ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ – ΕΡΓΑΣΙΑ 3 (ΑΝΑΦΟΡΑ ΠΑΡΑΔΟΣΗΣ)

ΣΥΝΤΑΚΤΗΣ ΕΡΓΑΣΙΑΣ – ΑΝΑΦΟΡΑΣ: ΧΑΡΑΛΑΜΠΙΔΗΣ ΝΑΠΟΛΕΩΝ (3220225)

ΕΠΕΞΗΓΗΣΗ ΣΥΝΑΡΤΗΣΕΩΝ ΜΕΡΟΥΣ Β

- **void insert (LargeDepositor item):** Υλοποιείται με τη βοήθεια των παρακάτω συναρτήσεων:
    - **insertAsRoot(LargeDepositor item, TreeNode node):** Αυτή η μέθοδος εισάγει έναν νέο κόμβο στο δέντρο, χρησιμοποιώντας έναν τυχαίο αλγόριθμο εισαγωγής. Ανάλογα με το τυχαίο αποτέλεσμα, ο νέος κόμβος εισάγεται στη ρίζα ή στο αριστερό ή δεξί υπόδεντρο. **Η πολυπλοκότητα είναι O(logn)** στη χειρότερη περίπτωση λόγω της αναζήτησης της κατάλληλης θέσης στο δέντρο.
    - **insertAtRoot(LargeDepositor item, TreeNode node):** Αυτή η μέθοδος είσαγει έναν νέο κόμβο στο δέντρο και τον κάνει ρίζα, εκτελώντας αντιστροφή ή περιστροφή ανάλογα με τη σχέση του με τον υπάρχοντα κόμβο. **Η πολυπλοκότητα είναι επίσης O(logn)** στην χειρότερη περίπτωση, καθώς ανάλογα με τη σχέση του νέου κόμβου με τον υπάρχοντα, ενδέχεται να χρειαστεί να διαπράξει περιστροφές.
- **void load(String filename):** Η εισαγωγή δεδομένων από ένα αρχείο επιτυγχάνεται με τη βοήθεια των ενσωματωμένων συναρτήσεων **BufferedReader,** και **FileReader.** Κάθε γραμμή διαβάζεται μία μία και χωρίζεται σε τμήματα βάσει κενών ενδιάμεσα απ'το κάθε δεδομένο (δηλαδή το πρώτο δεδομένο είναι το ΑΦΜ, μετά το κενό είναι το δεδομένο που περιέχει το όνομα, κ.ο.κ). Μόλις διαβαστούν όλα τα δεδομένα της γραμμής, δημιουργείται ένα αντικείμενο τύπου **LargeDepositor** και προστίθεται στο δέντρο. Αυτό γίνεται για κάθε γραμμή του αρχείου και η διαδικασία τερματίζει μόλις το πρόγραμμα δεν εντοπίσει επόμενη γραμμή. **Δεν υπάρχει κάποια απαίτηση πολυπλοκότητας για την συγκεκριμένη συνάρτηση.**
- **void updateSavings(int AFM, double savings, TreeNode node):** Αυτή η μέθοδος εντοπίζει έναν κόμβο με βάση το ΑΦΜ και ενημερώνει την μεταβλητή **savings** του εκάστοτε LargeDepositor. Σε περίπτωση που δεν είναι δυνατός ο εντοπισμός με το συγκεκριμένο ΑΦΜ, εμφανίζεται μήνυμα λάθους στην οθόνη. Στη συνέχεια, υπολογίζεται η διαφορά **cmp** μεταξύ του ΑΦΜ που δόθηκε ως όρισμα και του ΑΦΜ του καταθέτη στον τρέχοντα κόμβο **(node.item.key()).** Αν η διαφορά είναι αρνητική, τότε η αναδρομική κλήση γίνεται στο αριστερό υπόδεντρο για να ενημερωθούν οι εξοικονομήσεις του καταθέτη. Αν η διαφορά είναι θετική, τότε η αναδρομική κλήση γίνεται στο δεξιό υπόδεντρο. Αν η διαφορά είναι μηδέν, τότε οι εξοικονομήσεις του καταθέτη αντικαθίστανται με το νέο ποσό. **Η πολυπλοκότητα είναι O(logn). Ωστόσο, σε χειρότερη περίπτωση, μπορεί να φτάσει έως O(n), όπου n είνια ο αριθμός των κόμβων του δέντρου.**
- **LargeDepositor searchByAFM(int AFM, TreeNode node):** Η μέθοδος δέχεται τον ΑΦΜ που πρέπει να αναζητηθεί και τον τρέχοντα κόμβο ως ορίσματα. Σε περίπτωση που ο ΑΦΜ δεν βρεθεί στο δέντρο, εμφανίζεται αντίστοιχο μήνυμα λάθους στην οθόνη. Αν ο ζητούμενος ΑΦΜ είναι μικρότερος από τον ΑΦΜ του τρέχοντος κόμβου, τότε η αναζήτηση συνεχίζεται στο αριστερό υπόδεντρο του τρέχοντος κόμβου. Αν ο ζητούμενος ΑΦΜ είναι μεγαλύτερος, τότε η αναζήτηση συνεχίζεται στο δεξί υπόδεντρο. Αν οι δύο ΑΦΜ είναι ίσοι, τοτε η αναζήτηση ολοκληρώνεται και ο καταθέτης που αντιστοιχεί στον συγκεκριμένο ΑΦΜ

επιστρέφεται. **Η πολυπλοκότητα της μεθόδου εξαρτάται κυρίως απ΄το ύψος του δέντρου. Σε ένα ισορροπημένο δυαδικό δέντρο αναζήτησης, η πολυπλοκότητα ειναι O(logn). Ωστόσο, στη χειρότερη περίπτωση μπορεί να φτάσει έως O(n), όπου n είναι ο αριθμός των κόμβων του δέντρου.**

- **StringDoubleEndedQueueImpl<LargeDepositor> searchByLastName(String last_name, TreeNode node, StringDoubleEndedQueueImpl<LargeDepositor> result):** Η μέθοδος δέχεται το επώνυμο που πρέπει να αναζητηθεί, τον τρέχοντα κόμβο, και μία ουρά αποτελεσμάτων ως ορίσματα. Κάθε φορά που καλείται η μέθοδος αυτή, συγκρίνει το δοθέν επώνυμο με το επώνυμο του καταθέτη του τρέχοντος κόμβου. Αν ο κόμβος είναι **null,** τότε η αναζήτηση στο συγκεκριμένο υπόδεντρο τερματίζεται. Υπολογίζεται η διαφορά **cmp** μεταξύ του δοθέντος επωνύμου και του επωνύμου του τρέχοντος κόμβου. Αν το δοθέν επώνυμο είναι «μικρότερο» από εκείνο του τρέχοντος κόμβου, η αναζήτηση συνεχίζεται στο αριστερό υπόδεντρο. Αν είναι μεγαλύτερο, συνεχίζεται στο δεξί υπόδεντρο. Αν είναι ίδια, τότε προστίθεται ο καταθέτης του τρέχοντος κόμβου στην ουρά αποτελεσμάτων, και συνεχίζεται η αναζήτηση στο δεξί υπόδεντρο για να βρεθούν επιπλέον καταθέτες με το ίδιο επώνυμο. **Η πολυπλοκότητα ειναι O(logn), καθώς η μέθοδος έχει αποδοτική αναζήτηση, ακόμα και για μεγάλα δέντρα.**
- **TreeNode remove(int AFM, TreeNode node):** Η μέθοδος υλοποιεί την διαγραφή ενός κόμβου από το δυαδικό δέντρο. συγκρίνει το κλειδί του κόμβου με το κλειδί που αναζητούμε. Αν είναι μικρότερο, εκτελεί αναδρομικά τη μέθοδο **remove** στο αριστερό υποδέντρο. Αν είναι μεγαλύτερο, εκτελεί τη μέθοδο **remove** στο δεξί υποδέντρο. Αν το βρει, ενώνει τα αριστερά και δεξιά υποδέντρα για να δημιουργήσει ένα νέο υπόδεντρο χωρίς τον τρέχοντα κόμβο. Ενημερώνει το μέγεθος του κόμβου με βάση τα μεγέθη των αριστερών και δεξιών υποδέντρων και επιστρέφει τον ενημερωμένο κόμβο. Η πολυπλοκότητα χρόνου αυτής της μεθόδου εξαρτάται από το ύψος του δέντρου, οπότε είναι **O(logn)**
- **double getMeanSavings():** Η μέθοδος υπολογίζει τον μέσο όρο των αποταμιεύσεων όλων των καταθετών στο δυαδικό δέντρο. Αρχικά ελέγχει εάν το δέντρο είναι άδειο. Αν είναι, επιστρέφει 0.0, δεδομένου ότι δεν υπάρχουν καταθέτες για υπολογισμό. Διαφορετικά, υπολογίζει το άθροισμα των αποταμιεύσεων καλώντας την αναδρομική μέθοδο getSumSavings, η οποία δέχεται έναν κόμβο ως είσοδο και επιστρέφει το άθροισμα των αποταμιεύσεων όλων των καταθετών στο υπόδεντρο που ξεκινά από τον κόμβο αυτό. Το άθροισμα αυτό διαιρείται με τον αριθμό των κόμβων στο δέντρο (root.N), που αντιστοιχεί στον συνολικό αριθμό των καταθετών. **Η πολυπλοκότητα χρόνου της μεθόδου αυτής εξαρτάται απο τον αριθμό των κόμβων στο δέντρο, αρα είναι O(n).**
- **void printTopLargeDepositors(TreeNode node, int k):** Η μέθοδος δέχεται δύο ορίσματα, τον τρέχοντα κόμβο του δυαδικού δέντρου και έναν ακέραιο k που υποδηλώνει πόσους κορυφαίους καταθέτες να εκτυπωθούν. Η μέθοδος ελέγχει αν ο κόμβος είναι null. Σε περίπτωση που δεν είναι, λαμβάνει αντικείμενα απ΄την **topDepositorsQueue,** και συνεχίζει μέχρι να συμπληρωθεί η ουρά, μέχρι δηλαδή να φτάσει μέγεθος k. Η διαδικασία αυτή γίνεται στο δεξί υπόδεντρο. Έπειτα επαναλαμβάνεται και για το αριστερό υπόδεντρο. **Η πολυπλοκότητα είναι O(n), διότι ο χρόνος εκτέλεσης της συνάρτησης εξαρτάται απ'τον αριθμό των κόμβων στο δέντρο.**
- **void printByAFM(TreeNode node):** Μέθοδος που χρησιμοποιείται για να εκτυπώσει τους καταθέτες του δυαδικού δέντρου σε σειρά με βάση το ΑΦΜ τους. Η μέθοδος δέχεται έναν κόμβο node ως όρισμα. Ξεκινάει από το αριστερό υποδέντρο του τρέχοντος κόμβου. Εάν ο κόμβος δεν είναι null, τότε καλεί αναδρομικά τον εαυτό της για το αριστερό υποδέντρο του τρέχοντος κόμβου. Στη συνέχεια, εκτυπώνει τον καταθέτη που αντιστοιχεί στον τρέχοντα κόμβο. Τέλος, καλεί αναδρομικά τον εαυτό της για το δεξί υποδέντρο του τρέχοντος

κόμβου. **Η πολυπλοκότητα είναι O(n), διότι ο χρόνος εκτέλεσης της συνάρτησης εξαρτάται απ'τον αριθμό των κόμβων στο δέντρο.**