

ΑΝΑΦΟΡΑ PROJECT

Η υλοποίηση των προγραμμάτων έγινε σε C (GNU C++ 11).

I. Sorting and Searching Algorithms

Γενικές σχεδιαστικές αποφάσεις

- Το διάβασμα του αρχείου δεδομένων με τις ημερήσιες κινήσεις της μετοχής γίνεται γραμμή-γραμμή με τη συνάρτηση `int readFile (int argc, char *argv[])` η οποία επιστρέφει το πλήθος των data records που διαβάστηκαν στον πίνακα (κάθε γραμμή του αρχείου πλην της επικεφαλίδας αντιστοιχεί σε ένα data record).
- Το όνομα του αρχείου δεδομένων περνάει στο πρόγραμμα είτε μέσω του command line ή αν δεν έχει δοθεί παράμετρος στην command line διαβάζεται από το χρήστη στη `readFile()`.
- Ο πίνακας S στον οποίο διαβάζονται τα στοιχεία του αρχείου δεδομένων δημιουργείται στατικά με μέγεθος `MAX_SIZE = 3.240`. Θα μπορούσε να έχει δημιουργηθεί δυναμικά με μέγεθος ίσο με το πλήθος των γραμμών του αρχείου + 1: για το σκοπό αυτό θα απαιτούνταν η χρήση μιας συνάρτησης πχ. `int countFileLines(char fileName[])` που θα διάβαζε το αρχείο χαρακτήρα - χαρακτήρα και θα υπολόγιζε και θα επέστρεφε το πλήθος των χαρακτήρων `'\n'` του αρχείου (η συγκεκριμένη υλοποίηση έγινε αλλά δεν έχει περιληφθεί στα παραδοτέα του project).

Πειραματική μελέτη συμπεριφοράς των αλγορίθμων

- Για τη μελέτη της συμπεριφοράς κάθε αλγόριθμου χρησιμοποιούνται δύο τρόποι:
 - A.** Μέτρηση του χρόνου εκτέλεσης μέσω της C++ βιβλιοθήκης `chrono`.
 - B.** Μέτρηση των συγκρίσεων για τους συγκριτικούς αλγόριθμους ταξινόμησης ή των βημάτων για τον `countingsort` και τους αλγόριθμους αναζήτησης.
- Η βιβλιοθήκη `chrono` επιλέχθηκε επειδή είναι `portable`, λειτουργεί και σε Linux και σε Windows συστήματα και χρησιμοποιεί ρολόι με τη μέγιστη δυνατή ακρίβεια (μέτρηση `nanoseconds`). Παρότι δεν μετρά καθαρό CPU χρόνο, επειδή οι αλγόριθμοι που μελετούμε δεν περιέχουν

I/O εντολές αλλά εκτελούν συγκρίσεις, στοιχειώδεις αριθμητικές πράξεις και εντολές καταχώρησης, ο χρόνος που τελικώς μετριέται ισοδυναμεί περίπου με τον χρόνο που η CPU είναι απασχολημένη με την εκτέλεση των παραπάνω εντολών.

- Για τα μεγέθη των αρχείων που χρησιμοποιούμε και τις χρονικές πολυπλοκότητες των αλγορίθμων (από $O(\log \log n)$ έως $O(n^2)$ με $n \leq 3.240$) οι χρόνοι που μετράει το ρολόι της `chrono` στα περισσότερα runs είναι 0 seconds δεδομένου ότι η ακρίβεια του ρολογιού είναι περιορισμένη. Για το λόγο αυτό η μέτρηση του χρόνου εκτέλεσης έγινε επιλεκτικά για συγκεκριμένα στιγμιότυπα των αλγορίθμων, όπως αναφέρεται και στη συνέχεια της αναφοράς.

Παρατήρηση:

Ο έλεγχος λειτουργίας όλων των προγραμμάτων έγινε με τα πλήρη αρχεία των μετοχών. Εξαιτίας όμως του μεγάλου μεγέθους των αρχείων, πολλά από τα αποτελέσματα και τα screenshots που δίνονται παρακάτω αφορούν στο αρχείο `agn2005.us.txt` το οποίο περιλαμβάνει μόνο τις ημερήσιες κινήσεις του 2005 (252 data records).

I.1 mergeSort, quicksort

Πρόγραμμα PROJECT_PART_I_1

- Για τη σύγκριση των data records χρησιμοποιείται η συνάρτηση `cmpOpenDate()` που συγκρίνει τα ζεύγη τιμών των πεδίων `Open` και `Date` των δύο data records και επιστρέφει 1, 0 ή -1 ανάλογα με τη διάταξη των ζευγών.
- Για κάθε αλγόριθμο μετράμε τις συγκρίσεις μεταξύ των στοιχείων μέσω της global μεταβλητής `comps` καθώς και τον χρόνο εκτέλεσης του αλγόριθμου. Το πλήθος των συγκρίσεων και ο χρόνος εκτέλεσης τυπώνονται μετά την κλήση κάθε αλγόριθμου. Επισημαίνεται ότι οι μετρούμενοι χρόνοι εκτέλεσης δεν είναι ακριβείς και αλλάζουν σε κάθε run, εντούτοις δίνουν μια ενδεικτική εικόνα της συμπεριφοράς του αλγόριθμου.
- Αναφορικά με τον quicksort υλοποιήθηκαν τρεις παραλλαγές, `standard`, `randomized` και `medOfThree`.

Στη συνέχεια παρατίθενται ενδεικτικά screenshots με τα περιεχόμενα του ταξινομημένου πίνακα μετά την κλήση κάθε αλγόριθμου. Λόγω μεγέθους των αρχείων δεδομένων, τα screenshots απεικονίζουν μόνον τις ταξινομημένες

τιμές του πεδίου Open κάθε data record του αρχείου agn2005.us.txt χωρισμένες με το χαρακτήρα '|' (για το σκοπό αυτό χρησιμοποιείται ειδική συνάρτηση printArray() αντί της κανονικής που τυπώνει τα πλήρη στοιχεία κάθε data record κατά γραμμές). Ο χρόνος εκτέλεσης κάθε αλγόριθμου για το αρχείο agn2005.us.txt που υπολογίζεται από τη βιβλιοθήκη chrono είναι 0 λόγω του μικρού μεγέθους της εισόδου. Για το λόγο αυτό οι συγκεκριμένοι χρόνοι δεν τυπώνονται στα screenshots που ακολουθούν.

Ενδεικτικά screenshots

[MERGESORT] SORTED ARRAY:

```
28.048 | 28.663 | 28.742 | 28.821 | 28.900 | 28.990 | 29.029 | 29.039 | 29.039 |
29.158 | 29.158 | 29.198 | 29.207 | 29.217 | 29.237 | 29.267 | 29.326 | 29.326 |
29.366 | 29.366 | 29.376 | 29.426 | 29.426 | 29.426 | 29.446 | 29.456 | 29.456 |
29.485 | 29.505 | 29.505 | 29.515 | 29.525 | 29.604 | 29.614 | 29.614 | 29.624 |
29.653 | 29.683 | 29.703 | 29.713 | 29.723 | 29.753 | 29.763 | 29.783 | 29.802 |
29.842 | 29.852 | 29.872 | 29.872 | 29.921 | 29.921 | 29.951 | 29.961 | 29.980 |
30.030 | 30.040 | 30.060 | 30.060 | 30.070 | 30.090 | 30.129 | 30.139 | 30.169 |
30.179 | 30.189 | 30.229 | 30.229 | 30.278 | 30.298 | 30.327 | 30.337 | 30.357 |
30.407 | 30.447 | 30.456 | 30.506 | 30.506 | 30.506 | 30.526 | 30.546 | 30.576 |
30.724 | 30.724 | 30.724 | 30.823 | 30.843 | 30.843 | 30.873 | 30.873 | 30.873 |
30.971 | 31.001 | 31.021 | 31.021 | 31.031 | 31.041 | 31.071 | 31.120 | 31.140 |
31.190 | 31.260 | 31.260 | 31.299 | 31.318 | 31.318 | 31.328 | 31.358 | 31.368 |
31.517 | 31.527 | 31.537 | 31.547 | 31.606 | 31.616 | 31.695 | 31.735 | 31.794 |
32.012 | 32.062 | 32.082 | 32.111 | 32.231 | 32.290 | 32.300 | 32.310 | 32.340 |
32.428 | 32.448 | 32.498 | 32.498 | 32.508 | 32.528 | 32.567 | 32.567 | 32.577 |
32.597 | 32.647 | 32.647 | 32.765 | 32.805 | 32.825 | 32.855 | 32.904 | 32.924 |
33.063 | 33.092 | 33.112 | 33.142 | 33.202 | 33.311 | 33.401 | 33.419 | 33.419 |
33.519 | 33.618 | 33.688 | 33.698 | 33.698 | 33.816 | 33.955 | 33.995 | 34.045 |
34.262 | 34.262 | 34.272 | 34.272 | 34.292 | 34.292 | 34.312 | 34.322 | 34.342 |
34.411 | 34.431 | 34.441 | 34.470 | 34.480 | 34.490 | 34.500 | 34.529 | 34.549 |
34.589 | 34.609 | 34.629 | 34.639 | 34.689 | 34.699 | 34.709 | 34.738 | 34.778 |
34.856 | 35.075 | 35.085 | 35.085 | 35.085 | 35.135 | 35.183 | 35.203 | 35.283 |
35.303 | 35.333 | 35.402 | 35.452 | 35.550 | 35.560 | 36.076 | 36.086 | 36.135 |
```

Number of comparisons: 1448

Process exited after 0.03466 seconds with return value 0
Press any key to continue . . .

[STANDARD QUICKSORT] SORTED ARRAY:

```
28.048 | 28.663 | 28.742 | 28.821 | 28.900 | 28.990 | 29.029 | 29.039 | 29.039 |
29.158 | 29.158 | 29.198 | 29.207 | 29.217 | 29.237 | 29.267 | 29.326 | 29.326 |
29.366 | 29.366 | 29.376 | 29.426 | 29.426 | 29.426 | 29.446 | 29.456 | 29.456 |
29.485 | 29.505 | 29.505 | 29.515 | 29.525 | 29.604 | 29.614 | 29.614 | 29.624 |
29.653 | 29.683 | 29.703 | 29.713 | 29.723 | 29.753 | 29.763 | 29.783 | 29.802 |
29.842 | 29.852 | 29.872 | 29.872 | 29.921 | 29.921 | 29.951 | 29.961 | 29.980 |
30.030 | 30.040 | 30.060 | 30.060 | 30.070 | 30.090 | 30.129 | 30.139 | 30.169 |
30.179 | 30.189 | 30.229 | 30.229 | 30.278 | 30.298 | 30.327 | 30.337 | 30.357 |
30.407 | 30.447 | 30.456 | 30.506 | 30.506 | 30.506 | 30.526 | 30.546 | 30.576 |
30.724 | 30.724 | 30.724 | 30.823 | 30.843 | 30.843 | 30.873 | 30.873 | 30.873 |
30.971 | 31.001 | 31.021 | 31.021 | 31.031 | 31.041 | 31.071 | 31.120 | 31.140 |
31.190 | 31.260 | 31.260 | 31.299 | 31.318 | 31.318 | 31.328 | 31.358 | 31.368 |
31.517 | 31.527 | 31.537 | 31.547 | 31.606 | 31.616 | 31.695 | 31.735 | 31.794 |
32.012 | 32.062 | 32.082 | 32.111 | 32.231 | 32.290 | 32.300 | 32.310 | 32.340 |
32.428 | 32.448 | 32.498 | 32.498 | 32.508 | 32.528 | 32.567 | 32.567 | 32.577 |
32.597 | 32.647 | 32.647 | 32.765 | 32.805 | 32.825 | 32.855 | 32.904 | 32.924 |
33.063 | 33.092 | 33.112 | 33.142 | 33.202 | 33.311 | 33.401 | 33.419 | 33.419 |
33.519 | 33.618 | 33.688 | 33.698 | 33.698 | 33.816 | 33.955 | 33.995 | 34.045 |
34.262 | 34.262 | 34.272 | 34.272 | 34.292 | 34.292 | 34.312 | 34.322 | 34.342 |
34.411 | 34.431 | 34.441 | 34.470 | 34.480 | 34.490 | 34.500 | 34.529 | 34.549 |
34.589 | 34.609 | 34.629 | 34.639 | 34.689 | 34.699 | 34.709 | 34.738 | 34.778 |
34.856 | 35.075 | 35.085 | 35.085 | 35.085 | 35.135 | 35.183 | 35.203 | 35.283 |
35.303 | 35.333 | 35.402 | 35.452 | 35.550 | 35.560 | 36.076 | 36.086 | 36.135 |
```

Number of comparisons: 2865

Process exited after 0.03494 seconds with return value 0
Press any key to continue . . .

[RANDOMIZED QUICKSORT] SORTED ARRAY:

```
28.048 | 28.663 | 28.742 | 28.821 | 28.900 | 28.990 | 29.029 | 29.039 | 29.039 |
29.158 | 29.158 | 29.198 | 29.207 | 29.217 | 29.237 | 29.267 | 29.326 | 29.326 |
29.366 | 29.366 | 29.376 | 29.426 | 29.426 | 29.426 | 29.446 | 29.456 | 29.456 |
29.485 | 29.505 | 29.505 | 29.515 | 29.525 | 29.604 | 29.614 | 29.614 | 29.624 |
29.653 | 29.683 | 29.703 | 29.713 | 29.723 | 29.753 | 29.763 | 29.783 | 29.802 |
29.842 | 29.852 | 29.872 | 29.872 | 29.921 | 29.921 | 29.951 | 29.961 | 29.980 |
30.030 | 30.040 | 30.060 | 30.060 | 30.070 | 30.090 | 30.129 | 30.139 | 30.169 |
30.179 | 30.189 | 30.229 | 30.229 | 30.278 | 30.298 | 30.327 | 30.337 | 30.357 |
30.407 | 30.447 | 30.456 | 30.506 | 30.506 | 30.506 | 30.526 | 30.546 | 30.576 |
30.724 | 30.724 | 30.724 | 30.823 | 30.843 | 30.843 | 30.873 | 30.873 | 30.873 |
30.971 | 31.001 | 31.021 | 31.021 | 31.031 | 31.041 | 31.071 | 31.120 | 31.140 |
31.190 | 31.260 | 31.260 | 31.299 | 31.318 | 31.318 | 31.328 | 31.358 | 31.368 |
31.517 | 31.527 | 31.537 | 31.547 | 31.606 | 31.616 | 31.695 | 31.735 | 31.794 |
32.012 | 32.062 | 32.082 | 32.111 | 32.231 | 32.290 | 32.300 | 32.310 | 32.340 |
32.428 | 32.448 | 32.498 | 32.498 | 32.508 | 32.528 | 32.567 | 32.567 | 32.577 |
32.597 | 32.647 | 32.647 | 32.765 | 32.805 | 32.825 | 32.855 | 32.904 | 32.924 |
33.063 | 33.092 | 33.112 | 33.142 | 33.202 | 33.311 | 33.401 | 33.419 | 33.419 |
33.519 | 33.618 | 33.688 | 33.698 | 33.698 | 33.816 | 33.955 | 33.995 | 34.045 |
34.262 | 34.262 | 34.272 | 34.272 | 34.292 | 34.292 | 34.312 | 34.322 | 34.342 |
34.411 | 34.431 | 34.441 | 34.470 | 34.480 | 34.490 | 34.500 | 34.529 | 34.549 |
34.589 | 34.609 | 34.629 | 34.639 | 34.689 | 34.699 | 34.709 | 34.738 | 34.778 |
34.856 | 35.075 | 35.085 | 35.085 | 35.085 | 35.135 | 35.183 | 35.203 | 35.283 |
35.303 | 35.333 | 35.402 | 35.452 | 35.550 | 35.560 | 36.076 | 36.086 | 36.135 |
```

Number of comparisons: 2311

Process exited after 0.03381 seconds with return value 0
Press any key to continue . . .

[MEDIAN OF THREE QUICKSORT] SORTED ARRAY:

```
28.048 | 28.663 | 28.742 | 28.821 | 28.900 | 28.990 | 29.029 | 29.039 | 29.039 |
29.158 | 29.158 | 29.198 | 29.207 | 29.217 | 29.237 | 29.267 | 29.326 | 29.326 |
29.366 | 29.366 | 29.376 | 29.426 | 29.426 | 29.426 | 29.446 | 29.456 | 29.456 |
29.485 | 29.505 | 29.505 | 29.515 | 29.525 | 29.604 | 29.614 | 29.614 | 29.624 |
29.653 | 29.683 | 29.703 | 29.713 | 29.723 | 29.753 | 29.763 | 29.783 | 29.802 |
29.842 | 29.852 | 29.872 | 29.872 | 29.921 | 29.921 | 29.951 | 29.961 | 29.980 |
30.030 | 30.040 | 30.060 | 30.060 | 30.070 | 30.090 | 30.129 | 30.139 | 30.169 |
30.179 | 30.189 | 30.229 | 30.229 | 30.278 | 30.298 | 30.327 | 30.337 | 30.357 |
30.407 | 30.447 | 30.456 | 30.506 | 30.506 | 30.506 | 30.526 | 30.546 | 30.576 |
30.724 | 30.724 | 30.724 | 30.823 | 30.843 | 30.843 | 30.873 | 30.873 | 30.873 |
30.971 | 31.001 | 31.021 | 31.021 | 31.031 | 31.041 | 31.071 | 31.120 | 31.140 |
31.190 | 31.260 | 31.260 | 31.299 | 31.318 | 31.318 | 31.328 | 31.358 | 31.368 |
31.517 | 31.527 | 31.537 | 31.547 | 31.606 | 31.616 | 31.695 | 31.735 | 31.794 |
32.012 | 32.062 | 32.082 | 32.111 | 32.231 | 32.290 | 32.300 | 32.310 | 32.340 |
32.428 | 32.448 | 32.498 | 32.498 | 32.508 | 32.528 | 32.567 | 32.567 | 32.577 |
32.597 | 32.647 | 32.647 | 32.765 | 32.805 | 32.825 | 32.855 | 32.904 | 32.924 |
33.063 | 33.092 | 33.112 | 33.142 | 33.202 | 33.311 | 33.401 | 33.419 | 33.419 |
33.519 | 33.618 | 33.688 | 33.698 | 33.698 | 33.816 | 33.955 | 33.995 | 34.045 |
34.262 | 34.262 | 34.272 | 34.272 | 34.292 | 34.292 | 34.312 | 34.322 | 34.342 |
34.411 | 34.431 | 34.441 | 34.470 | 34.480 | 34.490 | 34.500 | 34.529 | 34.549 |
34.589 | 34.609 | 34.629 | 34.639 | 34.689 | 34.699 | 34.709 | 34.738 | 34.778 |
34.856 | 35.075 | 35.085 | 35.085 | 35.085 | 35.135 | 35.183 | 35.203 | 35.283 |
35.303 | 35.333 | 35.402 | 35.452 | 35.550 | 35.560 | 36.076 | 36.086 | 36.135 |
```

Number of comparisons: 2411

Process exited after 0.03392 seconds with return value 0
Press any key to continue . . .

Συμπεριφορά κάθε αλγόριθμου

- Ο mergeSort για n στοιχεία έχει χρονική πολυπλοκότητα μέσης και χειρότερης περίπτωσης $\Theta(n \log n)$, είναι δηλ. ασυμπτωτικά βέλτιστος αλλά χρησιμοποιεί $\Theta(n)$ έξτρα χώρο.

Τα αποτελέσματα της θεωρητικής ανάλυσης επιβεβαιώνονται από τα αποτελέσματα εκτέλεσης:

Πλήθος συγκρίσεων

- Για το αρχείο agn2005.us.txt με τα 252 data records το πλήθος των συγκρίσεων που εκτελούνται είναι 1.448.
- Για το αρχείο agn.us.txt με τα 3.239 data records το πλήθος των συγκρίσεων που εκτελούνται είναι 25.799. Για τα αρχεία ainv.us.txt και ale.us.txt με τα 3.201 data records έκαστο, το πλήθος των συγκρίσεων είναι 26.727 και 27.636 αντίστοιχα.

Οι λιγότερες συγκρίσεις εκτελούνται όταν ο πίνακας είναι ήδη ταξινομημένος κατά αύξουσα ή φθίνουσα σειρά.

Πράγματι, οι συγκρίσεις που κάνει ο mergeSort για την ταξινόμηση του ήδη ταξινομημένου με βάση το πεδίο Date αρχείου agn2005.us.txt είναι 996 ενώ για την ταξινόμηση του αρχείου agn.us.txt είναι 18.325 και των αρχείων ainv.us.txt και ale.us.txt 18.115.

Χρόνος εκτέλεσης για ταξινόμηση με βάση το πεδίο Open

Αρχείο agn.us.txt: 0,0009994 secs

Αρχείο ainv.us.txt: 0,0009984 secs

Αρχείο ale.us.txt: 0,0009993 secs

Χρόνος εκτέλεσης για ταξινόμηση του ήδη ταξινομημένου με βάση το πεδίο Date αρχείου

Αρχείο agn.us.txt: 0,0009996 secs

Αρχείο ainv.us.txt: 0,0009983 secs

Αρχείο ale.us.txt: 0,0009989 secs

- Ο standard quickSort για n στοιχεία έχει χρονική πολυπλοκότητα μέσης περίπτωσης $\Theta(n \log n)$ ($\leq 1,44 * (n+1) * \log n$) και χειρότερης περίπτωσης $O(n^2)$ ($\leq (n+1) * (n+2) / 2 - 3$). Η χειρότερη περίπτωση συμβαίνει όταν ο πίνακας είναι ήδη ταξινομημένος κατά αύξουσα ή φθίνουσα σειρά.

[Όπου στην αναφορά γράφεται $\log x$ ή $\log \log x$ εννοείται το ακέραιο μέρος τους.]

Στην περίπτωση μας, τα data records των αρχείων είναι τυχαία κατανομημένα με βάση το πεδίο Open. Επομένως, περιμένουμε και οι τρεις παραλλαγές του QuickSort να έχουν τη χρονική πολυπλοκότητα της μέσης περίπτωσης.

Πράγματι, οι συγκρίσεις που κάνει κάθε αλγόριθμος επιβεβαιώνουν τον ισχυρισμό μας:

Πλήθος συγκρίσεων

Αρχείο agn2005.us.txt:

- standard quickSort: $2.865 < 1,44 \cdot 253 \cdot \log 252 = 2.906$
- randomizedQuickSort: 2.648
- medOfThreeQuickSort: 2.411

Αρχείο agn.us.txt:

- standard quickSort: $194.435 < 4 \cdot 1,44 \cdot 3.240 \cdot \log 3.239 = 4 \cdot 54.408 = 217.632$
- randomizedQuickSort: $50.810 < 1,44 \cdot 3.240 \cdot \log 3.239 = 54.408$
- medOfThreeQuickSort: $53.778 < 1,44 \cdot 3.240 \cdot \log 3.239$

Αρχείο ainv.us.txt:

- standard quickSort: $49.417 < 1,44 \cdot 3.202 \cdot \log 3.201 = 53.691$
- randomizedQuickSort: $45.498 < 1,44 \cdot 3.202 \cdot \log 3.201$
- medOfThreeQuickSort: $46.265 < 1,44 \cdot 3.202 \cdot \log 3.201$

Αρχείο ale.us.txt:

- standard quickSort: $75.210 < 1,5 \cdot 1,44 \cdot 3.202 \cdot \log 3.201 = 1,5 \cdot 53.691 = 80.537$
- randomizedQuickSort: $46.715 < 1,44 \cdot 3.202 \cdot \log 3.201 = 53.691$
- medOfThreeQuickSort: $45.781 < 1,44 \cdot 3.202 \cdot \log 3.201$

Οι συγκρίσεις που κάνει κάθε αλγόριθμος για την ταξινόμηση του ήδη ταξινομημένου με βάση το πεδίο Date αρχείου agn2005.us.txt έχουν ως εξής:

- standard quickSort: 32.128 που είναι το ακριβές αποτέλεσμα της θεωρητικής ανάλυσης $(n+1) \cdot (n+2) / 2 - 3$ για $n = 252$
- randomizedQuickSort: 2.304
- medOfThreeQuickSort: 2.143

Για τα υπόλοιπα αρχεία οι συγκρίσεις είναι:

Αρχείο agn.us.txt

- standard quickSort: 5.250.417 που είναι το ακριβές αποτέλεσμα της θεωρητικής ανάλυσης $(n+1)*(n+2)/2-3$ για $n = 3.239$
- randomizedQuickSort: 44.339
- medOfThreeQuickSort: 39.213

Αρχείο ainv.us.txt:

- standard quickSort: 5.128.000 που είναι το ακριβές αποτέλεσμα της θεωρητικής ανάλυσης $(n+1)*(n+2)/2-3$ για $n = 3.201$
- randomizedQuickSort: 44.225
- medOfThreeQuickSort: 38.681

Αρχείο ale.us.txt:

- standard quickSort: 5.128.000
- randomizedQuickSort: 45.314
- medOfThreeQuickSort: 38.681

Όπως, φαίνεται παραπάνω, και οι δύο παραλλαγές randomizedQuickSort και medOfThreeQuickSort παρουσιάζουν συμπεριφορά μέσης περίπτωσης.

Επισημαίνουμε εδώ ότι το πλήθος των συγκρίσεων που εκτελεί ο randomizedQuickSort σε κάθε run αλλάζει εξαιτίας ακριβώς του randomization (μία 'κακή' τυχαία επιλογή του ρίντο στοιχείου μπορεί να δώσει αυξημένο πλήθος συγκρίσεων).

Συνοψίζοντας, μπορούμε να πούμε ότι ο medOfThreeQuickSort υπερτερεί σε σχέση με τις άλλες δύο παραλλαγές.

Χρόνος εκτέλεσης για ταξινόμηση με βάση το πεδίο Open

Αρχείο agn.us.txt:

- standard quickSort: 0,0010002 secs
- randomizedQuickSort: 0,0009994 secs
- medOfThreeQuickSort: 0,0009992 secs

Αρχείο ainv.us.txt:

- standard quickSort: 0,001006 secs
- randomizedQuickSort: 0,0009992 secs
- medOfThreeQuickSort: 0,0009993 secs

Αρχείο ale.us.txt:

- standard quickSort: 0,0009995 secs
- randomizedQuickSort: 0,0009989 secs
- medOfThreeQuickSort: 0,001 secs

Χρόνος εκτέλεσης για ταξινόμηση του ήδη ταξινομημένου με βάση το πεδίο Date αρχείου

Αρχείο agn.us.txt:

- standard quickSort: 0,0645686 secs
- randomizedQuickSort: 0,0009992 secs
- medOfThreeQuickSort: 0,0009994 secs

Αρχείο ainv.us.txt:

- standard quickSort: 0,0629803 secs
- randomizedQuickSort: 0,0009994 secs
- medOfThreeQuickSort: 0,0009993 secs

Αρχείο ale.us.txt:

- standard quickSort: 0,0640277 secs
- randomizedQuickSort: 0,0010001 secs
- medOfThreeQuickSort: 0,0009994 secs

Παρότι οι μετρούμενοι χρόνοι δεν είναι ακριβείς, δίνουν μια ενδεικτική εικόνα της συμπεριφοράς κάθε αλγόριθμου.

I.2 heapSort, countingSort

Πρόγραμμα PROJECT_PART_I_2

- Για τη σύγκριση των data records ο heapsort χρησιμοποιεί τη συνάρτηση cmpCloseDate() που συγκρίνει τα ζεύγη τιμών των πεδίων Open και Date των δύο data records και επιστρέφει 1, 0 ή -1 ανάλογα με τη διάταξη των ζευγών.
- Στον heapsort μετράμε μετράμε τις συγκρίσεις μεταξύ των στοιχείων μέσω της global μεταβλητής comps καθώς και τον χρόνο εκτέλεσης του

αλγόριθμου. Το πλήθος των συγκρίσεων και ο χρόνος εκτέλεσης τυπώνονται μετά την κλήση του αλγόριθμου.

- Για τον countingSort υλοποιήθηκαν δύο παραλλαγές, standard και extended.

Ο standard countingSort λειτουργεί πάνω στις ακέραιες τιμές του πεδίου Close αλλά δεν παράγει σωστό ταξινομημένο πίνακα (πχ. τα data records με τιμές Close 35,510, 35,243, 35,210 εμφανίζονται στον ταξινομημένο πίνακα με αυτήν τη σειρά).

Ο extended countingSort λειτουργεί πάνω στις τιμές $1.000 * \text{Close}$ που είναι ακέραιες δεδομένου ότι οι τιμές Close των data records έχουν τρία δεκαδικά ψηφία. Έτσι, παράγει σωστό ταξινομημένο πίνακα εις βάρος βέβαια του μεγαλύτερου χώρου που απαιτείται για τον πίνακα C ο οποίος έχει μέγεθος 1.000 φορές μεγαλύτερο από το μέγεθος του πίνακα C του standard countingSort. [Ακριβέστερα ισχύει: $1.000 * [(\text{int}(\text{max}) - \text{int}(\text{min}) - 1) + 1] \leq \text{μέγεθος πίνακα C} \leq 1.000 * [(\text{int}(\text{max}) - \text{int}(\text{min}) + 1) - 1]$, όπου max, min είναι η μεγαλύτερη και μικρότερη δεκαδική τιμή και $(\text{int})\text{max}$, $(\text{int})\text{min}$ η μεγαλύτερη και μικρότερη ακέραια τιμή (ακέραιο μέρος της δεκαδικής τιμής) του πεδίου Close αντίστοιχα.]

Και στις δύο παραλλαγές μετράμε και τυπώνουμε το χρόνο εκτέλεσης του αλγόριθμου.

Στη συνέχεια παρατίθενται ενδεικτικά screenshots με τα περιεχόμενα του ταξινομημένου πίνακα μετά την κλήση κάθε αλγόριθμου. Λόγω μεγέθους των αρχείων δεδομένων, τα screenshots απεικονίζουν μόνον τις ταξινομημένες τιμές του πεδίου Close κάθε data record του αρχείου agn2005.us.txt. Στα screenshots δεν περιλαμβάνονται οι χρόνοι εκτέλεσης που υπολογίζονται από τη chrono και οι οποίοι, λόγω μικρού μεγέθους εισόδου, είναι 0.

Ενδεικτικά screenshots

```
[HEAPSORT] SORTED ARRAY:
27.859 | 28.226 | 28.663 | 28.880 | 29.019 | 29.039 | 29.069 | 29.079 | 29.079 |
29.148 | 29.217 | 29.227 | 29.257 | 29.287 | 29.297 | 29.317 | 29.317 | 29.326 |
29.336 | 29.336 | 29.336 | 29.336 | 29.346 | 29.386 | 29.396 | 29.406 | 29.406 |
29.465 | 29.485 | 29.485 | 29.505 | 29.564 | 29.614 | 29.624 | 29.634 | 29.634 |
29.683 | 29.683 | 29.703 | 29.723 | 29.733 | 29.733 | 29.743 | 29.783 | 29.783 |
29.832 | 29.842 | 29.852 | 29.862 | 29.872 | 29.882 | 29.892 | 29.931 | 29.961 |
30.050 | 30.060 | 30.070 | 30.080 | 30.080 | 30.100 | 30.100 | 30.110 | 30.110 |
30.159 | 30.179 | 30.209 | 30.219 | 30.229 | 30.229 | 30.229 | 30.239 | 30.268 |
30.357 | 30.377 | 30.377 | 30.377 | 30.456 | 30.496 | 30.506 | 30.546 | 30.566 |
30.674 | 30.704 | 30.764 | 30.783 | 30.803 | 30.843 | 30.863 | 30.863 | 30.863 |
30.943 | 30.952 | 30.952 | 31.021 | 31.061 | 31.091 | 31.101 | 31.110 | 31.120 |
31.250 | 31.260 | 31.260 | 31.299 | 31.299 | 31.318 | 31.318 | 31.338 | 31.378 |
31.457 | 31.467 | 31.547 | 31.587 | 31.596 | 31.606 | 31.606 | 31.616 | 31.824 |
32.072 | 32.092 | 32.151 | 32.161 | 32.171 | 32.221 | 32.300 | 32.330 | 32.350 |
32.448 | 32.448 | 32.478 | 32.508 | 32.508 | 32.528 | 32.577 | 32.577 | 32.597 |
32.667 | 32.755 | 32.765 | 32.785 | 32.805 | 32.835 | 32.904 | 32.944 | 33.004 |
33.024 | 33.063 | 33.063 | 33.102 | 33.112 | 33.221 | 33.361 | 33.381 | 33.459 |
33.519 | 33.529 | 33.558 | 33.678 | 33.747 | 33.786 | 33.975 | 34.065 | 34.143 |
34.262 | 34.282 | 34.322 | 34.342 | 34.352 | 34.362 | 34.362 | 34.362 | 34.362 |
34.431 | 34.431 | 34.441 | 34.451 | 34.460 | 34.500 | 34.539 | 34.539 | 34.559 |
34.619 | 34.619 | 34.629 | 34.639 | 34.659 | 34.689 | 34.689 | 34.798 | 34.807 |
34.886 | 34.906 | 34.946 | 34.996 | 34.996 | 35.016 | 35.065 | 35.173 | 35.233 |
35.263 | 35.333 | 35.383 | 35.510 | 35.530 | 35.550 | 36.076 | 36.204 | 36.214 |

Number of comparisons: 3277

-----
Process exited after 0.03532 seconds with return value 0
Press any key to continue . . .
```

```
[COUNTINGSORT] SORTED ARRAY:
27.859 | 28.663 | 28.226 | 28.880 | 29.723 | 29.346 | 29.287 | 29.336 | 29.653 |
29.336 | 29.614 | 29.039 | 29.069 | 29.019 | 29.872 | 29.882 | 29.406 | 29.317 |
29.892 | 29.783 | 29.624 | 29.505 | 29.683 | 29.733 | 29.961 | 29.931 | 29.634 |
29.326 | 29.386 | 29.227 | 29.485 | 29.783 | 29.733 | 29.832 | 29.406 | 29.842 |
29.852 | 29.703 | 29.297 | 29.148 | 29.336 | 29.217 | 29.079 | 29.317 | 29.336 |
29.743 | 29.634 | 29.257 | 29.426 | 29.465 | 29.446 | 29.336 | 29.079 | 29.832 |
30.863 | 30.377 | 30.070 | 30.129 | 30.239 | 30.496 | 30.080 | 30.100 | 30.377 |
30.952 | 30.803 | 30.546 | 30.357 | 30.783 | 30.456 | 30.327 | 30.030 | 30.873 |
30.229 | 30.110 | 30.229 | 30.060 | 30.268 | 30.110 | 30.000 | 30.080 | 30.159 |
30.209 | 30.219 | 30.179 | 30.704 | 30.863 | 30.883 | 30.952 | 30.576 | 30.943 |
30.644 | 30.229 | 30.307 | 31.616 | 31.338 | 31.260 | 31.120 | 31.250 | 31.824 |
31.180 | 31.824 | 31.943 | 31.547 | 31.596 | 31.318 | 31.398 | 31.260 | 31.110 |
31.467 | 31.587 | 31.299 | 31.606 | 31.299 | 31.447 | 31.021 | 31.130 | 31.091 |
32.161 | 32.597 | 32.478 | 32.617 | 32.528 | 32.755 | 32.627 | 32.448 | 32.448 |
32.904 | 32.835 | 32.350 | 32.171 | 32.785 | 32.765 | 32.944 | 32.805 | 32.577 |
32.072 | 32.300 | 32.577 | 32.399 | 32.409 | 32.667 | 32.330 | 32.221 | 33.102 |
33.747 | 33.519 | 33.558 | 33.786 | 33.529 | 33.975 | 33.678 | 33.004 | 33.221 |
33.459 | 33.499 | 33.063 | 33.361 | 33.063 | 33.014 | 33.014 | 34.342 | 34.629 |
34.659 | 34.451 | 34.322 | 34.362 | 34.639 | 34.569 | 34.619 | 34.836 | 34.392 |
34.173 | 34.143 | 34.431 | 34.500 | 34.539 | 34.946 | 34.559 | 34.539 | 34.996 |
34.886 | 34.362 | 34.282 | 34.262 | 34.807 | 34.689 | 34.362 | 34.362 | 34.352 |
34.996 | 34.827 | 34.252 | 34.441 | 34.431 | 35.383 | 35.510 | 35.243 | 35.065 |
35.233 | 35.233 | 35.016 | 35.530 | 35.333 | 35.173 | 36.284 | 36.204 | 36.214 |

-----
Process exited after 0.03594 seconds with return value 0
Press any key to continue . . .
```

[Όπως φαίνεται στο screenshot, λόγω των δεκαδικών ψηφίων οι τιμές του πεδίου Close δεν ταξινομούνται ορθά με τον standard countingSort.]

[COUNTINGSORT EXTENDED] SORTED ARRAY:

```
27.859 | 28.226 | 28.663 | 28.880 | 29.019 | 29.039 | 29.069 | 29.079 | 29.079 |
29.148 | 29.217 | 29.227 | 29.257 | 29.287 | 29.297 | 29.317 | 29.317 | 29.326 |
29.336 | 29.336 | 29.336 | 29.336 | 29.346 | 29.386 | 29.396 | 29.406 | 29.406 |
29.465 | 29.485 | 29.485 | 29.505 | 29.564 | 29.614 | 29.624 | 29.634 | 29.634 |
29.683 | 29.683 | 29.703 | 29.723 | 29.733 | 29.733 | 29.743 | 29.783 | 29.783 |
29.832 | 29.842 | 29.852 | 29.862 | 29.872 | 29.882 | 29.892 | 29.931 | 29.961 |
30.050 | 30.060 | 30.070 | 30.080 | 30.080 | 30.100 | 30.100 | 30.110 | 30.110 |
30.159 | 30.179 | 30.209 | 30.219 | 30.229 | 30.229 | 30.229 | 30.239 | 30.268 |
30.357 | 30.377 | 30.377 | 30.456 | 30.496 | 30.506 | 30.546 | 30.566 | 30.576 |
30.674 | 30.704 | 30.764 | 30.783 | 30.803 | 30.843 | 30.863 | 30.863 | 30.863 |
30.943 | 30.952 | 30.952 | 31.021 | 31.061 | 31.091 | 31.101 | 31.110 | 31.120 |
31.250 | 31.260 | 31.260 | 31.299 | 31.299 | 31.318 | 31.318 | 31.338 | 31.378 |
31.457 | 31.467 | 31.547 | 31.587 | 31.596 | 31.606 | 31.606 | 31.616 | 31.824 |
32.072 | 32.092 | 32.151 | 32.161 | 32.171 | 32.221 | 32.300 | 32.330 | 32.350 |
32.448 | 32.448 | 32.478 | 32.508 | 32.508 | 32.528 | 32.577 | 32.577 | 32.597 |
32.667 | 32.755 | 32.765 | 32.785 | 32.805 | 32.835 | 32.904 | 32.944 | 33.004 |
33.024 | 33.063 | 33.063 | 33.102 | 33.112 | 33.221 | 33.361 | 33.381 | 33.459 |
33.519 | 33.529 | 33.558 | 33.678 | 33.747 | 33.786 | 33.975 | 34.065 | 34.143 |
34.262 | 34.282 | 34.322 | 34.342 | 34.352 | 34.362 | 34.362 | 34.362 | 34.362 |
34.431 | 34.431 | 34.441 | 34.451 | 34.460 | 34.500 | 34.539 | 34.539 | 34.559 |
34.619 | 34.619 | 34.629 | 34.639 | 34.659 | 34.689 | 34.689 | 34.798 | 34.807 |
34.886 | 34.906 | 34.946 | 34.996 | 34.996 | 35.016 | 35.065 | 35.173 | 35.233 |
35.263 | 35.333 | 35.383 | 35.510 | 35.530 | 35.550 | 36.076 | 36.204 | 36.214 |
```

Process exited after 0.03708 seconds with return value 0
Press any key to continue . . .

[Με τον extended CountingSort οι τιμές του πεδίου Close ταξινομούνται ορθά.]

Συμπεριφορά κάθε αλγόριθμου

- Κάθε φάση του heapsort (Δόμησης και Διαλογής που καλούν τη συνάρτηση `shiftDown()`) για n στοιχεία έχει χρονική πολυπλοκότητα χειρότερης περίπτωσης $\Theta(n \log n)$, άρα η χρονική πολυπλοκότητα χειρότερης περίπτωσης του αλγόριθμου είναι $\Theta(n \log n)$ (ο αλγόριθμος είναι ασυμπτωτικά βέλτιστος).

Τα αποτελέσματα της θεωρητικής ανάλυσης επιβεβαιώνονται από τα αποτελέσματα εκτέλεσης του προγράμματος:

Πλήθος συγκρίσεων

- Για το αρχείο `agn2005.us.txt` με τα 252 data records το πλήθος των συγκρίσεων που εκτελούνται είναι 3.277.
- Για το αρχείο `agn.us.txt` με τα 3.239 data records το πλήθος των συγκρίσεων που εκτελούνται είναι 66.516. Για τα αρχεία `ainv.us.txt` και `ale.us.txt` με τα 3.201 data records έκαστο, το πλήθος των συγκρίσεων είναι 62.751 και 64.671 αντίστοιχα.

Χρόνος εκτέλεσης

Αρχείο `agn.us.txt`: 0,0009995 secs

Αρχείο `ainv.us.txt`: 0,0010006 secs

Αρχείο `ale.us.txt`: 0,0010018 secs

- Ο `extendedCountingSort` για n στοιχεία έχει χρονική πολυπλοκότητα χειρότερης περίπτωσης $\Theta(n+k)$, όπου $k = (\text{int})\text{max} - (\text{int})\text{min}$, max , min είναι η μεγαλύτερη και μικρότερη δεκαδική τιμή και $(\text{int})\text{max}$, $(\text{int})\text{min}$ η μεγαλύτερη και μικρότερη ακέραια τιμή (ακέραιο μέρος της δεκαδικής τιμής) του πεδίου `Close` αντίστοιχα.

Πράγματι, έχουμε:

Πλήθος βημάτων

Ο αλγόριθμος εκτελεί ακριβώς $4*n+2*[(\text{int})\text{max}-(\text{int})\text{min}]+1$ βήματα, όπου κάθε βήμα αντιστοιχεί σε 4 το πολύ απλές πράξεις που μπορεί να είναι συγκρίσεις ή προσθαιρέσεις ή καταχωρήσεις ακεραίων.

Συγκεκριμένα:

- Για το αρχείο `agn2005.us.txt` με τα 252 data records ισχύει $(\text{int})\text{min} = 27$ και $(\text{int})\text{max} = 36$ και το πλήθος των βημάτων που εκτελούνται είναι 1.027.
- Για το αρχείο `agn.us.txt` με τα 3.239 data records ισχύει $(\text{int})\text{min} = 20$ και $(\text{int})\text{max} = 336$ και το πλήθος των βημάτων που εκτελούνται είναι 13.589.

Χρόνος εκτέλεσης

Αρχείο `agn.us.txt`: 0,0009995 secs

Αρχείο `ainv.us.txt`: 0,0010006 secs

Αρχείο `ale.us.txt`: 0,0010018 secs

Κλείνοντας, αναφέρουμε ότι ο αλγόριθμος χρησιμοποιεί τον πίνακα S και δύο βοηθητικούς πίνακες B και C μεγέθους n και $1.000*(\text{max}-\text{min})+2$ αντίστοιχα, δηλ. συνολικό χώρο $\Theta(n+k)$ με $k = (\text{int})\text{max} - (\text{int})\text{min}$.

I.3 binarySearch, interpolationSearch

Πρόγραμμα PROJECT_PART_I_3

- Στον αλγόριθμο `InterpolationSearch` και προκειμένου να υπολογίσουμε το δείκτη m (probe index) για τη σύγκριση με το στοιχείο αναζήτησης x χρησιμοποιούμε τη συνάρτηση `unsigned long val(char s[])` η οποία μετατρέπει την τιμή `Date` του data record σε ακέραια τιμή αφαιρώντας τον χαρακτήρα `'-'`.

- Σε κάθε αλγόριθμο μετράμε μέσω της global μεταβλητής steps τα βήματα που εκτελεί. Σε κάθε βήμα γίνεται υπολογισμός του probe index m, σύγκριση του στοιχείου S[m] με το στοιχείο αναζήτησης x και ενημέρωση του αριστερού (l) ή του δεξιού (r) ορίου του υποπίνακα στον οποίο θα συνεχιστεί η αναζήτηση στο επόμενο βήμα αν $S[m] \neq x$ (κάθε βήμα αντιστοιχεί σε μία επανάληψη του while loop η οποία απαιτεί $O(1)$ χρόνο).
- Οι χρόνοι εκτέλεσης και των δύο αλγόριθμων στα τρία αρχεία δεδομένων που υπολογίζονται από τη βιβλιοθήκη chrono είναι 0 λόγω του μικρού πλήθους βημάτων που εκτελούνται ($O(\log \log n) - O(\log n)$ για $n \leq 3.239$, έχουμε δηλ. συμπεριφορά μέσης περίπτωσης λόγω της σχεδόν ομοιόμορφης κατανομής των τιμών του πεδίου Date). Για το λόγο αυτό οι συγκεκριμένοι χρόνοι δεν τυπώνονται στα screenshots ακολουθούν.
- Οι χρόνοι εκτέλεσης έχουν φυσική σημασία και τυπώνονται για κάποια συγκεκριμένα στιγμιότυπα εισόδου που έχουμε δημιουργήσει για τα οποία ο interpolationSearch αλγόριθμος παρουσιάζει συμπεριφορά χειρότερης περίπτωσης, όπως αναλύεται στο αντίστοιχο τμήμα της αναφοράς.

Στη συνέχεια παρατίθενται ενδεικτικά screenshots με τα αποτελέσματα αναζήτησης κάθε αλγόριθμου στα αρχεία agn2005.us.txt και agn.us.txt.

Ενδεικτικά screenshots

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2006-01-03

[BINARY SEARCH]
Date 2006-01-03 not found.
Number of steps: 8

-----
Process exited after 15.1 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-01-03

[BINARY SEARCH]
Date 2005-01-03 found at array position 0. Volume: 1027044
Number of steps: 7

-----
Process exited after 14.1 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[BINARY SEARCH]

Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 1

-----
Process exited after 26.52 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2011-10-08

[BINARY SEARCH]

Date 2011-10-08 not found.
Number of steps: 12

-----
Process exited after 13.11 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-01-03

[BINARY SEARCH]

Date 2005-01-03 found at array position 0. Volume: 1027044
Number of steps: 11

-----
Process exited after 18.37 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[BINARY SEARCH]

Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 7

-----
Process exited after 15.55 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2006-01-03

[INTERPOLATION SEARCH]
Date 2006-01-03 not found.
Number of steps: 1

-----
Process exited after 10.9 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-01-03

[INTERPOLATION SEARCH]
Date 2005-01-03 found at array position 0. Volume: 1027044
Number of steps: 1

-----
Process exited after 12.38 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[INTERPOLATION SEARCH]
Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 5

-----
Process exited after 9.77 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2011-10-08

[INTERPOLATION SEARCH]
Date 2011-10-08 not found.
Number of steps: 21

-----
Process exited after 16.19 seconds with return value 0
Press any key to continue . . .
```



```

Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-01-03

[INTERPOLATION SEARCH]
Date 2005-01-03 found at array position 0. Volume: 1027044
Number of steps: 1

-----
Process exited after 16.74 seconds with return value 0
Press any key to continue . . .

```

```

Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[INTERPOLATION SEARCH]
Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 20

-----
Process exited after 29.46 seconds with return value 0
Press any key to continue . . .

```

Συμπεριφορά κάθε αλγόριθμου

- Ο αλγόριθμος `binarySearch` για n στοιχεία έχει χρονική πολυπλοκότητα $O(\log n)$. Στην καλύτερη περίπτωση εκτελεί 1 βήμα (επιτυχής αναζήτηση με την πρώτη σύγκριση) και στη χειρότερη περίπτωση $\log n$ βήματα (επιτυχής αναζήτηση με την τελευταία σύγκριση ή ανεπιτυχής αναζήτηση).

Πράγματι, όπως φαίνεται και στα screenshots που δίνονται παραπάνω:

- Για το αρχείο `agn2005.us.txt` με τα 252 data records και στοιχείο αναζήτησης $x = "2006-01-03"$ έχουμε 8 βήματα (ανεπιτυχής αναζήτηση, το x κείται εκτός ορίων του πίνακα) για $x = "2005-01-03"$ 7 βήματα και για $x = "2005-07-01"$ (data record στη μεσαία θέση του πίνακα) 1 βήμα.
- Για το αρχείο `agn.us.txt` με τα 3.239 data records και στοιχείο αναζήτησης $x = "2011-10-08"$ έχουμε 12 βήματα (ανεπιτυχής αναζήτηση) για $x = "2005-01-03"$ 11 βήματα και για $x = "2005-07-01"$ 7 βήματα.

Κατά μέσο όρο αναμένουμε ο αλγόριθμος να κάνει $\log n/2$ βήματα.

Επειδή σε κάθε βήμα ο δείκτης m υπολογίζεται από την ίδια σταθερή φόρμουλα $(l+r)/2$, το πλήθος των βημάτων του εκτελούνται εξαρτάται κυρίως από το στοιχείο αναζήτησης x και όχι από την κατανομή των στοιχείων στον ταξινομημένο πίνακα.

- Ο αλγόριθμος `interpolationSearch` σε κάθε βήμα προσπαθεί να 'μαντέψει' τη θέση του x με βάση την απόστασή του από τα ακριανά στοιχεία $S[1]$ και $S[l]$. Με βάση τα αποτελέσματα της θεωρητικής ανάλυσης, αν τα n στοιχεία του ταξινομημένου πίνακα είναι ομοιόμορφα κατανεμημένα μεταξύ του μικρότερου και του μεγαλύτερου, τότε ο μέσος αριθμός των βημάτων που εκτελούνται είναι $O(\log \log n)$ ($\leq 2,4 * \log \log n + 2$).

Όταν τα στοιχεία του πίνακα δεν ακολουθούν την ομοιόμορφη κατανομή, τότε ο αλγόριθμος μπορεί στη χειρότερη περίπτωση να γίνει γραμμικός ως προς n .

Στην περίπτωσή μας, τα `data records` των αρχείων που δίνονται είναι ταξινομημένα ως το πεδίο `Date` και απέχουν το καθένα από το γειτονικό του 1 ημέρα με εξαίρεση τα Σαβ/κα και τις αργίες που το Χρηματιστήριο είναι κλειστό. Άρα, τα στοιχεία του ταξινομημένου πίνακα είναι περίπου ομοιόμορφα κατανεμημένα μεταξύ της πρώτης ημέρας του 2005 και της τελευταίας ημέρας του 2017 για τις οποίες είχαμε κινήσεις της μετοχής. Το ίδιο ισχύει και για τα `data records` ενός έτους (αρχείο `agn2005.us.txt`).

Με βάση τα παραπάνω περιμένουμε ο αλγόριθμος να παρουσιάζει συμπεριφορά μέσης περίπτωσης. Πράγματι:

- Για το αρχείο `agn2005.us.txt` με τα 252 `data records` το πλήθος των βημάτων που εκτελούνται είναι για $x = "2006-01-03"$ 1 βήμα (ανεπιτυχής αναζήτηση, το x κείται εκτός ορίων του πίνακα), για $x = "2005-01-03"$ 1 βήμα και για $x = "2005-07-01"$ $5 < 2,4 * \log \log 252 + 2$ βήματα.
- Για το αρχείο `agn.us.txt` με τα 3.239 `data records` το πλήθος των βημάτων που εκτελούνται είναι για $x = "2011-10-08"$ $21 \approx 2 * (2,4 * \log \log 3.239 + 2)$ βήματα (ανεπιτυχής αναζήτηση), για $x = "2005-01-03"$ 1 βήμα και για $x = "2005-07-01"$ $20 \approx 2 * (2,4 * \log \log 3.239 + 2)$ βήματα.

Στιγμιότυπο χειρότερης περίπτωσης αλγόριθμου `interpolationSearch`

Το αρχείο `agn2005.us.txt` περιέχει τις ημερήσιες κινήσεις της μετοχής για το έτος 2005 με τελευταία ημερομηνία την '2005-30-12'. Στο τέλος του αρχείου αυτού προσθέτουμε την τελευταία κίνηση του αρχείου `agn.us.txt`, η οποία έχει ημερομηνία '2017-11-10' και δημιουργούμε το αρχείο `agn2005WorstCase.us.txt` (το νέο αρχείο έχει πλέον 253 `data records`).

Αναζητούμε στο νέο αρχείο την ημερομηνία '2005-30-12'. Το πλήθος των βημάτων που εκτελεί πλέον ο `interpolationSearch` είναι 205, δηλ. γραμμικό στο μέγεθος της εισόδου.

Το σχετικό `screenshot` στο οποίο εμφανίζεται και ο χρόνος εκτέλεσης είναι το εξής:

```

Give the stock data filename: agn2005WorstCase.us.txt

Give the date to search for (yyyy-mm-dd): 2005-12-30

[INTERPOLATION SEARCH]
Date 2005-12-30 found at array position 251. Volume: 538695
Number of steps: 205
Running time measured: 0 seconds

-----
Process exited after 27.7 seconds with return value 0
Press any key to continue . . .

```

Η προσθήκη του data record με ημερομηνία '2017-11-10' στο αρχείο agn2005WorstCase.us.txt χαλάει την περίπου ομοιόμορφη κατανομή των data records του αρχείου agn2005.us.txt (η τελευταία ημερομηνία απέχει πολύ από όλες τις προηγούμενες οι οποίες δημιουργούν ένα cluster με αποτέλεσμα ο δείκτης m (probe index) που υπολογίζει σε κάθε βήμα ο αλγόριθμος με βάση το στοιχείο αναζήτησης να προχωρά πολύ αργά προς τα δεξιά).

Για να δειχθεί καλύτερα και ο χρόνος εκτέλεσης του αλγόριθμου, δημιουργούμε ένα νέο αρχείο με το όνομα agnWorstCase.us.txt το οποίο περιέχει όλα τα data records του αρχείου agnWorstCase.us.txt και στο τέλος ένα νέο data record με ημερομηνία '9999-01-01' (θεωρείστε ότι μπορούμε με μια μηχανή του χρόνου να ταξιδέψουμε στο έτος 9999! ☺). Το νέο αρχείο έχει πλέον 3.240 data records.

Αναζητούμε στο νέο αρχείο την ημερομηνία '2017-11-10' που είναι η τελευταία ημερομηνία του 2017 με κίνηση της μετοχής. Το πλήθος των βημάτων που εκτελεί ο interpolationSearch είναι τώρα 2.070.

Το σχετικό screenshot είναι το εξής:

```

Give the stock data filename: agnWorstCase.us.txt

Give the date to search for (yyyy-mm-dd): 2017-11-10

[INTERPOLATION SEARCH]
Date 2017-11-10 found at array position 3238. Volume: 3251449
Number of steps: 2070
Running time measured: 0.0010057 seconds

-----
Process exited after 16.99 seconds with return value 0
Press any key to continue . . .

```

```

Give the stock data filename: agnWorstCase.us.txt

Give the date to search for (yyyy-mm-dd): 2017-11-10

[INTERPOLATION SEARCH]
Date 2017-11-10 found at array position 3238. Volume: 3251449
Number of steps: 2070
Running time measured: 0 seconds

-----
Process exited after 12.98 seconds with return value 0
Press any key to continue . . .

```

I.4 binInterpolationSearch, binInterpolationSearchImproved

Πρόγραμμα PROJECT_PART_I_4

- Και σε αυτές τις παραλλαγές του interpolationSearch χρησιμοποιούμε τη συνάρτηση unsigned long val(char s[]) για τον υπολογισμό του δείκτη m.
- Και στις δύο παραλλαγές, όταν το μέγεθος του block αναζήτησης γίνει ≤ 5 , καλείται ο αλγόριθμος linearSearch (στον κώδικα που παραδίδεται η δυνατότητα αυτή έχει απενεργοποιηθεί – τεθεί σε σχόλια).

Στη συνέχεια παρατίθενται ενδεικτικά screenshots με τα αποτελέσματα αναζήτησης κάθε αλγόριθμου στα αρχεία agn2005.us.txt και agn.us.txt. Και σε αυτά τα screenshots δεν τυπώνεται ο χρόνος εκτέλεσης που υπολογίζεται από τη chrono, ο οποίος, λόγω του μικρού αριθμού βημάτων, είναι 0.

Ενδεικτικά screenshots

```

Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-12-25

[BINARY INTERPOLATION SEARCH]
Date 2005-12-25 not found.
Number of steps: 4

-----
Process exited after 25.45 seconds with return value 0
Press any key to continue . . .

```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[BINARY INTERPOLATION SEARCH]

Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 5

-----
Process exited after 20 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-09-30

[BINARY INTERPOLATION SEARCH]

Date 2005-09-30 found at array position 188. Volume: 1607613
Number of steps: 4

-----
Process exited after 14.5 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2011-10-08

[BINARY INTERPOLATION SEARCH]

Date 2011-10-08 not found.
Number of steps: 4

-----
Process exited after 15.64 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[BINARY INTERPOLATION SEARCH]

Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 5

-----
Process exited after 10.94 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-09-30

[BINARY INTERPOLATION SEARCH]
Date 2005-09-30 found at array position 188. Volume: 1607613
Number of steps: 8

-----
Process exited after 17.96 seconds with return value 0
Press any key to continue . . .
```

```
-----
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-12-25

[BINARY INTERPOLATION SEARCH IMPROVED]
Date 2005-12-25 not found.
Number of steps: 4

-----
Process exited after 19.88 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[BINARY INTERPOLATION SEARCH IMPROVED]
Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 5

-----
Process exited after 16.13 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agn2005.us.txt

Give the date to search for (yyyy-mm-dd): 2005-09-30

[BINARY INTERPOLATION SEARCH IMPROVED]
Date 2005-09-30 found at array position 188. Volume: 1607613
Number of steps: 4

-----
Process exited after 18.34 seconds with return value 0
Press any key to continue . . .
```

```

Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2011-10-08

[BINARY INTERPOLATION SEARCH IMPROVED]

Date 2011-10-08 not found.
Number of steps: 4

-----
Process exited after 20.45 seconds with return value 0
Press any key to continue . . .

```

```

Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-07-01

[BINARY INTERPOLATION SEARCH IMPROVED]

Date 2005-07-01 found at array position 125. Volume: 479771
Number of steps: 5

-----
Process exited after 15.96 seconds with return value 0
Press any key to continue . . .

```

```

Give the stock data filename: agn.us.txt

Give the date to search for (yyyy-mm-dd): 2005-09-30

[BINARY INTERPOLATION SEARCH IMPROVED]

Date 2005-09-30 found at array position 188. Volume: 1607613
Number of steps: 9

-----
Process exited after 12.69 seconds with return value 0
Press any key to continue . . .

```

Συμπεριφορά κάθε αλγόριθμου

- Και στις δύο παραλλαγές η χρονική πολυπλοκότητα μέσης περίπτωσης παραμένει $O(\log n \log n)$.
- Η παραλλαγή `binInterpolationSearch` για n στοιχεία εκτελεί άλματα μεγέθους \sqrt{n} μέχρι να εντοπίσει το block μεγέθους \sqrt{n} που ενδεχομένως περιέχει το στοιχείο αναζήτησης x . Στο επόμενο βήμα τα άλματα έχουν μέγεθος $\sqrt{\sqrt{n}}$ κ.ο.κ. Με τον τρόπο αυτό η πολυπλοκότητα χειρότερης περίπτωσης βελτιώνεται από $O(n)$ σε $O(\sqrt{n})$.
- Η παραλλαγή `binInterpolationSearchImproved` για n στοιχεία εκτελεί εκθετικά μεγάλα άλματα (μεγέθους \sqrt{n} , $2\sqrt{n}$, $2^2\sqrt{n}$, κ.ο.κ.) μέχρι να εντοπίσει το block που μπορεί να περιέχει το x . Στη συνέχεια καλεί τον αλγόριθμο `binarySearch` για το block αυτό με στοιχεία στις θέσεις

πχ. $m+2^{i-1}*\sqrt{n}$ μέχρι $m+2^i*\sqrt{n}$ του πίνακα S ώστε να εντοπίσει το block μεγέθους \sqrt{n} που ενδεχομένως περιέχει το x . Έτσι, σε ένα βήμα εκτελούνται το πολύ $\log n$ άλματα και ο αλγόριθμος `binarySearch` κάνει το πολύ $\log \sqrt{n}$ συγκρίσεις, άρα απαιτείται συνολικός χρόνος $O(\log n)$, στο επόμενο βήμα απαιτείται συνολικός χρόνος $O(\log \sqrt{n})$ κ.ο.κ. που δίνει χρονική πολυπλοκότητα χειρότερης περίπτωσης $O(\log n)$.

Η συμπεριφορά χειρότερης περίπτωσης και των δύο παραλλαγών δεν είναι δυνατόν να παρατηρηθεί με τα δοθέντα αρχεία δεδομένων, εξαιτίας της περίπου ομοιόμορφης κατανομής των τιμών `Date` των `data records`, όπως αναφέρθηκε και στο μέρος I.3.

Πράγματι, όπως φαίνεται και στα screenshots που δίνονται παραπάνω:

- Για το αρχείο `agn2005.us.txt` με τα 252 `data records` το πλήθος των βημάτων που εκτελούν και οι δύο αλγόριθμοι για $x = "2005-12-25"$ είναι 4 βήματα (ανεπιτυχής αναζήτηση), για $x = "2005-07-01"$ 5 βήματα και για $x = "2005-09-30"$ 4 βήματα.
- Για το αρχείο `agn.us.txt` με τα 3.239 `data records` το πλήθος των βημάτων που εκτελούν και οι δύο αλγόριθμοι για $x = "2011-10-08"$ είναι 4 βήματα και για $x = "2005-07-01"$ 5 βήματα. Για $x = "2005-09-30"$ ο `binInterpolationSearch` εκτελεί 8 βήματα ενώ ο `binInterpolationImproved` 9.

Παρατήρηση:

Τα βήματα που μετράμε και στους δύο αλγόριθμους μέσω της `global` μεταβλητής `steps` αντιστοιχούν στα συνολικά βήματα για τον υπολογισμό του δείκτη m (`probe index`) και τα απαιτούμενα άλματα για τον εντοπισμό του block που μπορεί να περιέχει το στοιχείο αναζήτησης. Στον αλγόριθμο `binInterpolationSearchImproved` μετράμε επιπλέον και τα βήματα (συγκρίσεις) που γίνονται κατά τη δυαδική αναζήτηση στο εκθετικού μεγέθους block.

Για να δείξουμε τη βελτίωση της συμπεριφοράς χειρότερης περίπτωσης του αλγόριθμου `interpolationSearch` που επιτυγχάνουν οι δύο παραλλαγές χρησιμοποιούμε το ειδικό στιγμιότυπο εισόδου που δημιουργήσαμε νωρίτερα.

Στιγμιότυπο βελτίωσης συμπεριφοράς χειρότερης περίπτωσης αλγόριθμου `interpolationSearch`

Χρησιμοποιούμε το αρχείο `agnWorstCase.us.txt` με τα 3.240 `data records` και εκτελούμε τις παραλλαγές `binInterpolationSearch` και `binInterpolationSearchImproved` για το αρχείο αυτό με στοιχείο αναζήτησης το `'2017-11-10'`. Υπενθυμίζεται ότι ο `interpolationSearch`

αλγόριθμος για το συγκεκριμένο στιγμιότυπο έκανε 2.070 βήματα. Τα βήματα που κάνουν οι δύο παραλλαγές είναι:

- binInterpolationSearch: 68 βήματα, δηλ. το πλήθος των βημάτων είναι $O(\sqrt{\text{μέγεθος της εισόδου}})$.
- binInterpolationSearchImproved: 21 βήματα, δηλ. το πλήθος των βημάτων είναι λογαριθμικό στο μέγεθος της εισόδου.

Παρατίθενται τα σχετικά screenshots:

```
Give the stock data filename: agnWorstCase.us.txt

Give the date to search for (yyyy-mm-dd): 2017-11-10

[BINARY INTERPOLATION SEARCH]
Date 2017-11-10 found at array position 3238. Volume: 3251449
Number of steps: 68
Running time measured: 0.0156642 seconds

-----
Process exited after 21.2 seconds with return value 0
Press any key to continue . . .
```

```
Give the stock data filename: agnWorstCase.us.txt

Give the date to search for (yyyy-mm-dd): 2017-11-10

[BINARY INTERPOLATION SEARCH IMPROVED]
Date 2017-11-10 found at array position 3238. Volume: 3251449
Number of steps: 21
Running time measured: 0 seconds

-----
Process exited after 18.01 seconds with return value 0
Press any key to continue . . .
```

II. Binary Search Trees και Hashing

II.A Binary Search Tree με βάση το πεδίο Date υλοποιημένο ως AVL και ως RED-BLACK δένδρο
Προγράμματα PROJECT_PART_II_A_AVL και PROJECT_PART_II_A_RB

Γενικές σχεδιαστικές αποφάσεις

- Το Δυαδικό Δένδρο Αναζήτησης (ΔΔΑ - BST) υλοποιήθηκε ως AVL και ως RED-BLACK κομβοπροσανατολισμένο δένδρο.

- Δομή κόμβου AVL δένδρου:

```
struct dateVolume // Data record stored in binary tree node
{
    char Date[11];
    int Volume;
};
typedef struct dateVolume dataItem;

struct binaryTreeNode // Binary Search Tree node implemented as AVL
tree node
{
    dataItem data;
    struct binaryTreeNode *left;
    struct binaryTreeNode *right;
    int height;
};
typedef struct binaryTreeNode btNode;

btNode *root = NULL; // Root of the tree initially empty
```

- Δομή κόμβου RED-BLACK δένδρου:

```
struct dateVolume // Data record stored in a binary tree node
{
    char Date[11];
    int Volume;
};
typedef struct dateVolume dataItem;

struct binaryTreeNode // Binary Search Tree node implemented as RED-
BLACK tree node
{
    dataItem data;
    struct binaryTreeNode *left;
    struct binaryTreeNode *right;
    struct binaryTreeNode *parent;
    char color;
};
typedef struct binaryTreeNode btNode;

btNode *root = NULL; // Root of the tree initially empty
```

- Το διάβασμα του αρχείου δεδομένων γίνεται στη συνάρτηση `void readFileToBinTree(int argc, char *argv[])`.
- Η δημιουργία του δένδρου γίνεται με τη βοήθεια της αντίστοιχης συνάρτησης `insertToBinTree(dataItem x)` η οποία καλείται από την

`readFileToBinTree()` κάθε φορά που διαβάζεται ένα `data record` (στην περίπτωση του AVL δένδρου η συνάρτηση επιστρέφει δείκτη σε κόμβο του δένδρου ενώ στην περίπτωση του RED-BLACK δένδρου δεν επιστρέφει κάποια τιμή). Δεν επιτρέπεται η εισαγωγή διπλοεγγραφών (`data records` με την ίδια τιμή στο πεδίο `Date`). Πρακτικά, επειδή όλες οι τιμές του πεδίου `Date` είναι μοναδικές, δεν έχουμε διπλοεγγραφές (η πρόβλεψη για τη μη εισαγωγή διπλοεγγραφών υπάρχει για λόγους πληρότητας).

- Οι ζητούμενες λειτουργίες υλοποιούνται μέσω των συναρτήσεων:

```
void inorderBinTree(btNode *r)
btNode *searchBinTree(btNode *r, char x[11])
deleteFromBinTree(btNode *r, char x[11])
```

(Η `deleteFromBinTree()` στην περίπτωση του AVL δένδρου επιστρέφει δείκτη σε κόμβο του δένδρου ενώ στην περίπτωση του RED-BLACK δένδρου δεν επιστρέφει κάποια τιμή).

Δεν υλοποιήθηκε ξεχωριστή συνάρτηση `modifyBinTree()`.

- Για κάθε διαφορετική δομή (AVL, RED-BLACK) χρησιμοποιούνται ένα σύνολο βοηθητικών συναρτήσεων (πχ. `rotateL()`, `rotateR()`, `btNodeHeight()`, `btNodeBalance()`, `uncle()`, `sibling()`, `swapColors()`, `fixRedRed()`, `fixDoubleBlack()`, κλπ.) για την υλοποίηση των παραπάνω βασικών πράξεων.

- Η επαναζύγισή του υπόδενδρου με ρίζα τον κόμβο `T1` γίνεται μέσω των συναρτήσεων `btNode *rotateL(btNode *T1)` και `btNode *rotateR(btNode *T1)` οι οποίες υλοποιούν την απλή αριστερή και δεξιά περιστροφή. Η διπλή περιστροφή περιλαμβάνει είτε μια απλή αριστερή και μια απλή δεξιά περιστροφή ή μία απλή δεξιά και μια απλή αριστερή περιστροφή.

Στην περίπτωση του AVL δένδρου, η εισαγωγή ενός νέου κόμβου τερματίζεται με μία το πολύ απλή ή διπλή περιστροφή (τερματική περιστροφή) ενώ στη διαγραφή ενός κόμβου οι περιστροφές μπορεί να διαδοθούν μέχρι τη ρίζα του δένδρου.

Στην περίπτωση του RED-BLACK δένδρου, σε κάθε πράξη εισαγωγής ή διαγραφής ενός κόμβου εκτελούνται το πολύ μία απλή και μία διπλή τερματική περιστροφή.

- Όταν στον κόμβο `r` του του RED-BLACK δένδρου παραβιάζεται κάποια από τις ιδιότητες του δένδρου καλούνται οι συναρτήσεις `void fixRedRed(btNode *r)` και `void fixDoubleBlack(btNode *r)` για να διορθώσουν το πρόβλημα.

Η `fixRedRed()` καλείται κατά την εισαγωγή ενός νέου κόμβου στο δένδρο όταν δημιουργούνται δύο κόκκινοι κόμβοι στη σειρά (κόκκινος πατέρας και κόκκινος γιος).

Η `fixDoubleBlack()` καλείται κατά τη διαγραφή ενός μαύρου κόμβου που οδηγεί σε μείωση κατά 1 του πλήθους των μαύρων κόμβων στα μονοπάτια από τη ρίζα του δένδρου τα οποία πριν τη διαγραφή διέρχονταν από τον κόμβο που διαγράφηκε. Και οι δύο συναρτήσεις είναι αναδρομικές, λειτουργούν ανά επίπεδο και εκτελούν εναλλαγές χρωμάτων στους κόμβους οι οποίες μπορεί να διαδοθούν μέχρι τη ρίζα του δένδρου και $O(1)$ περιστροφές.

- Για την απεικόνιση των κόμβων κάθε δένδρου χρησιμοποιούνται οι συναρτήσεις `inorderBinTree()` και `printBinTree()`:
 - Η `inorderBinTree()` εμφανίζει τη βασική πληροφορία (Date, Volume) κάθε data record που αποθηκεύεται στους κόμβους του δένδρου καθώς και ένα σύνολο βοηθητικών πληροφοριών κυρίως για λόγους testing. Στην υλοποίηση με AVL δένδρο απεικονίζονται επιπλέον για κάθε κόμβο το ύψος και η ζύγιση του κόμβου ενώ στην υλοποίηση με RED-BLACK δένδρο απεικονίζονται για κάθε κόμβο με έναν ή κανέναν γιο (φύλλο) το χρώμα του κόμβου, το πλήθος των μαύρων κόμβων στο μονοπάτι από τον κόμβο προς τη ρίζα του δένδρου καθώς και το συνολικό πλήθος κόκκινων και μαύρων κόμβων του μονοπατιού.
 - Η `printBinTree()` τυπώνει τη δομή του δένδρου από αριστερά προς τα δεξιά με την πληροφορία Volume κάθε κόμβου. Στην υλοποίηση με AVL δένδρο τυπώνεται επιπλέον η ζύγιση του κόμβου και στην υλοποίηση με RED-BLACK δένδρο το χρώμα του κόμβου.

Στη συνέχεια παρατίθενται ενδεικτικά screenshots από την εκτέλεση των λειτουργιών για κάθε υλοποίηση για το περιορισμένου μεγέθους αρχείο `agn2005.us.txt`.

Ενδεικτικά screenshots

AVL δένδρο

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
Enter your choice <1 - 5>: 6
Wrong option, try again ...

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
Enter your choice <1 - 5>:
```

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 1

Date	Volume	Height	Balance
2005-01-03	1027044	0	0
2005-01-04	1927762	1	0
2005-01-05	943399	0	0
2005-01-06	662398	2	0
2005-01-07	1087886	0	0
2005-01-10	896381	1	0
2005-01-11	835841	0	0
2005-01-12	1476041	3	0
2005-01-13	772578	0	0
2005-01-14	453538	1	0
2005-01-18	592979	0	0
2005-01-19	694685	2	0
2005-01-20	858644	0	0
2005-01-21	1040463	1	0
2005-01-24	920193	0	0
2005-01-25	555647	4	0
2005-01-26	874385	0	0
2005-01-27	636871	1	0
2005-01-28	771367	0	0
2005-01-31	884878	2	0
2005-02-01	925137	0	0
2005-02-02	699124	1	0
2005-02-03	962772	0	0
2005-02-04	1040161	3	0
2005-02-07	683485	0	0
2005-02-08	1209972	1	0
2005-02-09	1229041	0	0
2005-02-10	1645651	2	0
2005-02-11	5309066	0	0
2005-02-14	1961665	1	0
2005-02-15	1142068	0	0
2005-02-16	1269704	5	0
2005-02-17	776211	0	0
2005-02-18	972962	1	0
2005-02-22	2638995	0	0
2005-02-23	1322474	2	0
2005-02-24	824540	0	0
2005-02-25	2066397	1	0
2005-02-28	1010193	0	0
2005-03-01	1699430	3	0
2005-03-02	1244379	0	0
2005-03-03	919082	1	0
2005-03-04	616589	0	0
2005-03-07	824440	2	0
2005-03-08	1966609	0	0
2005-03-09	785796	1	0
2005-03-10	1174354	0	0

2005-03-11	664718	4	0
2005-03-14	764203	0	0
2005-03-15	815158	1	0
2005-03-16	524066	0	0
2005-03-17	661590	2	0
2005-03-18	1134299	0	0
2005-03-21	841694	1	0
2005-03-22	804765	0	0
2005-03-23	558673	3	0
2005-03-24	517609	0	0
2005-03-28	1632937	1	0
2005-03-29	699527	0	0
2005-03-30	594594	2	0
2005-03-31	670974	0	0
2005-04-01	923422	1	0
2005-04-04	1001314	0	0
2005-04-05	1431646	6	0
2005-04-06	749070	0	0
2005-04-07	715873	1	0
2005-04-08	745638	0	0
2005-04-11	624560	2	0
2005-04-12	936033	0	0
2005-04-13	497428	1	0
2005-04-14	461710	0	0
2005-04-15	684091	3	0
2005-04-18	969430	0	0
2005-04-19	639089	1	0
2005-04-20	766726	0	0
2005-04-21	607509	2	0
2005-04-22	861571	0	0
2005-04-25	646153	1	0
2005-04-26	924431	0	0
2005-04-27	711635	4	0
2005-04-28	975283	0	0
2005-04-29	1455256	1	0
2005-05-02	1386040	0	0
2005-05-03	1405412	2	0
2005-05-04	2600048	0	0
2005-05-05	1581076	1	0
2005-05-06	2645856	0	0
2005-05-09	1618611	3	0
2005-05-10	1557364	0	0
2005-05-11	1215724	1	0
2005-05-12	528001	0	0
2005-05-13	941684	2	0
2005-05-16	693574	0	0
2005-05-17	704573	1	0
2005-05-18	784888	0	0
2005-05-19	1126327	5	0
2005-05-20	1202707	0	0
2005-05-23	959240	1	0
2005-05-24	934722	0	0
2005-05-25	1145598	2	0
2005-05-26	755728	0	0
2005-05-27	499345	1	0
2005-05-31	523864	0	0
2005-06-01	459793	3	0
2005-06-02	950260	0	0

2005-06-03	747656	1	0
2005-06-06	605188	0	0
2005-06-07	620928	2	0
2005-06-08	511656	0	0
2005-06-09	531532	1	0
2005-06-10	453941	0	0
2005-06-13	612553	4	0
2005-06-14	387853	0	0
2005-06-15	408639	1	0
2005-06-16	382909	0	0
2005-06-17	923925	2	0
2005-06-20	638989	0	0
2005-06-21	636062	1	0
2005-06-22	339925	0	0
2005-06-23	509133	3	0
2005-06-24	686209	0	0
2005-06-27	728284	1	0
2005-06-28	633944	0	0
2005-06-29	420342	2	0
2005-06-30	499851	0	0
2005-07-01	479771	1	0
2005-07-05	749673	0	0
2005-07-06	424984	7	0
2005-07-07	932401	0	0
2005-07-08	827063	1	0
2005-07-11	742612	0	0
2005-07-12	814955	2	0
2005-07-13	633540	0	0
2005-07-14	429625	1	0
2005-07-15	438604	0	0
2005-07-18	627385	3	0
2005-07-19	649382	0	0
2005-07-20	598630	1	0
2005-07-21	741200	0	0
2005-07-22	470690	2	0
2005-07-25	933208	0	0
2005-07-26	946829	1	0
2005-07-27	881145	0	0
2005-07-28	2684702	4	0
2005-07-29	2215828	0	0
2005-08-01	1949355	1	0
2005-08-02	851379	0	0
2005-08-03	1308146	2	0
2005-08-04	661489	0	0
2005-08-05	1040463	1	0
2005-08-08	967514	0	0
2005-08-09	927457	3	0
2005-08-10	1465245	0	0
2005-08-11	1014936	1	0
2005-08-12	1188177	0	0
2005-08-15	806782	2	0
2005-08-16	642924	0	0
2005-08-17	941381	1	0
2005-08-18	951672	0	0
2005-08-19	1038042	5	0
2005-08-22	1391488	0	0
2005-08-23	1014834	1	0
2005-08-24	1273438	0	0

2005-08-25	392393	2	0
2005-08-26	867625	0	0
2005-08-29	1047929	1	0
2005-08-30	1268998	0	0
2005-08-31	1630416	3	0
2005-09-01	1064679	0	0
2005-09-02	1138838	1	0
2005-09-06	1699631	0	0
2005-09-07	938051	2	0
2005-09-08	812735	0	0
2005-09-09	1110789	1	0
2005-09-12	1238627	0	0
2005-09-13	873376	4	0
2005-09-14	616488	0	0
2005-09-15	686209	1	0
2005-09-16	1010899	0	0
2005-09-19	434770	2	0
2005-09-20	571386	0	0
2005-09-21	1491075	1	0
2005-09-22	794877	0	0
2005-09-23	480377	3	0
2005-09-26	546162	0	0
2005-09-27	966101	1	0
2005-09-28	1071640	0	0
2005-09-29	824642	2	0
2005-09-30	1607613	0	0
2005-10-03	965092	1	0
2005-10-04	1023815	0	0
2005-10-05	1143278	6	0
2005-10-06	911414	0	0
2005-10-07	1020889	1	0
2005-10-10	727174	0	0
2005-10-11	795078	2	0
2005-10-12	892243	0	0
2005-10-13	809507	1	0
2005-10-14	1060239	0	0
2005-10-17	586421	3	0
2005-10-18	836750	0	0
2005-10-19	1142269	1	0
2005-10-20	916458	0	0
2005-10-21	811423	2	0
2005-10-24	902737	0	0
2005-10-25	806782	1	0
2005-10-26	943399	0	0
2005-10-27	635457	4	0
2005-10-28	1597725	0	0
2005-10-31	1078199	1	0
2005-11-01	869037	0	0
2005-11-02	1307642	2	0
2005-11-03	2398655	0	0
2005-11-04	1305826	1	0
2005-11-07	1500155	0	0
2005-11-08	795987	3	0
2005-11-09	1041472	0	0
2005-11-10	856222	1	0
2005-11-11	471700	0	0
2005-11-14	743418	2	0
2005-11-15	890931	0	0

2005-11-16	389871	1	0
2005-11-17	538898	0	0
2005-11-18	1050452	5	0
2005-11-21	494704	0	0
2005-11-22	587834	1	0
2005-11-23	462215	0	0
2005-11-25	295733	2	0
2005-11-28	942996	0	0
2005-11-29	988501	1	0
2005-11-30	1013422	0	0
2005-12-01	984869	3	0
2005-12-02	677835	0	0
2005-12-05	912222	1	0
2005-12-06	1752199	0	0
2005-12-07	1024925	2	0
2005-12-08	1409650	0	0
2005-12-09	1460300	1	0
2005-12-12	859048	0	0
2005-12-13	653317	4	0
2005-12-14	684494	0	0
2005-12-15	810717	1	0
2005-12-16	768341	0	0
2005-12-19	749776	2	0
2005-12-20	989610	0	0
2005-12-21	619515	1	0
2005-12-22	461912	0	0
2005-12-23	607609	3	0
2005-12-27	542430	0	0
2005-12-28	508628	2	-1
2005-12-29	392090	1	-1
2005-12-30	538695	0	0

Tree structure:

```

                    538695 <0>
                   392090 <-1>
                  508628 <-1>
                  542430 <0>
                 607609 <0>
                   461912 <0>
                   619515 <0>
                   989610 <0>
                  749776 <0>
                   768341 <0>
                   810717 <0>
                   684494 <0>
                 653317 <0>
                   859048 <0>
                   1460300 <0>
                   1409650 <0>
                  1024925 <0>
                   1752199 <0>
                   912222 <0>
                   677835 <0>
                 984869 <0>
                   1013422 <0>
                   988501 <0>
                   942996 <0>
                  295733 <0>
                   462215 <0>
                   587834 <0>
                   494704 <0>
                1050452 <0>
                   538898 <0>
                   389871 <0>
                   890931 <0>
                  743418 <0>
                   471700 <0>
                   856222 <0>
                   1041472 <0>
                 795987 <0>
                   1500155 <0>
                   1305826 <0>
                   2398655 <0>
                  1307642 <0>
                   869037 <0>
                   1078199 <0>
                   1597725 <0>
                635457 <0>
                   943399 <0>
                   806782 <0>
                   902737 <0>
                  811423 <0>
                   916458 <0>
                   1142269 <0>
                   836750 <0>
                  586421 <0>
                   1060239 <0>
                   809507 <0>
                   892243 <0>
                  795078 <0>
```



```

727174 <0>
1020889 <0>
911414 <0>
1143278 <0>
1023815 <0>
965092 <0>
1607613 <0>
824642 <0>
1071640 <0>
966101 <0>
546162 <0>
480377 <0>
794877 <0>
1491075 <0>
571386 <0>
434770 <0>
1010899 <0>
686209 <0>
616488 <0>
873376 <0>
1238627 <0>
1110789 <0>
812735 <0>
938051 <0>
1699631 <0>
1138838 <0>
1064679 <0>
1630416 <0>
1268998 <0>
1047929 <0>
867625 <0>
392393 <0>
1273438 <0>
1014834 <0>
1391488 <0>
1038042 <0>
951672 <0>
941381 <0>
642924 <0>
806782 <0>
1188177 <0>
1014936 <0>
1465245 <0>
927457 <0>
967514 <0>
1040463 <0>
661489 <0>
1308146 <0>
851379 <0>
1949355 <0>
2215828 <0>
2684702 <0>
881145 <0>
946829 <0>
933208 <0>
470690 <0>
741200 <0>
598630 <0>
649382 <0>

```

	627385	<0>		438604	<0>
				429625	<0>
				633540	<0>
	814955	<0>		742612	<0>
				827063	<0>
				932401	<0>
424984	<0>			749673	<0>
				479771	<0>
				499851	<0>
	420342	<0>		633944	<0>
				728284	<0>
				686209	<0>
	509133	<0>		339925	<0>
				636062	<0>
				638989	<0>
	923925	<0>		382909	<0>
				408639	<0>
				387853	<0>
	612553	<0>		453941	<0>
				531532	<0>
				511656	<0>
	620928	<0>		605188	<0>
				747656	<0>
				950260	<0>
	459793	<0>		523864	<0>
				499345	<0>
				755728	<0>
	1145598	<0>		934722	<0>
				959240	<0>
				1202707	<0>
1126327	<0>			784888	<0>
				704573	<0>
				693574	<0>
	941684	<0>		528001	<0>
				1215724	<0>
				1557364	<0>
	1618611	<0>		2645856	<0>
				1581076	<0>
				2600048	<0>
	1405412	<0>		1386040	<0>
				1455256	<0>
				975283	<0>
	711635	<0>		924431	<0>
				646153	<0>
				861571	<0>

607509 <0>
 766726 <0>
 639089 <0>
 969430 <0>
 684091 <0>
 461710 <0>
 497428 <0>
 936033 <0>
 624560 <0>
 745638 <0>
 715873 <0>
 749070 <0>
 1431646 <0>
 1001314 <0>
 923422 <0>
 670974 <0>
 594594 <0>
 699527 <0>
 1632937 <0>
 517609 <0>
 558673 <0>
 804765 <0>
 841694 <0>
 1134299 <0>
 661590 <0>
 524066 <0>
 815158 <0>
 764203 <0>
 664718 <0>
 1174354 <0>
 785796 <0>
 1966609 <0>
 824440 <0>
 616589 <0>
 919082 <0>
 1244379 <0>
 1699430 <0>
 1010193 <0>
 2066397 <0>
 824540 <0>
 1322474 <0>
 2638995 <0>
 972962 <0>
 776211 <0>
 1269704 <0>
 1142068 <0>
 1961665 <0>
 5309066 <0>
 1645651 <0>
 1229041 <0>
 1209972 <0>
 683485 <0>
 1040161 <0>
 962772 <0>
 699124 <0>
 925137 <0>
 884878 <0>
 771367 <0>
 636871 <0>
 874385 <0>
 555647 <0>
 920193 <0>
 1040463 <0>
 858644 <0>
 694685 <0>
 592979 <0>
 453538 <0>
 772578 <0>
 1476041 <0>
 835841 <0>
 896381 <0>
 1087886 <0>
 662398 <0>
 943399 <0>
 1927762 <0>
 1027044 <0>

Όπως φαίνεται και στα screenshots, το ύψος του AVL δένδρου για το αρχείο agn2005.us.txt είναι 7.

Εκτελέσαμε το πρόγραμμα και για τα 3 δοθέντα αρχεία agn.us.txt, ainv.us.txt και ale.us.txt και το ύψος του AVL δένδρου που δημιουργήθηκε από κάθε αρχείο ήταν 11.

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
```

Enter your choice <1 - 5>: 2

Give the date <yyyy-mm-dd>: 2005-01-03

Volume for the given date is: 1027044

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
```

Enter your choice <1 - 5>: 3

Give the date <yyyy-mm-dd>: 2005-01-03

Current record: 2005-01-03 : 1027044

Give the new volume <>= 0>: 9999999

Volume modified

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
```

Enter your choice <1 - 5>: 2

Give the date <yyyy-mm-dd>: 2005-01-03

Volume for the given date is: 9999999

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
```

Enter your choice <1 - 5>: 1

Date	Volume	Height	Balance
2005-01-03	9999999	0	0
2005-01-04	1927762	1	0
2005-01-05	943399	0	0
2005-01-06	662398	2	0

```

1269704 <0>
      1142068 <0>
      1961665 <0>
      5309066 <0>
      1645651 <0>
      1229041 <0>
      1209972 <0>
      683485 <0>
      1040161 <0>
      962772 <0>
      699124 <0>
      925137 <0>
      884878 <0>
      771367 <0>
      636871 <0>
      874385 <0>
      555647 <0>
      920193 <0>
      1040463 <0>
      858644 <0>
      694685 <0>
      592979 <0>
      453538 <0>
      772578 <0>
      1476041 <0>
      835841 <0>
      896381 <0>
      1087886 <0>
      662398 <0>
      943399 <0>
      1927762 <0>
      9999999 <0>

```

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 4

Give the date <yyyy-mm-dd>: 2005-01-03

Date found and deleted

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 1

Date	Volume	Height	Balance
2005-01-04	1927762	1	-1
2005-01-05	943399	0	0
2005-01-06	662398	2	0
2005-01-07	1087886	0	0
2005-01-10	896381	1	0
2005-01-11	835841	0	0

```

1269704 <0>
      1142068 <0>
      1961665 <0>
      5309066 <0>
    1645651 <0>
      1229041 <0>
      1209972 <0>
      683485 <0>
    1040161 <0>
      962772 <0>
      699124 <0>
      925137 <0>
      884878 <0>
      771367 <0>
      636871 <0>
      874385 <0>
    555647 <0>
      920193 <0>
      1040463 <0>
      858644 <0>
      694685 <0>
      592979 <0>
      453538 <0>
      772578 <0>
    1476041 <0>
      835841 <0>
      896381 <0>
      1087886 <0>
      662398 <0>
      943399 <0>
      1927762 <-1>

```

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 2

Give the date <yyyy-mm-dd>: 2005-07-10

This date does not exist in the tree

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 5

 Process exited after 27.29 seconds with return value 0
 Press any key to continue . . .

RED-BLACK δένδρο

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice (1 - 5): 6

Wrong option, try again ...

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice (1 - 5):

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice (1 - 5): 1

Date	Volume	Color	BNodes	TNodes
2005-01-03	1027044	B	7	7
2005-01-04	1927762	B	-	-
2005-01-05	943399	B	7	7
2005-01-06	662398	B	-	-
2005-01-07	1087886	B	7	7
2005-01-10	896381	B	-	-
2005-01-11	835841	B	7	7
2005-01-12	1476041	B	-	-
2005-01-13	772578	B	7	7
2005-01-14	453538	B	-	-
2005-01-18	592979	B	7	7
2005-01-19	694685	B	-	-
2005-01-20	858644	B	7	7
2005-01-21	1040463	B	-	-
2005-01-24	920193	B	7	7
2005-01-25	555647	B	-	-
2005-01-26	874385	B	7	7
2005-01-27	636871	B	-	-
2005-01-28	771367	B	7	7
2005-01-31	884878	B	-	-
2005-02-01	925137	B	7	7
2005-02-02	699124	B	-	-
2005-02-03	962772	B	7	7
2005-02-04	1040161	B	-	-
2005-02-07	683485	B	7	7
2005-02-08	1209972	B	-	-
2005-02-09	1229041	B	7	7
2005-02-10	1645651	B	-	-
2005-02-11	5309066	B	7	7
2005-02-14	1961665	B	-	-
2005-02-15	1142068	B	7	7
2005-02-16	1269704	B	-	-
2005-02-17	776211	B	7	7
2005-02-18	972962	B	-	-
2005-02-22	2638995	B	7	7
2005-02-23	1322474	B	-	-
2005-02-24	824540	B	7	7
2005-02-25	2066397	B	-	-
2005-02-28	1010193	B	7	7
2005-03-01	1699430	B	-	-
2005-03-02	1244379	B	7	7
2005-03-03	919082	B	-	-
2005-03-04	616589	B	7	7
2005-03-07	824440	B	-	-
2005-03-08	1966609	B	7	7
2005-03-09	785796	B	-	-
2005-03-10	1174354	B	7	7
2005-03-11	664718	B	-	-
2005-03-14	764203	B	7	7
2005-03-15	815158	B	-	-

2005-03-16	524066	B	7	7
2005-03-17	661590	B	-	-
2005-03-18	1134299	B	7	7
2005-03-21	841694	B	-	-
2005-03-22	804765	B	7	7
2005-03-23	558673	B	-	-
2005-03-24	517609	B	7	7
2005-03-28	1632937	B	-	-
2005-03-29	699527	B	7	7
2005-03-30	594594	B	-	-
2005-03-31	670974	B	7	7
2005-04-01	923422	B	-	-
2005-04-04	1001314	B	7	7
2005-04-05	1431646	B	-	-
2005-04-06	749070	B	7	8
2005-04-07	715873	B	-	-
2005-04-08	745638	B	7	8
2005-04-11	624560	B	-	-
2005-04-12	936033	B	7	8
2005-04-13	497428	B	-	-
2005-04-14	461710	B	7	8
2005-04-15	684091	B	-	-
2005-04-18	969430	B	7	8
2005-04-19	639089	B	-	-
2005-04-20	766726	B	7	8
2005-04-21	607509	B	-	-
2005-04-22	861571	B	7	8
2005-04-25	646153	B	-	-
2005-04-26	924431	B	7	8
2005-04-27	711635	B	-	-
2005-04-28	975283	B	7	8
2005-04-29	1455256	B	-	-
2005-05-02	1386040	B	7	8
2005-05-03	1405412	B	-	-
2005-05-04	2600048	B	-	8
2005-05-05	1581076	B	-	-
2005-05-06	2645856	B	7	8
2005-05-09	1618611	B	-	-
2005-05-10	1557364	B	-	8
2005-05-11	1215724	B	7	-
2005-05-12	528001	B	7	8
2005-05-13	941684	B	-	-
2005-05-16	693574	B	7	8
2005-05-17	704573	B	-	-
2005-05-18	784888	B	7	8
2005-05-19	1126327	R	-	-
2005-05-20	1202707	B	7	8
2005-05-23	959240	B	-	-
2005-05-24	934722	B	7	8
2005-05-25	1145598	B	-	-
2005-05-26	755728	B	7	8
2005-05-27	499345	B	-	-
2005-05-31	523864	B	7	8
2005-06-01	459793	B	-	-
2005-06-02	950260	B	7	8
2005-06-03	747656	B	-	-
2005-06-06	605188	B	7	8
2005-06-07	620928	B	-	-
2005-06-08	511656	B	7	8
2005-06-09	531532	B	-	-
2005-06-10	453941	B	7	8
2005-06-13	612553	B	-	-

2005-06-14	387853	B	7	8
2005-06-15	408639	B	-	-
2005-06-16	382909	B	7	8
2005-06-17	923925	B	-	-
2005-06-20	638989	B	7	8
2005-06-21	636062	B	-	-
2005-06-22	339925	B	7	8
2005-06-23	509133	B	-	-
2005-06-24	686209	B	7	8
2005-06-27	728284	B	-	-
2005-06-28	633944	B	7	8
2005-06-29	420342	B	-	-
2005-06-30	499851	B	7	8
2005-07-01	479771	B	-	-
2005-07-05	749673	B	7	8
2005-07-06	424984	B	-	-
2005-07-07	932401	B	7	8
2005-07-08	827063	B	-	-
2005-07-11	742612	B	7	8
2005-07-12	814955	B	-	-
2005-07-13	633540	B	7	8
2005-07-14	429625	B	-	-
2005-07-15	438604	B	7	8
2005-07-18	627385	B	-	-
2005-07-19	649382	B	7	8
2005-07-20	598630	B	-	-
2005-07-21	741200	B	7	8
2005-07-22	470690	B	-	-
2005-07-25	933208	B	7	8
2005-07-26	946829	B	-	-
2005-07-27	881145	B	7	8
2005-07-28	2684702	B	-	-
2005-07-29	2215828	B	7	8
2005-08-01	1949355	B	-	-
2005-08-02	851379	B	7	8
2005-08-03	1308146	B	-	-
2005-08-04	661489	B	7	8
2005-08-05	1040463	B	-	-
2005-08-08	967514	B	7	8
2005-08-09	927457	B	-	-
2005-08-10	1465245	B	7	8
2005-08-11	1014936	B	-	-
2005-08-12	1188177	B	7	8
2005-08-15	806782	B	-	-
2005-08-16	642924	B	7	8
2005-08-17	941381	B	-	-
2005-08-18	951672	B	7	8
2005-08-19	1038042	R	-	-
2005-08-22	1391488	B	7	9
2005-08-23	1014834	B	-	-
2005-08-24	1273438	B	7	9
2005-08-25	392393	B	-	-
2005-08-26	867625	B	7	9
2005-08-29	1047929	B	-	-
2005-08-30	1268998	B	7	9
2005-08-31	1630416	B	-	-
2005-09-01	1064679	B	7	9
2005-09-02	1138838	B	-	-
2005-09-06	1699631	B	7	9
2005-09-07	938051	B	-	-
2005-09-08	812735	B	7	9
2005-09-09	1110789	B	-	-

2005-09-12	1238627	B	7	9
2005-09-13	873376	R	-	-
2005-09-14	616488	B	7	9
2005-09-15	686209	B	-	-
2005-09-16	1010899	B	7	9
2005-09-19	434770	B	-	-
2005-09-20	571386	B	7	9
2005-09-21	1491075	B	-	-
2005-09-22	794877	B	7	9
2005-09-23	480377	B	-	-
2005-09-26	546162	B	7	9
2005-09-27	966101	B	-	-
2005-09-28	1071640	B	7	9
2005-09-29	824642	B	-	-
2005-09-30	1607613	B	7	9
2005-10-03	965092	B	-	-
2005-10-04	1023815	B	7	9
2005-10-05	1143278	B	-	-
2005-10-06	911414	B	7	9
2005-10-07	1020889	B	-	-
2005-10-10	727174	B	7	9
2005-10-11	795078	B	-	-
2005-10-12	892243	B	7	9
2005-10-13	809507	B	-	-
2005-10-14	1060239	B	7	9
2005-10-17	586421	B	-	-
2005-10-18	836750	B	7	9
2005-10-19	1142269	B	-	-
2005-10-20	916458	B	7	9
2005-10-21	811423	B	-	-
2005-10-24	902737	B	7	9
2005-10-25	806782	B	-	-
2005-10-26	943399	B	7	9
2005-10-27	635457	R	-	-
2005-10-28	1597725	B	7	10
2005-10-31	1078199	B	-	-
2005-11-01	869037	B	7	10
2005-11-02	1307642	B	-	-
2005-11-03	2398655	B	7	10
2005-11-04	1305826	B	-	-
2005-11-07	1500155	B	7	10
2005-11-08	795987	R	-	-
2005-11-09	1041472	B	7	10
2005-11-10	856222	B	-	-
2005-11-11	471700	B	7	10
2005-11-14	743418	B	-	-
2005-11-15	890931	B	7	10
2005-11-16	389871	B	-	-
2005-11-17	538898	B	7	10
2005-11-18	1050452	B	-	-
2005-11-21	494704	B	7	10
2005-11-22	587834	B	-	-
2005-11-23	462215	B	7	10
2005-11-25	295733	B	-	-
2005-11-28	942996	B	7	10
2005-11-29	988501	B	-	-
2005-11-30	1013422	B	7	10
2005-12-01	984869	R	-	-
2005-12-02	677835	B	7	11
2005-12-05	912222	B	-	-
2005-12-06	1752199	B	7	11
2005-12-07	1024925	R	-	-
2005-12-08	1409650	B	7	11
2005-12-09	1460300	B	-	-

2005-12-12	859048	B	7	11
2005-12-13	653317	B	-	-
2005-12-14	684494	B	7	11
2005-12-15	810717	B	-	-
2005-12-16	768341	B	7	11
2005-12-19	749776	R	-	-
2005-12-20	989610	B	7	12
2005-12-21	619515	R	-	-
2005-12-22	461912	B	7	12
2005-12-23	607609	B	-	-
2005-12-27	542430	B	7	12
2005-12-28	508628	R	-	-
2005-12-29	392090	B	7	12
2005-12-30	538695	R	7	13

Tree structure:

```

                                     538695 (R)
                                     392090 (B)
                                     508628 (R)
                                     542430 (B)
                                607609 (B)
                                461912 (B)
                                619515 (R)
                                989610 (B)
                                749776 (R)
                                768341 (B)
                                810717 (B)
                                684494 (B)
                                653317 (B)
                                859048 (B)
                                1460300 (B)
                                1409650 (B)
                                1024925 (R)
                                1752199 (B)
                                912222 (B)
                                677835 (B)
                                984869 (R)
                                1013422 (B)
                                988501 (B)
                                942996 (B)
                                295733 (B)
                                462215 (B)
                                587834 (B)
                                494704 (B)
                                1050452 (B)
                                538898 (B)
                                389871 (B)
                                890931 (B)
                                743418 (B)
                                471700 (B)
                                856222 (B)
                                1041472 (B)
                                795987 (R)
                                1500155 (B)
                                1305826 (B)
                                2398655 (B)
                                1307642 (B)
                                869037 (B)
                                1078199 (B)
                                1597725 (B)
                                635457 (R)
                                943399 (B)
                                806782 (B)
                                902737 (B)
                                811423 (B)
                                916458 (B)
                                1142269 (B)
                                836750 (B)
                                586421 (B)
                                1060239 (B)
                                809507 (B)
                                892243 (B)
                                795078 (B)
                                727174 (B)
                                1020889 (B)
                                911414 (B)
                                1143278 (B)
                                1023815 (B)

```

	965092 (B)
	1607613 (B)
824642 (B)	
	1071640 (B)
	966101 (B)
	546162 (B)
480377 (B)	
	794877 (B)
	1491075 (B)
	571386 (B)
434770 (B)	
	1010899 (B)
	686209 (B)
	616488 (B)
873376 (R)	
	1238627 (B)
	1110789 (B)
	812735 (B)
938051 (B)	
	1699631 (B)
	1138838 (B)
	1064679 (B)
1630416 (B)	
	1268998 (B)
	1047929 (B)
	867625 (B)
392393 (B)	
	1273438 (B)
	1014834 (B)
	1391488 (B)
1038042 (R)	
	951672 (B)
	941381 (B)
	642924 (B)
806782 (B)	
	1188177 (B)
	1014936 (B)
	1465245 (B)
927457 (B)	
	967514 (B)
	1040463 (B)
	661489 (B)
1308146 (B)	
	851379 (B)
	1949355 (B)
	2215828 (B)
2684702 (B)	
	881145 (B)
	946829 (B)
	933208 (B)
470690 (B)	
	741200 (B)
	598630 (B)
	649382 (B)
627385 (B)	
	438604 (B)
	429625 (B)
	633540 (B)
814955 (B)	
	742612 (B)
	827063 (B)
	932401 (B)
424984 (B)	
	749673 (B)
	479771 (B)

	420342 (B)	633944 (B)
	728284 (B)	686209 (B)
509133 (B)		339925 (B)
	636062 (B)	638989 (B)
	923925 (B)	382909 (B)
	408639 (B)	387853 (B)
612553 (B)		453941 (B)
	531532 (B)	511656 (B)
	620928 (B)	605188 (B)
	747656 (B)	950260 (B)
459793 (B)		523864 (B)
	499345 (B)	755728 (B)
	1145598 (B)	934722 (B)
	959240 (B)	1202707 (B)
1126327 (B)		784888 (B)
	704573 (B)	693574 (B)
	941684 (B)	528001 (B)
	1215724 (B)	1557364 (B)
1618611 (B)		2645856 (B)
	1581076 (B)	2600048 (B)
	1405412 (B)	1386040 (B)
	1455256 (B)	975283 (B)
711635 (B)		924431 (B)
	646153 (B)	861571 (B)
	607509 (B)	766726 (B)
	639089 (B)	969430 (B)
684091 (B)		461710 (B)
	497428 (B)	936033 (B)
	624560 (B)	745638 (B)
	715873 (B)	749070 (B)
1431646 (B)		1001314 (B)
	923422 (B)	670974 (B)
	594594 (B)	

```

699527 <B>
1632937 <B>
517609 <B>
558673 <B>
804765 <B>
841694 <B>
1134299 <B>
661590 <B>
524066 <B>
815158 <B>
764203 <B>
664718 <B>
1174354 <B>
785796 <B>
1966609 <B>
824440 <B>
616589 <B>
919082 <B>
1244379 <B>
1699430 <B>
1010193 <B>
2066397 <B>
824540 <B>
1322474 <B>
2638995 <B>
972962 <B>
776211 <B>
1269704 <B>
1142068 <B>
1961665 <B>
5309066 <B>
1645651 <B>
1229041 <B>
1209972 <B>
683485 <B>
1040161 <B>
962772 <B>
699124 <B>
925137 <B>
884878 <B>
771367 <B>
636871 <B>
874385 <B>
555647 <B>
920193 <B>
1040463 <B>
858644 <B>
694685 <B>
592979 <B>
453538 <B>
772578 <B>
1476041 <B>
835841 <B>
896381 <B>
1087886 <B>
662398 <B>
943399 <B>
1927762 <B>
1027044 <B>

```

Όπως φαίνεται και στα screenshots, κάθε μονοπάτι από τη ρίζα σε ένα κόμβο με έναν ή κανέναν γιο (φύλλο) έχει 7 μαύρους κόμβους. Το μακρύτερο μονοπάτι έχει 13 συνολικά κόμβους (7 μαύρους και 6 κόκκινους) και το ύψος του δένδρου είναι 12.

Εκτελέσαμε το πρόγραμμα και για τα 3 δοθέντα αρχεία `agn.us.txt`, `ainv.us.txt`, `ale.us.txt`. Και στα 3 δένδρα που δημιουργήθηκαν κάθε μονοπάτι από τη ρίζα σε ένα κόμβο με έναν ή κανέναν γιο (φύλλο) είχε 11 μαύρους κόμβους. Το μακρύτερο μονοπάτι είχε 21 συνολικά κόμβους (11 μαύρους και 10 κόκκινους) και το ύψος του δένδρου ήταν 20.

Γενικότερα ισχύει ότι για κάθε κόμβο r του δένδρου οποιοδήποτε μονοπάτι από τον r προς ένα κόμβο με έναν ή κανέναν γιο (φύλλο) έχει το ίδιο πλήθος μαύρων κόμβων και κάθε κόκκινος κόμβος έχει μαύρο πατέρα. [Επειδή ο NULL κόμβος θεωρείται μαύρος όπως απαιτεί ο ορισμός του RED-

BLACK δένδρου, η προηγούμενη πρόταση μπορεί να διατυπωθεί ισοδύναμα ως εξής: για κάθε κόμβο r του δένδρου οποιοδήποτε μονοπάτι από τον r προς ένα NULL κόμβο έχει το ίδιο πλήθος μαύρων κόμβων και κάθε κόκκινος κόμβος έχει μαύρο πατέρα. Στο ύψος του δένδρου που υπολογίζεται παραπάνω, δεν προσμετρώνται οι NULL κόμβοι, πχ. ένα φύλλο έχει ύψος 0.]

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
```

Enter your choice <1 - 5>: 2

Give the date <yyyy-mm-dd>: 2005-01-03

Volume for the given date is: 1027044

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
```

Enter your choice <1 - 5>: 3

Give the date <yyyy-mm-dd>: 2005-01-03

Current record: 2005-01-03 : 1027044

Give the new volume <>= 0>: 9999999

Volume modified

```
1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit
```

Enter your choice <1 - 5>: 1

Date	Volume	Color	BNodes	TNodes
2005-01-03	9999999	B	?	?
2005-01-04	1927762	B	-	-
2005-01-05	943399	B	?	?
2005-01-06	662398	B	-	-
2005-01-07	1087886	B	?	?
2005-01-10	896381	B	-	-
2005-01-11	835841	B	?	?
2005-01-12	1476041	B	-	-

1269704

1142068

1961665

5309066

1645651

1229041

1209972

683485

1040161

962772

699124

925137

884878

771367

636871

874385

555647

920193

1040463

858644

694685

592979

453538

772578

1476041

835841

896381

1087886

662398

943399

1927762

9999999

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 4

Give the date <yyyy-mm-dd>: 2005-01-03

Date found and deleted

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 1

Date	Volume	Color	BNodes	TNodes
2005-01-04	1927762	B	7	7
2005-01-05	943399	R	7	8
2005-01-06	662398	B	-	-
2005-01-07	1087886	B	7	8
2005-01-10	896381	R	-	-
2005-01-11	835841	B	7	8
2005-01-12	1476041	B	-	-
2005-01-13	772578	B	7	8
2005-01-14	453538	B	-	-

```

1269704 <B>
1142068 <B>
1961665 <B>
5309066 <B>
1645651 <B>
1229041 <B>
1209972 <B>
683485 <B>
1040161 <R>
962772 <B>
699124 <B>
925137 <B>
884878 <B>
771367 <B>
636871 <B>
874385 <B>
555647 <B>
920193 <B>
1040463 <B>
858644 <B>
694685 <R>
592979 <B>
453538 <B>
772578 <B>
1476041 <B>
835841 <B>
896381 <R>
1087886 <B>
662398 <B>
943399 <R>
1927762 <B>

```

```

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 2

Give the date <yyyy-mm-dd>: 2005-12-25

This date does not exist in the tree

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>: 5

-----
Process exited after 315.4 seconds with return value 0
Press any key to continue . . .

```

II.B Binary Search Tree με βάση τα πεδία Volume υλοποιημένο ως AVL και RED-BLACK δένδρο

Προγράμματα PROJECT_PART_II_B_AVL και PROJECT_PART_II_B_RB

- Η δημιουργία του δένδρου γίνεται και εδώ με τη βοήθεια της αντίστοιχης συνάρτησης insertToBinTree(). Δεν επιτρέπεται η εισαγωγή διπλοεγγραφών (data records με τις ίδιες τιμές στα πεδία Date, Volume). Όπως αναφέρθηκε και νωρίτερα, επειδή όλες οι τιμές του

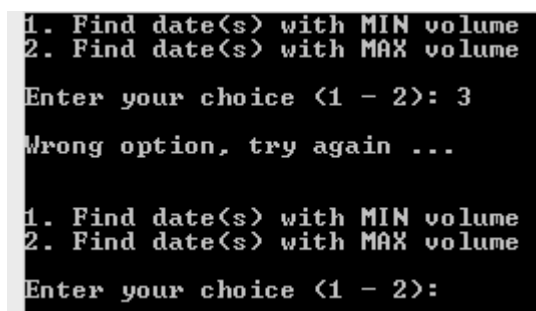
πεδίου Date είναι μοναδικές, δεν έχουμε διπλοεγγραφές (η πρόβλεψη για τη μη εισαγωγή διπλοεγγραφών υπάρχει για λόγους πληρότητας). Οι ζητούμενες λειτουργίες υλοποιούνται μέσω των συναρτήσεων:

- `btNode *minValuebtNode(btNode *r)` που επιστρέφει δείκτη στον κόμβο του δένδρου με την μικρότερη τιμή Volume (αριστερότερος κόμβος του δένδρου που έχει μόνο δεξί γιο ή είναι φύλλο) ή NULL αν το δένδρο είναι άδειο,
- `btNode *maxValuebtNode(btNode *r)` που επιστρέφει δείκτη στον κόμβο του δένδρου με τη μεγαλύτερη τιμή Volume (δεξιότερος κόμβος του δένδρου που έχει μόνο αριστερό γιο ή είναι φύλλο) ή NULL αν το δένδρο είναι άδειο και
- `void reportBinTree(btNode *r, int x)` που τυπώνει τις τιμές Date των data records που έχουν τιμή Volume ίση με x. [Η `reportBinTree()` είναι αρκετά γενική: μπορεί εύκολα να τροποποιηθεί ώστε να τυπώνει τις τιμές Date των data records με τιμή Volume μεταξύ των τιμών x_1 και x_2 με $x_1 < x_2$. Η χρονική πολυπλοκότητα για δένδρο με n data records είναι $O(\log n + k)$, όπου k το μέγεθος της απάντησης (πλήθος των data records με τιμή Date εντός του δοθέντος διαστήματος).]

- Στην υλοποίηση με το RED-BLACK δένδρο χρησιμοποιείται επιπλέον η συνάρτηση `searchBinTree()` η οποία καλείται από την `insertToBinTree()` για τον έλεγχο διπλοεγγραφών.
- Στη συνέχεια παρατίθενται ενδεικτικά screenshots από την εκτέλεση των λειτουργιών για κάθε υλοποίηση. Για το αρχείο `agn2005.us.txt` δίνονται επίσης τμήματα από screenshots της `inorder` διαπέρασης και της δομής του δένδρου όπου απεικονίζονται οι κόμβοι με τη μικρότερη και μεγαλύτερη τιμή Volume.

Ενδεικτικά screenshots

AVL δένδρο



```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume
Enter your choice <1 - 2>: 3
Wrong option, try again ...

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume
Enter your choice <1 - 2>:
```



```

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice (1 - 2): 1

Dates with MIN volume: 2005-11-25
MIN volume: 295733

-----
Process exited after 1.917 seconds with return value 0
Press any key to continue . . .

```

Date	Volume	Height	Balance
2005-11-25	295733	0	0
2005-06-22	339925	1	0
2005-06-16	382909	0	0
2005-06-14	387853	3	-1
2005-11-16	389871	1	-1
2005-12-29	392090	0	0
2005-08-25	392393	2	1
2005-06-15	408639	0	0
2005-06-29	420342	4	1
2005-07-06	424984	0	0
2005-07-14	429625	2	-1
2005-09-19	434770	0	0
2005-07-15	438604	1	1
2005-01-14	453538	5	1
2005-06-10	453941	0	0
2005-06-01	459793	2	-1

```

497428 (1)
      494704 (0)
      480377 (-1)
      479771 (-1)
      471700 (0)
      470690 (0)
      462215 (0)
      461912 (0)
      461710 (0)
      459793 (-1)
      453941 (0)
      453538 (1)
      438604 (1)
      434770 (0)
      429625 (-1)
      424984 (0)
      420342 (1)
      408639 (0)
      392393 (1)
      392090 (0)
      389871 (-1)
      387853 (-1)
      382909 (0)
      339925 (0)
      295733 (0)

```

```

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice (1 - 2): 2

Dates with MAX volume: 2005-02-11
MAX volume: 5309066

-----
Process exited after 2.597 seconds with return value 0
Press any key to continue . . .

```

2005-09-06	1699631	2	0
2005-12-06	1752199	0	0
2005-01-04	1927762	1	0
2005-08-01	1949355	0	0
2005-02-14	1961665	5	1
2005-03-08	1966609	0	0
2005-02-25	2066397	2	-1
2005-07-29	2215828	0	0
2005-11-03	2398655	1	0
2005-05-04	2600048	0	0
2005-02-22	2638995	3	1
2005-05-06	2645856	0	0
2005-07-28	2684702	1	0
2005-02-11	5309066	0	0

Tree structure:

```

                    5309066 <0>
              2684702 <0>
            2645856 <0>
    2638995 <1>
          2600048 <0>
        2398655 <0>
        2215828 <0>
    2066397 <-1>
      1966609 <0>
1961665 <1>
      1949355 <0>
      1927762 <0>
      1752199 <0>
    1699631 <0>
      1699430 <1>
        1645651 <0>
    1632937 <1>
          1630416 <0>
          1618611 <1>
          1607613 <1>
          1597725 <0>

```

Το ύψος του AVL δένδρου που δημιουργείται από το αρχείο agn2005.us.txt είναι 9.

Αρχείο agn.us.txt

```

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 1

Dates with MIN volume: 2010-04-26
MIN volume: 0

-----
Process exited after 1.925 seconds with return value 0
Press any key to continue . . .

```

```

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2016-04-05
MAX volume: 36807460

-----
Process exited after 2.302 seconds with return value 0
Press any key to continue . . .

```

Αρχείο ainv.us.txt

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 1

Dates with MIN volume: 2005-11-25
MIN volume: 127347

-----

Process exited after 1.932 seconds with return value 0
Press any key to continue . . .
```

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2014-02-28
MAX volume: 57522365

-----

Process exited after 1.712 seconds with return value 0
Press any key to continue . . .
```

Αρχείο ale.us.txt

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 1

Dates with MIN volume: 2005-11-25
MIN volume: 29473

-----

Process exited after 2.297 seconds with return value 0
Press any key to continue . . .
```

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2014-02-27
MAX volume: 2118295

-----

Process exited after 2.239 seconds with return value 0
Press any key to continue . . .
```

Προκειμένου να ελεγχθεί η ορθότητα της συνάρτησης `reportBinTree()`, τροποποιήσαμε σε 4 data records του αρχείου `agn2005.us.txt` την τιμή του πεδίου `Volume` με την τιμή 1 (μικρότερη τιμή που αποθηκεύεται στο δένδρο) και σε άλλα 3 data records την τιμή του πεδίου `Volume` με την τιμή 9999999 (μεγαλύτερη τιμή που αποθηκεύεται στο δένδρο).

Το νέο AVL δένδρο που δημιουργήθηκε έχει πάλι ύψος 9.

Στη συνέχεια παρατίθενται τα screenshots από την εκτέλεση του προγράμματος.

```

461710 <-1>
    459793 <0>
    453941 <1>
        438604 <0>
        434770 <0>
        429625 <0>
    424984 <0>
        420342 <0>
        408639 <1>
            392393 <0>
            392090 <0>
            389871 <0>
    387853 <0>
        382909 <0>
        339925 <1>
            295733 <0>
            1 <-1>
    1 <-1>
        1 <1>
            1 <0>

```

Dates with MIN volume: 2005-03-08 2005-03-02 2005-01-14 2005-04-05
MIN volume: 1

Process exited after 6.593 seconds with return value 0
Press any key to continue . . .

```

Tree structure:
-----
    9999999 <0>
    9999999 <0>
    9999999 <0>
    5309066 <0>
        2684702 <0>
        2645856 <-1>
    2638995 <0>
        2600048 <0>
        2398655 <0>
        2215828 <0>
    2066397 <0>
        1961665 <1>
        1949355 <0>
    1927762 <0>
        1752199 <0>
        1699631 <0>
        1699430 <0>
    1645651 <0>
        1632937 <1>
        1630416 <0>
    1618611 <0>

```

Dates with MAX volume: 2005-01-24 2005-01-11 2005-02-03
MAX volume: 9999999

Process exited after 2.033 seconds with return value 0
Press any key to continue . . .

RED BLACK δένδρο

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 3

Wrong option, try again ...

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>:
```

Αρχείο agn2005.us.txt

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 1

Dates with MIN volume: 2005-11-25
MIN volume: 295733

-----
Process exited after 2.358 seconds with return value 0
Press any key to continue . . .
```

Date	Volume	Color	BNodes	TNodes
2005-11-25	295733	R	5	8
2005-06-22	339925	B	-	-
2005-06-16	382909	R	5	8
2005-06-14	387853	B	-	-
2005-11-16	389871	B	5	8
2005-12-29	392090	R	5	9
2005-08-25	392393	R	-	-
2005-06-15	408639	B	5	8
2005-06-29	420342	B	-	-
2005-07-06	424984	B	5	8
2005-07-14	429625	R	-	-
2005-09-19	434770	R	5	9
2005-07-15	438604	B	5	8
2005-01-14	453538	B	-	-
2005-06-10	453941	B	5	7
2005-06-01	459793	R	-	-
2005-04-14	461710	R	5	9

```
497428 (R)
                                     494704 (R)
                                     480377 (B)
                                     479771 (R)
                                     471700 (B)
470690 (B)
                                     462215 (R)
                                     461912 (B)
                                     461710 (R)
459793 (R)
                                     453941 (B)
453538 (B)
                                     438604 (B)
                                     434770 (R)
429625 (R)
424984 (B)
420342 (B)
408639 (B)
392393 (R)
392090 (R)
389871 (B)
387853 (B)
382909 (R)
339925 (B)
295733 (R)
```

```

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2005-02-11
MAX volume: 5309066

-----
Process exited after 21.66 seconds with return value 0
Press any key to continue . . .

```

2005-08-31	1630416	R	5	8
2005-03-28	1632937	B	5	7
2005-02-10	1645651	R	-	-
2005-03-01	1699430	R	5	8
2005-09-06	1699631	B	-	-
2005-12-06	1752199	R	5	8
2005-01-04	1927762	B	-	-
2005-08-01	1949355	R	5	8
2005-02-14	1961665	B	5	7
2005-03-08	1966609	B	-	-
2005-02-25	2066397	B	5	7
2005-07-29	2215828	R	-	-
2005-11-03	2398655	R	5	8
2005-05-04	2600048	B	5	7
2005-02-22	2638995	B	-	-
2005-05-06	2645856	R	5	7
2005-07-28	2684702	B	-	-
2005-02-11	5309066	R	5	7

```

Tree structure:
-----
                    5309066 <R>
                2684702 <B>
                2645856 <R>
            2638995 <B>
                2600048 <B>
                    2398655 <R>
                2215828 <R>
                    2066397 <B>
            1966609 <B>
                1961665 <B>
                    1949355 <R>
                1927762 <B>
                    1752199 <R>
                    1699631 <B>
                    1699430 <R>
            1645651 <R>
                1632937 <B>
                    1630416 <R>
            1618611 <B>
                    1607613 <R>
                    1597725 <B>

```

Κάθε μονοπάτι του RED-BLACK δένδρου από τη ρίζα σε ένα κόμβο με έναν ή κανέναν γιο (φύλλο) έχει 5 μαύρους κόμβους. Το μακρύτερο μονοπάτι έχει 10 συνολικά κόμβους (5 μαύρους και 5 κόκκινους) και το ύψος του δένδρου είναι 9.

Αρχείο agn.us.txt

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 1

Dates with MIN volume: 2010-04-26
MIN volume: 0

-----

Process exited after 1.805 seconds with return value 0
Press any key to continue . . .
```

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2016-04-05
MAX volume: 36807460

-----

Process exited after 1.214 seconds with return value 0
Press any key to continue . . .
```

Αρχείο ainv.us.txt

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 1

Dates with MIN volume: 2005-11-25
MIN volume: 127347

-----

Process exited after 2.191 seconds with return value 0
Press any key to continue . . .
```

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2014-02-28
MAX volume: 57522365

-----

Process exited after 1.676 seconds with return value 0
Press any key to continue . . .
```

Αρχείο ale.us.txt

```
1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 1

Dates with MIN volume: 2005-11-25
MIN volume: 29473

-----

Process exited after 1.659 seconds with return value 0
Press any key to continue . . .
```

```

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2014-02-27
MAX volume: 2118295

-----
Process exited after 1.826 seconds with return value 0
Press any key to continue . . .

```

```

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>: 2

Dates with MAX volume: 2014-02-27
MAX volume: 2118295

-----
Process exited after 1.826 seconds with return value 0
Press any key to continue . . .

```

Στα RED-BLACK δένδρα που δημιουργήθηκαν από τα 2 αρχεία `agn.us.txt` και `ainv.us.txt` κάθε μονοπάτι από τη ρίζα σε ένα κόμβο με έναν ή κανέναν γιο (φύλλο) είχε 8 μαύρους κόμβους. Το μακρύτερο μονοπάτι είχε 15 συνολικά κόμβους (8 μαύρους και 7 κόκκινους) και το ύψος του δένδρου ήταν 14.

Στο RED-BLACK δένδρο με τα `records` του αρχείου `ale.us.txt` κάθε μονοπάτι από τη ρίζα σε ένα κόμβο με έναν ή κανέναν γιο (φύλλο) είχε 7 μαύρους κόμβους. Το μακρύτερο μονοπάτι είχε 14 συνολικά κόμβους (7 μαύρους και 7 κόκκινους) και το ύψος του δένδρου ήταν 13.

Ενοποίηση προγραμμάτων `PROJECT_PART_II_A` και `PROJECT_PART_II_B`:
Πρόγραμμα `PROJECT_PART_II_AB` (`PROJECT_PART_II_AB_AVL` και `PROJECT_PART_II_AB_RB`)

Η συνάρτηση `insertToBinTree()` χρησιμοποιείται και στα δύο προγράμματα `PROJECT_PART_II_A` και `PROJECT_PART_II_B` με την εξής διαφοροποίηση:

Στο πρόγραμμα `PROJECT_PART_II_A` το δένδρο δημιουργείται με βάση το πεδίο `Date` (η διάταξη των `data records` στο δένδρο καθορίζεται με βάση το πεδίο `Date`) ενώ στο πρόγραμμα `PROJECT_PART_II_B` με βάση το πεδίο `Volume` (η διάταξη των `data records` στο δένδρο καθορίζεται με βάση το πεδίο `Volume`).

Προκειμένου λοιπόν η `insertToBinTree()` να έχει ενιαία μορφή και να αποφύγουμε δύο διαφορετικές συναρτήσεις εισαγωγής, υλοποιήσαμε τις συναρτήσεις σύγκρισης `cmpDate(d1, d2)` και `cmpVolumeDate(d1, d2)` οι οποίες παίρνουν ως ορίσματα δύο `data records` `d1`, `d2` και επιστρέφουν 1, 0, -1 ανάλογα με τη διάταξη των `d1`, `d2`.

Ισχύει:

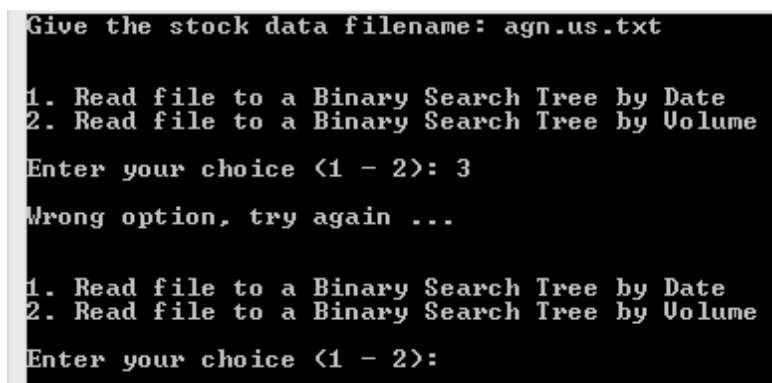
- `cmpDate(d1, d2) = 0` αν `d1.Date = d2.Date` και `cmpDate(d1, d2) = 1` ή `-1` αν `d1.Date > d2.Date` ή `d1.Date < d2.Date` αντίστοιχα
- `cmpVolumeDate(d1, d2) = 0` αν `d1.Volume == d2.Volume` και `d1.Date == d2.Date`, `cmpDate(d1, d2) = 1` αν `d1.Volume > d2.Volume` ή `d1.Volume == d2.Volume` και `d1.Date > d2.Date` και `-1` διαφορετικά.

Στη συνέχεια χρησιμοποιούμε το δείκτη σε συνάρτηση `cmpPtr()` ο οποίος, πριν την κλήση της `insertToBinTree()`, τίθεται στην τιμή `&cmpDate()` ή `&cmpVolume()` ανάλογα με το αν το δένδρο δημιουργείται με βάση το πεδίο `Date` (πρόγραμμα `PROJECT_PART_II_A`) ή το πεδίο `Volume` (πρόγραμμα `PROJECT_PART_II_B`). Η τιμή του δείκτη `cmpPtr()` (διεύθυνση του κώδικα της συνάρτησης) χρησιμοποιείται κατόπιν στην `insertToBinTree()` για τη σύγκριση μεταξύ των `data records`.

Ο δείκτης `cmpPtr()` αξιοποιείται επίσης στη συνάρτηση `searchBinTree()` που χρησιμοποιούν και τα δύο προγράμματα `PROJECT_PART_II_A` και `PROJECT_PART_II_B` με την υλοποίηση του RED-BLACK δένδρου.

Παρακάτω δίνεται το screenshot από το ενοποιημένο menu των δύο προγραμμάτων. Πριν την εμφάνιση του menu, το πρόγραμμα ανοίγει το αρχείο δεδομένων (το οποίο δίνεται είτε ως παράμετρος στην `command line` ή από το χρήστη) ή τυπώνει σχετικό μήνυμα σε περίπτωση λάθους.

Ανάλογα με την επιλογή του χρήστη, καλείται το αντίστοιχο υπομενού με τις λειτουργίες κάθε προγράμματος A και B, όπως περιγράφηκαν παραπάνω στα κεφ. II.A και II.B της αναφοράς.



```
Give the stock data filename: agn.us.txt

1. Read file to a Binary Search Tree by Date
2. Read file to a Binary Search Tree by Volume
Enter your choice <1 - 2>: 3
Wrong option, try again ...

1. Read file to a Binary Search Tree by Date
2. Read file to a Binary Search Tree by Volume
Enter your choice <1 - 2>:
```

```

Give the stock data filename: agn.us.txt

1. Read file to a Binary Search Tree by Date
2. Read file to a Binary Search Tree by Volume

Enter your choice <1 - 2>: 1

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice <1 - 5>:

```

```

Give the stock data filename: agn.us.txt

1. Read file to a Binary Search Tree by Date
2. Read file to a Binary Search Tree by Volume

Enter your choice <1 - 2>: 2

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>:

```

II.Γ Hashing με αλυσίδες

Πρόγραμμα PROJECT_PART_II_C

Γενικές σχεδιαστικές αποφάσεις

- Δομή δεδομένων:

```

#define M 11 // Size of Hash table

struct dateVolume // Data record stored in Hash table Bucket list
{
    char Date[11];
    int Volume;
};
typedef struct dateVolume dataItem;

struct listNode // Bucket list node
{
    dataItem data;
    struct listNode *next;
};
typedef struct listNode lNode;

lNode *hashTable[M] = {NULL}; // Hash table of M buckets initially
empty

```

- Το διάβασμα του αρχείου δεδομένων γίνεται στη συνάρτηση `void readFileToHashTable(int argc, char *argv[])`.
- Η δημιουργία της hash δομής (buckets με λίστες) γίνεται με τη βοήθεια της συνάρτησης `void insertToHashTable(dataItem x)` η οποία καλείται από την `readFileToHashTable()` κάθε φορά που διαβάζεται ένα data record. Δεν επιτρέπεται η εισαγωγή διπλοεγγραφών (data records με το ίδιο Date). Όπως αναφέρθηκε ήδη, επειδή όλες οι τιμές του πεδίου Date είναι μοναδικές, δεν έχουμε διπλοεγγραφές (η πρόβλεψη για τη μη εισαγωγή διπλοεγγραφών υπάρχει και εδώ για λόγους πληρότητας).
- Οι ζητούμενες λειτουργίες υλοποιήθηκαν μέσω των πράξεων:


```
lNode *searchHashTable(char x[11])
void deleteFromHashTable(char x[11])
```
- Πριν την έξοδο του χρήστη από το πρόγραμμα τυπώνουμε τα περιεχόμενα του hashTable μέσω της `void displayHashTable()` και στη συνέχεια τερματίζεται η εκτέλεση.

Στη συνέχεια παρατίθενται ενδεικτικά screenshots από την εκτέλεση του προγράμματος για τη δομή hashing που δημιουργείται από τα data records του αρχείου `agn2005.us.txt`.

1. Search volume for a given date
2. Modify volume for a given date
3. Delete data record of a given date
4. Display Hash table contents and Exit

Enter your choice <1 - 4>: 4

Hash Table contents:

```

Bucket [0] -> 2005-12-29, 392090 ! 2005-11-10, 856222 ! 2005-11-01, 869037 ! 2005-
10-20, 916458 ! 2005-10-11, 795078 ! 2005-09-23, 480377 ! 2005-09-14, 616488 ! 2005-
08-24, 1273438 ! 2005-08-15, 806782 ! 2005-07-25, 933208 ! 2005-07-07, 932401 ! 2005-
06-17, 923925 ! 2005-06-08, 511656 ! 2005-05-27, 499345 ! 2005-05-18, 784888 ! 2005-
05-09, 1618611 ! 2005-04-28, 975283 ! 2005-04-19, 639089 ! 2005-03-29, 699527 ! 2005-
02-10, 1645651 ! 2005-02-01, 925137 ! 2005-01-20, 858644 ! 2005-01-11, 835841 !

```

```

Bucket [1] -> 2005-12-01, 984869 ! 2005-11-11, 471700 ! 2005-11-02, 1307642 ! 2005-
10-21, 811423 ! 2005-10-12, 892243 ! 2005-10-03, 965092 ! 2005-09-15, 686209 ! 2005-
09-06, 1699631 ! 2005-08-25, 392393 ! 2005-08-16, 642924 ! 2005-07-26, 946829 ! 2005-
07-08, 827063 ! 2005-06-27, 728284 ! 2005-06-09, 531532 ! 2005-05-19, 1126327 ! 2005-
04-29, 1455256 ! 2005-03-10, 1174354 ! 2005-03-01, 1699430 ! 2005-02-11, 5309066 !
2005-02-02, 699124 ! 2005-01-21, 1040463 ! 2005-01-12, 1476041 ! 2005-01-03, 1027044 !

```

```

Bucket [2] -> 2005-12-20, 989610 ! 2005-12-02, 677835 ! 2005-11-30, 1013422 ! 2005-
11-21, 494704 ! 2005-11-03, 2398655 ! 2005-10-31, 1078199 ! 2005-10-13, 809507 ! 2005-
10-04, 1023815 ! 2005-09-16, 1010899 ! 2005-09-07, 938051 ! 2005-08-26, 867625 !
2005-08-17, 941381 ! 2005-08-08, 967514 ! 2005-07-27, 881145 ! 2005-07-18, 627385 !
2005-06-28, 633944 ! 2005-04-01, 923422 ! 2005-03-11, 664718 ! 2005-03-02, 1244379 !
2005-02-03, 962772 ! 2005-01-31, 884878 ! 2005-01-13, 772578 ! 2005-01-04, 1927762 !

```

```

Bucket [3] -> 2005-12-30, 538695 ! 2005-12-21, 619515 ! 2005-12-12, 859048 ! 2005-
11-22, 587834 ! 2005-11-04, 1305826 ! 2005-10-14, 1060239 ! 2005-10-05, 1143278 ! 2005-
09-26, 546162 ! 2005-09-08, 812735 ! 2005-08-18, 951672 ! 2005-08-09, 927457 ! 2005-
07-28, 2684702 ! 2005-07-19, 649382 ! 2005-06-29, 420342 ! 2005-05-10, 1557364 !
2005-04-20, 766726 ! 2005-04-11, 624560 ! 2005-03-30, 594594 ! 2005-03-21, 841694 !
2005-03-03, 919082 ! 2005-02-22, 2638995 ! 2005-02-04, 1040161 ! 2005-01-14, 453538 !
2005-01-05, 943399 !

```

```

Bucket [4] -> 2005-12-22, 461912 ! 2005-12-13, 653317 ! 2005-11-23, 462215 ! 2005-
11-14, 743418 ! 2005-10-24, 902737 ! 2005-10-06, 911414 ! 2005-09-27, 966101 ! 2005-
09-09, 1110789 ! 2005-08-19, 1038042 ! 2005-07-29, 2215828 ! 2005-06-10, 453941 ! 2005-
06-01, 459793 ! 2005-05-20, 1202707 ! 2005-05-11, 1215724 ! 2005-05-02, 1386040 !
2005-04-21, 607509 ! 2005-04-12, 936033 ! 2005-03-31, 670974 ! 2005-03-22, 804765 !
2005-03-04, 616589 ! 2005-02-23, 1322474 ! 2005-02-14, 1961665 ! 2005-01-24, 920193 !
2005-01-06, 662398 !

```

```

Bucket [5] -> 2005-12-23, 607609 ! 2005-12-14, 684494 ! 2005-12-05, 912222 ! 2005-
11-15, 890931 ! 2005-10-25, 806782 ! 2005-10-07, 1020889 ! 2005-09-28, 1071640 ! 200
5-09-19, 434770 ! 2005-08-29, 1047929 ! 2005-07-01, 479771 ! 2005-06-20, 638989 ! 20
05-06-02, 950260 ! 2005-05-12, 528001 ! 2005-05-03, 1405412 ! 2005-04-22, 861571 ! 2
005-04-13, 497428 ! 2005-04-04, 1001314 ! 2005-03-23, 558673 ! 2005-03-14, 764203 !
2005-02-24, 824540 ! 2005-02-15, 1142068 ! 2005-01-25, 555647 ! 2005-01-07, 1087886
!

Bucket [6] -> 2005-12-15, 810717 ! 2005-12-06, 1752199 ! 2005-11-25, 295733 ! 2005
-11-16, 389871 ! 2005-11-07, 1500155 ! 2005-10-26, 943399 ! 2005-10-17, 586421 ! 200
5-09-29, 824642 ! 2005-08-10, 1465245 ! 2005-08-01, 1949355 ! 2005-07-20, 598630 ! 2
005-07-11, 742612 ! 2005-06-30, 499851 ! 2005-06-21, 636062 ! 2005-06-03, 747656 ! 2
005-05-31, 523864 ! 2005-05-13, 941684 ! 2005-05-04, 2600048 ! 2005-04-14, 461710 !
2005-04-05, 1431646 ! 2005-03-24, 517609 ! 2005-03-15, 815158 ! 2005-02-25, 2066397
! 2005-02-16, 1269704 ! 2005-02-07, 683485 ! 2005-01-26, 874385 !

Bucket [7] -> 2005-12-16, 768341 ! 2005-12-07, 1024925 ! 2005-11-17, 538898 ! 2005
-11-08, 795987 ! 2005-10-27, 635457 ! 2005-10-18, 836750 ! 2005-09-01, 1064679 ! 200
5-08-11, 1014936 ! 2005-08-02, 851379 ! 2005-07-21, 741200 ! 2005-07-12, 814955 ! 20
05-06-22, 339925 ! 2005-06-13, 612553 ! 2005-05-23, 959240 ! 2005-05-05, 1581076 ! 2
005-04-15, 684091 ! 2005-04-06, 749070 ! 2005-03-16, 524066 ! 2005-03-07, 824440 ! 2
005-02-17, 776211 ! 2005-02-08, 1209972 ! 2005-01-27, 636871 ! 2005-01-18, 592979 !

Bucket [8] -> 2005-12-08, 1409650 ! 2005-11-18, 1050452 ! 2005-11-09, 1041472 ! 20
05-10-28, 1597725 ! 2005-10-19, 1142269 ! 2005-09-20, 571386 ! 2005-09-02, 1138838 !
2005-08-30, 1268998 ! 2005-08-12, 1188177 ! 2005-08-03, 1308146 ! 2005-07-22, 47069
0 ! 2005-07-13, 633540 ! 2005-06-23, 509133 ! 2005-06-14, 387853 ! 2005-05-24, 93472
2 ! 2005-05-06, 2645856 ! 2005-04-25, 646153 ! 2005-04-07, 715873 ! 2005-03-17, 6615
90 ! 2005-03-08, 1966609 ! 2005-02-18, 972962 ! 2005-02-09, 1229041 ! 2005-01-28, 77
1367 ! 2005-01-19, 694685 !

Bucket [9] -> 2005-12-27, 542430 ! 2005-12-09, 1460300 ! 2005-11-28, 942996 ! 2005
-09-30, 1607613 ! 2005-09-21, 1491075 ! 2005-09-12, 1238627 ! 2005-08-31, 1630416 !
2005-08-22, 1391488 ! 2005-08-04, 661489 ! 2005-07-14, 429625 ! 2005-07-05, 749673 !
2005-06-24, 686209 ! 2005-06-15, 408639 ! 2005-06-06, 605188 ! 2005-05-25, 1145598
! 2005-05-16, 693574 ! 2005-04-26, 924431 ! 2005-04-08, 745638 ! 2005-03-18, 1134299
! 2005-03-09, 785796 ! 2005-02-28, 1010193 !

Bucket [10] -> 2005-12-28, 508628 ! 2005-12-19, 749776 ! 2005-11-29, 988501 ! 2005-
10-10, 727174 ! 2005-09-22, 794877 ! 2005-09-13, 873376 ! 2005-08-23, 1014834 ! 2005
-08-05, 1040463 ! 2005-07-15, 438604 ! 2005-07-06, 424984 ! 2005-06-16, 382909 ! 200
5-06-07, 620928 ! 2005-05-26, 755728 ! 2005-05-17, 704573 ! 2005-04-27, 711635 ! 200
5-04-18, 969430 ! 2005-03-28, 1632937 ! 2005-01-10, 896381 !

```

```

Process exited after 3.423 seconds with return value 0
Press any key to continue . . .

```

```

1. Search volume for a given date
2. Modify volume for a given date
3. Delete data record of a given date
4. Display Hash table contents and Exit

```

Enter your choice <1 - 4>: 1

Give the date <yyyy-mm-dd>: 2005-01-03

Volume for the given date is: 1027044

```

1. Search volume for a given date
2. Modify volume for a given date
3. Delete data record of a given date
4. Display Hash table contents and Exit

```

Enter your choice <1 - 4>: 2

Give the date <yyyy-mm-dd>: 2005-01-03

Current record: 2005-01-03 : 1027044

Give the NEW volume <>= 0>: 9999999

Volume modified

```

1. Search volume for a given date
2. Modify volume for a given date
3. Delete data record of a given date
4. Display Hash table contents and Exit

```

Enter your choice <1 - 4>: 1

Give the date <yyyy-mm-dd>: 2005-01-03

Volume for the given date is: 9999999

```

.....
Bucket [1] -> 2005-12-01, 984869 : 2005-11-11, 471700 : 2005-11-02, 1307642 : 2005-
-10-21, 811423 : 2005-10-12, 892243 : 2005-10-03, 965092 : 2005-09-15, 686209 : 2005-
-09-06, 1699631 : 2005-08-25, 392393 : 2005-08-16, 642924 : 2005-07-26, 946829 : 2005-
-07-08, 827063 : 2005-06-27, 728284 : 2005-06-09, 531532 : 2005-05-19, 1126327 : 2005-
-04-29, 1455256 : 2005-03-10, 1174354 : 2005-03-01, 1699430 : 2005-02-11, 5309066 :
: 2005-02-02, 699124 : 2005-01-21, 1040463 : 2005-01-12, 1476041 : 2005-01-03, 99999
99 :

```

```

1. Search volume for a given date
2. Modify volume for a given date
3. Delete data record of a given date
4. Display Hash table contents and Exit

```

Enter your choice <1 - 4>: 3

Give the date <yyyy-mm-dd>: 2005-01-03

Date found and deleted

```

.....
Bucket [1] -> 2005-12-01, 984869 : 2005-11-11, 471700 : 2005-11-02, 1307642 : 2005-
-10-21, 811423 : 2005-10-12, 892243 : 2005-10-03, 965092 : 2005-09-15, 686209 : 2005-
-09-06, 1699631 : 2005-08-25, 392393 : 2005-08-16, 642924 : 2005-07-26, 946829 : 2005-
-07-08, 827063 : 2005-06-27, 728284 : 2005-06-09, 531532 : 2005-05-19, 1126327 : 2005-
-04-29, 1455256 : 2005-03-10, 1174354 : 2005-03-01, 1699430 : 2005-02-11, 5309066 :
: 2005-02-02, 699124 : 2005-01-21, 1040463 : 2005-01-12, 1476041 :

```

```

1. Search volume for a given date
2. Modify volume for a given date
3. Delete data record of a given date
4. Display Hash table contents and Exit

Enter your choice (1 - 4): 1

Give the date (yyyy-mm-dd): 2005-12-25

This date does not exist in Hash table

```

Ενοποίηση προγραμμάτων Binary Search Tree (υλοποιήσεις με AVL και RED-BLACK δένδρα) και Hashing με αλυσίδες

Προγράμματα PROJECT_PART_II_ABC_AVL και PROJECT_PART_II_ABC_RB

Παρακάτω δίνονται screenshots από το ενοποιημένο menu των δύο προγραμμάτων που διαχειρίζονται τις λειτουργίες του Binary Search Tree και της Hashing δομής. Πριν την εμφάνιση του menu, το πρόγραμμα ανοίγει το αρχείο δεδομένων (το οποίο δίνεται είτε ως παράμετρος στην command line ή από το χρήστη) ή τυπώνει σχετικό μήνυμα σε περίπτωση λάθους.

Ανάλογα με την επιλογή του χρήστη, καλείται το αντίστοιχο υπομενού με τις λειτουργίες κάθε προγράμματος AB και C, όπως περιγράφηκαν παραπάνω στα κεφ. II.A, II.B και II.Γ της αναφοράς. Τα screenshots είναι ίδια και για τις δύο υλοποιήσεις του δένδρου αναζήτησης ως AVL και ως RED-BLACK δένδρο.

Ενδεικτικά screenshots

```

1. Read file to a Binary Search Tree
2. Read file to a Hash Table

Enter your choice (1 - 2): 1

1. Read file to a Binary Search Tree by Date
2. Read file to a Binary Search Tree by Volume

Enter your choice (1 - 2): 1

1. Inorder traversal of BST
2. Search volume for a given date
3. Modify volume for a given date
4. Delete BST node of a given date
5. Exit

Enter your choice (1 - 5):

```

1. Read file to a Binary Search Tree
2. Read file to a Hash Table

Enter your choice <1 - 2>: 1

1. Read file to a Binary Search Tree by Date
2. Read file to a Binary Search Tree by Volume

Enter your choice <1 - 2>: 2

1. Find date(s) with MIN volume
2. Find date(s) with MAX volume

Enter your choice <1 - 2>:

1. Read file to a Binary Search Tree
2. Read file to a Hash Table

Enter your choice <1 - 2>: 2

1. Search volume for a given date
2. Modify volume for a given date
3. Delete data record of a given date
4. Display Hash table contents and Exit

Enter your choice <1 - 4>: