ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

Τμήμα: Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Αναφορά στο project του μαθήματος Δομές Δεδομένων

Όνομα: Νικόλαος Μπακάλης
ΑΜ: 1054316

e-mail (Gmail): bakalis.nikolas@gmail.com

e-mail (CEID): nbakalis@ceid.upatras.gr

e-mail (Upatras): up1054316@upnet.gr

Γλώσσα προγραμματισμού: Java

Μέρος Α

Σας δίνεται το αρχείο integers.txt το οποίο περιέχει ακέραιους αριθμούς, έναν σε κάθε γραμμή του. Σας ζητείται να γράψετε ένα πρόγραμμα που θα διαβάζει το αρχείο και θα ταξινομεί τα περιεχόμενά του με χρήση του αλγόριθμου Merge Sort.

Απάντηση:
Το διάβασμα του αρχείου γίνεται στην “mainClass” Κλάση του προγράμματος με την χρήση μιας Try – Catch και ενός “BufferReader” με όνομα “br” ο οποίος όσο το αρχείο έχει γραμμές το διαβάζει και το αποθηκεύει σε ένα ArrayList από integers με όνομα “lines”. Τον αλγόριθμο Merge Sort τον υλοποιώ στην Κλάση “MergeSort” με την μέθοδο “mergeSort” η οποία έχει ως ορίσματα έναν πίνακα από integers με όνομα “array”, ένα low “l” και ένα high “h” για το μικρότερο και μεγαλύτερο στοιχείο του πίνακα αντίστοιχα. Αρχικά σπάω τον πίνακα σε 2 κομμάτια, μετά σε 4, μετά σε 8 κλπ., μέχρι κάθε integer του πίνακα να είναι ένας καινούριος πίνακας. Έπειτα ελέγχω αν ο προηγούμενος αριθμός είναι μεγαλύτερος από τον επόμενο ή όχι και ταξινομώ αναλόγως. Συγχωνεύω τους υποπίνακες και έχω ολόκληρο τον αρχικό πίνακα ταξινομημένο.

Μέρος Β

Επεκτείνετε το πρόγραμμα του Μέρους Α έτσι ώστε ο χρήστης να μπορεί να αναζητεί κάποιον ακέραιο διαλέγοντας κάθε φορά έναν από τους ακόλουθους τρεις τρόπους:

1. Γραμμική αναζήτηση
2. Binary search
3. Interpolation search

Απάντηση:

1. Γραμμική αναζήτηση: Διατρέχουμε όλον τον ταξινομημένο πίνακα με τους αριθμούς και ψάχνουμε μέχρι να βρούμε το στοιχείο που θέλουμε. (Μέσα στην Κλάση: “SerialSearch”)

2. Binary search: Βρίσκουμε το μεσαίο στοιχείο του πίνακα. Ελέγχουμε αν το στοιχείο της επιθυμίας μας βρίσκεται αριστερά ή δεξιά του μεσαίου στοιχείου του πίνακα και ψάχνουμε ένα από τα δυο κομμάτια αναλόγως της προηγούμενης απάντησής μας. (Μέσα στην Κλάση: “BinarySearch”)

3. Interpolation search: Ίδια με την Binary search μόνο που τώρα το “μεσαίο” στοιχείο υπολογίζεται από τον τύπο του βιβλίου: `(search - first) / (last – first)` έπειτα η αναζήτηση γίνεται ως εξής υπολογίζουμε με τον τύπο της interpolation την θέση του στοιχείου που αναζητούμε όσο το αριστερό όριο είναι μικρότερο του δεξιού και όσο το νούμερο προς αναζήτηση είναι μεγαλύτερο από το στοιχείο στο αριστερό όριο και μικρότερο από αυτό στο δεξί. Αν στη θέση που βρίσκουμε είναι το στοιχείο προς αναζήτηση τότε βρήκαμε το στοιχείο, αλλιώς το στοιχείο δεν υπάρχει στον πίνακα. (Μέσα στην Κλάση: “InterpolationSearch”)

Μέρος Γ

Χρησιμοποιώντας ξανά το αρχείο integers.txt κατασκευάστε ένα Red-Black Tree που θα υποστηρίζει τις πράξεις της εισαγωγής και αναζήτησης.

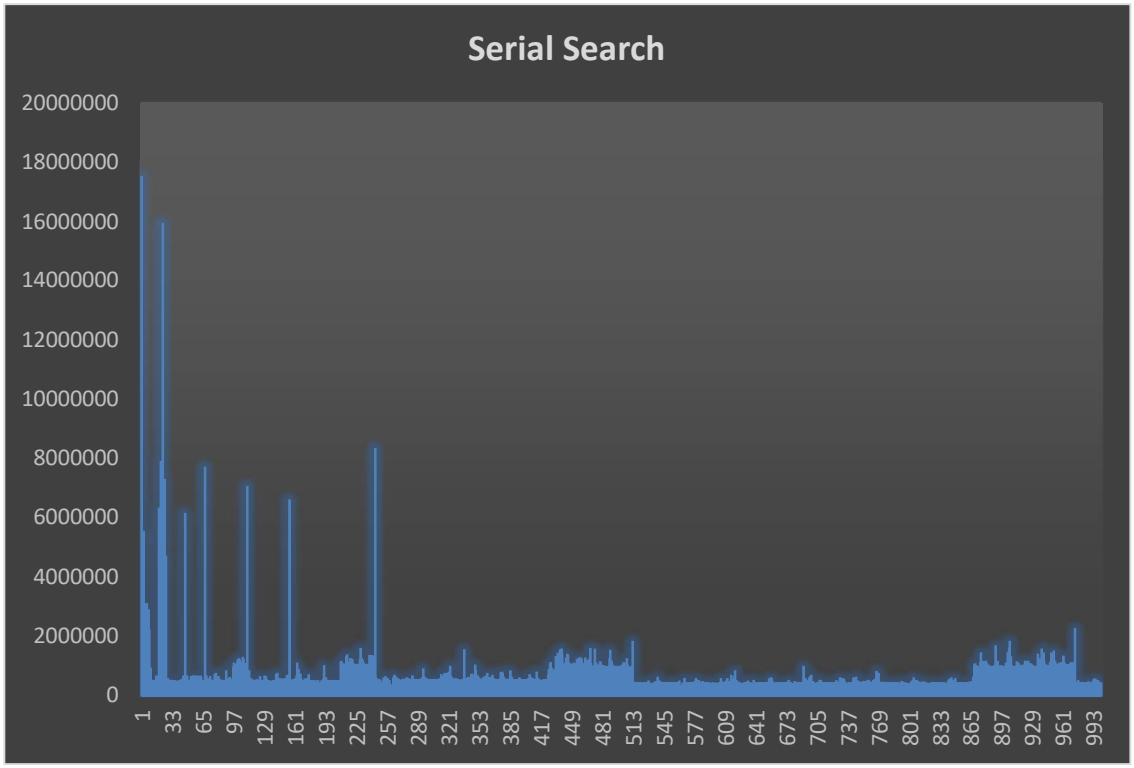
Απάντηση:

Ξεκινάω δημιουργώντας έναν κόμβο ο οποίος είναι και ρίζα του δέντρου. Μετά από αυτό το βήμα αρχίζω την εισαγωγή στοιχείων σε κόμβους που συνδέω με την ρίζα κάνω τις απαραίτητες περιστροφές και αλλαγές χρωμάτων όπου χρειάζεται και φτάνω στο τελικό μου δέντρο. Η αναζήτηση ξεκινά από την ρίζα και αναλόγως το περιεχόμενο του κόμβου που ψάχνουμε προχωράμε δεξιά ή αριστερά της ρίζας αν ο αριθμός που ψάχνουμε είναι μικρότερος ή μεγαλύτερος αντίστοιχα (όμοια λογική με την Binary Search) συνεχίζουμε για κάθε επόμενο κόμβο με την ίδια λογική μέχρι να βρούμε ή όχι τον αριθμό της αναζήτησής μας. (Μέσα στην Κλάση: “RedBlackTree”)

Μέρος Δ

Γράψτε ένα πρόγραμμα που θα εκτελεί ένα μεγάλο αριθμό αναζητήσεων με κάθε μία εκ των τεσσάρων μεθόδων που αναφέρονται στα Μέρη Β, Γ και μετρήστε το μέσο αλλά και το χειρότερο χρόνο αναζήτησης κάθε μεθόδου. Παρουσιάστε τα αποτελέσματά των πειραμάτων σας με τη βοήθεια πινάκων και γραφικών παραστάσεων και εξηγήστε αν αυτά συνάδουν με τη θεωρία.

Απάντηση:



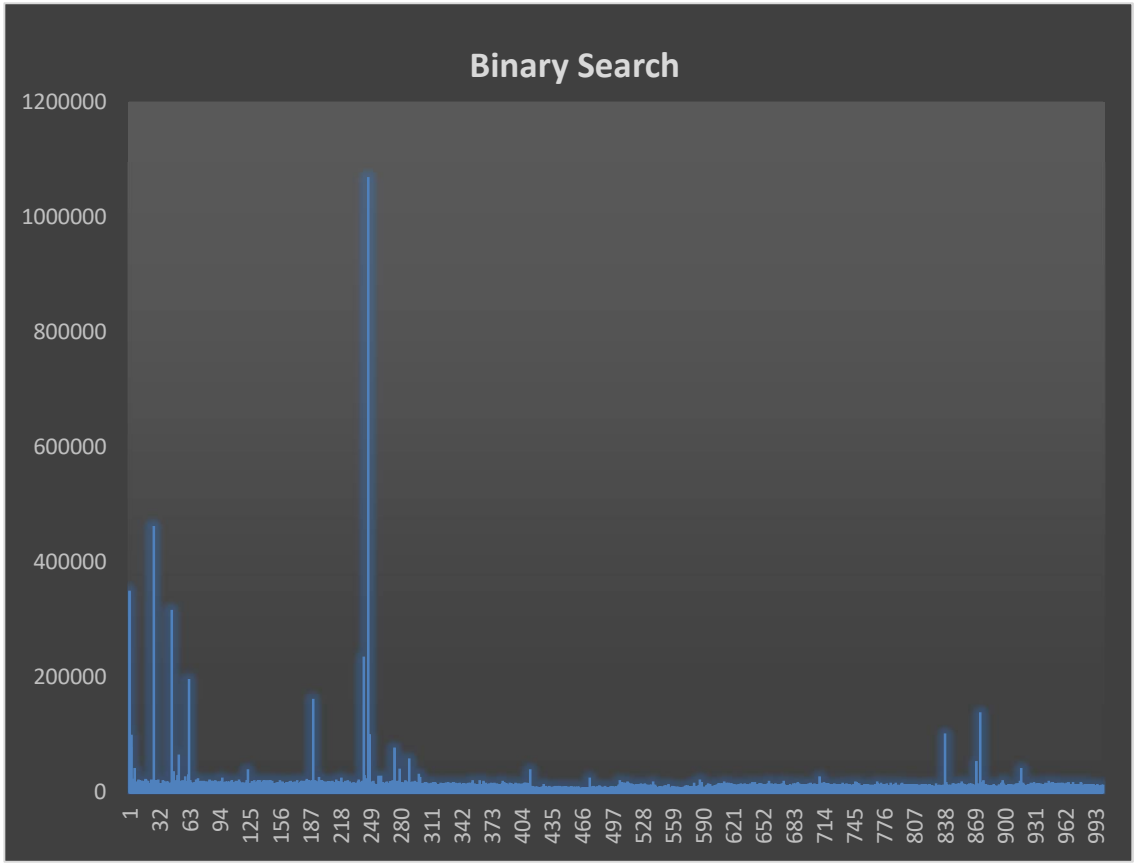
Στην Serial Search 1000 αναζητήσεων παρατηρώ ότι:

Ο χρόνος της Serial Search είναι O(n)

Ο χειρότερος χρόνος είναι ο: 17498402 στον αριθμό 336985

Ο καλύτερος: 27632 στον αριθμό 27968

Ενώ ο μέσος όρος είναι: 744328.1



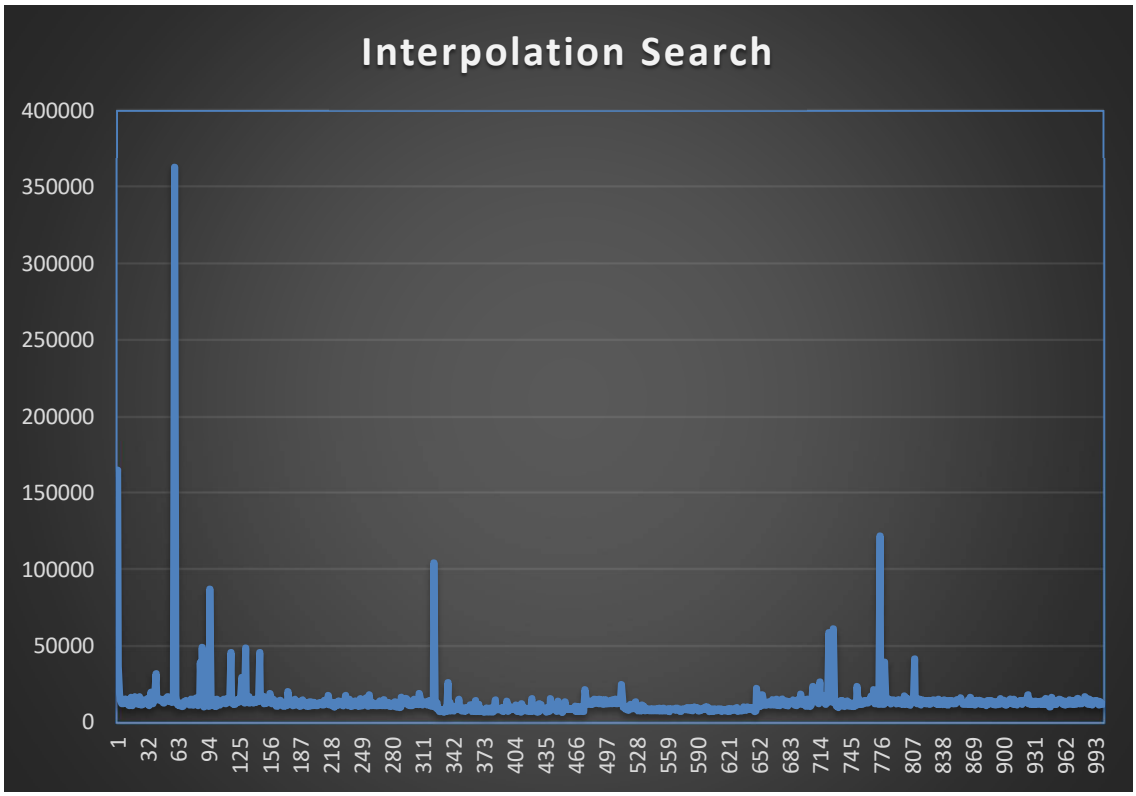
Στην Binary Search 1000 αναζητήσεων παρατηρώ ότι:

Ο χειρότερος χρόνος της Binary Search είναι O(nlogn)

Ο χειρότερος χρόνος είναι ο: 1067770 στον αριθμό 811485

Ο καλύτερος: 6710 και εμφανίζεται σε αρκετούς αριθμούς

Ενώ ο μέσος όρος είναι: 17065.36



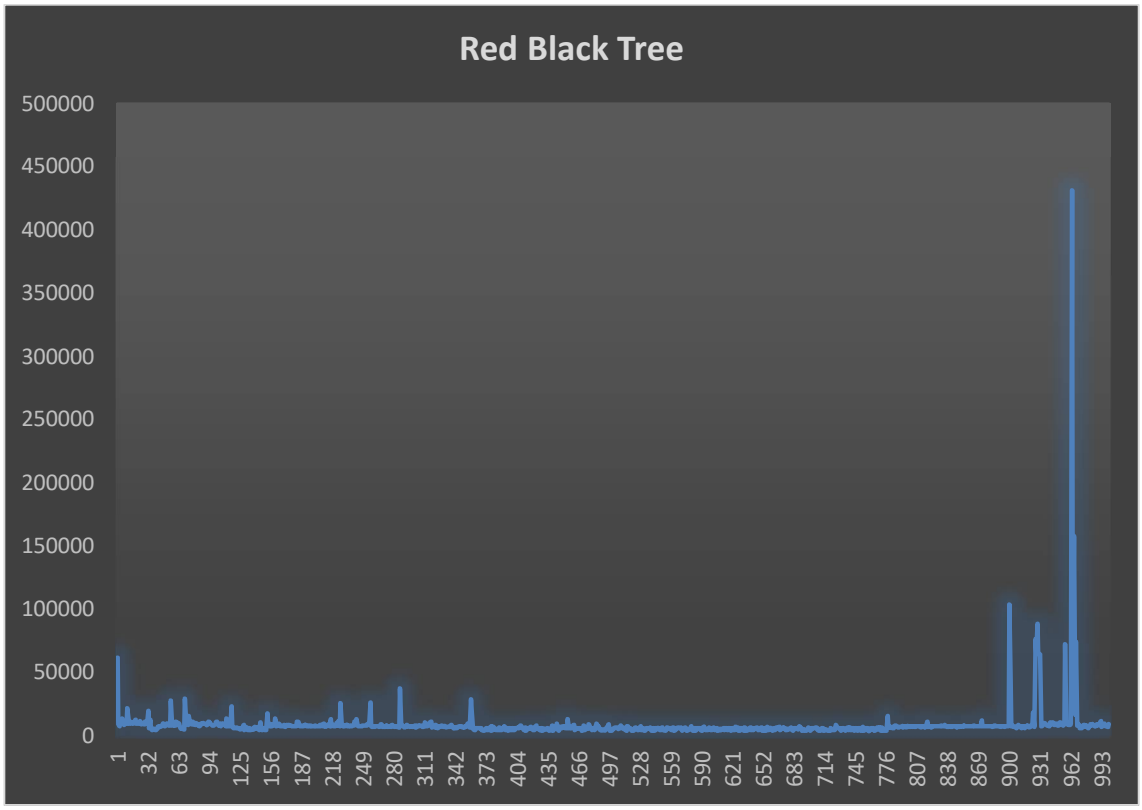
Στην Interpolation Search 1000 αναζητήσεων παρατηρώ ότι:

Ο χειρότερος χρόνος της Interpolation Search είναι O(n)

Ο χειρότερος χρόνος είναι ο: 363161στον αριθμό 445307

Ο καλύτερος: 6710 και εμφανίζεται σε αρκετούς αριθμούς

Ενώ ο μέσος όρος είναι: 13203.25



Στο Red Black Tree 1000 αναζητήσεων παρατηρώ ότι:

Ο χειρότερος χρόνος της αναζήτησης του Red Black Tree είναι O(logn)

Ο χειρότερος χρόνος είναι ο: 431450 στον αριθμό 984622

Ο καλύτερος: 3947 στον αριθμό 815565

Ενώ ο μέσος όρος είναι: 8614.387

Μέρος Ε

Σας δίνεται το αρχείο words.txt το οποίο περιέχει λέξεις αποτελούμενες από χαρακτήρες του λατινικού αλφαβήτου, μία σε κάθε γραμμή του. Γράψτε ένα πρόγραμμα που με βάση αυτή την είσοδο θα κατασκευάζει ένα Digital Tree (Trie) και θα επιτρέπει στο χρήστη να αναζητήσει, να προσθέσει ή και να διαγράψει κάποια λέξη από τη δομή.

Φτιάχνω έναν αρχικό κόμβο που λέω “root” και συνεχίζω κάνοντας εισαγωγή των υπόλοιπων λέξεων που έχω διαβάσει από την “mainClass” στο δέντρο μου. Η αναζήτηση γίνεται ξεκινώντας από την “root” και ερευνώντας αν υπάρχει μονοπάτι των γραμμάτων που πληκτρολογήσαμε για αναζήτηση. Η διαγραφή γίνεται αφού βρούμε ότι η λέξη που αναζητήσαμε υπάρχει. Αν υπάρχει αλλάζουμε την bool μεταβλητή του “leaf” σε false και διαγράφουμε τα υπόλοιπα γράμματα μέχρι να βρούμε κάποιο άλλο “leaf” με τιμή true. Αν δεν υπάρχει leaf με τιμή true τότε διαγράφουμε όλα τα γράμματα της λέξης που πληκτρολογήσαμε μέχρι την “root”.

Περισσότερες λεπτομέρειες για τον κώδικα υπάρχουν με σχόλια μέσα στις κλάσεις του κώδικα καθώς και στο αναλυτικό User Interface.