Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Μάθημα: Βάσεις Δεδομένων
Εργασία Ακ. Έτος 2016-2017

Ονοματεπώνυμο	**Α.Μ.**
Σταυρινάκης Παναγιώτης	6217
E-mail	**Γλώσσα**
stavrinakis@ceid.upatras.gr	C#

Μέρος Α)

Προκειμένου να μεταφορτώνονται όλες οι εγγραφές από το αρχείο χρησιμοποιώ έναν StreamReader() και με την εντολή ReadLine() διαβάζω κάθε γραμμή του αρχείου και εξάγω τα δεδομένα τα οποία έπειτα εισάγονται στον πίνακα hotel του οποίου το μέγεθος έχει ήδη οριστεί από τον αριθμό των ξενοδοχείων. Για να το πετύχω αυτό ελέγχω την περίπτωση που συναντώ τον comma separator ώστε να γνωρίζω σε ποιο σημείο τελειώνει μια λέξη και την αποθηκεύω στην αντίστοιχη θέση. Μόλις φτάσει σε κενή σειρά κλείνει το file και μπορούμε να συνεχίσουμε στις επόμενες λειτουργίες. Παρόμοιος με τον StreamReader() είναι ο StreamWriter() που με την εντολή WriteLine() εξάγω τα δεδομένα σε γραμμές του file μέχρι να μην υπάρχουν άλλα ξενοδοχεία για εξαγωγή.

Για την εισαγωγή στοιχείων αρχικά αυξάνω τον αριθμό των ξενοδοχείων κατά 1 κάθε φορά που κάνω εισαγωγή ξενοδοχείων, δημιουργώ έναν νέο προσωρινό πίνακα με το νέο μέγεθος όπου εισάγω τα νέα στοιχεία στην τελευταία θέση του πίνακα. Έπειτα, μόλις τελειώσω με τις επιπλέον εγγραφές αλλάζω το μέγεθος του αρχικού πίνακα hotel ανάλογα με το πόσες εγγραφές συνολικά έγιναν και περνάω τις τιμές στις τελευταίες θέσεις χρησιμοποιώντας τις εντολές Array.Resize & Array.Copy. Κάθε φορά γίνεται έλεγχος σε περίπτωση που εισάγω ID το οποίο υπάρχει ήδη.

Για να γίνει η αναζήτηση με βάση το ID πρώτα ταξινομώ τα στοιχεία με βάση το ID χρησιμοποιώντας την συνάρτηση BubbleSort(), αν και δεν είναι απαραίτητο για την γραμμική αναζήτηση. Με ένα for loop διατρέχω τον πίνακα και αν αντιστοιχεί το ID με κάποιο από αυτά του πίνακα τότε εκτυπώνεται το αντίστοιχο ξενοδοχείο με τα στοιχεία του.

Για να γίνει η αναζήτηση με βάση το όνομα κράτησης χρησιμοποιώ 2 for loops όπου ψάχνω σε κάθε ξενοδοχείο όλα τα ονόματα κρατήσεων και αν αντιστοιχούν σε αυτό που θέλουμε τότε εκτυπώνονται τα στοιχεία που ζητάει.
Τέλος, με την τελευταία επιλογή το πρόγραμμα θα τερματίζει αφού ζητήσει από τον χρήστη να αποθηκεύσει πρώτα εισαγωγές που τυχόν έκανε.

Μέρος Β)

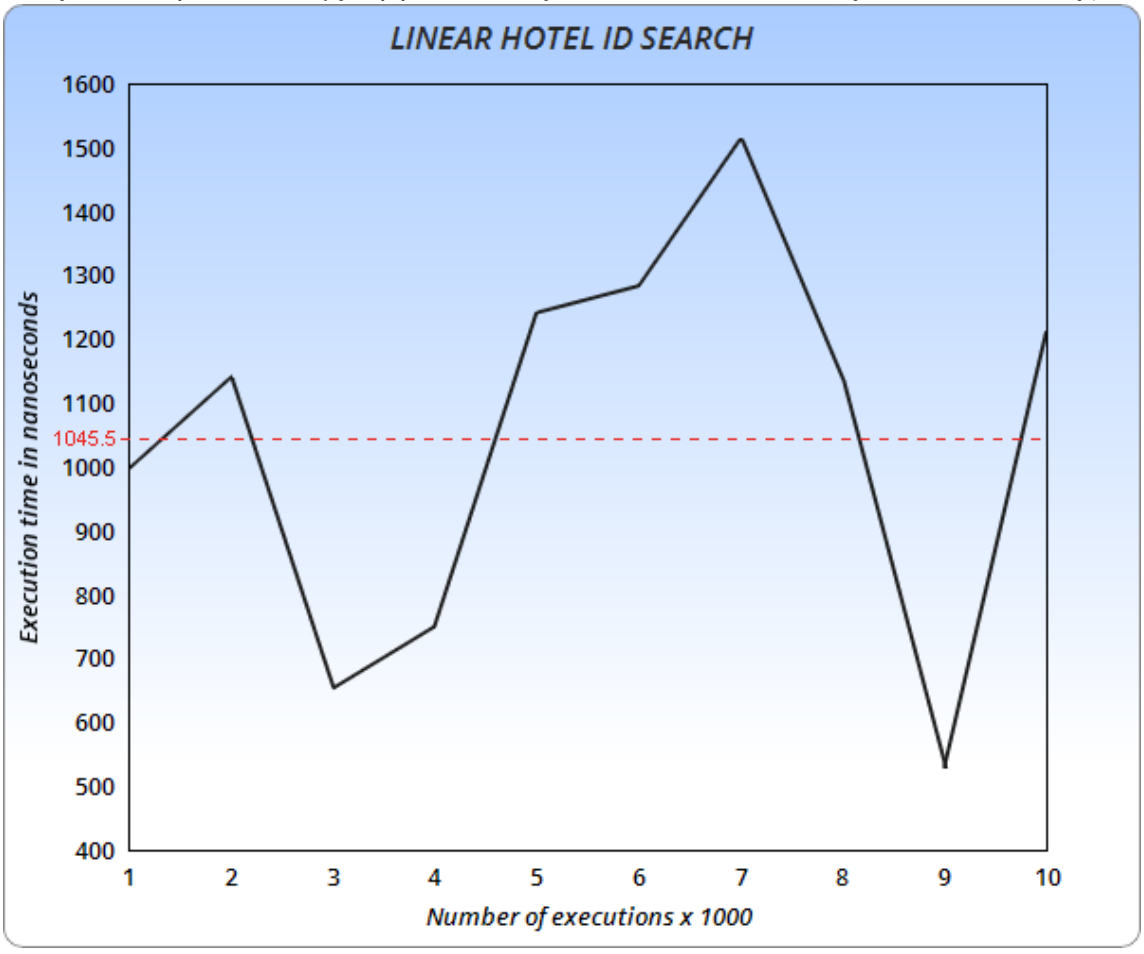
Για την υλοποίηση της αναζήτησης με βάση το ID με χρήση δυαδικής αναζήτησης καθώς και αναζήτησης παρεμβολής θα πρέπει ο πίνακας hotel να είναι ταξινομημένος ως προς το ID, για το λόγο αυτό χρησιμοποιώ τη συνάρτηση BubbleSort(). Για την αναζήτηση έχω ορίσει κατάλληλους ελέγχους BinarySearch & InterpolationSearch ώστε να γίνονται σωστά οι αναζητήσεις και αν επιτύχουν να εμφανίζουν τα αντίστοιχα στοιχεία.

Μέρος Γ & Μέρος Δ)

Δυστυχώς, δεν υλοποιήθηκαν.

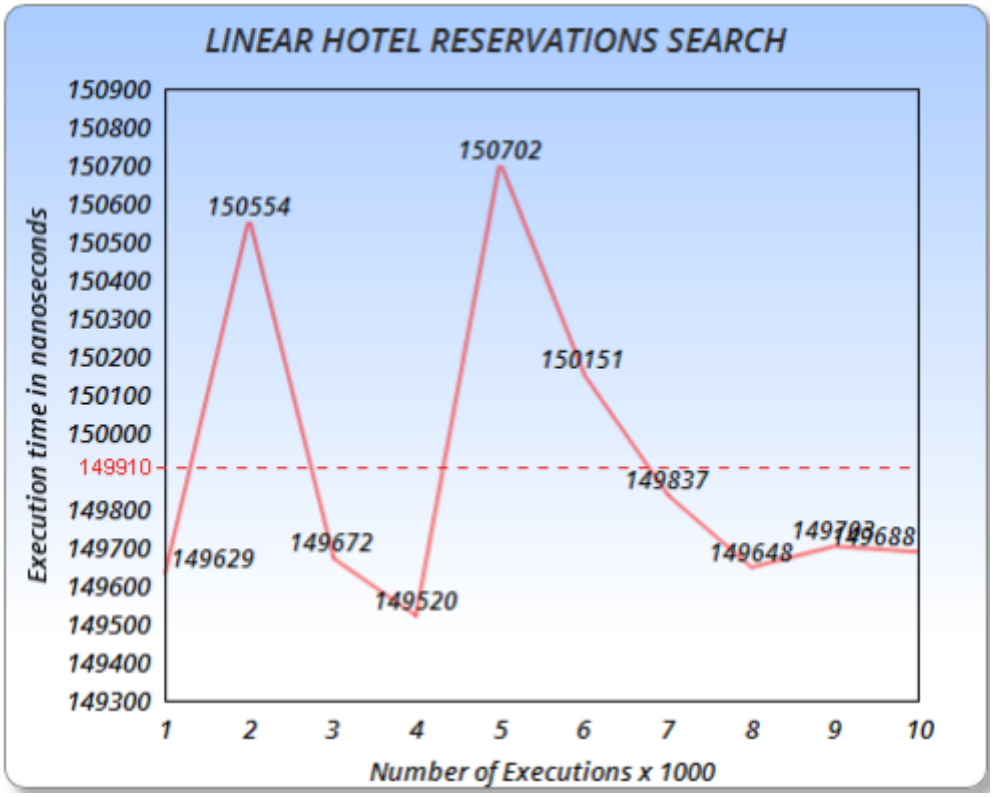
Μέρος Ε)

Linear Search(ID): Στον αλγόριθμο γραμμικής αναζήτησης τα στοιχεία προσπελαύνονται το ένα μετά το άλλο μέχρι να βρεθεί το στοιχείο που ψάχνουμε ή μέχρι να μην υπάρχουν άλλα στοιχεία. Η μέση πολυπλοκότητα του είναι O((n+1)/2) ενώ η πολυπλοκότητα χειρότερης περίπτωσης είναι O(n).
Παρακάτω βλέπουμε το διάγραμμα που προκύπτει από τον χρόνο εκτέλεσης.



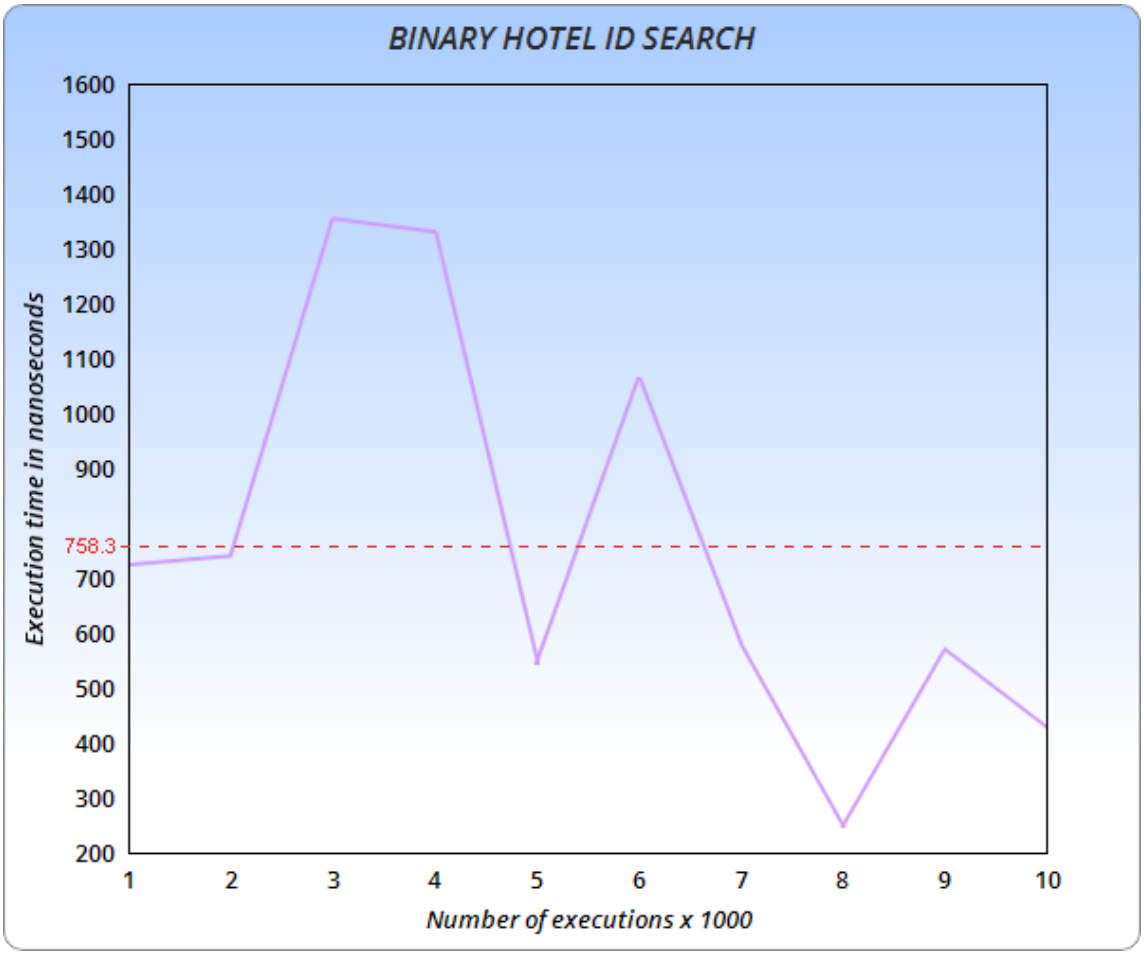
Οι χρόνοι που προκύπτουν είναι μεγαλύτεροι από τους υπόλοιπους αλγόριθμους όπως θα δούμε διότι έχει τον μεγαλύτερο χρόνο χειρότερης περίπτωσης. Ο μέσος χρόνος εκτέλεσης είναι περίπου 1045.5 ns.

Linear Search(Name): Με τον ίδιο τρόπο όπως πριν βλέπουμε το διάγραμμα που προκύπτει από τον χρόνο εκτέλεσης.



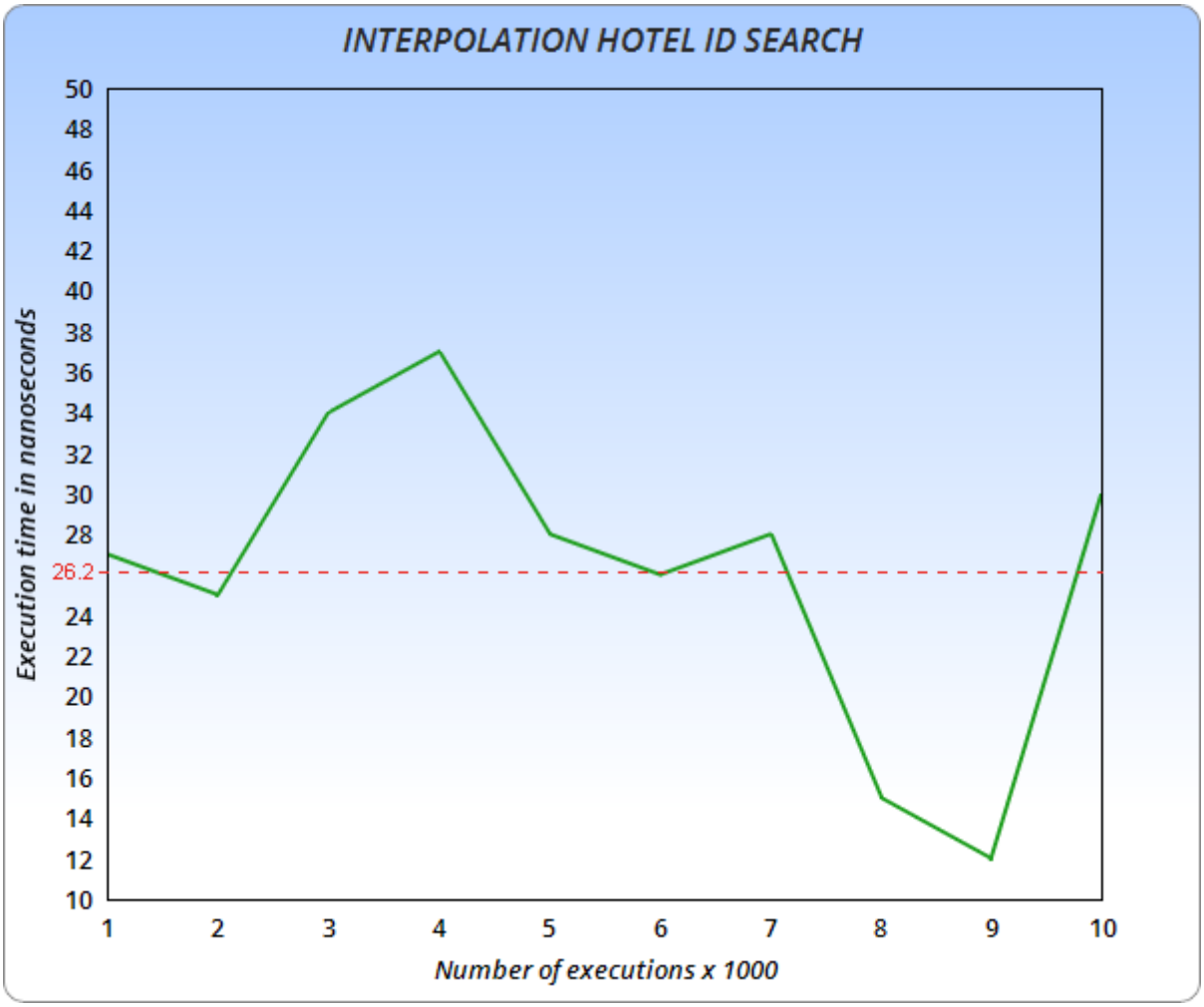
Για τον ίδιο λόγο με πριν οι χρόνοι που προκύπτουν θα είναι μεγαλύτεροι από τους άλλους αλγορίθμους, συν το γεγονός ότι η αναζήτηση είναι πολύ μεγαλύτερη από πριν διότι ο αλγόριθμος αναζητά πολύ περισσότερα δεδομένα. Ο μέσος χρόνος εκτέλεσης είναι περίπου 149910 ns.

Binary Search: Στον αλγόριθμο δυαδικής αναζήτησης χρησιμοποιούμε την τεχνική Διαίρει και Βασίλευε. Η αναζήτηση σε έναν μη ταξινομημένο πίνακα γίνεται σε χρόνο γραμμικό ως προς το μέγεθος της εισόδου, συγκρίνοντας ένα προς ένα τα στοιχεία εισόδου με το κλειδί. Η δυαδική αναζήτηση στηρίζεται στο γεγονός ότι ο πίνακας είναι ταξινομημένος, μειώνοντας σημαντικά τον αριθμό συγκρίσεων που απαιτούνται. Η μέση πολυπλοκότητα είναι O(log n), ενώ η πολυπλοκότητα χειρότερης περίπτωσης είναι O(n*log n).
Παρακάτω βλέπουμε το διάγραμμα που προκύπτει από τον χρόνο εκτέλεσης.



Όπως εξηγήσαμε πριν, οι χρόνοι που προκύπτουν θα είναι μικρότεροι από τον προηγούμενο αλγόριθμο και επειδή έχει μικρότερο χρόνο χειρότερης περίπτωσης O(n*log n). Ο μέσος χρόνος εκτέλεσης είναι περίπου 758.3 ns.

Interpolation Search: Στην αναζήτηση παρεμβολής η εύρεση της εγγραφής στον ταξινομημένο πίνακα γίνεται υπολογίζοντας την πιθανή θέση εγγραφής σε σχέση με τις τιμές των στοιχείων στα άκρα της περιοχής αναζήτησης. Ανάλογα με τη σύγκριση της τιμής της εγγραφής με την πιθανή θέση αναπροσαρμόζεται το αντίστοιχο άκρο. Η πολυπλοκότητα χειρότερης περίπτωσης είναι O(log log n).
Παρακάτω βλέπουμε το διάγραμμα που προκύπτει από τον χρόνο εκτέλεσης.



Σε αυτή τη περίπτωση οι χρόνοι εκτέλεσης είναι πάρα πολύ μικροί αφού ο χρόνος χειρότερης περίπτωσης είναι O(log log n). Ο μέσος χρόνος εκτέλεσης είναι περίπου 26.2 ns.