ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ - Η.Μ.Μ.Υ

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΡΧΕΙΩΝ – ΠΛΗ202

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2

ΑΤΟΜΙΚΗ ΑΝΑΦΟΡΑ ΥΛΟΠΟΙΗΣΗΣ

Παντελής Κωνσταντίνος
2015030070

Σκοπός Εργασίας:

Εξοικείωση με πασιφηλείς πρακτικές συγγραφής κώδικα σε Java(ή άλλη γλώσσα O.O.P) , A.D.T Δένδρα, Δυαδικά Δένδρα Αναζήτησης, Τεχνικές Αναζήτησης σε Δένδρα, Σύγκριση Απόδοσης Δενδρικών Δομών.

Περι Υλοποίησης:

Ο κώδικας για την παρούσα άσκηση γράφθηκε στην γλώσσα Java (***jdk 13.0.1***) στο IDE Eclipse, ενώ η ανάλυση και υλοποίηση αυτής κατατμίζεται σε 3 διακριτά υποτμήματα:

- ***Μέρος Α : Στατική Υλοποίηση Δυαδικού Δένδορυ Αναζήτησης – Δ.Δ.Ε (Binary Search Tree - BST)***

- ***Μέρος Β:*** ***Δυναμική Υλοποίηση Δυαδικού Δένδορυ Αναζήτησης – Δ.Δ.Ε (Binary Search Tree - BST)***

- ***Μέρος Γ:*** ***Πειράματα - Σύγκριση τεχνικών υλοποίησης - δομών.***

Κατακερματισμός Ζητουμένων - Υλοποίσης:

Μέρος Α:

- Δημιουργία Binary Search Tree με χρήση πίνακα δεικτοδότητησης - στατικά
- Υλοποίση βασικών λειτουργιών στο δένδρο(εισαγωγή, αναζήτηση, ενθεματική διάσχιση, αναζήτηση εύρους τιμών).
- Διάβασμα αρχείων απο Αρχείο τιμών.

Μέρος Β:

- Δημιουργία Binary Search Tree δυναμικά – με χρήση ενθηλάκωσης κόμβων (encapsulation – parent,child relationships)
- Υλοποίση βασικών λειτουργιών στο δένδρο(εισαγωγή, αναζήτηση, ενθεματική διάσχιση, αναζήτηση εύρους τιμών).

Μέρος Γ:

- Δοκιμαστικές Αναζητήσεις υπαρκτών και μη κλειδιών, Καταγραφή χρόνων εκτέλεσης.
- Ανάλυση και επεξήγηση αποτελεσμάτων.

Οργάνωση Κώδικα και Documentation:

Ο πηγαίος κώδικας βρίσκεται οργανωμένος σε 3 πακέτα (packages) τα περιεχόμενα των οποίων περιγράφονται αναλυτικά στην παραχθείσα βιβλιογραφία (μορφή Javadoc) ενώ εντός του συμπιεσμένου (.zip) παραδοτέου υπάρχει και αρχείο με διάγραμμα κλάσεων (UML). Επιπλέον δομήθηκε ένα console based menu(τρέχει σε run time και όχι με την εκκίνση) για πλοήγηση στο πρόγραμμα, συλλογή στατιστικών και άλλες χρήσιμες λειτουργίες.

(Σημαντικό: Στην πρώτη εκτέλεση επιλέξτε “help” για πληροφορίες σχετικά με το format εντολών κ.α).

Testing:

$ file -s	[Static BST from file]	$ file -d	[Dynamic BST from file]
$ 1000000	[1000000 keys imported]	$ 1000000	[1000000 keys imported]

stats [Run tests given in exercise via ***TestCase1.class***]

Package:	**Summary:**
treeUtillities	Υλοποιήσεις χρήσιμων "εργαλείων" (Node.class used for dynamic BST implementation)
trees	Interface BST και δύο κλάσεις υλοποίησης για Δένδρα (Δυναμική, Στατική).
test	Κλάση για παραγωγή και αναζήτηση τυχαίων κλειδιών(TestCase.class) και κυρίως κλάση (treeGenerator.java).

Υλοποίηση:

<u>Μέρος Α - Β:</u>

ΔΔΕ Στατικό – A static Binary Search Tree.

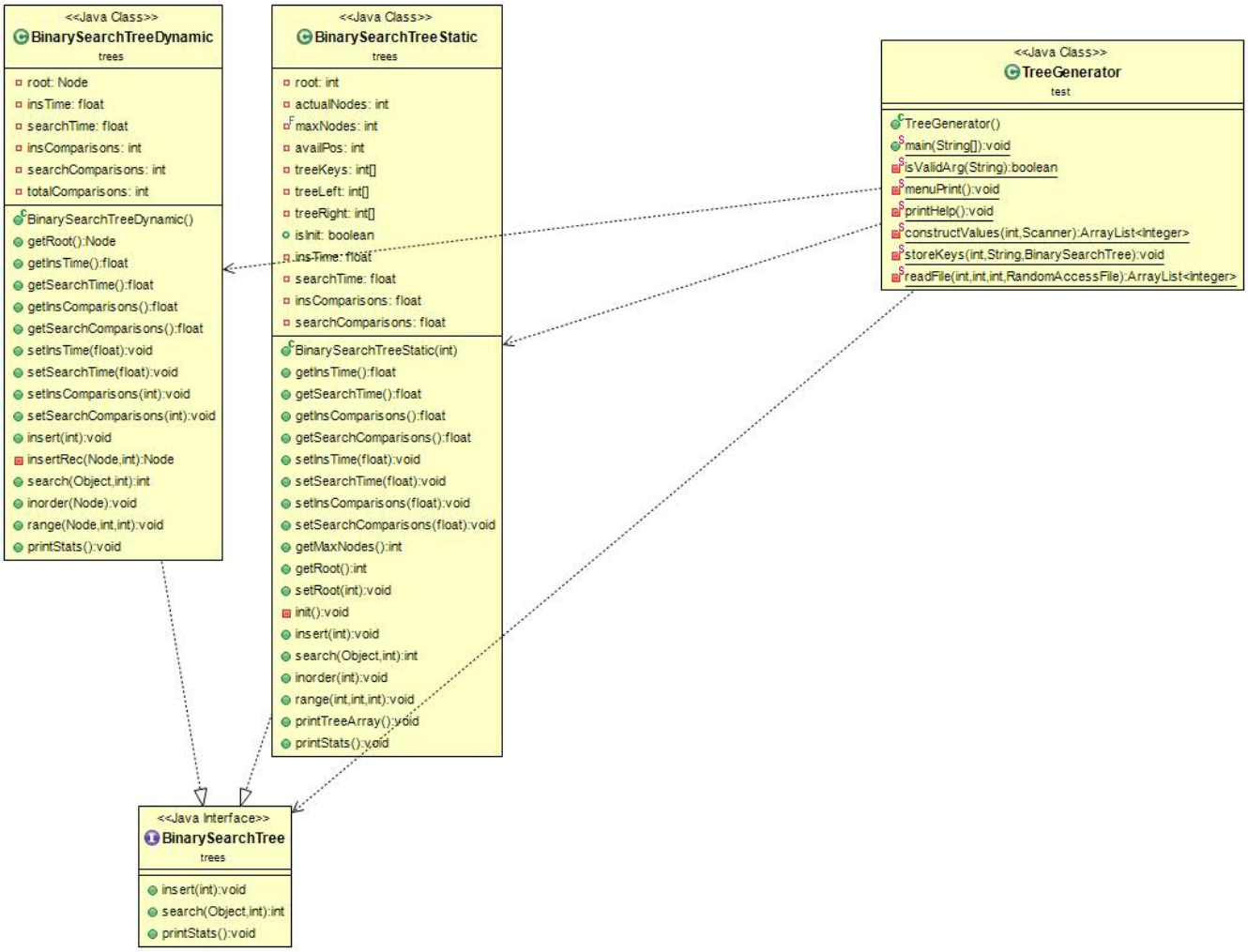
- ***BinarySearchTree.java, BinarySearchTreeStatic.java ,BinarySearchTreeDynamic.java***

Υλοποιήθηκε μια εκδοχή ενός στατικού και ενός δυναμικού Δυαδικού Δένδρου Έρευνας:

- Ορισμός Διεπαφής BinarySearchTreeStatic, -Dynamic (Interface BinarySearchTree), έτσι ώστε οι δυναμική με την στατική υλοποίηση να "μοιράζονται" κοινές μεθόδους κ.α (χρήση Wrappers, Object.class).
- Ορισμός των βασικών μεθόδων στο Δένδρο (Εισαγωγή, Αναζήτηση, Αναζήτηση Εύρους Τιμών, Ενθεματική Διάσχιση)
- Μέθοδοι για διάβασμα στοιχείων απο τα αρχεία με χρήση σελίδων-datapages και *RandomAccessFile* ***(*) .***
- Οι δύο υλοποιήσεις διαφέρουν κυρίως στον τρόπο που δεικτοδοτούν τους κόμβους (μέσω 3 πινάκων η στατική, μέσω ενθηλάκωσης κόμβων η δυναμική). Προφανώς η δεύτερη/δυναμική δεν φέρει περιορισμούς αναφορικά με το πλήθος των στοιχείων που δύναται να αποθηκευθούν, ενώ έχει εν γένει πιο abstract υλοιποίηση, υστερεί παρα ταύτα σε ταχύτητα λόγω δυναμικής δεσμ. κόμβων.

(*): Η περιγραφείσα τεχνική αναπτύχθηκε εκτενός στην Άσκηση 1, και άυξηση κατακόρυφα την ταχύτητα διαβάσματος ειδικά για "μεγάλα" αρχεία (10^7 κλειδιά).

Σχηματικά*:*



Range Search on Sorted Array:

- ***TestCase1.class***

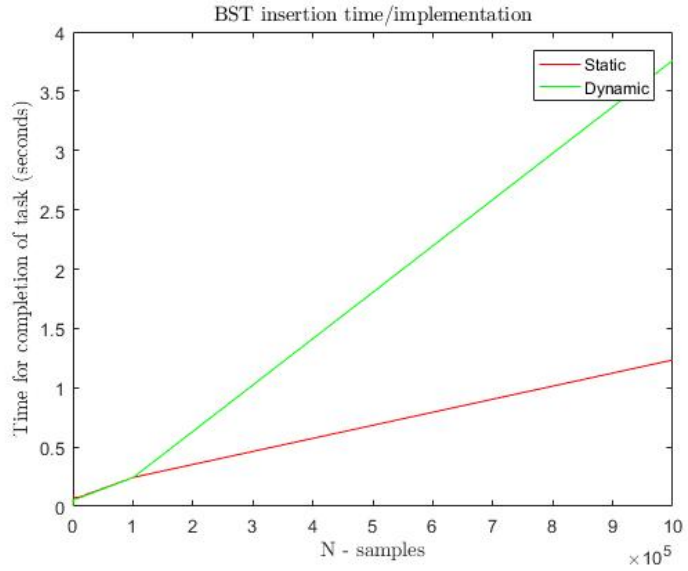
Σημαντική επιπλέον και η υλοποίηση αναζήτησης εύρους τιμών (range search) στο ταξινομημένο πεδίο. Εν γένει διαβάζονται τα όρια(low, high), γίνεται δυαδική αναζήτηση για τα index των ορίων και εκτυπώνονται τα στοιχεία στο (lowIndex, highIndex).

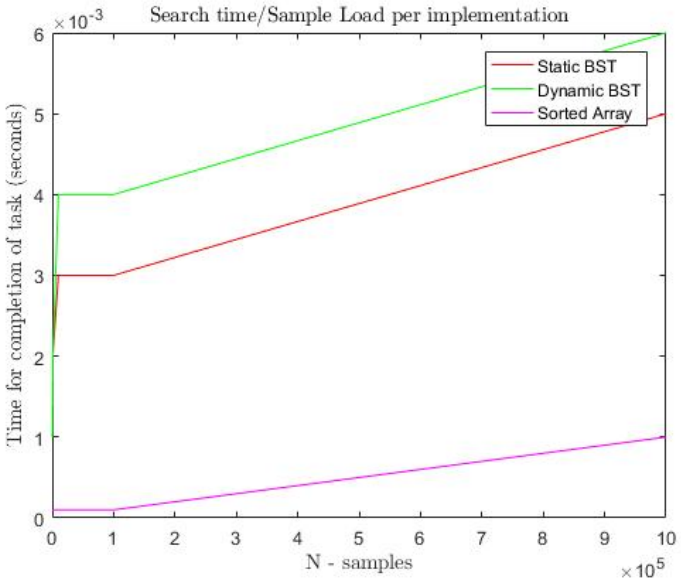
Το παραπάνω εκμεταλλεύται πλήρως την αύξουσα ταξινόμηση του πεδίου πετυχαίνοντας σχεδόν 2 τάξεις μεγέθους μείωση στον αριθμό συγκρίσεων, καθότι μετά την εύρεση των αρχικών δεικτών, το μόνο που συμβαίνει είναι προσπέλαση.

Συλλογή Δεδομένων:

Φορτώθηκαν 1.000.000 ακέραια κλειδιά απο το αρχείο και πραγματοποιήθηκαν τα εξής και στις δύο υλοποιήσεις (στατική, δυναμική).

	Χρόνος Εισαγωγής (sec)	***Συγκρίσεις/ Εισαγωγή***	***Συγκρίσεις/ Αναζήτηση -N κλειδιών (N = 100)***	***Χρόνος αναζήτησης-N κλειδιών (N = 100) (sec)***	***Μέσος αριθμός συγκρίσεων range K, K +100***	***Μέσος αριθμός συγκρίσεων range K, K +1000***
Static BST	1.724 – 2.011	21.777216	1235.5 – 1265	0.004	1241.5	1253.5
Dynamic BST	3.795 – 3.921	26.122075	1286.0 - 1316	0.005	1266.5	1271.4
Ταξινομημένος Array	Δ.Ο	Δ.Ο	19	0.001	39	39





Μέρος Γ:

Σχολιασμός Αποτελεσμάτων

Εν γένει καθίσταται προφανές οτι ανάμεσα σε δύο δενδρικές δομές, αποδοτικότερο χρονικά είναι το στατικό δυαδικό δένδρο, καθότι δεν καταναλώνεται χρόνος για δημιουργία κόμβου, αλλα και για προσπέλαση ενθυλακωμένων δεδομένων (όπως στο δυναμικό ανάλογο του). Τα παραπάνω σαφώς αποτελούν σχεδιαστικές επιλογές καθότι το δυναμικό δένδρο πετυχαίνει μγαλύτερο βαθμό αφαίρεσης στην υλοποίηση του αλλά και μηδενική εξάρτηση απο στατικά μεγέθη (απεριόριστος αριθμός κόμβων), υστερώντας σε ταχύτητα και space – complexity. Αντιθέτως η στατική δομή παρότι πετυχαίνει καλύτερα στοιχεία χωροχρονικής απόδοσης, άπτεται στατικών δεδομένων και φέρει περιορισμούς. Εν κατακλείδι η επιλογή βαραίνει το σχεδιαστή.

Αναφορικά τώρα με τους χρόνους αναζήτησης δεδομένων, οι δενδρικές δομές - ΔΔΕ εμφανίζονται σημαντικά πιο αργές απο την δυαδική αναζήτηση ταξινομημένων πεδίων, καθότι παρότι φέρουν έννοια διάταξης, τα δεδομένα τους δεν είναι σε καμία περίπτωση ταξινομημένα.

Το προαναφερθέν κοστίζει σε προσβάσεις – συγκρίσεις στοιχείων, άρα προφανώς και σε χρονική πολυπλοκότητα ***[O(n) worst case]***, έναντι σαφώς του ταξινομημένου πεδίου ***[O(logn) worst case]***, το οποίο εμφανίζεται να είναι ***η βέλτιστη λύση εαν κανείς επιθυμεί να αναζητεί δεδομένα, ή πεδία τιμών***.

Στην “αντίπερα όχθη” οι ΔΔΕ – BinarySearchTrees αποτελούν την αποδοτικότερη επιλογή δομής σε περιπτώση που επιθυμούμε την τμηματική εισαγωγή ή διαγραφή δεδομένων με διατήρηση της έννοιας διάταξης αυτών ***[O(logn) – O(n) size dependable on BST vs O(n) Array].***

Παραπομπές – Πηγές:

1. **https://www.geeksforgeeks.org**
2. **Φροντιστήριο Μαθήματος.**
3. **Σελίδα Μαθήματος Courses.**