ΠΛΗ202 – ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ & ΑΡΧΕΙΩΝ
2Η ΑΣΚΗΣΗ

ΠΕΤΡΟΥ ΔΗΜΗΤΡΙΟΣ - 2018030070

Δυαδικά Δέντρα Αναζήτησης

1. Σύντομη περιγραφή

Μετά από επιτυχημένο compilation μέσω της εντολής **javac *.java** εντός του πακέτου **mypackage** το πρόγραμμα μπορεί να εκκινηθεί. Καλείται με τη χρήση της εντολής **java mypackage.MyTree**. Κατά την εκκίνησή του, γίνεται ερώτηση στο χρήστη για το path του αρχείου με τους αριθμούς. Σημειώνεται, πως έχουν γίνει οι απαραίτητες ενέργειες για την καλύτερη διαχείριση εξαιρέσεων και σφαλμάτων που ίσως προκύψουν. Αφού η εφαρμογή εκτελέσει μερικά απαραίτητα βήματα. Αυτόματα, έχουν ήδη δημιουργηθεί τα 2 δυαδικά δέντρα και το μονοδιάστατο πεδίο και είναι έτοιμα προς χρήση. Εμφανίζεται ένα απλό μενού χρήστη το οποίο δίνει πρόσβαση στις λειτουργίες που είχαν προδιαγραφεί στην εκφώνηση. Για τις λειτουργίες αναζήτησης η εφαρμογή ρωτάει το χρήστη για τις επιθυμητές τιμές (κλειδιά ή έυρος κλειδιών) και στην συνέχεια καλεί τις αντίστοιχες μεθόδους και εκτυπώνει τα αποτελέσματά τους. Για την διάσχιση inorder εκτυπώνονται στην οθόνη τα στοιχεία των δύο δέντρων σε ταξινομημένη σειρά. Το πρόγραμμα μετά την ολοκλήρωση κάθε ενέργειας επιστρέφει στο μενού χρήστη μέχρι να δοθεί η εντολή “e”, όποτε και τερματίζει.

2. Διαδικασία λύσης – Παρατηρήσεις

Η υλοποίηση του ΔΔΕ με δυναμική παραχώρηση μνήμης χρησιμοποιήθηκε έτοιμη από το διαδίκτυο. Ωστόσο, η λογική υλοποίησης των αλγορίθμων αναζήτησης, εισαγωγής και διάσχισης δεν διαφέρει σε τίποτα από αυτή του στατικού ΔΔΕ.

Για το ΔΔΕ με στατική παραχώρηση μνήμης (υλοποίηση με πεδίο):

a) Δεσμεύεται αριθμός κόμβων που να αντιστοιχεί στον αριθμό των στοιχείων του παρεχόμενου αρχείου (υπολογίζεται: μέγεθος αρχείου στο δίσκο σε byte / 4 byte).
b) Αρχικοποιούνται όλα τα κελιά/στοιχεία των πινάκων που συνθέτουν το δέντρο στην συμβατική τιμή NULL=-1.
c) Οι κόμβοι θα αποθηκευτούν στο δέντρο σειριακά με έναν αυξανόμενο δείκτη διαθεσιμότητας.

a) Εισαγωγή τυχαίου κλειδιού

Πραγματοποιείται αρχικά έλεγχος αν υπάρχει ρίζα στο δέντρο ωστέ να είναι δυνάτη η διάσχιση για την εύρευση του κατάλληλου σημείο για εισαγωγή του νέου κλειδιού. Αν δεν υπάρχει ρίζα, τότε το νέο κλειδί γίνεται ρίζα του δέντρου, αποθήκευεται στον πίνακα και η μέθοδος επιστρέφει. Αν υπάρχει ρίζα και υφίσταται δέντρο, τότε σε μια επαναληπτική δομή ανάλογα τη ανισοτική σχέση μεταξύ κλειδιού και εκάστοτε κόμβου μετακινούμαστε δεξιά ή αριστερά. Σε κάθε κόμβο που περνάμε, ελέγχεται αν έχει παιδί το οποίο να μας ενδιαφέρει. Αν δεν έχει και οι συνθήκες πληρούνται τότε το νέο κλειδί γίνεται το κατάλληλο παιδί του κόμβου, αποθηκεύεται στον πίνακα και η μέθοδος επιστρέφει. Σε άλλη περίπτωση συνεχίζουμε την διαδικάσια μετακίνησης μέχρι να βρούμε κόμβο που να ταιριάζει στις απαιτήσεις μας.

b) Αποθήκευση κλειδιών

Όπως αναφέρθηκε πριν, η θέση στην οποία θα αποθηκεύεται κάθε επόμενο κλειδί, καθορίζεται από μια μεταβλητή που αυξάνεται σειριακά. Από την συνάρτηση εισαγωγής κλειδιού ζητείται κάθε φορά στην αρχή η επόμενη διαθέσιμη θέση, ώστε να είναι έτοιμη να αποθηκεύσει εκεί τον νέο κόμβο. Ωστόσο όταν στην συνάρτηση φτάσει τιμή NULL=-1 τότε η διαδικασία εισαγωγής σταματάει διότι ο πίνακας είναι πλήρης και δεν δύναται να δεχθεί άλλα στοιχεία.

c) Αναζήτηση τυχαίου κλειδιού

Ξεκινώντας από την ρίζα του δέντρου με μια επαναληπτική δομή διατρέχουμε τους κόμβους με κατάλληλο μονοπάτι. Αυτό, καθορίζεται ως εξής:

a) Αν το κλειδί του κόμβου είναι ίσο με αυτό που αναζητούμε, τότε η αναζήτηση θεωρείται επιτυχής και η μέθοδος επιστρέφει.
b) Αν το κλειδί του κόμβου είναι μεγαλύτερο από αυτό που αναζητούμε, τότε συνεχίζουμε προς τα αριστερά πάλι με την διαδικασία (από το βήμα a).
c) Αν το κλειδί του κόμβου είναι μικρότερο από αυτό που αναζητούμε, τότε συνεχίζουμε προς τα δεξιά πάλι με την διαδικασία (από το βήμα a).
d) Αν κανένας από τους προηγούμενους ελέγχους δεν ήταν επιτυχής και πλέον έχουμε φτάσει σε ανύπαρκτο κόμβο, τότε η αναζήτηση θεωρείται ανεπιτυχής και η μέθοδος επιστρέφει.

d) Διάσχιση inorder

a) Ξεκινώντας από τη ρίζα του δέντρου, με την βεβαιότητα ότι υπάρχει, αναδρομικά καθοδηγούμαστε συνεχώς προς τα αριστερά.
b) Με την επιστροφή από την αναδρομή εκτυπώνουμε το κλειδί του εκάστοτε κόμβου μέχρι να φτάσουμε τη ρίζα.
c) Μετά τη ρίζα εκτυπώνουμε το κλειδί του κάθε κόμβου και συνεχίζουμε αναδρομικά προς τα δεξιά.

e) Αναζήτηση εύρους τιμών

Η υλοποιήση της συγκεκριμένης λειτουργικότητας, θυμίζει αρκετά την διάσχιση inorder. Ωστόσο, εδώ επιλέγεται να διασχισθούν και να χρησιμοποιηθούν υποδέντρα που βρίσκονται εντός ενός συγκεκριμένου εύρους τιμών.

a) Ελέγχουμε αρχικά αν υφίσταται δέντρο ή υποδέντρο για την εκάστοτε κλήση της μεθόδου.
b) Αν το κλειδί του εκάστοτε κόμβου είναι μεγαλύτερο από το κάτω όριο του εύρους τότε αναδρομικά συνεχίζουμε αριστερά στο δέντρο.
c) Αν το κλειδί του εκάστοτε κόμβου βρίσκεται ανάμεσα στο κάτω και στο άνω όριο του εύρους τότε μας ενδιαφέρει και το εκτυπώνουμε.
d) Αν το κλειδί του εκάστοτε κόμβου είναι μικρότερο από το ανω όριο του εύρους τότε αναδρομικά συνεχίζουμε δεξιά στο δέντρο.

Στην υλοποίηση έχουν μετατραπεί σε σχόλια οι δύο **System.out.println** στις inrange των δύο δέντρων προκειμένου να εκτελεστούν οι δοκιμές για την απόδοση. Για την εκτέλεση χειροκίνητης αναζήτησης inrange με εκτύπωση παρακαλώ «ξεμαρκάρετε» τις από σχόλια.

3. Πηγές κώδικα & αλγορίθμων

Inrange: https://www.geeksforgeeks.org/print-bst-keys-in-the-given-range/
BinarySearchArray: https://www.geeksforgeeks.org/binary-search/
Dynamic BST: https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/
Inrange Search Array: https://stackoverflow.com/questions/15291363/how-to-use-a-binarysearch-on-a-sorted-array-to-find-the-number-of-integers-withi