαρακτηριστικό kwFreq κάθε στοιχείου εγγραφής, που αντιπροσωπεύει το πλήθος των υπαρκτών λέξεων κλειδιών στα πεδία κειμένου της εγγραφής αυτής. Συνεπώς μια εγγραφή A θα κυριαρχείται από κάποια εγγραφή B αν, επιπλέον, η τιμή του χαρακτηριστικού kwFreq της A είναι μικρότερη ή ίση της αντίστοιχης μεταβλητής του B.

```
77      public static ArrayList<Integer> RunBNL(Float pointX, Float pointY ,String[] keywords){
78          String path= "C:\\Users\\George\\Desktop\\THESIS\\src\\0Dataset\\", filename= path+
            "PreProcessingResult.csv";// "parsedSmall.csv" ; //
79          CsvCacher cache=new CsvCacher(filename,pointX,pointY,keywords);
```

Σε ό,τι αφορά τη μέθοδο RunBNL η μόνη τροποποίηση αφορά την εισροή πλειάδας στοιχείων κειμένου keywords η οποία και μεταβιβάζεται στην κλάση διαχωρισμού CsvCacher όπου και θα ακολουθήσει η επεξεργασία της με σκοπό τον υπολογισμό των κοινών στοιχείων με τα πεδία κειμένου της κάθε εγγραφής που η κλάση αυτή προσπελαύνει.

```
5   public class CsvCacher{
6       public Boolean isTemporary;
7       private String inputFile;
8       public XXFile file_dataset = new XXFile();
9       private String[] keywords;
10      private Integer[] defaultDominated={0 ,0,Integer.MAX_VALUE,0,Integer.MAX_VALUE,500,500};
11      public double pX,pY;
12      public CsvCacher(String inputFile, double pX,double pY, String[] keywords){
13          this.inputFile=inputFile;
14          this.pX=pX;
15          this.pY=pY;
16          this.keywords=keywords;
17      }
```

Η κλάση CsvCacher επιδέχεται τροποποιήσεις στην κατασκευαστική μέθοδο με το ίδιο όνομα καθώς και στη δημιουργίας αντικειμένου εγγραφής createLine. Πιο συγκεκριμένα αρχικοποιείται σειρά στοιχείων κειμένου keywords στη σειρά 9 και αντίστοιχη λογική εφαρμόζεται για την εισροή και θέση τους εντός της οδηγού μεθόδου CsvCacher.

```
105         double kwFreq=0;
106         if(keywords.length!=0){
107             for (int i = 0; i < keywords.length; i++) {
108                 for (int j = 7; j <=17; j++) {
109                     if(j!=8&&keywords[i].equals(separated[j])){
110                         kwFreq++;
111                         break;
112                     }
113                 }
114             }
115             kwFreq/=keywords.length;
116         }
```

Σε ό,τι αφορά τη μέθοδο createLine αρχικοποιείται με τιμή μηδέν στη σειρά 105 μεταβλητή δεκαδικών τιμών kwFreq, η οποία θα αποθηκεύει το πλήθος των λέξεων κλειδιών που εμφανίζονται στα πεδία κειμένου της αντίστοιχης εγγραφής. Για τον λόγο αυτό στις σειρές 106-116 εφόσον το πλήθος των δοθέντων λέξεων κλειδιών είναι θετικό, ο βρόγχος for της σειράς 107 συγκρίνει κάθε keyword με τα

αντίστοιχα πεδία κειμένου της εγγραφής, μέσω του βρόγχου for της σειράς 108. Σε περίπτωση που βρεθεί πεδίο που να περιέχει την εκάστοτε επεξεργαζόμενη λέξη κλειδί, η τιμή της μεταβλητής kwFreq αυξάνεται κατά μονάδα, και εξετάζεται η επόμενη λέξη κλειδί. Πρέπει επίσης να σημειωθεί πως δεδομένου του ότι τα σημαντικά πεδία κειμένου περιέχουν τιμές μονολεκτικές, οι λέξεις κλειδιά μπορούν είτε να ταιριάζουν εξ' ολοκλήρου με την τιμή του αντίστοιχου πεδίου, είτε όχι. Για τον λόγο αυτό στη σειρά 109 χρησιμοποιείται η συνάρτηση σύγκρισης κειμένου equals της java που επιστρέφει αληθή τιμή για την περίπτωση που τα προς σύγκριση κείμενα είναι πανομοιότυπα και ψευδή τιμή σε διαφορετική περίπτωση. Τέλος, σε περίπτωση ύπαρξης λέξεων κλειδιών, η σειρά 115 υπολογίζει την τιμή kwFreq διαιρώντας με το συνολικό πλήθος της σειράς στοιχείων κειμένου keywords, ούτως ώστε να συνάδει με τον ορισμό του αντίστοιχου μεγέθους της §6.1.

```
117        return  new EntryLine(id,city,price,year,odometer,lat,dlong,manufacturer,make,condition,cylinders,
           fuel,title_status,transmission,drive,size,type,paint_color,distanceFromInput,kwFreq);
```

Επιπροσθέτως, κατά την δημιουργία αντικειμένου εγγραφής μέσω της κλάσης EntryLine, προσφέρεται ως εισροή η νέα μεταβλητή kwFreq με σκοπό την πρόσβαση στην τιμή αυτή της κάθε εγγραφής κατά το στάδιο υπολογισμού της κορυφογραμμής και την σύγκριση των εγγραφών βάσει της τιμής αυτής.

```
12        public double DistanceFromInput,kwFreq;
```

```
15    EntryLine(Integer  id,String city,  Long Price, Integer  Year, Long Odometer ,
16    double dLong, double lat, String manufacturer, String make, String condition, String cylinders,
      String fuel, String title_status, String transmission, String drive, String size, String type,String
      paint_color,double DistanceFromInput,double kwFreq){
17        this.id=id;
18        this.year=Year;
19        this.price=Price;
20        this.odometer=Odometer;
21        this.dlong=dlong;
22        this.lat= lat;
23        this.manufacturer=manufacturer;
24        this.make=make;
25        this.condition=condition;
26        this.cylinders=cylinders;
27        this.fuel=fuel;
28        this.title_status=title_status;
29        this.transmission=transmission;
30        this.drive=drive;
31        this.size=size;
32        this.type=type;
33        this.paint_color=paint_color;
34        this.DistanceFromInput=DistanceFromInput;
35        this.kwFreq=kwFreq;
36    }
```

Σε ό,τι αφορά την κλάση αντικειμένων εγγραφής EntryLine, η τροποποίηση αφορά την αποθήκευση του χαρακτηριστικού kwFreq δεκαδικής τιμής, που αφορά το πλήθος των λέξεων κλειδιών που εμφανίζονται στα σημαντικά πεδία κειμένου της εγγραφής αυτής. Για τον λόγο αυτό στη σειρά 12 αρχικοποιείται η μεταβλητή kwFreq δεκαδικών τιμών, η οποία μέσω της κατασκευαστικής μεθόδου EntryLine στη σειρά 35 αποθηκεύει την τιμή του μεγέθους αυτού.

7.3 *Πειραματικά δεδομένα*

Ο τρόπος με τον οποίο θα διενεργηθούν τα πειράματα πάνω στην υλοποίηση της §6.2 είναι παρόμοιος με αυτόν της §5.3. Δεδομένου πως η υλοποίηση της λογικής της §6.1 αποτελεί τροποποίηση της αντίστοιχης λογικής της §5, το αρχείο εκροών της κλάσης BNL επέκτασης .csv αποτελείται από 100 εγγραφές στις οποίες αναγράφονται τα σημεία συντεταγμένων που χρησιμοποιήθηκαν ως εισροή, οι υπολογιστικοί πόροι που δαπανήθηκαν για τον υπολογισμό της κορυφογραμμής για καθένα από αυτά καθώς και το μήκος της κορυφογραμμής αυτής. Πιο συγκεκριμένα, οι υπολογιστικοί πόροι αφορούν τον χρόνο σε ms που διήρκησε ο υπολογισμός της κορυφογραμμής με γνώμονα την απόσταση από το αντίστοιχο σημείο συντεταγμένων καθώς και τον χώρο που δεσμεύτηκε κατά την διάρκεια της διαδικασίας αυτής από την RAM του συστήματος. Ωστόσο, ο τρόπος με τον οποίον θα διενεργηθεί η εκτέλεση της υλοποίησης της §6.2 θα αφορά την μεταβολή του πλήθους των λέξεων κλειδιών που ο χρήστης δίνει ως εισροή, και την επίδραση του πάνω στα σημαντικά μεγέθη του χώρου και του χρόνου που δαπανώνται σε κάθε επερώτημα κορυφογραμμής BNL. Τα πλήθη των λέξεων κλειδιών που θα επιχειρηθούν είναι 2,4,6,8 και 10.

	TIME(ms)	SPACE(gb)	LENGTH(pts)
MAX	3978	0.128842	647
MIN	3180	0.001793	225
AVG	3547.31	0.064653	351.95
MEDIAN	3575	0.064292	328

Πίνακας 20 Στατιστικά μετρικών για χρήση συνόλου λέξεων κλειδιών 2 στοιχείων

	TIME(ms)	SPACE(gb)	LENGTH(pts)
MAX	4231	0.125106	783
MIN	3386	0.001997	353
AVG	3558.54	0.066932	567.52
MEDIAN	3489.5	0.067353	584.5

Πίνακας 21 Στατιστικά μετρικών για χρήση συνόλου λέξεων κλειδιών 4 στοιχείων

	TIME(ms)	SPACE(gb)	LENGTH(pts)
MAX	4487	0.124804	765
MIN	3353	0.001853	326
AVG	3563.69	0.058943	529.01
MEDIAN	3455.5	0.060438	536.5

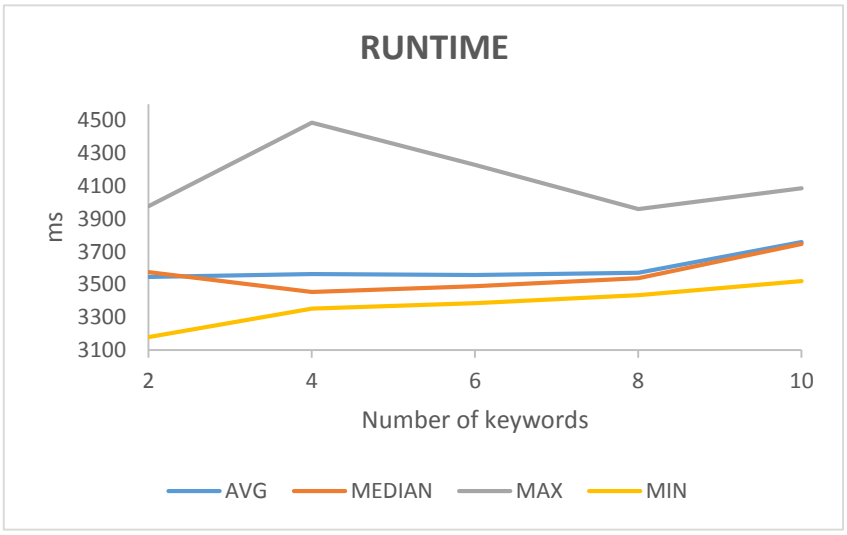
Πίνακας 22 Στατιστικά μετρικών για χρήση συνόλου λέξεων κλειδιών 6 στοιχείων

	TIME(ms)	SPACE(gb)	LENGTH(pts)
MAX	3961	0.113457	1063
MIN	3436	0.001999	598
AVG	3572.04	0.062021	827.21
MEDIAN	3539	0.062631	832

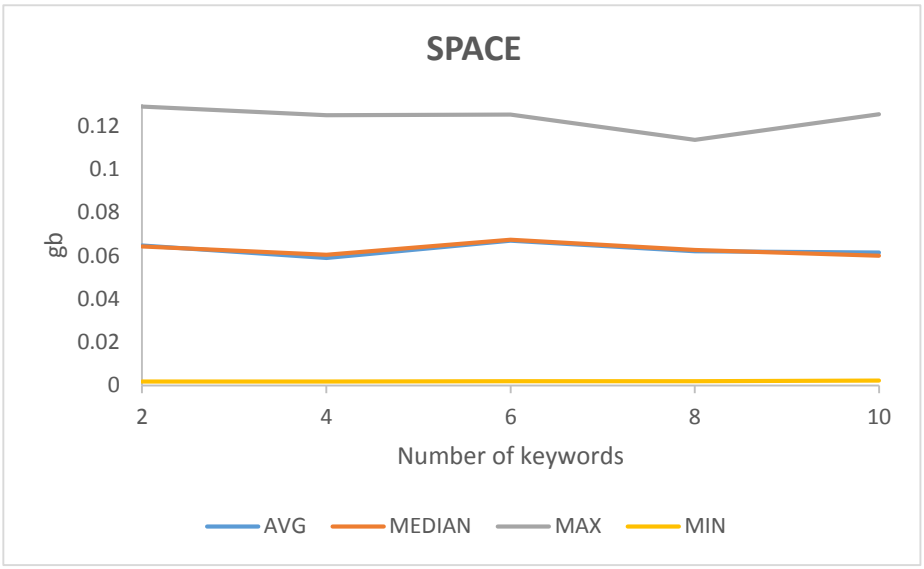
Πίνακας 23 Στατιστικά μετρικών για χρήση συνόλου λέξεων κλειδιών 8 στοιχείων

	TIME(ms)	SPACE(gb)	LENGTH(pts)
MAX	4087	0.125216	1178
MIN	3522	0.002256	661
AVG	3760.14	0.061362	898.53
MEDIAN	3747	0.059957	910

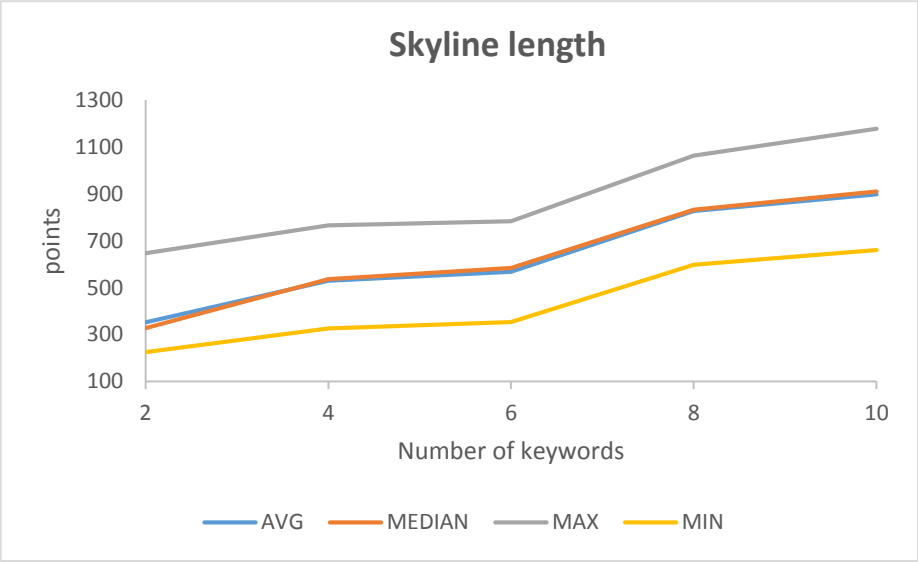
Πίνακας 24 Στατιστικά μετρικών για χρήση συνόλου λέξεων κλειδιών 10 στοιχείων



Εικόνα 35 Διάγραμμα των στατιστικών χρόνου για τα διάφορα πλήθη λέξεων κλειδιών



Εικόνα 36 Διάγραμμα των στατιστικών χώρου για τα διάφορα πλήθη λέξεων κλειδιών



Εικόνα 37 Διάγραμμα των στατιστικών μήκους κορυφογραμμής για τα διάφορα πλήθη λέξεων κλειδιών

7.4 *Συμπεράσματα*

Ο νέος παράγοντας ομοιότητας κειμένου kwFreq ανάμεσα στις εγγραφές και το δοθέν σύνολο λέξεων κλειδιών φαίνεται να μην επηρεάζει σημαντικά τον χώρο που δαπανάται για την περάτωση των επερωτημάτων κορυφογραμμής. Σε ό,τι αφορά τον δαπανηθέντα χρόνο, σημειώνεται μια αισθητή αύξηση του με την αύξηση του πλήθους των λέξεων κλειδιών που δίνονται ως εισροή. Επιπροσθέτως, αν αναλογιστεί κανείς την πρώτη εφαρμογή του BNL και την περίπτωση με απόσταση του προηγούμενου κεφαλαίου, ο χρόνος αυξάνεται περίπου κατά 400ms και 200ms αντίστοιχα. Το γεγονός αυτό θα μπορούσε να δικαιολογήσει η εισαγωγή ενός επιπλέον σημαντικού παράγοντα βάσει του οποίου καλείται να υπολογιστεί η κορυφογραμμή. Το μήκος της κορυφογραμμής σημειώνει επίσης σημαντική αύξηση με μεγαλύτερο πλήθος των δοθέντων λέξεων κλειδιών. Σε σύγκριση με τις πρώτες δυο επερωτήσεις των προηγούμενων κεφαλαίων το μέσο μήκος της κορυφογραμμής είναι κατά πολύ μεγαλύτερο. Η εισαγωγή περισσότερων παραγόντων δείχνει να έχει άμεση σχέση με το μήκος της κορυφογραμμής που παράγεται ως αποτέλεσμα. Η ύπαρξη περισσότερων στοιχείων μέσα στις εκροές περιορίζει την πιθανότητα ύπαρξης έκτοπων

στοιχείων. Συμπερασματικά, όταν η κορυφογραμμή αποτελείται από περισσότερα στοιχεία, ελαχιστοποιείται η επίδραση που δύνανται να έχουν τυχούσες έκτοπες εγγραφές, επάνω στα παραγόμενα αποτελέσματα. Έτσι η εισαγωγή ενός ακόμη αντικειμενικού παράγοντα όπως η ομοιότητα κειμένου kwFreq αυξάνει την αξιοπιστία της κορυφογραμμής.

8 *ΣΥΜΠΕΡΑΣΜΑΤΑ*

8.1 *Σύνοψη*

Ο αλγόριθμος προ-επεξεργασίας που αναπτύχθηκε στο δεύτερο κεφάλαιο με σκοπό την αφαίρεση των περιττών στοιχείων του αρχικού συνόλου δεδομένων οδήγησε στην επιθυμητή μορφή του αρχείου εισροών για τον αλγόριθμο BNL. Η υλοποίηση του αλγορίθμου BNL σε java βρίσκει τα αντικειμενικώς βέλτιστα στοιχεία εκείνα, που δεν κυριαρχούνται από κανένα στοιχείο εντός του συνόλου των εγγραφών των αρχικών δεδομένων. Κατόπιν της εφαρμογής της ανεπτυγμένης προσέγγισης του BNL, παρατηρήθηκαν έκτοπες εγγραφές εντός του συνόλου των εξαγώγιμων στοιχείων. Η παρουσία εγγραφών τέτοιου είδους που δεν ανταποκρίνονται σε αντικειμενικώς ρεαλιστικά δεδομένα υποδεικνύει την ευαισθησία του αλγορίθμου σε ό,τι αφορά εσφαλμένες ή αναληθείς εγγραφές εντός των εισροών. Κατ' επέκταση, όταν γίνεται λόγος για πραγματικά δεδομένα, διαφαίνεται η ανάγκη ύπαρξης μηχανισμού εκκαθάρισης του προς χρήση συνόλου εισροών ούτως ώστε να εξαλειφθούν οι έκτοπες εγγραφές εκείνες που ενδεχομένως θα αλλοιώσουν τα επιθυμητά αποτελέσματα των εκροών του αλγορίθμου BNL. Σε αντίθετη περίπτωση το σύνολο των εγγραφών το οποίο εξάγεται βάσει του αλγορίθμου αυτού είναι υποκείμενο σε υπερβολικά τέλειες εγγραφές που, αν και δεν κυριαρχούνται από τις υπόλοιπες, ωστόσο δεν είναι ουσιαστικής σημασίας σε πραγματικές συνθήκες. Ως αποτέλεσμα, τίθεται ως προαπαιτούμενη η εμπιστοσύνη του συνόλου εισροών για να εκμεταλλευτεί κανείς την αντικειμενικότητα που παρέχει ο αλγόριθμος BNL και κατόπιν να κάνει εφαρμογή του.

Αρωγός στην προσπάθεια αντιμετώπισης έκτοπων εγγραφών παρουσιάζεται στο πέμπτο κεφάλαιο ο αλγόριθμος k-Skyband. Η προσέγγιση αυτή οδηγεί σε εγγραφές με φραγμένο πλήθος κυρίαρχων επι αυτών στοιχείων. Παρατηρείται πως η σταδιακή αύξηση της μεταβλητής K των κυρίαρχων στοιχείων επι εγγραφής, οδηγεί σταδιακά σε σύνολα σημείων που περιέχουν τα αντικειμενικώς βέλτιστα στοιχεία που δεν αποτελούν έκτοπες εγγραφές. Παρόλα αυτά κρίνεται απαραίτητη η επανειλημμένη εφαρμογή του αλγορίθμου τροποποιημένου για k-Skyband αλγορίθμου BNL ούτως ώστε να αποφανθεί κανείς για την τιμή της μεταβλητής k εκείνης, που θα επιφέρει τα επιθυμητά αποτελέσματα. Συνεπώς είναι προαπαιτούμενη η επεξεργασία του συνόλου εκροών και η εκ των

υστέρων κρίση του κάθε σημείου που ανήκει στο k-Skyband ως επιθυμητού ή όχι. Η επανειλημμένη εκτέλεση, μάλιστα, του αλγορίθμου k-Skyband για μεγαλύτερες τιμές του k προς αναζήτηση των βέλτιστων πραγματικών εγγραφών θα οδηγήσει και σε ανάλογη με αυτό χρήση πολύτιμων υπολογιστικών πόρων σε ό,τι αφορά τον χρόνο επεξεργασίας και τον χώρο που αφιερώνεται στην κύρια μνήμη με σκοπό τις συγκρίσεις πολλαπλών μεταξύ τους στοιχείων. Ως αποτέλεσμα, σε ό,τι αφορά την εφαρμογή της τροποποιημένης εκδοχής για k-Skyband του BNL, τίθεται το δίλλημα χρήσης μεγάλου όγκου υπολογιστικών πόρων με σκοπό να καταλήξει κανείς στα αντικειμενικώς βέλτιστα πραγματικά στοιχεία ανάμεσα σε σύνολα δεδομένων που ενδεχομένως περιέχουν έκτοπες εγγραφές.

Ως εναλλακτικός τρόπος αντιμετώπισης έκτοπων εγγραφών παρουσιάζεται η περιορισμένη κορυφογραμμή που προτείνεται εντός του πέμπτου κεφαλαίου. Η προσέγγιση αυτή τροποποιεί την υλοποίηση του BNL με τέτοιον τρόπο ούτως ώστε κάθε εγγραφή του αρχικού συνόλου εισροών να ελέγχεται πριν θεωρηθεί ως υποψήφια για το τελικό σύνολο κορυφογραμμής. Πιο συγκεκριμένα, δοθέντος συνόλου περιορισμών για κάθε σημαντικό πεδίο, οι τιμές των επιμέρους εγγραφών για τα εν λόγω πεδία κρίνουν το αν οι εγγραφές αυτές πρόκειται να θεωρηθούν έγκυρες ή όχι και κατ' επέκταση υποψήφιες για το σύνολο εκροών κορυφογραμμής του αλγορίθμου. Ως αποτέλεσμα, η κορυφογραμμή που υποδεικνύει ως αποτέλεσμα η συγκεκριμένη προσέγγιση εξαρτάται άμεσα από την επιλογή των κατάλληλων περιορισμών εκείνων, που θα αποκλείσουν υποσύνολο των εισροών και κατόπιν θα οδηγήσουν στον υπολογισμό της κορυφογραμμής βάσει των υπολειπόμενων εγγραφών. Η προσέγγιση αυτή οδηγεί σε αντικειμενικά αποτελέσματα ανεξαρτήτως της ύπαρξης έκτοπων εγγραφών. Το γεγονός αυτό δικαιολογεί η ένταξη όλων των στοιχείων της κορυφογραμμής εντός των δοθέντων περιορισμών. Κατ' επέκταση η σωστή επιλογή κατάλληλων περιορισμών θα αναιρέσει την ευαισθησία του BNL σε έκτοπες εγγραφές, εξαλείφοντας την δυνατότητα τους να ενταχθούν στο σύνολο εκροών κορυφογραμμής. Συμπερασματικά, προαπαιτούμενη είναι η δημιουργία συνόλου περιορισμών που θα αποκλείει έκτοπες εγγραφές βάσει ανεδαφικών τιμών και θα επιφέρει έγκυρα αποτελέσματα. Πρέπει ωστόσο να σημειωθεί πως ελλοχεύει ο κίνδυνος αποκλεισμού πραγματικών στοιχείων που βρίσκονται εκτός των δοθέντων περιορισμών ή και η αγνόηση έκτοπων στοιχείων που χρίζουν αποκλεισμού σε περίπτωση λανθασμένης επιλογής των περιορισμών αυτών. Διαφαίνεται, ως αποτέλεσμα, η ανάγκη προτίμησης κατάλληλων περιορισμών που θα αποτρέψουν την απόρριψη πραγματικών εγγραφών και συνάμα θα θωρακίσουν τα αποτελέσματα από τυχόντα έκτοπα στοιχεία.

8.2 *Πειραματικά συμπεράσματα*

Οι τρείς περιπτώσεις εφαρμογής επερωτημάτων BNL εξετάζουν την εισαγωγή διαφορετικών παραγόντων ως προς του οποίους καλείται να υπολογιστεί το αποτέλεσμα της κορυφογραμμής. Στο 4ο κεφάλαιο, χρησιμοποιώντας τρία από τα ήδη υπάρχοντα πεδία χαρακτηριστικών του συνόλου πραγματικών δεδομένων το επερώτημα BNL καταλήγει σε κορυφογραμμή μήκους τριών στοιχείων. Τα στοιχεία της κορυφογραμμής ωστόσο συμπεραίνεται πως αποτελούν έκτοπες εγγραφές, η παρουσία των οποίων έχει αδιαμφισβήτητη επιρροή τόσο πάνω στο τελικό αποτέλεσμα του κάθε επερωτήματος κορυφογραμμής BNL όσο και στους υπολογιστικούς πόρους που διατίθενται για την περάτωση του. Πιο συγκεκριμένα τα πειραματικά δεδομένα παρουσιάζουν τον μικρότερο χρόνο παραγωγής εκροών που έχει παρατηρηθεί στην περίπτωση παρουσίας έκτοπων στοιχείων. Το γεγονός αυτό δικαιολογείται εύκολα λόγω του συγκεντρωτικά μειωμένου πλήθους συγκρίσεων που θα πρέπει να γίνουν προκειμένου να παραχθεί το τελικό αποτέλεσμα. Πιθανά έκτοπα στοιχεία έχουν επίσης άμεση επιρροή στο μήκος της κορυφογραμμής του αποτελέσματος, δεδομένης της κυριαρχίας τους επι του συνόλου των υπόλοιπων στοιχείων των εισροών. Έτσι, πολλά από τα στοιχεία που υπο κανονικές συνθήκες ανήκουν στο αποτέλεσμα της κορυφογραμμής, στην περίπτωση έκτοπων στοιχείων αποκλείονται μειώνοντας το πλήθος των επιστρεφόμενων εγγραφών. Συμπερασματικά, ο BNL μπορεί να χρησιμοποιηθεί για την αντικειμενική εύρεση της κορυφογραμμής σε πραγματικά δεδομένα, η χρησιμότητα των εκροών και η διάθεση υπολογιστικών πόρων όμως εξαρτώνται από την ύπαρξη μηχανισμού που διασφαλίζει την αντιμετώπιση ενδεχόμενων έκτοπων εγγραφών.

Το έκτο κεφάλαιο στοχεύει στην προσθήκη επιπλέον σημαντικού παράγοντα, μέσω μιας εισροής 100 στοιχείων συντεταγμένων, βάσει των οποίων δημιουργείται μια συνιστώσα απόστασης που χαρακτηρίζει κάθε μια από της εγγραφές. Η προσθήκη του νέου αυτού παράγοντα φαίνεται να επηρεάζει πρωτίστως τον χρόνο που απαιτείται για την περάτωση του καθενός από τα 100 επερωτήματα σε σύγκριση με την απλή εφαρμογή του BNL. Μια υποτυπώδης αύξηση της τάξης των 200ms υποδηλώνει πως μεγαλύτερο πλήθος σημαντικών παραγόντων ενδεχομένως οδηγεί σε περισσότερους πόρους που σχετίζονται με τον χρόνο. Σημαντική είναι, δε, η αλλαγή επάνω στο μέσο μήκος της κορυφογραμμής των αποτελεσμάτων το οποίο έχει εκτοξευθεί σε σύγκριση με το

επερώτημα του 4ου κεφαλαίου. Η αύξηση του πλήθους των επιστρεφόμενων υποδεικνύει πως η επιρροή των έκτοπων περιορίζεται με την εισαγωγή ενός νέου αντικειμενικού, δίχως έκτοπες τιμές, παράγοντα και κατ' επέκταση η αξιοπιστία των εκροών αυξάνεται. Ως αποτέλεσμα, το μήκος και η αξιοπιστία της κορυφογραμμής καθώς και ο χρόνος περάτωσης των επερωτημάτων εξαρτώνται από το πλήθος των σημαντικών παραγόντων και την ενδεχόμενη παρουσία έκτοπων σε αυτούς τιμών.

Το έβδομο κεφάλαιο εισάγει μια νέα συνιστώσα υπολογισμού της κορυφογραμμής που σχετίζεται με την ομοιότητα ανάμεσα στα σημαντικά πεδία κειμένου της κάθε εγγραφής και δεδομένου συνόλου λέξεων κλειδιών. Σε σύγκριση τόσο με την απλή εφαρμογή του BNL όσο και με τις επερωτήσεις του κεφαλαίου 6, παρουσιάζεται αυξημένος ο μέσος χρόνος για την παραγωγή του αποτελέσματος της κορυφογραμμής. Συμπερασματικά, επιβεβαιώνεται πως το πλήθος των σημαντικών πεδίων έχει επιδράσεις στους υπολογιστικούς πόρους χρόνου που διατίθενται από το σύστημα για τις τελικές εκροές. Τέλος, αναμφισβήτητη είναι και η επιρροή της προσθήκης της νέας συνιστώσας ομοιότητας κειμένου επάνω στο μέσο μήκος του αποτελέσματος της κορυφογραμμής, το οποίο εκτοξεύεται σε σύγκριση με τις δυο προηγούμενες περιπτώσεις επερωτημάτων. Συνεπώς η επιρροή πιθανών έκτοπων εγγραφών επάνω στο τελικό αποτέλεσμα ελαττώνεται διαμορφώνοντας έτσι πιο αξιόπιστες εκροές κορυφογραμμής.

9 *ΒΙΒΛΙΟΓΡΑΦΙΑ*

- Anon., n.d. *oracle.com.* [Ηλεκτρονικό] Available at: https://docs.oracle.com/javase/7/docs/api/java/io/ObjectInputStream.html
- Borzsonyi, K. S., 2001. *The skyline operator.* s.l., s.n., p. 256–273.
- Godfrey Parke, S. R. G. J., 2006. Algorithms and Analyses for Maximal Vector Computation. *VLDB Journal.*
- Kacem Imed, H. S. B. P., 2002. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation,* Τόμος 60, pp. 245-276.
- Kalyvas Christos, T. T., 2017. *A Survey of Skyline Query Processing,* Karlovassi, Samos: s.n.
- Neil C. Schwertman, R. d. S., 2007. Identifying outliers with sequential fences. *Computational Statistics & Data Analysis* .
- Papadias D., T. Y. F. G. S. B., 2003. *Progressive Skyline Computation in Database Systems,* s.l.: s.n.
- Wagner, I., 2020. *New vehicle average selling price in the U.S. 2016 -2019,* s.l.: s.n.