2η Άσκηση – Δυαδικά Δέντρα
Βαγγέλης Αθανασάκης - 2019030118

Σκοπός της 2ης εργαστηριακής άσκησης ήταν η διερεύνηση της αποδοτικότητας του απλού δυαδικού δέντρου, του threaded δυαδικού δέντρου καθώς και του απλού ταξινομημένου πεδίου σε λειτουργίες όπως: 1) Εισαγωγή στοιχείων 2) Αναζήτηση τυχαίας τιμής κλειδιού 3) Αναζήτηση εύρους τιμών κλειδιών.

Περιγραφή υλοποίησης κώδικα:

- Αρχικά, **δημιουργείται το interface BinarySearchTree** το οποίο έχει τις μεθόδους insert (για εισαγωγή τυχαίου στοιχείου σε δέντρο), find (για αναζήτηση τυχαίου κλειδιού σε δέντρο) και printRange (για εκτύπωση εύρους τιμών δέντρου).
- Στην συνέχεια, **δημιουργείται η κλάση BST** η οποία υλοποιεί το παραπάνω interface. Η κλάση αυτή αναπαριστά το δυαδικό δέντρο αναζήτησης το οποίο υλοποιείται από array. Αρχικά, μέσω του constructor 'BST' και με την βοήθεια της μεθόδου 'setup' αρχικοποιείται το ΔΔΕ και η stack με τις διαθέσιμες θέσεις για εισαγωγή στοιχείων.
- Για την **εισαγωγή τυχαίου στοιχείου**, χρησιμοποιείται η μέθοδος 'insert' η οποία καλεί την μέθοδο 'inserthelp'. Αυτή, δημιουργεί ένα νέο δέντρο αν το δέντρο είναι άδειο αλλιώς αν το δέντρο υπάρχει, βρίσκει την σωστή θέση που πρέπει να αποθηκευτεί το νέο κλειδί μέσω αναδρομικών κλήσεων, αποθηκεύει το κλειδί και ενημερώνει την στοίβα με τις διαθέσιμες θέσεις ώστε να δείχνει στην αμέσως επόμενη διαθέσιμη θέση προς εισαγωγή.
- Για την **αναζήτηση τυχαίου κλειδιού**, χρησιμοποιείται η μέθοδος 'find' η οποία καλεί την μέθοδο 'findhelp'. Αυτή, εξετάζει την πληροφορία της ρίζας, αν είναι η τιμή που ψάχνει τότε επιστρέφει την τιμή αλλιώς κινείται στο κατάλληλο υποδέντρο και καλεί αναδρομικά τον εαυτό της αυτή την φορά με ρίζα το παιδί της αρχικής ρίζας. Η διαδικασία επαναλαμβάνεται έως ότου βρεθεί το κλειδί.
- Για την **αναζήτηση εύρους τιμών**, χρησιμοποιείται η μέθοδος 'printRange' η οποία καλεί την 'printrangehelp'. Αυτή η μέθοδος ακολουθεί την ίδια λογική με την 'findhelp' για την εύρεση της κατάλληλης θέσης του δέντρου η οποία πρέπει να τυπωθεί, μέσω αναδρομικών κλήσεων του εαυτού της.
- Έπειτα, **δημιουργείται η κλάση Threaded BST** η οποία υλοποιεί το interface 'BinarySearchTree'. Αυτή η κλάση αναπαριστά το νηματοειδές δέντρο αναζήτησης το οποίο υλοποιείται με την χρήση array. Αρχικά, όπως και στο ΔΔΕ, μέσω του constructor 'BST' και με την βοήθεια της μεθόδου 'setup' αρχικοποιείται το νηματοειδές ΔΔΕ και η stack με τις διαθέσιμες θέσεις για εισαγωγή στοιχείων.

- Για την **<u>εισαγωγή τυχαίου στοιχείου στο Νηματοειδές ΔΔΕ</u>**, χρησιμοποιείται η 'insert' η οποία καλεί την μέθοδο 'insertHelp'. Αυτή μέσω ενός while loop βρίσκει τον κόμβο-πατέρα ο οποίος είναι κατάλληλος για την εισαγωγή του καινούργιου στοιχείου ως παιδί του(σταματάει όταν ο πατέρας έχει thread αντί για παιδί στην θέση που πρέπει να γίνει η εισαγωγή). Στην συνέχεια, αφού βρεθεί ο κατάλληλος κόμβος-πατέρας γίνεται η εισαγωγή του καινούργιου κλειδιού και ενημερώνεται η stack με τις διαθέσιμες θέσεις μνήμης.
- Για την **<u>αναζήτηση τυχαίου κλειδιού</u>**, χρησιμοποιείται η μέθοδος 'find' η οποία καλεί την μέθοδο 'findHelp'. Αυτή μέσω ενός while loop αρχίζει να ψάχνει το κλειδί από την ρίζα του δέντρου και ανάλογα με την τιμή του κόμβου, επιστρέφει την θέση του κόμβου αν είναι ίση με την τιμή που ψάχνει, αλλιώς κινείται στο αριστερό παιδί αν η τιμή του κόμβου είναι μεγαλύτερη από το κλειδί, διαφορετικά κινείται στο δεξί παιδί. Η διαδικασία επαναλαμβάνεται έως ότου βρεθεί το κλειδί.
- Για την **<u>αναζήτηση εύρους τιμών,</u>** χρησιμοποιείται η 'printRange' η οποία καλεί την μέθοδο 'printRangeHelp' ώστε να τυπώσει ένα εύρος τιμών από 'min' έως 'max' . Αυτή, αρχικά, καλεί την μέθοδο 'find' η οποία βρίσκει την τιμή 'min' και επιστρέφει την θέση της μέσα στο array του δέντρου. Αν αυτή η τιμή δεν υπάρχει τότε η τιμή του 'min' αυξάνεται κάθε φορά κατά ένα μέχρι να βρεθεί η αμέσως μεγαλύτερη τιμή του στο δέντρο. Ακολούθως, τυπώνεται η τιμή του κόμβου στην θέση 'min', προχωράει στο αμέσως μεγαλύτερο στοιχείο που είναι το δεξί παιδί του κόμβου (min)-τυπώνεται η τιμή του και η διαδικασία επαναλαμβάνεται μέχρι η τιμή του κόμβου να είναι μεγαλύτερη από την τιμή του 'max'.
- **<u>Δημιουργείται, επίσης, η κλάση 'BinarySearch'</u>** η οποία εφαρμόζει δυαδική αναζήτηση σε ταξινομημένο πεδίο ακεραίων μέσω της μεθόδου 'search΄ η οποία καλεί την private μέθοδο 'doSearch'.
- **<u>Για την ταξινόμηση του array</u>**, χρησιμοποιείται η μέθοδος 'inorder' της κλάσης 'ThreadedBST' η οποία παίρνει ως όρισμα έναν buffer στον οποίο αποθηκεύει ταξινομημένα τα κλειδιά του νηματοειδούς ΔΔΕ. Η Inorder διάσχιση ενός νηματοειδούς δέντρου εκτυπώνει τα στοιχεία του δέντρου σε αύξουσα σειρά χάρη στην ύπαρξη των threads στο νηματοειδές δέντρο.
- **<u>Για την αναζήτηση εύρους τιμών</u>** από min ως max σε ταξινομημένο πεδίο ακεραίων η κλάση αυτή χρησιμοποιεί την μέθοδο 'searchRange' η οποία αρχικά αναζητεί μέσω της μεθόδου 'doSearch' τις θέσεις των 'min' και 'max' μέσα στο ταξινομημένο array. Αν δεν υπάρχουν, βρίσκει την αμέσως μεγαλύτερη διαθέσιμη τιμή από το 'min' και την αμέσως μικρότερη διαθέσιμη τιμή του 'max' και τυπώνει όλες τις τιμές από την θέση του 'min' έως την θέση 'max'.
- Τέλος, **<u>δημιουργείται η κλάση 'BSTmain'</u>** η οποία περιέχει την main συνάρτηση καθώς **<u>και η κλάση 'MoultiCounter'</u>** η οποία βοηθάει στην συλλογή μετρήσεων.

- Για την εκπόνηση της άσκησης χρησιμοποιήθηκε υλικό από τα φροντιστήρια το οποίο τροποποιήθηκε σύμφωνα με τις ανάγκες της άσκησης (κώδικας κλάσης MultiCounter, κώδικας BinarySearch -> doSearch για δυαδική αναζήτηση, κώδικας για την υλοποίηση της κλάσης BST ->find, insert, printrange, κώδικας για την inorder διάσχιση δέντρου) καθώς και υλικό από το διαδίκτυο το οποίο επίσης τροποποιήθηκε (κώδικας για την insert της κλάσης ThreadedBST https://www.geeksforgeeks.org/threaded-binary-tree-insertion/,κώδικας για την findHelp της κλάσης ThreadedBST https://www.geeksforgeeks.org/threaded-binary-search-tree-deletion/?ref=rp)

Τεκμηρίωση Αποτελεσμάτων:

Μέθοδος	Μέσος αριθμός συγκρίσεων/εισαγωγή	Μέσος αριθμός συγκρίσεων/τυχαία αναζήτηση	Μέσος αριθμός συγκρίσεων/αναζήτηση εύρους (K=100)	Μέσος αριθμός συγκρίσεων/αναζήτηση εύρους(K=1000)
ΔΔΕ Α	**84**	**79**	**132**	**576**
Νηματοειδές ΔΔΕ Β	**130**	**137**	**1328**	**1669**
Ταξινομημένο Πεδίο	**-**	**68**	**1321**	**1670**

```
Counter 1(Insertion At BST): 84
Counter 2(Insertion At Threaded BST): 130

Counter 3(Random Key Search BST): 79
Counter 4(Random Key Search Threaded BST): 137
Counter 9(Binary Random Search): 68
```

```
Counter 5(Random Range(100) Search BST): 132
Counter 6(Random Range(100) Search TBST): 1328
Counter 10(Binary Random Range(100) Search): 1321

Counter 7(Random Range(1000) Search BST): 576
Counter 8(Random Range(1000) Search TBST): 1669
Counter 11(Binary Random Range(1000) Search) 1670
```

- Από τον παραπάνω πίνακα παρατηρούμε ότι το ΔΔΕ είναι πιο αποδοτικό στην εισαγωγή στοιχείων από το Νηματοειδές ΔΔΕ καθώς πραγματοποιούνται λιγότερες συγκρίσεις κατά μέσο όρο για την κάθε εισαγωγή σε αυτό. Αυτό συμβαίνει γιατί αποφεύγονται οι συγκρίσεις οι οποίες εξετάζουν αν ένας κόμβος έχει παιδί ή thread. Ωστόσο, το ΔΔΕ λόγω τον αναδρομικών κλήσεων απαιτεί περισσότερη μνήμη από το σύστημα.
- Στην αναζήτηση τυχαίου κλειδιού, η σειρά αποδοτικότητας είναι η ακόλουθη: Νηματοειδές ΔΔΕ < ΔΔΕ <= Ταξινομημένο πεδίο. Παρατηρούμε δηλαδή ότι ο πιο αποδοτικός τρόπος αναζήτησης είναι στο ταξινομημένο πεδίο (με πολύ μικρή όμως διαφορά από το BST) στο οποίο εφαρμόζεται δυαδική αναζήτηση ενώ ο λιγότερο αποδοτικός τρόπος αναζήτησης είναι στο threaded BST. Αυτό, συμβαίνει γιατί και πάλι πραγματοποιούνται έλεγχοι στο νηματοειδές δέντρο για τον προσδιορισμό του παιδιού ενός κόμβου(αν είναι παιδί ή thread).

- Στις αναζητήσεις εύρους (K=100) και εύρους (K=1000) παρατηρείται και πάλι ότι το ΔΔΕ είναι κατά πολύ πιο αποδοτικό ενώ το threaded BST και το Ταξινομημένο Πεδίο έχουν αποδοτικότητα σχεδόν ίδιας τάξεως. Αυτό συμβαίνει γιατί τόσο στο Threaded BST όσο και στο Ταξινομημένο Πεδίο στην μέθοδο που υλοποιεί την εκτύπωση εύρους (printRangeHelp και searchRange αντίστοιχα) χρησιμοποιούνται μέθοδοι για τον προσδιορισμό των δύο άκρων του εύρους (στο Ταξινομημένο Πεδίο βρίσκεται μόνο το κάτω άκρο) οι οποίες πραγματοποιούν πολλές συγκρίσεις-αναθέσεις. Για παράδειγμα, αν το εύρος είναι min=100,max=200 και τα διαθέσιμα κλειδιά είναι από 150 έως 300 τότε για τον προσδιορισμό του min' και του max' (Θα πρέπει να πάρουν την τιμή min=150,max=200) θα πρέπει τα εκτελεστεί 150 φορές η find στο threaded BST και 50 φορές η doSearch στο Ταξινομημένο Πεδίο.

<u>Συνοψίζοντας:</u>

- Το threaded BST δεν χρησιμοποιεί αναδρομικές κλήσεις γεγονός που συμβάλει στην χρησιμοποίηση λιγότερης μνήμης από το BST ο αλγόριθμος του οποίου χρησιμοποιεί αναδρομή. Ωστόσο, ο αλγόριθμός του Threaded είναι πιο περίπλοκος πράγμα που φαίνεται και στον μέσο αριθμό συγκρίσεων/αναθέσεων.
- Το Ταξινομημένο Πεδίο είναι το αποδοτικότερο στην αναζήτηση ενός τυχαίου κλειδιού αλλά η αποδοτικότητα του στις αναζητήσεις εύρους κυμαίνεται στα ίδια επίπεδα με το Threaded BST.