ΠΛΗΡΟΦΟΡΙΕΣ 2^{ΗΣ} ΑΣΚΗΣΗΣ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΜΙΧΑΗΛ ΚΡΑΤΗΜΕΝΟΣ

Αρχικά δημιουργήθηκε μια διεπαφή BinarySearchTree η οποία περιέχει τις μεθόδους ενός ΔΔΕ που ζητήθηκαν να υλοποιηθούν (οι insert, search, range επιστρέφουν το συνολικό αριθμό συγκρίσεων που έγιναν μέσα στην κάθε μία μέθοδο ξεχωριστά) . Ύστερα υλοποιήθηκε μια αφηρημένη κλάση BSTree η οποία υλοποιεί τη BinarySearchTree, ενώ διαθέτει ακόμη και κάποιες βασικές μεθόδους που χρησιμοποιούν τα ΔΔΕ.

Δυαδικό Δένδρο Έρευνας με array

Μετά δημιουργήθηκε και η κλάση BSTArray η οποία είναι το ΔΔΕ με array (3 x N) που ζητήθηκε και είναι υποκλάση της BSTree.

- Η εισαγωγή τυχαίου κλειδιού έγινε μέσω της μεθόδου insert, η οποία αφού ελέγξει αν το δέντρο είναι γεμάτο, εισάγει τον καινούργιο κόμβο στην πρώτη διαθέσιμη θέση (αν δεν υπάρχει κάποια ρίζα, τότε γίνεται αυτός ρίζα και η μέθοδος επιστρέφει). Ύστερα, με επαναληπτικό τρόπο ψάχνει κάτω από ποιο φύλο του δέντρου θα μπει ο καινούργιος κόμβος και μετά επιστρέφει αφού έχει αλλάξει ο πρώτος διαθέσιμος κόμβος.
- Η αναζήτηση τυχαίου κλειδιού έγινε μέσω της μεθόδου search, η οποία ψάχνει επαναληπτικά από τη ρίζα του δέντρου έναν κόμβο, χωρίζοντας το δέντρο σε υποδέντρα μέχρι να τον βρει και αν επιτύχει, **κανονικά** τον τυπώνει στην οθόνη μαζί με τον αριστερό και το δεξιό του κόμβο, εφόσον υπάρχουν.
- Η διάσχιση inorder έγινε μέσω της μεθόδου inorder, η οποία αναδρομικά τυπώνει τα κλειδιά του δέντρου ταξινομημένα.
- Η αναζήτηση εύρους τιμών έγινε μέσω της μεθόδου range, η οποία καλεί τη rangeRec και αυτή αναδρομικά **κανονικά** τυπώνει τα κλειδιά του δέντρου που βρίσκονται ανάμεσα στο δοσμένο εύρος τιμών

ΣΗΜΕΙΩΣΗ: Ο τρόπος με τον οποίο ανιχνευόταν ο επόμενος κόμβος για την μέθοδο insert ήταν με το να φορτώνω στα πεδία right [2][N] του array το επόμενο index του, αφού βέβαια είχε αρχικοποιηθεί ολόκληρο στο -1. Στις εκτυπώσεις γράφω «κανονικά» επειδή ζητήθηκε να μην γίνονται, γιαυτό και τα system out prints είναι σε σχόλια.

Για να υπολογιστεί η απόδοση του ΔΔΕ με array, δημιουργήθηκε και ένα δυναμικό ΔΔΕ, μέσω της κλάσης BSTDynamic, καθώς και ένα arrayList το οποίο διέθετε τα κλειδιά του δέντρου ταξινομημένα (sortedTree), κάτι το οποίο επετεύχθη μέσω της διαδικασίας sortTreeToAL του BSTArray (η οποία ακολουθούσε τον κώδικα της inorder, όπου αντί να τυπώνει τα κλειδιά τα αποθήκευσε στο arrayList).

Οι ζητούμενες μέθοδοι στο BSTDynamic έγιναν παρόμοια με αυτές στο BSTArray με τη διαφορά ότι όλες έγιναν αναδρομικά και με τη χρήση παραπάνω μεθόδων (

insertRec, searchRec, inorderRec, rangeRec) καθώς χρειαζόταν να βρεθεί ο κόμβος (Node) του δοσμένου κλειδιού (int) (κάτι το οποίο έγινε μέσω της μεθόδου findRoot της BSTDynamic) και για να μπορούσαν να μετρηθούν σωστά οι συνολικές συγκρίσεις.

Τα αρχεία κλειδιών για τα δύο ΔΔΕ διαβάστηκαν μέσω της μεθόδου readNumbers, με τη χρήση της έτοιμης κλάσης RandomAccessFile, η οποία αποθηκεύει αρχικά τα κλειδιά σε ένα arrayList και μετά τα περνά στο δέντρο μέσω της μεθόδου calcCompSum, με σκοπό να μετρηθούν σωστά οι ζητούμενοι συνολικοί χρόνοι και οι συνολικές συγκρίσεις. Η μέθοδος printInsertAverage τυπώνει το μέσο αριθμό συγκρίσεων ανα εισαγωγή

Οι 100 αναζητήσεις τυχαίων αριθμών, η μέτρηση του μέσου αριθμού συγκρίσεων ανά αναζήτηση και του συνολικού χρόνου για τα δύο ΔΔΕ έγιναν με τη μέθοδο calcSearchAverage. Για το ταξινομημένο array έγιναν με τη μέθοδο calcAverageCompAL και οι αναζητήσεις με τη μέθοδο binSearch, η οποία αναζητά δυαδικά ένα δοσμένο κλειδί σε ένα arrayList.

Οι 100 τυχαίες αναζητήσεις εύρους τιμών και η μέτρηση του μέσου αριθμού συγκρίσεων ανά αναζήτηση για τα δύο ΔΔΕ έγιναν με τη μέθοδο calcRangeComps. Για το ταξινομημένο array έγιναν με τη μέθοδο calcSTRange, η οποία ζητά από το χρήστη να επιλέξει ένα από τα δύο διαθέσιμα εύρη τιμών (100 και 1000), μέσω της μεθόδου giveRightInput και αναζητά τις ακραίες τιμές (τυχαία τιμή και τυχαία τιμή + το δοσμένο εύρος) με τη μέθοδο binSearch και **κανονικά** τυπώνει τις ενδιάμεσες.

ΣΗΜΕΙΩΣΗ: Οι τυχαίες τιμές παράγονται από τη μέθοδο getRandomNumberInRange. Επίσης όλοι οι ζητούμενοι χρόνοι μετριούνται σε nanoseconds.

ΛΑΘΗ - ΠΑΡΑΠΑΝΩ ΜΕΘΟΔΟΙ:

- Η χρήση της RandomAccessFile στη μέθοδο readNumbers, παρόλο που δουλεύει σωστά, προκάλεσε μια πολύ μεγάλη καθυστέρηση στην αρχή του προγράμματος, κάτι το οποίο θα μπορούσε να αποφευχθεί.
- Έχει υλοποιηθεί και μια διαδικασία εκτύπωσης των δέντρων (print, printRec) για προσωπική διευκόλυνση.
- Έχουν υλοποιηθεί και κάποια μενού επιλογών και οι αντίστοιχες συναρτήσεις που τα τυπώνουν (printMenu, printMenu1, printMenu2), για να μπορεί ο χρήστης να επιλέγει ποιανής δομής τα στοιχεία να δει.
- Έχει χρησιμοποιηθεί και μια κλάση για να διαβάζει πληροφορίες από το χρήστη (StandardInputRead).

ΠΗΓΕΣ

- inorder, range, rangeRec (στο BSTArray): χρησιμοποιήθηκαν σημειώσεις από ένα μαγνητοσκοπημένο φροντιστήριο του κυρίου Καρασαββίδη.
- BSTDynamic: https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/
- printRec (στο BSTDynamic): https://stackoverflow.com/questions/47510048/printing-java-binary-search-tree
- binSearch: https://www.youtube.com/watch?v=hKI93hOfeIk
- getRandomNumberInRange: https://mkyong.com/java/java-generate-random-integers-in-a-range/
- StandardInputRead: αυτή η κλάση μας δόθηκε στο 2ο εξάμηνο από τους εργαστηριακούς βοηθούς στο μάθημα Δομημένος Προγραμματισμός

Οι πηγές αναφέρονται και στα σχόλια σε Javadoc των κλάσεων και συναρτήσεων στο πεδίο @version.