Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2020

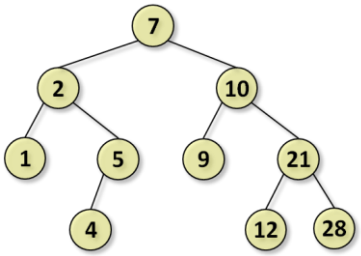
6η Εργαστηριακή Άσκηση
Δυαδικά Δένδρα Αναζήτησης: Κοκκινόμαυρα Δένδρα
Παράδοση έως Παρασκευή 10/12, 12:00 από το eCourse

ΠΡΟΣΟΧΗ: Γράψτε σε κάθε αρχείο που παραδίδετε τα ονόματα και τους Α.Μ. των μελών της ομάδας σας. Συμπεριλάβετε όλα τα αρχεία σας (κώδικας Java και lab6results.txt) σε ένα zip αρχείο το οποίο να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.

Το πρόγραμμα BinarySearchTree.java υλοποιεί μια ένα απλό (μη ισορροπημένο) δυαδικό δένδρο αναζήτησης, το οποίο αποθηκεύει αντικείμενα γενικού τύπου Item με κλειδιά γενικού τύπου Key. Τροποποιήστε το παραπάνω πρόγραμμα έτσι ώστε να υλοποιεί ένα κοκκινόμαυρο δένδρο.

Το BinarySearchTree.java περιλαμβάνει τις ακόλουθες μεθόδους:

Item search(Key key)	Επιστρέφει το αντικείμενο με κλειδί key.
void insert(Key key, Item item)	Εισάγει αντικείμενο item με κλειδί key.
void print()	Τυπώνει το δυαδικό δένδρο όπως δείχνει το παρακάτω σχήμα. Η εκτύπωση γίνεται από επάνω προς τα κάτω με τη σειρά που εμφανίζονται οι κόμβοι από τα δεξιά προς τα αριστερά, δηλαδή το δένδρο εμφανίζεται περιστραμμένο αριστερά κατά 90^o. Σε κάθε κόμβο εμφανίζει το κλειδί, καθώς και μέσα σε αγκύλες το ύψος και το πλήθος των απογόνων του κόμβου.




```
                    28[1,1]
          21[2,3]
                    12[1,1]
  10[3,5]
          9[1,1]
7[4,10]
          5[2,2]
                    4[1,1]
  2[3,4]
          1[1,1]
```

Επίσης περιλαμβάνει τις βοηθητικές μεθόδους:

BSTreeNode searchNode(Key key)	Επιστρέφει τον κόμβο που περιέχει το κλειδί key. Αν το key δεν υπάρχει στο δένδρο τότε επιστρέφει τον τελευταίο κόμβο στο μονοπάτι αναζήτησης.
BSTreeNode insertNode(Key key, Item item)	Εισάγει αντικείμενο item με κλειδί key και επιστρέφει τον κόμβο που περιέχει το key.

Κοκκινόμαυρο δένδρου

Τροποποιήστε το πρόγραμμα BinarySearchTree.java έτσι ώστε να υλοποιεί ένα κοκκινόμαυρο δένδρο αναζήτησης. Κάθε κόμβος v αποθηκεύει μια επιπλέον Boolean μεταβλητή isRed, έτσι ώστε

v.isRed == true αν ο κόμβος v είναι κόκκινος

v. isRed == false αν ο κόμβος v είναι μαύρος.

Κατά την εισαγωγή ενός νέου στοιχείου, πριν ολοκληρωθεί η εκτέλεση της insert(key, item), πρέπει να ελέγξουμε αν παραβιάζεται η αναλλοίωτη συνθήκη του κοκκινόμαυρου δένδρου η οποία επιβάλει τα παιδιά ενός κόκκινου κόμβου να είναι μαύρα. Αυτό συμβαίνει όταν ο αλγόριθμος εισαγωγής τοποθετεί ένα νέο (κόκκινο) κόμβο (με κλειδί key και αντικείμενο item) ως παιδί ενός κόκκινου κόμβου. Σε αυτήν την περίπτωση θα πρέπει να εκτελέσουμε τη μέθοδο αποκατάστασης fix() του κοκκινόμαυρου δένδρου, την οποία θα πρέπει να υλοποιήσετε. (Δείτε το συνοδευτικό αρχείο RedBlackTree.pdf.)

Υπόδειξη: Θα χρειαστεί να υλοποιήσετε βοηθητικές μεθόδους, όπως οι rotateLeft() και rotateRight(). (Δείτε το συνοδευτικό αρχείο RedBlackTree.pdf.)

Ανανέωση του ύψους και του πλήθους των απογόνων των κόμβων

Το πρόγραμμα BinarySearchTree.java περιέχει μερικές ακόμα βοηθητικές μεθόδους οι οποίες ανανεώνουν το ύψος και το πλήθος των απογόνων των κόμβων μετά από μια εισαγωγή.

void updateNode(BSTreeNode x)	Ανανεώνει τα πεδία x. height και x. N ενός κόμβου x, τα οποία διατηρούν το ύψος του κόμβου και το πλήθος των απογόνων του, αντίστοιχα.
void updateHeights(BSTreeNode x)	Ανανεώνει το ύψος και το πλήθος των απογόνων κάθε κόμβου στο μονοπάτι από τον κόμβο x προς τη ρίζα.

Χρησιμοποιήστε αυτές τις μεθόδους στη fix() έτσι ώστε να διατηρούνται τα σωστά ύψη και το πλήθος των απογόνων των κόμβων μετά τη διαδικασία αποκατάστασης του κοκκινόμαυρου δένδρου.

Αποθηκεύστε το τροποποιημένο πρόγραμμα σας ως RedBlackTree.java.

Εκτέλεση προγραμμάτων

Η main() μέθοδος διαβάζει από την είσοδο το πλήθος n των αντικειμένων που θα εισαχθούν στο δένδρο, όπου για κάθε αντικείμενο επιλέγεται ένα τυχαίο κλειδί στο διάστημα $[0,2n)$. Αν δύο ή περισσότερα αντικείμενα έχουν το ίδιο κλειδί k, τότε μετά τις εισαγωγές το δένδρο θα περιέχει μόνο το τελευταίο αντικείμενο με κλειδί k που θα εισαχθεί. (Άρα, το δένδρο μπορεί να περιέχει λιγότερους από n κόμβους όταν ολοκληρωθούν οι εισαγωγές.) Μετά την εισαγωγή, εκτελεί διαδοχικές αναζητήσεις αυτών των n κλειδιών.

Εκτελέστε το πρόγραμμα RedBlackTree. java για $n = 16$ αντικείμενα και αποθηκεύστε τα αποτελέσματα της εκτέλεσης στο αρχείο lab6results. txt. Πρέπει να τροποποιήσετε τη μέθοδο print() έτσι ώστε να επισημαίνονται ποιοι κόμβοι είναι μαύροι και ποιοι κόκκινοι.

Για την εκτέλεση του προγράμματος γράψτε

java RedBlackTree 16.

Επαναλάβατε το ίδιο για $n = 1000$ και $n = 1000000$ και τυπώστε μόνο το ύψος του τελικού δένδρου. Θυμηθείτε ότι ένα κοκκινόμαυρο δένδρο με n κλειδιά έχει ύψος το πολύ ίσο με $2\lg(n+1)$.

- Συγκρίνετε τα ύψη και τους χρόνους εκτέλεσης που λάβατε με τις αντίστοιχες τιμές του προγράμματος BinarySearchTree.java.

- Συγκρίνετε, επίσης, τους χρόνους εκτέλεσης των προγραμμάτων BinarySearchTree.java και RedBlackTree.java με το έτοιμο πρόγραμμα TreeMapTest.java

Παραδοτέα

Ανεβάστε στο eCourse ένα zip αρχείο με το τελικό πρόγραμμα σας RedBlackTree.java καθώς και με το αρχείο των αποτελεσμάτων lab6results. txt. Το zip αρχείο πρέπει να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.

Προαιρετικό μέρος (bonus +0,5 μονάδα)

Υλοποιήστε στα προγράμματα BinarySearchTree.java και RedBlackTree.java την παρακάτω μέθοδο:

$range(k, m)$: Επιστρέφει το πλήθος των κλειδιών του δένδρου τα οποία βρίσκονται στο διάστημα $[k, m]$.

Για παράδειγμα, στο δένδρο του σχήματος (στην πρώτη σελίδα) έχουμε $range(2,9) = 5$.

Η μέθοδος θα πρέπει να εκτελείται σε χρόνο ανάλογο του ύψους του δένδρου. *Υπόδειξη: Θυμηθείτε τη ρουτίνα $rank(Key\ k)$ που είδαμε στο μάθημα.*