Αναφορά 2ης Εργαστηριακής Άσκησης

Δυαδικό Δένδρο Έρευνας με πεδίο

Δομές Δεδομένων & Αρχείων

Μπέτσιος Θωμάς

A. M. 2013030104

Για το δυαδικό δέντρο έρευνας με χρήση πεδίου:

Αρχικά δημιούργησα το πακέτο BST και μια κλάση με το ίδιο όνομα και δημιούργησα έναν δυσδιάστατο πίνακα τύπου **int** . Στην συνέχεια χρησιμοποίησα τις μεθόδους **insert**, **find** και **findRange** που δόθηκαν καθώς και τις αντίστοιχες private **help** μεθόδους για να υλοποιήσω την λειτουργικότητα του δέντρου, όπως μας ζητήθηκε απο την άσκηση. Στην ουσία έκανα μια προσαρμογή του έτοιμου κώδικα σε δομή array σύμφωνα με αυτά που είπαμε και στοφροντηστήριο. Στο μόνο σημείο που πρόσθεσα κώδικα ήταν στην δημιουργία κόμβου στην μέθοδο **insertHelp** που γινόταν με την χρήση του **new,** ενώ τωρα καλώ μια συνάρτηση **getFirstAvailable (ή αλλιώς η getNode που μας ζητήθηκε)** για να βρώ ποια είναιη πρώτη διαθέσιμη θέση ωστε να γράψω το key στην συγκεκριμένη θέση.

Επίσης δημιούργησα και δύο μεθόδους **initializeBstArray** και **initializeStack** για την αρχικοποίηση τουπίνακα και της στοίβας με τις σωστές τιμές. Για την αρχικοποίηση της στοίβας επειδήο πίνακας είναι άδειος αρχικά ή στοίβα δείχνει κάθε φορά την επόμενη θέση μέχρι το τέλος του πίνακα (στην πρώτη σειρά είναι η θέση 2, στην δεύτερη σειρά η θέση 3 ....). Προσπάθησα να βάλω τυχαίους αριθμούς κατα την αρχικοποίηση αλλά μου έβγαζε ένα out of bounds error.

Τέλος χρησιμοποίησα την κλάση **generateRandomKeys** για την δημιουργία 10^5 κλειδιών για την εισαγωγή τους στον πίνακα καθώς και 100 τυχαίων κλειδιών για την αναζήτηση τους μετά την δημιουργία του πίνακα.

- Για την εκτέλεση του κώδικα δημιούργησα μια νέα κλάση **BST_Main** που περιέχει την main function. Δημιουργώ αρχικά 10^5 κλειδιά και τα εισάγω στον πίνακα, μετά υπολογίζω το μ.ο των συγκρίσεων ανα εισαγωγή και το εκτυπώνω. Στην συνέχεια κάνω αναζήτηση 100 τυχαίων κλειδιών καθώς και αναζήτηση εύρους 2 φορές για K=100 και K=1000, ακολουθώντας την ίδια διαδικασία.
- Για την μέτρηση του αριθμού των συγκρίσεων και αναθέσεων χρησιμοποιώ την κλάση **MultiCounter** που μας έχει δοθεί.

Για την υλοποίηση του νηματοειδούς δυαδικού δέντρου αναζήτησης:

Αρχικά να πώ οτι διαχώρισα τις δύο υλοποιήσεις, μετα απο ερώτηση που σας έκανα, παρολο που ενα μεγάλο μέρος του κώδικα είναι ίδιο, γιατί ήθελα να έχω την κάθε δομή ξεχωριστά και οργανωμένα.

Στην ουσία το μόνο κομμάτι που αλλάζει είναι στις μεθόδους **insertHelp**, **findHelp** και **findRangeHelp.** Για την υλοποίηση τωνμεθόδων αυτών χρησιμοποίησα την συνάρτηση που μας δώσατε απο το GeeksForGeeks στην σελίδα (https://www.geeksforgeeks.org/threaded-binary-tree-insertion/) και

προσάρμοσα τον κώδικα ωστε να είναι συμβατός και να υλοποείται με την δομή πίνακα που χρησιμοποιούμε.

Η μέθοδος **insertHelp** δεν επιτρέπει την ύπαρξη διπλότυπων κλειδιών, κάνοντας έλεγχο στην αρχή και στην συνέχεια ανάλογα με την τιμή του κλειδιού πηγαίνει στο δεξί ή αριστερό υποδέντρο μέχρι να βρεί την θέση που θα γίνει η εισαγωγή. Υπάρχουν τρείς περιπτώσεις οι οποίες παίρνονται υπόψην και αυτές είναι η εισαγωγή στη ρίζα, εισαγωγή ως δεξί παιδί η ως αριστερό παιδί. Σε όλες τις περιπτώσεις κοιτάμε τα πεδία του πίνακα **leftThread** και **rightThread** να παίρνουν τις κατάλληλες τιμές. Επίσης να πώ οτι στα πεδία του **leftThread** και **rightThread** χρησιμοποίω τις τιμές **Integer.MIN_VALUE** ως **False** και την τιμή **Integer.MAX_VALUE** ως **True**.

Για την μέθοδο **insertFindHelp** χρησιμοποίησα το πρώτο κομμάτι του κώδικα της **insertHelp** οπου κάνει αναζήτηση στο αριστερό ή στο δεξί υποδέντρο ανάλογα με την τιμή που έχει το κλειδί.

Δεν πρόλαβα να υλοποιήσω την μέθοδο RangeSearch για το ThreadedBST.

- Για την εκτέλεση του κώδικα δημιούργησα μια κλάση **ThreadedBST_Main** που περιέχει και την main function, η οποία ακολουθεί την ίδια ακριβώς λογική και λειτουργικότητα με την κλάση **BST_Main** όπως ανέλυσα παραπάνω.

<u>Τέλος για το ταξινομημένο πεδίο:</u>

Αρχικά δημιούργησα ένα καινούριο package με όνομα **SortedArray** καθώς και μια κλάση με το ίδιο όνομα. Στην συνέχεια δημιούργησα μια κλάση **BinarySearch**, η οποία κάνει δυαδική αναζήτηση στο ταξινομημένο πεδιο το κλειδί που παίρνει ως όρισμα.

Στην συνέχεια δημιούργησα μια κλάση για την αναζήτηση εύρους με όνομα **rangeSearch,** η οποία με δυαδική αναζήτηση ψάχνει το κάτω άκρο και σειριακά ψάχνει τα κλειδί μέχρι η τιμή του κλειδιού να είναι μεγαλύετρη απο το πάνω άκρο που παίρνει ως όρισμα. Αν δεν υπάρχει το κάτω άκρο ψάχνει το πάνω άκρο και ακολουθεί την ίδια δαδικασία μέχρι η τιμή του κλειδιού να είναι μικρότερη απο την τιμή του κάτω άκρου.

- Για την εκτέλεση δημιούργησα μια κλάση SortedArray_Main, η οποία δημιουργεί 10^5 κλειδιά σε πίνακα και τα ταξινομεί σε με την χρήση της κλάσης Arrays.sort. Στην συνέχεια δημιουργεί 100 τυχαία κλειδιά για αναζήτηση και κάνει τις 3 αναζητήσεις(1 δυαδική αναζήτηση και 2 εύρους για K=100 & K=1000).Τέλος υπολογίζει το μ.ο. των συγκρίσεων και τα εκτυπώνει για κάθε περίπτωση.