ΠΛΗ 201 – Άσκηση 2

Στην παρούσα έκθεση περιγράφεται η γενική ιδέα επίλυσης της άσκησης 3, η παρουσίαση των μεθόδων που εφαρμόζει και η ερμηνεία των αποτελεσμάτων.

Binary Search Tree(Array implementation)

Η κλάση αυτή εισάγει n κλειδιά σε έναν πίνακα n x 3 ξεκινώντας από την αρχή του πίνακα(1η στήλη-κλειδί, 2η στήλη-θέση αριστερού παιδιού, 3η στήλη-θέση δεξιού παιδιού), χωρίς να αφήνει να εισαχθούν περισσότερα κλειδιά από τη δεσμευμένη μνήμη. Όλα τα κλειδιά εισάγονται διαδοχικά μία-μία γραμμή. Γραμμή πίνακα χωρίς αποθηκευμένο κόμβο έχει στην $3^{η}$ στήλη τιμή που δείχνει την επόμενη άδεια θέση για αποθήκευση κόμβου.

Κόμβος, αποθηκευμένος στη γραμμή K, χωρίς παιδί έχει στη θέση [K][2], [K][3] την τιμή -1.

Εκτός από την εισαγωγή τυχαίου κλειδιού υποστηρίζει τις εξής λειτουργίες

- Αναζήτηση τυχαίου κλειδιού(αναδρομική κλήση μεθόδου)
- Αναζήτηση εύρους κλειδιών(αναδρομική κλήση μεθόδου)
- Εκτύπωση δένδρου σε μορφή πίνακα(public μέθοδος)
- Ενθεματική διάσχιση (inorder traversal) του δένδρου
- Εύρεση ρίζας του δένδρου(public μέθοδος)

Threaded Binary Search Tree(Array implementation)

Η κλάση αυτή είναι μία τροποποίηση της BST δημιουργώντας πίνακα n x 5. Οι 3 πρώτες στήλες είναι αντίστοιχες με αυτές του BST. Η $4^{η}$ στήλη έχει τιμή 1 αν αριστερά έχει νήμα, 0 αν έχει παιδί. Η $5^{η}$ έχει τιμή 1 αν το δεξιά έχει νήμα, 0 αν έχει παιδί.

Η κλάση αυτή υποστηρίζει τις ίδιες λειτουργίες με την BST και μία επιπλέον για την εύρεση του inorder successor ενός κλειδιού.

Επίσης σημαντική διαφορά στην υλοποίηση αυτής της κλάσης αποτελεί το γεγονός ότι η inorder traversal γίνεται χωρίς αναδρομή και έτσι μειώνεται κατανάλωση μνήμης και αυξάνεται η ταχύτητα εκτέλεσης του προγράμματος.

Multicounter

Κλάση που δημιουργεί static int[10] και χρησιμοποιείται για την μέτρηση δεικτών απόδοσης, όπως παρουσιάσθηκε σε φροντιστήριο του μαθήματος.

SortedArray

Κλάση που ταξινομεί ένα int array και το χρησιμοποιεί σαν δομή δεδομένων. Υποστηρίζει τις λειτουργίες της αναζήτησης τυχαίου κλειδιού(binary), αναζήτηση εύρους κλειδιών(binary για να βρω το min και serial traversal μέχρι τον max), εκτύπωση όλου του array.

Main

Εκεί βρίσκεται η main μέθοδος μέσω της οποίας γίνεται η δημιουργία τυχαίων μοναδικών κλειδιών για εισαγωγή και αναζήτηση, η αρχικοποίηση των δομών δεδομένων και οι ζητούμενες αναζητήσεις .

Αποτελέσματα

Η αξιολόγηση των δομών δεδομένων έγινε σε δύο φάσεις. Η 1η έγινε με καταμέτρηση των εντολών συγκρίσεως που περιέχει το πρόγραμμα(πχ. while, if, if else. !=, < κλπ). Η 2η έγινε με επιπλέον καταμέτρηση του αριθμού κλήσεων των μεθόδων για τις ζητούμενες λειτουργίες. Οι παρακάτω μετρήσεις προκύπτουν έπειτα από εκτέλεση του προγράμματος (10^5 κλειδιά).

*Οι πληροφορίες αυτές εμφανίζονται και στην κονσόλα(ενδέχεται να διαφέρουν λόγω τυχαιότητας των παραγόμενων κλειδιών).

Πίνακας 1: Αποτελέσματα 1ης μεθόδου καταμέτρησης(συγκρίσεις)

Μέσος αρθμός προσβάσεων δίσκου ανά εισαγωγή	Μέσος αρθμός προσβάσεων δίσκου ανά τυχαία αναζήτηση	Μέσος αρθμός προσβάσεων δίσκου ανά διαγραφή	Μέσος αρθμός προσβάσεων δίσκου για τυχαία αναζήτηση εύρους (K=10)	Μέσος αρθμός προσβάσεων δίσκου για τυχαία αναζήτηση εύρους (K=1000)
ΔΔΕ Α	77	61	98	463
Νηματοειδές ΔΔΕ Β	115	84	126	490
Ταξινομημένο πεδίο		52	74	255

Πίνακας 2: Αποτελέσματα 2ης μεθόδου καταμέτρησης(συγκρίσεις, κλήσεις μεθόδων)

Μέθοδος	Μέσος αριθμός συγκρίσεων/ εισαγωγή	Μέσος αριθμός συγκρίσεων/ τυχαία αναζήτηση	Μέσος αριθμός συγκρίσεων/ τυχαία αναζήτηση εύρους (K=100)	Μέσος αριθμός συγκρίσεων/ τυχαία αναζήτηση εύρους (K=1000)
ΔΔΕ Α	102	86	139	678
Νηματοειδές ΔΔΕ Β	143	107	148	507
Ταξινομημένο πεδίο		68	88	269

<u>Ερμηνεία αποτελεσμάτων</u>

Η 1η καταμέτρηση κρύβει το βασικό μειονέκτημα του BST, δηλαδή την αναδρομή. Έτσι η εισαγωγή, αναζήτηση τυχαίου κλειδιού /εύρους κλειδιών του Threaded BST απαιτεί περισσότερες συγκρίσεις γιατί πριν μετακινηθεί σε αριστερό/δεξί κόμβο πρέπει να ελεγχθεί αν είναι παιδί ή νήμα.

Η 2η καταμέτρηση φανερώνει την ανωτερότητα του Threaded BST στην αναζήτηση εύρους κλειδιών. Μάλιστα φαίνεται ότι όσο μεγαλώνει το εύρος αναζήτησης, τόσο καλύτερη είναι η απόδοση του Threaded BST έναντι του BST. Αυτό είναι αναμενόμενο και από τον τρόπο δημιουργίας του πρώτου, αφού μπορεί να βρεθεί εύκολα ο inorder successor κάθε κόμβου, χωρίς αναδρομικές αναζητήσεις σε κάθε υποδένδρο.

Εκ πρώτης όψεως το ταξινομημένο πεδίο φαίνεται το πιο αποδοτικό. Αυτό όμως ισχύει μόνο σε περίπτωση που γνωρίζουμε όλα τα κλειδιά εκ των προτέρων και δεν γίνονται επιπλεόν εισαγωγές/διαγραφές. Αν απαιτούνταν κάτι τέτοιο θα η απόδοση θα έπεφτε καθώς θα χρειαζόταν εκ νέου ταξινόμηση-μετακίνηση κλειδιών. Επομένως είναι δομή δεδομένων με ελάχιστες πραγματικές εφαρμογές.

<u>Πηγές πληροφόρησης</u>

- Project countingdemo από το 2ο φροντιστήριο
- Κώδικας από το useful_for_2nd_project.java από το φροντιστήριο
- https://www.geeksforgeeks.org/print-bst-keys-in-given-range-o1-space/
- https://www.geeksforgeeks.org/inorder-successor-in-binary-search-tree/