Αναφορά Εργαστηριακής Άσκησης Δομών Δεδομένων 2015 – 16

Ονοματεπώνυμο: Παναγιώτης Ντενέζος

A.M: 5853

E-mail: ntenezos@ceid.upatras.gr

Η γλώσσα προγραμματισμού που επιλέχθηκε για την άσκηση είναι η *C++* και η υλοποίηση έγινε στο *Visual Studio 2015*

Εισαγωγή

Για την γενική υλοποίηση του προγράμματος δημιουργήθηκαν τα παρακάτω αρχεία. Δίνεται μια σύντομη εξήγηση για το καθένα. Περισσότερη ανάλυση θα γίνει στο αντίστοιχο κομμάτι για το κάθε ένα.

- *Main.cpp*: Στο συγκεκριμένο αρχείο περιέχεται η συνάρτηση main, η οποία υλοποιεί όλη τη διεπαφή με τον χρήστη αλλά και διαχειρίζεται τα αντικείμενα που δημιουργήθηκαν, προκειμένου να επιτευχθεί η λειτουργικότητα του προγράμματος.
- *File Actions.h*: Στο αρχείο αυτό περιέχονται οι συναρτήσεις loadFile και writeFile που υλοποιούν την φόρτωση και την αποθήκευση ενός .csv αρχείου αντίστοιχα.
- *Searching Methods.h*: Εδώ υλοποιούνται οι συναρτήσεις LinearSearch, BinarySearch και InterpolationSearch που είναι για γραμμική αναζήτηση, δυαδική αναζήτηση και δυαδική αναζήτηση παρεμβολής αντίστοιχα. Ακόμα δημιουργούνται οι κλάσεις Trie και RBtree για την υλοποίηση της αναζήτησης με χρήση των ψηφιακών και των RED-BLACK δέντρων αντίστοιχα.

Στα αρχεία *HotelsVectors.h*, *Hotel.h* και *Reservation.h* περιέχονται κλάσεις για την δημιουργία κατάλληλων vectros, ώστε να επιτευχθεί η δυναμική διαχείριση των δομών Hotel και Reservation που δίνονται στην εκφώνηση.

Μέρος Α

Με την εκτέλεση του προγράμματος εμφανίζεται ένα μενού επιλογών.

Με την επιλογή 1, ζητείται από τον χρήστη να εισάγει ένα όνομα αρχείου .csv. Μετά την εισαγωγή του ονόματος καλείται η συνάρτηση loadFile, η οποία (αν το αρχείο υπάρχει) φορτώνει σε έναν vector (Hotels) τύπου Hotel όλα τα ξενοδοχεία με τα στοιχεία τους και για κάθε ένα ξενοδοχείο φορτώνει σε ένα vector (Reservations) τύπου Reservation όλες τις κρατήσεις του. Στην περίπτωση που το αρχείο είναι άδειο, τότε φορτώνεται στο αρχείο με επιτυχία. Αν πάλι δεν υπάρχει το αρχείο τότε εμφανίζεται αντίστοιχο μήνυμα λάθους.

Με την επιλογή 2, ζητείται πάλι από τον χρήστη να εισάγει ένα όνομα αρχείου .csv, στο οποίο επιθυμεί να αποθηκεύσει ό,τι υπάρχει στην κύρια μνήμη του προγράμματος. Μετά την εισαγωγή του ονόματος καλείται η συνάρτηση writeFile, η οποία (αν το αρχείο υπάρχει) τότε αποθηκεύει στο αρχείο όλα τα δεδομένα. Αν πάλι το αρχείο δεν υπάρχει, τότε δημιουργείται και μετά αποθηκεύονται τα δεδομένα.

Με την επιλογή 3, ζητείται από τον χρήστη να εισάγει το id το ξενοδοχείου που θέλει να προσθέσει. Αφού γίνει έλεγχος ότι το id δεν υπάρχει, αυξάνει το μέγεθος του vector Hotels και ζητάει από τον χρήστη να εισάγει το όνομα, τα αστέρια και των αριθμό των δωματίων του νέου ξενοδοχείου. Στην συνέχεια, γίνεται ερώτηση στον χρήστη για το αν θέλει να προσθέσει κάποια κράτηση. Αν ο χρήστης απαντήσει καταφατικά, τότε αυξάνεται το μέγεθος του vector Reservations και εισάγονται τα στοιχεία της κράτησης. Μετά εμφανίζεται πάλι μήνυμα για το αν ο χρήστης θέλει να εισάγει άλλη κράτηση. Η παραπάνω διαδικασία γίνεται μέχρις ότου ο χρήστης να μην θέλει να προσθέσει άλλη κράτηση.

Με την επιλογή 4, εμφανίζεται ένα δεύτερο μενού επιλογών όπου ο χρήστης μπορεί να επιλέξει την μέθοδο αναζήτησης που επιθυμεί. Με την επιλογή 1 (γραμμική αναζήτηση), ζητείται από τον χρήστη να εισάγει το id, το οποίο επιθυμεί να αναζητήσει. Έπειτα, καλείται η συνάρτηση LinearSearch, η οποία ψάχνει όλο το vector Hotels από την αρχή μέχρι να βρει το id που δόθηκε.

Με την επιλογή 5, ζητείται από τον χρήστη να εισάγει τα αστέρια και των αριθμό των κρατήσεων που θέλει να έχει το ξενοδοχείο. Στην συνέχεια, γίνεται γραμμική

αναζήτηση σε όλο το vector Hotels και κάθε φορά που βρίσκεται ξενοδοχείο που να πληρεί τις παραπάνω προϋποθέσεις τότε εμφανίζεται το όνομα του.

Με την επιλογή 6, εμφανίζεται ένα δεύτερο μενού επιλογών όπου ο χρήστης μπορεί να επιλέξει την μέθοδο αναζήτησης που επιθυμεί. Με την επιλογή 1 (γραμμική αναζήτηση), ζητείται από τον χρήστη να εισάγει το επώνυμο, το οποίο επιθυμεί να αναζητήσει. Έπειτα, γίνεται γραμμική αναζήτηση σε όλο το vector Reservations για κάθε ξενοδοχείο και κάθε φορά που βρίσκεται μια κράτηση στο συγκεκριμένο επώνυμο εμφανίζονται όλα τα στοιχεία της κράτησης.

Με την επιλογή 7, επιτυγχάνεται έξοδος του προγράμματος. Αφού πρώτα επιβεβαιωθεί ότι όντως ο χρήστης θέλει να τερματίσει την εφαρμογή, αδειάζουν όλα τα δεδομένα που δημιούργησε το πρόγραμμα στην κύρια μνήμη και έπειτα τερματίζει με επιτυχία.

Μέρος Β

Για την υλοποίηση αυτού του μέρους της άσκησης, προστέθηκαν στο μενού επιλογών της 4^{ης} επιλογής δυο ακόμα επιλογές αναζήτησης. Μια για δυαδική αναζήτηση και μια για αναζήτηση παρεμβολής. Αρχικά, γίνεται ταξινόμηση με βάση το id στο vector Hotels.

Στην περίπτωση της δυαδικής αναζήτησης καλείται η συνάρτηση BinarySearch. Η συνάρτηση αυτή υποδιαιρεί τον χώρο της αναζήτησης κάθε φορά μέχρι να βρει το ζητούμενο στοιχείο ή ο δείκτης του αριστερού άκρου να γίνει μεγαλύτερος από αυτόν του δεξιού και άρα να μην υπάρχει το ζητούμενο στοιχείο.

Στην περίπτωση της αναζήτησης παρεμβολής καλείται η συνάρτηση InterpolationSearch. Η συνάρτηση αυτή εξετάζει το μεγέθους του vector και κάνει εκτίμηση για το που στο περίπου μπορεί να είναι το ζητούμενο στοιχείο.

Μέρος Γ

Σε αυτό το ερώτημα προστέθηκε άλλη μια επιλογή αναζήτησης στο μενού επιλογών της 6^{ης} επιλογής. Όπως και παραπάνω (Μέρος Β) ο vector Hotels είναι ταξινομημένος.

Για την υλοποίηση των RED-BLACK δέντρων δημιουργήθηκε μια δομή node, η οποία αναπαριστά τον κάθε κόμβο. Η δομή αυτή περιέχει pointers τύπου node πού δείχνουν για κάθε κόμβο τον κόμβο που είναι γονιός, το αριστερό παιδί του κόμβου και το δεξί παιδί του κόμβου. Ακόμα δημιουργήθηκε η κλάση RBtree, η οποία αναπαριστά το RED-BLACK δέντρο. Σε αυτή την κλάση αναπτύχθηκαν οι εξής μέθοδοι:

- insert: δέχεται σαν όρισμα το id που επιθυμεί ο χρήστης να αναζητήσει. Ξεκινώντας δημιουργεί έναν κόμβο x και στην μεταβλητή του key αποθηκεύει το id και στο χρώμα του το κόκκινο. Στην πορεία ελέγχει, αν ο κόμβος root δεν έχει οριστεί τότε γίνεται αυτός ο κόμβος root και ο πατέρας του δεν υφίσταται, αλλιώς ελέγχει με βάση το id που πρέπει να πάει και ορίζονται οι τιμές right και left. Στην συνέχεια, καλείται η συνάρτηση inserfix(x).
- insertfix: δέχεται σαν όρισμα τον κόμβο x. Αν ο κόμβος αυτός είναι ο root τότε γίνεται χρώμα μαύρο (η ρίζα έχει πάντα μαύρο χρώμα), αλλιώς ανάλογα με την θέση που προστέθηκε ο νέος κόμβος παίρνει και το αντίστοιχο χρώμα, αν κριθεί απαραίτητο γίνονται οι απαραίτητες πράξεις (left ή right rotation) ώστε το δέντρο να ζυγιστεί.
- leftrotate: ανάλογα με την περίπτωση του insertfix, δέχεται σαν είσοδο τον κόμβο του πατέρα ή τον κόμβο του "παππού" και εκτελεί την πράξη της αριστερής περιστροφής.
- rightrotate: λειτουργεί, όπως η συνάρτηση leftrotate, μόνο που εκτελεί την πράξη της δεξιάς πειστροφής.
- search: δέχεται σαν όρισμα έναν vector τύπου Hotel και το id, το οποίο επιθυμεί ο χρήστης να αναζητήσει. Ξεκινάει από την ρίζα του δέντρου και αν είναι κενή επιστρέφει αντίστοιχο μήνυμα, αλλιώς δημιουργεί έναν κόμβο tmp και κάνοντας τους απαραίτητους ελέγχους κινείται είτε αριστερά είτε δεξιά μέχρι να βρεθεί το ζητούμενο id.

Μέρος Δ

Σε αυτό το ερώτημα προστέθηκε άλλη μια επιλογή αναζήτησης στο μενού επιλογών της 4ης επιλογής. Όπως και παραπάνω (Μέρος Β) ο vector Hotels είναι ταξινομημένος.

Για την υλοποίηση των ψηφιακών δέντρων δημιουργήθηκε μια κλάση TrieNode, η οποία αναπαριστά τον κάθε κόμβο. Η κλάση αυτή περιέχει ένα πίνακα από pointers τύπου TrieNode προκειμένου να κατασκευαστούν τα επίπεδα του δέντρου και ένα vector reserve τύπου Reservation προκειμένου να αποθηκευτούν οι κρατήσεις που έχει κάνει αυτός με το ζητούμενο όνομα. Παράλληλα, δημιουργήθηκε η κλάση Trie, η οποία αναπαριστά το ψηφιακό δέντρο. Σε αυτή την κλάση αναπτύχθηκαν οι εξής μέθοδοι:

- insert: δέχεται σαν όρισμα έναν vector τύπου Reservation και το επώνυμο, το οποίο επιθυμεί ο χρήστης να αναζητήσει. Ξεκινάει από την ρίζα του δέντρου και αφού κάνει μετατροπή το κάθε γράμμα του επωνύμου σε ακέραιο, ώστε να δεικτοδοτήσει τον πίνακα, και να προσπελάσει κάθε επίπεδο είτε δημιουργώντας ένα νέο κόμβο όπου δεν υπάρχει είτε απλά προσπερνώντας το μέχρι να φτάσει στο τέλος του επιθέτου όπου και προσθέτει στο vector του τελευταίου κόμβου την εκάστοτε κράτηση.
- search: όπως και παραπάνω ακολουθεί μια πορεία μέσω δεικτών από την ρίζα μέχρι κάποιο κόμβο. Αν το επώνυμο το οποίο αναζητείται υπάρχει, τότε εμφανίζει όλα τα στοιχεία της κράτησης.

Μέρος Ε

Για την μέτρηση των χρόνων εκτέλεσης έγινε χρήση του τύπου *clock_t* που ουσιαστικά εκφράζεται σε κύκλους ρολογιού.

Έγιναν συνολικά 10 μετρήσεις με σύνολο μεγέθους 1100 (όσο και το αρχείο data.csv που δίνεται) έτσι ώστε τα αποτελέσματα να είναι όσον το δυνατόν πιο αξιόπιστα. Οι χρόνοι και ο αριθμός των συγκρίσεων (μαζί με τους μέσους όρους τους) φαίνονται στους πίνακες παρακάτω:

	Γραμμική Αναζήτηση	Δυαδική Αναζήτηση	Αναζήτηση Παρεμβολής	RED-BLACK δέντρα
1	0,792	1,371	0,981	0,431
2	0,776	1,394	1,375	0,436
3	0,761	1,374	1,767	0,414
4	0,756	1,376	1,296	0,431
5	0,738	1,349	1,701	0,413
6	0,769	1,358	1,632	0,444
7	0,779	1,359	0,998	0,426
8	0,767	1,383	1,223	0,429
9	0,736	1,343	0,986	0,419
10	0,745	1,364	1,653	0,435
ΜΟ	0,6874	1,2307	1,1959	0,3414

Πίνακας 1ος: Χρόνοι (sec) για αναζήτηση με βάση το id (επιλογή 4)

	Γραμμική Αναζήτηση	Δυαδική Αναζήτηση	Αναζήτηση Παρεμβολής	RED-BLACK δέντρα
1	601765	20771	3300	1517905
2	628549	20640	4400	1561604
3	616493	20482	4400	1514476
4	624118	20704	3300	1515843
5	612332	20570	3300	1545468
6	619601	20596	4400	1557077
7	601765	20582	3300	1535070
8	628549	20781	4400	1554053
9	616493	20349	3300	1513498
10	624118	20604	4400	1542298
ΜΟ	617378,3	20607,9	3850	1535729,2

Πίνακας 2ος: Συγκρίσεις στην αναζήτηση με βάση το id (επιλογή 4)

	Γραμμική Αναζήτηση
1	0,318
2	0,335
3	0,347
4	0,339
5	0,348
6	0,348
7	0,345
8	0,346
9	0,344
10	0,341
ΜΟ	0,3411

Πίνακας 3ος: Χρόνοι (sec) για αναζήτηση με βάση τα αστέρια και των αριθμό των κρατήσεων (επιλογή 5)

	Γραμμική Αναζήτηση
1	1210000
2	1210000
3	1210000
4	1210000
5	1210000
6	1210000
7	1210000
8	1210000
9	1210000
10	1210000
ΜΟ	1210000

Πίνακας 4ος: Συγκρίσεις στην αναζήτηση με βάση τα αστέρια και των αριθμό των κρατήσεων (επιλογή 5)

	Γραμμική Αναζήτηση	Digital Tries
1	17,26	0,006
2	17,179	0,006
3	17,208	0,007
4	17,219	0,007
5	17,404	0,009
6	17,135	0,006
7	17,353	0,006
8	17,046	0,007
9	17,331	0,005
10	17,486	0,006
ΜΟ	17,2621	0,0065

Πίνακας 5ος: Χρόνοι (sec) για αναζήτηση με βάση το επώνυμο (επιλογή 6)

	Γραμμική Αναζήτηση	Δυαδική Αναζήτηση
1	19360000	8800
2	19360000	19800
3	19360000	13200
4	19360000	11000
5	19360000	13200
6	19360000	11000
7	19360000	13200
8	19360000	11000
9	19360000	13200
10	19360000	8800
ΜΟ	19360000	12320

Πίνακας 6ος: Συγκρίσεις στην αναζήτηση με βάση το επώνυμο (επιλογή 6)

Ανάλυση Αποτελεσμάτων

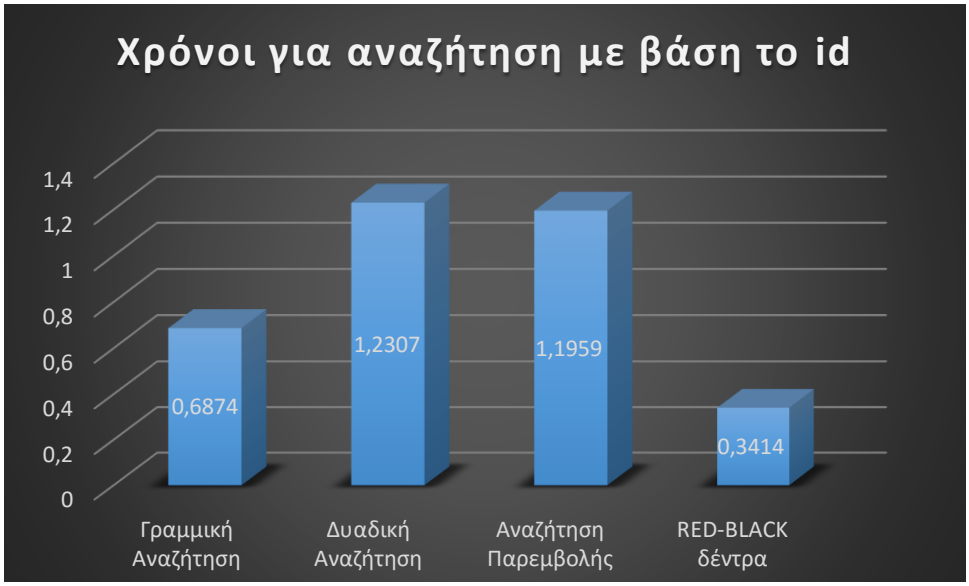
Γενικά

- Αναζήτηση με βάση το id
  - Γραμμική αναζήτηση: Η θεωρία δίνει O(n), το οποίο και επαληθεύεται καθώς $T(n) \leq c*f(n) \rightarrow 617378.3 \leq c*1100$, στο οποίο αν θέσουμε c=1 προκύπτει $617378.3 \leq 1100$ που ισχύει.
  - Δυαδική αναζήτηση: Η θεωρία δίνει O(logn), το οποίο και επαληθεύεται καθώς $T(n) \leq c*f(n) \rightarrow 4.31 \leq c*3.04$, στο οποίο αν θέσουμε c=1 προκύπτει $4.31 \leq c*3.04$ που ισχύει
  - Αναζήτηση παρεμβολής: Η θεωρία δίνει O(loglogn), το οποίο και επαληθεύεται καθώς $T(n) \leq c*f(n) \rightarrow 0.55 \leq c*0.48$, στο οποίο αν θέσουμε c=2 προκύπτει $0.55 \leq 0.96$ που ισχύει
  - Αναζήτηση σε RED-BLACK δέντρο: Η θεωρία δίνει O(logn), το οποίο και επαληθεύεται καθώς $T(n) \leq c*f(n) \rightarrow 4.31 \leq c*3.04$, στο οποίο αν θέσουμε c=1 προκύπτει $4.31 \leq c*3.04$ που ισχύει

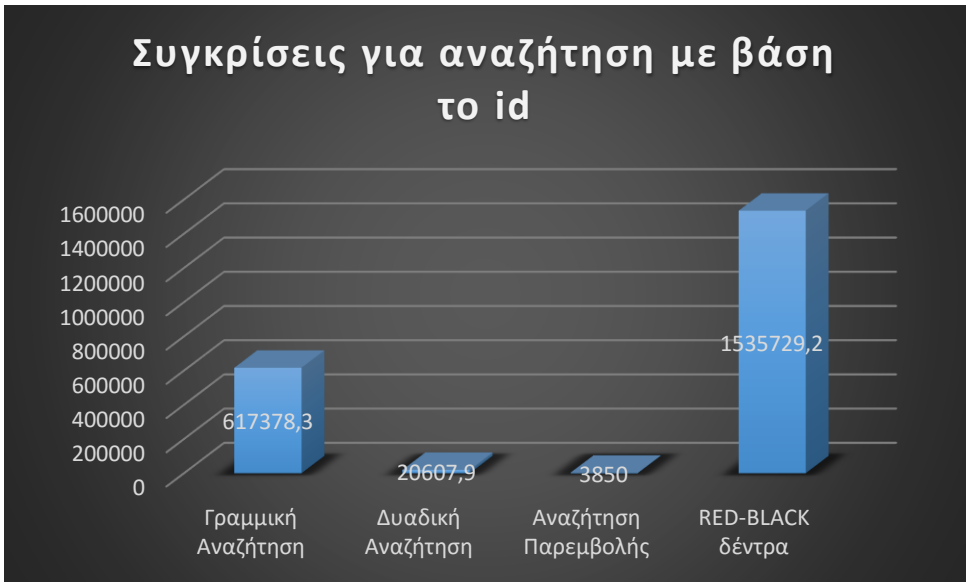
- Αναζήτηση με βάση τα αστέρια και των αριθμό των κρατήσεων
  - Γραμμική αναζήτηση: Η θεωρία δίνει O(n), το οποίο και επαληθεύεται καθώς $T(n) \leq c*f(n) \rightarrow 1210000 \leq c*1100$, στο οποίο αν θέσουμε c=1 προκύπτει $1210000 \leq 1100$ που ισχύει.

- Αναζήτηση με βάση το επώνυμο
  - Γραμμική αναζήτηση: Σε αυτή την αναζήτηση, με δεδομένο ότι ένα επώνυμο μπορεί να εμφανίζεται παραπάνω από μια φορές, το πρόγραμμα που δημιουργήθηκε προσπελαύνει όλες τις εταιρίες μια προς μια και συνεπώς ο αριθμός των συγκρίσεων κάθε φορά είναι σταθερός.
  - Αναζήτηση σε Ψηφιακό δέντρο: Η θεωρία δίνει O(logkn), όπου k = 26 (όσο το αγγλικό αλφάβητο), το οποίο και επιβεβαιώνεται καθώς : $T(n) \leq c*f(n) \rightarrow 2.89 \leq c*2.14$, στο οποίο αν θέσουμε c=2 προκύπτει $2.89 \leq 4.28$ που ισχύει.

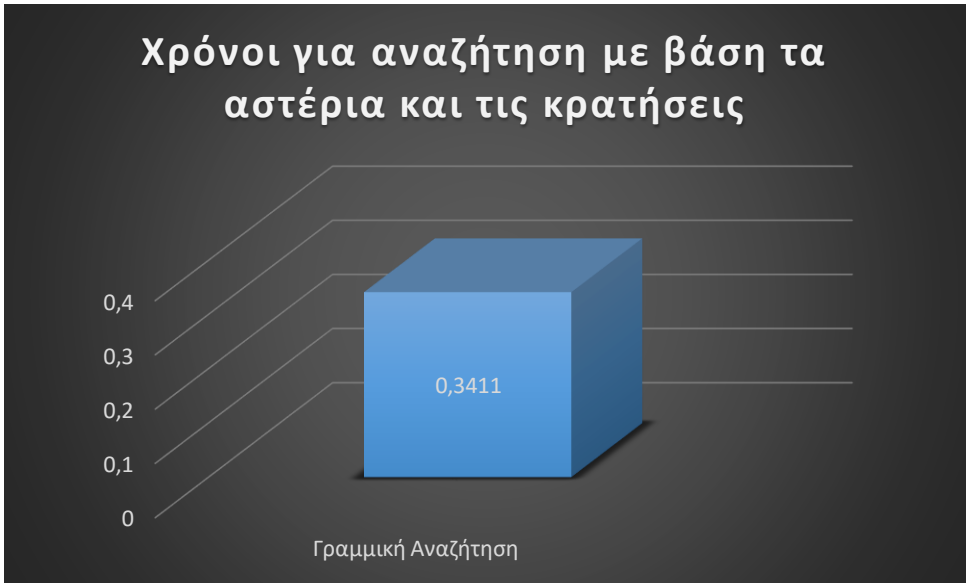
Η συσχέτιση όλων των παραπάνω φαίνεται στα παρακάτω γραφήματα, στα οποία έχει τροποποιηθεί κατάλληλα η κλίμακα στον κατακόρυφο άξονα προκειμένου να είναι εμφανής η διαφορά.



Εικόνα 1η: Χρόνοι (sec) για αναζήτηση με βάση το id (επιλογή 4)



Εικόνα 2η: Συγκρίσεις στην αναζήτηση με βάση το id (επιλογή 4)

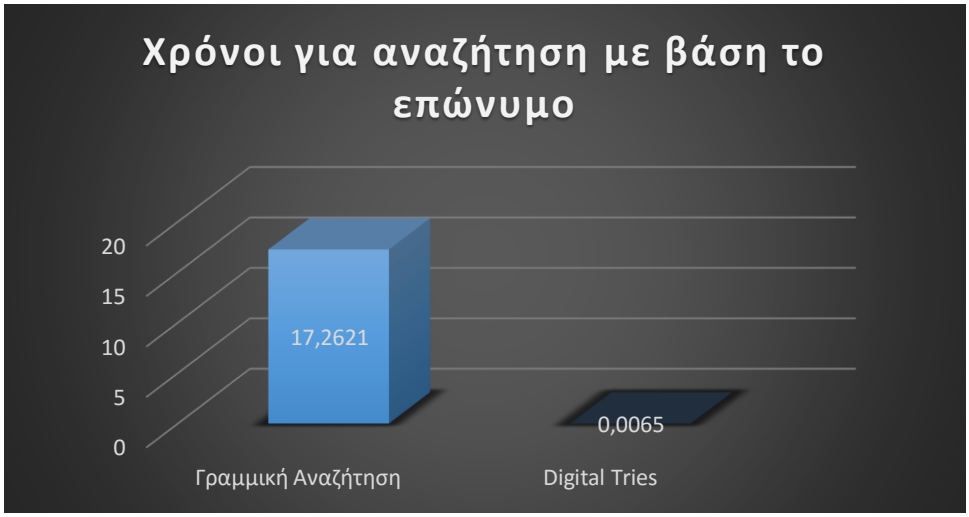


Εικόνα 3ης: Χρόνοι (sec) για αναζήτηση με βάση τα αστέρια και των αριθμό των κρατήσεων

(επιλογή 5)

Εικόνα 4η: Συγκρίσεις στην αναζήτηση με βάση τα αστέρια και των αριθμό των κρατήσεων

(επιλογή 5)



Εικόνα 5η: Χρόνοι (sec) για αναζήτηση με βάση το επώνυμο (επιλογή 6)

Εικόνα 6ος: Συγκρίσεις στην αναζήτηση με βάση το επώνυμο (επιλογή 6)