ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
DEPARTMENT OF INFORMATICS

Δομές Δεδομένων

Διδάσκουσα: Κ. Παπακωνσταντινοπούλου

Χειμερινό Εξάμηνο 2023

Εργασία 3
Δέντρα Δυαδικής Αναζήτησης
Προθεσμία υποβολής: 3/2/2024, 23:59

Σκοπός της εργασίας αυτής είναι η εξοικείωση με τις μεθόδους επεξεργασίας Δέντρων Δυαδικής Αναζήτησης (Binary Search Trees) και η υλοποίηση πίνακα συμβόλων με τέτοιες δομές. Διαβάστε πολύ προσεκτικά την εκφώνηση και τα ζητούμενα της εργασίας.

Προκειμένου να περιοριστεί η φοροδιαφυγή στη χώρα μας, η κυβέρνηση σκέφτεται να εξετάσει τα οικονομικά στοιχεία των πολιτών που δεν δικαιολογούνται από τα επαγγέλματά τους. Αρχικά θα εστιάσει στη συστηματική καταχώρηση Ελλήνων μεγαλοκαταθετών σε χώρες του εξωτερικού (σε συνεργασία με τις αρχές της Ελβετίας και άλλων κρατών). Για να γίνει αυτό όμως, χρειάζεται να υλοποιηθεί μια αποδοτική δομή δεδομένων στην οποία θα καταχωρούνται όλοι οι μεγαλοκαταθέτη. Μια τέτοια δομή μπορεί να έχει τη μορφή ενός πίνακα συμβόλων.

Στόχος της εργασίας αυτής είναι να υλοποιήσετε έναν **πίνακα συμβόλων** με την μορφή δέντρου δυαδικής αναζήτησης. Συγκεκριμένα, θα πρέπει να υλοποιήσετε το παρακάτω interface (οι μέθοδοι περιγράφονται στο Μέρος Β).

```
public interface TaxEvasionInterface {
void insert(LargeDepositor item);
void load(String filename);
void updateSavings(int AFM, double savings);
LargeDepositor searchByAFM(int AFM);
List searchByLastName(String last_name);
void remove(int AFM);
double getMeanSavings();
void printTopLargeDepositors(int k);
void printByAFM();
}
```

Μέρος Α [10 μονάδες]. Οι κλάσεις LargeDepositor και TreeNode. Η υλοποίηση του interface θα γίνει χρησιμοποιώντας τυχαιοποιημένο δέντρο δυαδικής αναζήτησης. Πριν την υλοποίηση των μεθόδων όμως, θα πρέπει να ορίσετε 2 βασικές κλάσεις που αφορούν τους καταθέτες και τους κόμβους του δέντρου. Κάθε ιδιοκτήτης τραπεζικών λογαριασμών στην Ελβετία, που θεωρείται

ύποπτος για φοροδιαφυγή στην Ελλάδα, θα αντιστοιχεί σε ένα αντικείμενο της κλάσης `LargeDepositor` που φαίνεται παρακάτω. Η `LargeDepositor` θα πρέπει να περιέχει και μια μέθοδο `key()` που θα επιστρέφει το κλειδί με το οποίο θα φτιαχτεί εν τέλει το δέντρο, δηλαδή τον ΑΦΜ (Αριθμό Φορολογικού Μητρώου). Μπορείτε επίσης εδώ να υπερφορτώσετε κατάλληλα την μέθοδο `toString` για να σας χρησιμεύσει για την εκτύπωση αποτελεσμάτων, καθώς και να προσθέσετε όποιες άλλες βοηθητικές μεθόδους χρειαστείτε.

```
class LargeDepositor {
   private int AFM; // ΑΦΜ (ελληνικός)
   private String firstName; // Όνομα
   private String lastName; // Επώνυμο
   private double savings;  // Συνολικό ύψος γνωστών καταθέσεων σε άλλες
χώρες
   private double taxedIncome; //Συνολικό δηλωμένο εισόδημα στην Ελλάδα
                               //την τελευταία 3ετία
   int key() {return AFM;} //μέθοδος για πρόσβαση στο κλειδί
// + getters, setters για τα πεδία
// + ό,τι άλλο χρειαστεί να προσθέσετε
}
```

Η κλάση TreeNode. Κάθε κόμβος του τυχαιοποιημένου ΔΔΑ θα είναι ένα αντικείμενο τύπου TreeNode. Η κλάση `TreeNode` θα υλοποιηθεί εντός της κλάσης RandomizedBST του Μέρους Β. Κάθε αντικείμενο `TreeNode` πρέπει να περιέχει ένα αντικείμενο τύπου LargeDepositor, καθώς και τους δείκτες προς το αριστερό και δεξιό υποδέντρο. Τέλος, πρέπει να περιέχει και ένα πεδίο που θα δηλώνει πόσους κόμβους έχει το υποδέντρο που ξεκινά από αυτόν τον κόμβο, όπως συζητήσαμε στο αντίστοιχο μάθημα. Επομένως στην κλάση `TreeNode` πρέπει να υπάρχουν τουλάχιστον τα εξής πεδία (και ενδεχομένως ό,τι άλλο θέλετε εσείς να προσθέσετε):

```
private class TreeNode {
   LargeDepositor item;
   TreeNode left; // pointer to left subtree
   TreeNode right; // pointer to right subtree
   int N;  //number of nodes in the subtree rooted at this TreeNode
   ...
   ...
}
```

Η πρόσβαση στο κλειδί του κόμβου θα πρέπει να γίνεται μέσω της μεθόδου `key()` του `item`. Αν h είναι ένα αντικείμενο τύπου `TreeNode`, το κλειδί του κόμβου θα το παίρνετε από την κλήση `h.item.key()`.

Μέρος Β [65 μονάδες]. Υλοποίηση του interface. Ο πίνακας συμβόλων θα υλοποιηθεί χρησιμοποιώντας τις παραπάνω κλάσεις, και φτιάχνοντας ένα τυχαιοποιημένο ΔΔΑ. Η κλάση που υλοποιεί το interface θα ονομάζεται `RandomizedBST` και θα ακολουθεί το παρακάτω υπόδειγμα:

```
class RandomizedBST implements TaxEvasionInterface  {
private class TreeNode {
...
};
private TreeNode root; //ρίζα στο δέντρο
//Υλοποίηση μεθόδων από εδώ και κάτω
```

```
...
}
```

Ακολουθεί συνοπτική περιγραφή των απαιτούμενων μεθόδων:

- `void insert(LargeDepositor item)`: εισάγει στο δέντρο έναν νέο κόμβο που περιέχει τον μεγαλοκαταθέτη του αντικειμένου `item`. Εάν υπάρχει ήδη μεγαλοκαταθέτης με τον ίδιο ΑΦΜ στο δέντρο, θα πρέπει να τυπώνει μήνυμα λάθους και να τερματίζει, χωρίς να κάνει καμία εισαγωγή. Η μέθοδος θα πρέπει να δουλεύει όπως η εισαγωγή σε τυχαιοποιημένο δέντρο από τις διαφάνειες: *με πιθανότητα 1/(N+1) θα πρέπει να γίνεται εισαγωγή στη ρίζα (μέσω περιστροφών), διαφορετικά θα πρέπει να γίνεται αναδρομικά εισαγωγή στο κατάλληλο υποδέντρο*. Μην ξεχνάτε την **ενημέρωση** του πεδίου N σε κάθε κόμβο. Για την πιο δομημένη υλοποίηση της μεθόδου, απαιτείται να υλοποιήσετε μια βοηθητική ιδιωτική μέθοδο `insertAsRoot(LargeDepositor item, Node h)`, η οποία θα κάνει εισαγωγή στη ρίζα του αντικειμένου item στο υποδέντρο που ξεκινά από τον κόμβο h.
- `void load(String filename)`: διαβάζει ένα αρχείο εισόδου με όνομα `filename` και ενημερώνει το δέντρο κάνοντας εισαγωγές με χρήση της `insert` (1 εισαγωγή για κάθε γραμμή του αρχείου εισόδου). Το αρχείο εισόδου θα είναι στην εξής μορφή (δεν χρειάζεται να σας απασχολήσει τι πρέπει να γίνει αν δεν είναι σε αυτή τη μορφή):
  ```
  AFM1 firstName lastName savings taxedIncome
  AFM2 firstName lastName savings taxedIncome
  AFM3 firstName lastName savings taxedIncome
  .
  .
  .
  ```
- `void updateSavings(int AFM, double savings)`: Ενημερώνει τις αποταμιεύσεις ενός υπάρχοντος υπόπτου (δηλαδή που υπάρχει ήδη στο δέντρο) στην τιμή `savings`. Αν δεν υπάρχει ο ΑΦΜ στο δέντρο, τυπώνει αντίστοιχο μήνυμα και τερματίζει.
- `LargeDepositor searchByAFM(int AFM)`: Ψάχνει στο δέντρο για την ύπαρξη υπόπτου με τον συγκεκριμένο ΑΦΜ. Αν υπάρχει, τυπώνει όλα τα στοιχεία του (ον/μο, καταθέσεις, δηλωμένο εισόδημα). Αν δεν υπάρχει, τυπώνει αντίστοιχο μήνυμα. Η μέθοδος search θα δουλεύει όπως και η αντίστοιχη μέθοδος που κάναμε στο μάθημα. Υπενθυμίζεται ότι ο ΑΦΜ είναι μοναδικός για κάθε χρήστη, επομένως δεν μπορεί να υπάρχει παραπάνω από ένας χρήστης με ένα συγκεκριμένο ΑΦΜ.
- `List searchByLastName(String last_name)`: Ψάχνει στο δέντρο για την ύπαρξη μεγαλοκαταθετών με το συγκεκριμένο επίθετο. Επειδή ενδέχεται να υπάρχουν περισσότερα του ενός άτομα με το ίδιο επίθετο, η μέθοδος θα πρέπει να επιστρέφει μια λίστα μονής σύνδεσης με αντικείμενα τύπου LargeDepositor. Αν η λίστα περιέχει το πολύ 5 υπόπτους, η μέθοδος θα πρέπει να τυπώνει τα στοιχεία τους πριν τερματίσει. Αν δεν υπάρχει κάποιος με αυτό το επίθετο, απλά επιστρέφει null. **Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμη δομή λίστας της Java.** Μπορείτε να φτιάξετε την δική σας ή να χρησιμοποιήσετε κώδικα από τα εργαστήρια για την λίστα ή να χρησιμοποιήσετε κώδικά σας από τις Εργασίες 1 και 2.
- `void remove(int AFM)`: Η μέθοδος αυτή αφαιρεί το αντικείμενο με τον συγκεκριμένο ΑΦΜ (αν υπάρχει στο δέντρο). Θα πρέπει να χρησιμοποιήσετε την μέθοδο αφαίρεσης που είδαμε για τυχαιοποιημένα δέντρα (η οποία είναι απλή παραλλαγή του τρόπου αφαίρεσης που είδαμε σε ΔΔΑ). Προσοχή στην ενημέρωση του πεδίου N, όπου αυτή απαιτείται!
- `double getMeanSavings()`: Η μέθοδος αυτή υπολογίζει το μέσο ποσό καταθέσεων με βάση τους καταθέτες που είναι αποθηκευμένοι στο δέντρο. Μπορείτε να την υλοποιήσετε με κάποια μέθοδο διάσχισης.
- `void printTopLargeDepositors(int k)`: Η μέθοδος αυτή βρίσκει και τυπώνει τα στοιχεία για τους k **πιο ύποπτους για φοροδιαφυγή μεγαλοκαταθέτες** (σε αύξουσα ή φθίνουσα σειρά ως προς το πόσο ύποπτοι είναι). Ένας βολικός τρόπος εδώ είναι να ανατίθεται ένα σκορ για το πόσο ύποπτος για φοροδιαφυγή είναι κάποιος μεγαλοκαταθέτης. Ως μετρική για την υποψία φοροδιαφυγής, το Υπουργείο Οικονομικών θα χρησιμοποιήσει τα εξής κριτήρια: Αν για κάποιον

καταθέτη ισχύει ότι `taxedIncome < 8000`, τότε θεωρείται ότι έχει το υψηλότερο δυνατό σκορ υποψίας, π.χ. Double.MAX_VALUE. Σε διαφορετική περίπτωση, θεωρούμε την ποσότητα `savings − taxedIncome`, ως μέτρο σύγκρισης για να καθορίσουμε αν ένας καταθέτης είναι περισσότερο ύποπτος από κάποιον άλλο. Όσο μεγαλύτερη δηλαδή είναι η απόκλιση μεταξύ των καταθέσεων στο εξωτερικό και του δηλωθέντος εισοδήματος στην Ελλάδα, τόσο περισσότερο ύποπτος είναι κάποιος. Δεν είναι υποχρεωτικό να αναθέσετε ένα σκορ σε κάθε καταθέτη, αρκεί να μπορείτε να συγκρίνετε 2 υπόπτους με βάση τα παραπάνω και να μπορείτε να αποφασίζετε ποιος από τους 2 είναι περισσότερο ύποπτος. Για την υλοποίηση μπορείτε να χρησιμοποιήσετε ιδέες από την Εργασία 2 (και μπορείτε να έχετε την κλάση LargeDepositor να υλοποιεί το interface Comparable μέσω κατάλληλης υλοποίησης της compareTo) ή μπορεί η μέθοδος σας να φτιάχνει ένα νέο δέντρο με κατάλληλο κλειδί και να χρησιμοποιήσετε διασχίσεις. Σε περίπτωση ισοβαθμιών επιλέξτε αυθαίρετα ποιους θα τυπώσετε. Π.χ. αν υπάρχουν παραπάνω από k καταθέτες με `taxedIncome < 8000`, δεν έχει σημασία ποιους θα επιλέξετε να τυπώσετε τα στοιχεία τους. Για την υλοποίηση, επιτρέπεται να χρησιμοποιήσετε έξτρα μνήμη O(k) (μπορείτε δηλαδή να αποθηκεύσετε μέχρι k αντικείμενα σε κάποιον πίνακα ή ουρά προτεραιότητας ή όποια άλλη δομή χρησιμοποιήσετε).

- `void printByAFM()`: Τυπώνει τα στοιχεία όλων των κόμβων του δέντρου, ταξινομημένα σε αύξουσα σειρά ως προς τον ΑΦΜ. Χρησιμοποιήστε κατάλληλη διάσχιση του δέντρου για την υλοποίηση αυτής της μεθόδου.

Απαιτήσεις πολυπλοκότητας: Εκτός από την `load` και την `printTopLargeDepositors`, για τις οποίες δεν υπάρχει κάποια απαίτηση χρόνου, όλες οι άλλες μέθοδοι θα πρέπει στην χειρότερη περίπτωση να τρέχουν σε χρόνο O(N), σε δέντρα με N αντικείμενα.

Για την υλοποίηση των μεθόδων ισχύει ότι και στις προηγούμενες εργασίες, δηλαδή, μπορείτε να χρησιμοποιήσετε δικές σας υλοποιήσεις ή έτοιμα κομμάτια από το εργαστήριο, **αλλά όχι έτοιμες δομές της Java.**

Μέρος Γ [15 μοναδες]. Μενού διαχείρισης. Στην μέθοδο main της κλάσης `RandomizedBST`, κατασκευάστε ένα απλό μενού διαχείρισης (δεν χρειάζεται γραφικό περιβάλλον) μέσα από το οποίο θα μπορεί κάποιος να εκτελεί όλες τις λειτουργίες της δομής σας. Το πρόγραμμα θα πρέπει να αλληλεπιδρά με το χρήστη, έτσι ώστε ο χρήστης να μπορεί να εισάγει και να διαγράφει υπόπτους, να αναζητά καταθέτες καθώς και να τρέχει τις άλλες μεθόδους που περιγράφονται στο Μέρος Β (το πρόγραμμα αυτό είναι ούτως ή άλλως χρήσιμο να το φτιάξετε για debugging). Δεν υπάρχει περιορισμός στο format για το πώς να εμφανίζονται οι επιλογές στον χρήστη. Είστε ελεύθεροι να το υλοποιήσετε όπως σας φαίνεται εσάς πιο βολικό και πιο user-friendly. Π.χ. θα μπορούσατε να έχετε ένα αρχικό μενού στο οποίο θα εμφανίζετε ως επιλογές τις λειτουργίες που περιγράφονται στο Μέρος Β. Κατόπιν, αν ο χρήστης επιλέξει π.χ. να κάνει αναζήτηση κάποιου καταθέτη θα μπορούσε το πρόγραμμά σας να ζητάει το ΑΦΜ ή το επίθετο και μετά να εκτελεί την αντίστοιχη μέθοδο. Ή αν ο χρήστης ζητήσει εισαγωγή, μετά το πρόγραμμα θα ζητήσει από τον χρήστη να πληκτρολογήσει το όνομα, το επίθετο, το ΑΦΜ, κτλ.

Μέρος Δ - Αναφορά παράδοσης [10 μονάδες]. Ετοιμάστε μία σύντομη αναφορά σε pdf αρχείο (μην παραδώσετε Word ή txt αρχεία!) με όνομα project3-report.pdf, στην οποία θα αναφερθείτε στα εξής:

a. Εξηγήστε συνοπτικά πώς υλοποιήσατε κάθε μέθοδο του Μέρους Β (αρκούν 5-10 γραμμές για κάθε μέθοδο).

b. Σχολιάστε την πολυπλοκότητα των μεθόδων αυτών, σαν συνάρτηση του πλήθους των κόμβων του δέντρου.

Το συνολικό μέγεθος της αναφοράς θα πρέπει να είναι τουλάχιστον 2 σελίδες. Μην ξεχνάτε τα ονοματεπώνυμα και τους ΑΜ σας στην αναφορά.

Οδηγίες Παράδοσης

Η εργασία σας θα πρέπει να μην έχει συντακτικά λάθη και να μπορεί να μεταγλωττίζεται. Εργασίες που δεν μεταγλωττίζονται χάνουν το 50% της συνολικής αξίας.

Η εργασία θα αποτελείται από:

1. Τον πηγαίο κώδικα (source code). Τοποθετήστε σε ένα φάκελο με όνομα **src** τα αρχεία java που έχετε φτιάξει. Χρησιμοποιήστε τα ονόματα των κλάσεων όπως δίνονται. Επιπλέον, φροντίστε να συμπεριλάβετε όποια άλλα αρχεία πηγαίου κώδικα φτιάξατε και απαιτούνται για να μεταγλωττίζεται η εργασία σας. Φροντίστε επίσης να προσθέσετε επεξηγηματικά σχόλια όπου κρίνετε απαραίτητο στον κώδικά σας.

2. Την αναφορά παράδοσης

Όλα τα παραπάνω αρχεία θα πρέπει να μπουν σε ένα αρχείο zip. Το όνομα που θα δώσετε στο αρχείο αυτό θα είναι ο αριθμός μητρώου σας πχ. 3130056_3130066.zip ή 3130056.zip (αν δεν είστε σε ομάδα). Στη συνέχεια, θα υποβάλετε το zip αρχείο σας στην περιοχή του μαθήματος «Εργασίες» στο e-class. Δεν χρειάζεται υποβολή και από τους 2 φοιτητές μιας ομάδας.

Η προθεσμία παράδοσης της εργασίας είναι Σάββατο, 3 Φεβρουαρίου 2024 και ώρα 23:59.