

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Μάθημα: Δομές Δεδομένων

Εργασία Ακ. Έτος 2015-2016

Ονοματεπώνυμο: Πατρώνη Σωτηρία

ΑΜ: 5399

e-mail: patron@ceid.upatras.gr

Η παρούσα εργασία υλοποιήθηκε χρησιμοποιώντας την γλώσσα C++ χρησιμοποιώντας το IDE DEV-C++. Σε κάθε ερώτημα γίνεται αναφορά της/των συνάρτησης/συναρτήσεων που το υλοποιούν. Για τον πλήρη κώδικα του προγράμματος (π.χ. Επικοινωνία με τον χρήστη) ανατρέξτε στα αρχεία πηγαίου κώδικα που περιλαμβάνονται στο .rar αρχείο.

ΜΕΡΟΣ Α (LINEAR SEARCH)

Η υλοποίηση της δομής του ξενοδοχείου έγινε με τον τρόπο που υποδεικύεται στην εκφώνηση της άσκησης και για κάθε δομή του ξενοδοχείου χρησιμοποιήθηκε ένας vector για την αποθήκευση των κρατήσεων και συνολικά άλλος ένας για την αποθήκευση όλων των ξενοδοχείων για τη διευκόλυνση των αναζητήσεων.

Η ολοκλήρωση της πρώτης επιλογής του μενού, δηλαδή της φόρτωσης των δεδομένων από το αρχείο υλοποιείται από τη συνάρτηση: **void loadHotels_Res(char *sf)**, η οποία δέχεται ως όρισμα ένα αρχείο με δεδομένα (στην προκειμένη περίπτωση το αρχείο data.csv που δόθηκε με την εκφώνηση) και εκτελεί τους κατάλληλους χειρισμούς των χαρακτήρων του αρχείου έτσι ώστε να αποθηκευτούν στα πεδία της δομής και να εισαχθούν στα ζητούμενα δέντρα με τον σωστό τρόπο.

Τη δεύτερη επιλογή του μενού της αποθήκευσης των δεδομένων της δομής σε αρχείο την υλοποιεί η συνάρτηση: **void save(char *sf)**, η οποία δέχεται σαν όρισμα το αρχείο προορισμού των δεδομένων. Η συνάρτηση αυτή διατρέχει όλα τα πεδία του αρχείου και τα αποθηκεύει καταλλήλως στο αρχείο, με τέτοιο τρόπο ώστε να μπορεί να διαβαστεί το αρχείο αυτό από την παραπάνω συνάρτηση φόρτωσης.

Την τρίτη επιλογή την υλοποιεί η συνάρτηση **void addNew()**, η οποία δημιουργεί μία νέα θέση στο vector με τα ξενοδοχεία και ζητώντας από τον χρήστη τα κατάλληλα δεδομένα, τα αποθηκεύει στην δομή με τον ίδιο τρόπο που αποθηκεύονται και στη συνάρτηση φόρτωσης.

Η επιλογή τέσσερα υλοποιείται με γραμμική αναζήτηση στην δομή με χρήση της συνάρτησης: **void displayID_LINEAR()**, η οποία δέχεται από τον χρήστη το id ενός ξενοδοχείου, διατρέχει γραμμικά όλη την δομή και εμφανίζει το αποτέλεσμα στο τέλος της αναζήτησης.

Η επιλογή πέντε υλοποιείται με γραμμική αναζήτηση στην δομή με χρήση της συνάρτησης: **void display_by_stars_res()**, η οποία δέχεται από τον χρήστη τα αστέρια του ξενοδοχείου και το όνομα μίας κράτησης, διατρέχει γραμμικά όλη την δομή και εμφανίζει όλα τα ξενοδοχεία με αυτά τα χαρακτηριστικά στο τέλος της αναζήτησης.

Η επιλογή έξι υλοποιείται με γραμμική αναζήτηση στην δομή με χρήση της συνάρτησης: **void display_res_by_name_LIN()**, η οποία δέχεται από τον χρήστη το όνομα μίας κράτησης

ενός ξενοδοχείου, διατρέχει γραμμικά όλη την δομή και εμφανίζει το αποτέλεσμα στο τέλος της αναζήτησης.
Η τελευταία επιλογή τερματίζει το πρόγραμμα της εφαρμογής.

ΜΕΡΟΣ Β (BINARY SEARCH, INTERPOLATION SEARCH)

Οι συναρτήσεις που υλοποιούν τις αναζητήσεις του ερωτήματος αυτού είναι οι εξής:

- **int binarySearch(int value, int left, int right)** , η οποία υλοποιεί τον αλγόριθμο της δυαδικής αναζήτησης και καλείται από τη συνάρτηση **void displayID_BIN()** , η οποία λαμβάνει τα αποτελέσματα της αναζήτησης και εμφανίζει στον χρήστη τα κατάλληλα στοιχεία.
- **int interpolation_search (int value)**, η οποία υλοποιεί τον αλγόριθμο της αναζήτησης παρεμβολής και καλείται από τη συνάρτηση **void displayID_IS()**, η οποία λαμβάνει τα αποτελέσματα της αναζήτησης και εμφανίζει στον χρήστη τα κατάλληλα στοιχεία.

Οι δύο παραπάνω συναρτήσεις για να υλοποιηθούν πρέπει η δομή στην οποία θα εφαρμόσουν την αναζήτηση να είναι ταξινομημένες με βάση το πεδίο αναζήτησης. Γι' αυτό τον λόγο επειδή η δομή δεν είναι ταξινομημένη εξ' αρχής όταν φορτώνονται τα δεδομένα, υλοποιείται ο αλγόριθμος της γρήγορης ταξινόμησης πριν κληθούν οι παραπάνω συναρτήσεις αναζήτησης, από τη συνάρτηση:

void quickSort(int left, int right) .

ΜΕΡΟΣ Γ (RED-BLACK ΔΕΝΤΡΑ)

Η αναζήτηση με χρήση Red-Black δέντρων έγινε με χρήση της δομής **class RBtree**, όπως έχει υλοποιηθεί στο header αρχείο της εργασίας και χρησιμοποιεί τις κύριες συναρτήσεις εισαγωγής : **void RBtree::insert(Hotel* z)** , διόρθωσης εισαγωγής **void RBtree::insertfix(RBnode *z)** , αριστερής περιστροφής **void RBtree::leftrotate(RBnode *p)** , δεξιής περιστροφής **void RBtree::rightrotate(RBnode *p)** και αναζήτησης **int RBtree::search(int x)** των Red-Black δέντρων. Με τη χρήση τις τελευταίας από τις παραπάνω συναρτήσεις γίνεται η αναζήτηση σε αυτό το ερώτημα.

ΜΕΡΟΣ Δ (TRIES)

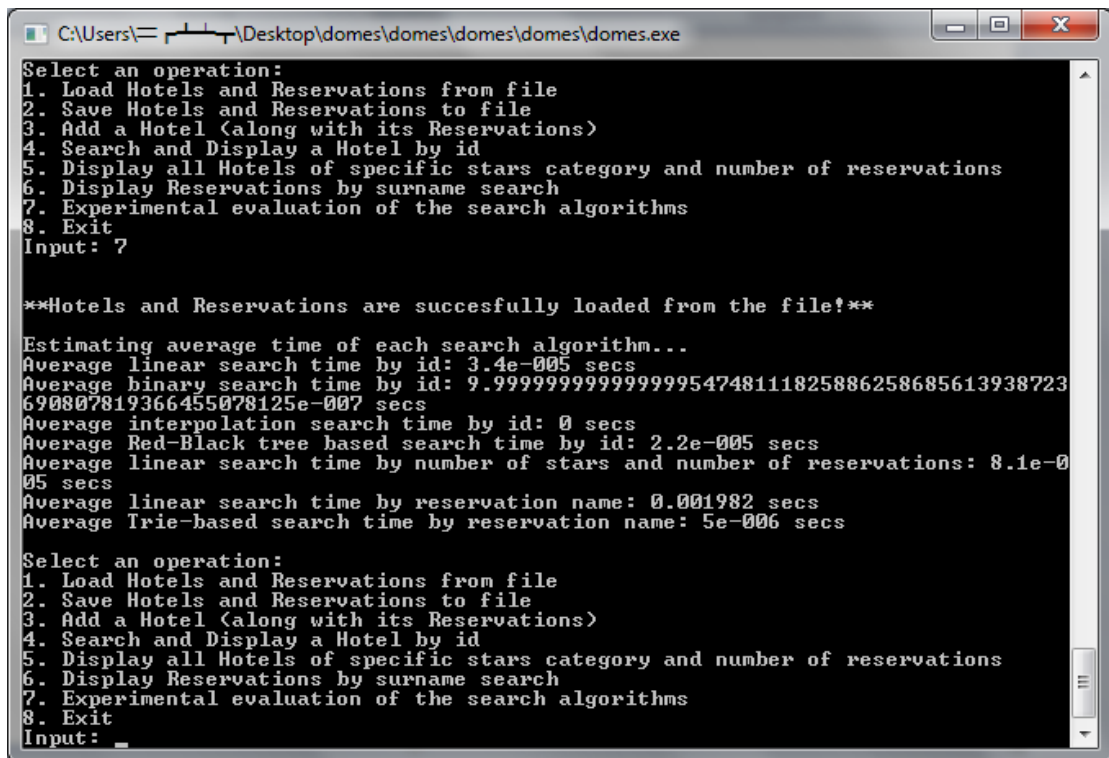
Η αναζήτηση με χρήση Tries έγινε με χρήση της δομής **class digital_trie**, όπως έχει υλοποιηθεί στο header αρχείο της εργασίας και χρησιμοποιεί τις κύριες συναρτήσεις εισαγωγής : **void insertTrie(Hotel* hotel, string key)** και αναζήτησης **bool searchTrie(string key)** των Tries. Με τη χρήση τις τελευταίας συνάρτησης γίνεται η αναζήτηση σε αυτό το ερώτημα.

ΜΕΡΟΣ Ε ΑΠΟΔΟΣΗ ΑΛΓΟΡΙΘΜΩΝ()

Για κάθε είδος αναζήτησης είναι γνωστοί από την θεωρία οι παρακάτω χρόνοι:

Γραμμική αναζήτηση	$O(n)$
Δυαδική αναζήτηση	$O(\log n)$
Αναζήτηση παρεμβολής	$O(\log \log n)$
TRIE	$O(m)$
RED-BLACK Δέντρα	$O(\log n)$

Μετά την εκτέλεση της επιλογής 7 του μενού λήφθηκαν οι παρακάτω μέσοι χρόνοι εκτέλεσης αναζητήσεων:



```
C:\Users\...\Desktop\domes\domes\domes\domes.exe
Select an operation:
1. Load Hotels and Reservations from file
2. Save Hotels and Reservations to file
3. Add a Hotel (along with its Reservations)
4. Search and Display a Hotel by id
5. Display all Hotels of specific stars category and number of reservations
6. Display Reservations by surname search
7. Experimental evaluation of the search algorithms
8. Exit
Input: 7

**Hotels and Reservations are succesfully loaded from the file!**

Estimating average time of each search algorithm...
Average linear search time by id: 3.4e-005 secs
Average binary search time by id: 9.99999999999999954748111825886258685613938723
690807819366455078125e-007 secs
Average interpolation search time by id: 0 secs
Average Red-Black tree based search time by id: 2.2e-005 secs
Average linear search time by number of stars and number of reservations: 8.1e-0
05 secs
Average linear search time by reservation name: 0.001982 secs
Average Trie-based search time by reservation name: 5e-006 secs

Select an operation:
1. Load Hotels and Reservations from file
2. Save Hotels and Reservations to file
3. Add a Hotel (along with its Reservations)
4. Search and Display a Hotel by id
5. Display all Hotels of specific stars category and number of reservations
6. Display Reservations by surname search
7. Experimental evaluation of the search algorithms
8. Exit
Input: _
```

Όπως παρατηρείται το μέσο χρονικό κόστος μιας δυαδικής αναζήτησης προσεγγίζει το 0, με βάση την ορισμένη ακρίβεια αναπαράστασης. Παρατηρούμε ότι το μέσο χρονικό κόστος της αναζήτησης παρεμβολής είναι και αυτό κοντά στο 0.

Όπως βλέπουμε όλες οι υπόλοιπες μετρήσεις ικανοποιούν τους θεωρητικούς χρόνους.