Void updateSavings(int AFM, double savings)

Αναζητούμε τον LargeDepositor με το δοσμένο ΑΦΜ μέσα από την μέθοδο foundByAFM(int AFM), η οποία εκτελεί δυαδική αναζήτηση με κλειδί το ΑΦΜ. Αν τον βρούμε ενημερώνουμε το πεδίο savings αλλιώς εκτυπώνουμε ανάλογο μήνυμα. Σε κάθε περίπτωση λόγω της δυαδικής αναζήτησης η μέθοδος έχει Time Complexity -> O(logn), όπου n ο αριθμός κόμβων του δέντρου.

LargeDepositor searchByAFM(int AFM)

Αναζητούμε τον LargeDepositor με το δοσμένο ΑΦΜ μέσα από την μέθοδο foundByAFM(int AFM), η οποία εκτελεί δυαδική αναζήτηση με κλειδί το ΑΦΜ. Αν τον βρούμε τον επιστρέφουμε αλλιώς εκτυπώνουμε ανάλογο μήνυμα και επιστρέφουμε null. Σε κάθε περίπτωση λόγω της δυαδικής αναζήτησης η μέθοδος έχει Time Complexity -> O(logn), όπου n ο αριθμός κόμβων του δέντρου.

Void remove(int AFM)

Αναζητούμε τον LargeDepositor με το δοσμένο ΑΦΜ μέσα από την μέθοδο foundByAFM(int AFM). Αν δεν τον βρούμε, δεν γίνεται τίποτα. Αν όμως τον βρούμε, τότε θα πρέπει να πάρουμε μια τυχαία απόφαση για το ποιο από τα παιδιά του θα μπει στην θέση του. Για την διαδικασία της ένωσης χρησιμοποιούμε την μέθοδο TreeNode joinNode(TreeNode a, TreeNode b), ενώ η ταχύτητα της μεθόδου εξαρτάται από το ύψος του δέντρου και άρα αφού έχουμε Τυχαιοποιημένο ΔΔΑ ( άρα ισοζυγισμένο δέντρο) το ύψος θα είναι περίπου logn άρα Time Complexity -> O(logn)