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 2024

Εργασία 3
Δέντρα Δυαδικής Αναζήτησης
Προθεσμία υποβολής: Κυριακή 16/2/2025, 23:59

Σκοπός της εργασίας αυτής είναι η εξοικείωση με τις δομές που χρησιμοποιούνται για την υλοποίηση πίνακα συμβόλων, όπως τα δέντρα δυαδικής αναζήτησης (σχετικές ενότητες διαφανειών: 12-14). Το ζητούμενο της εργασίας είναι να κατασκευάσετε ένα Μετρητή Λέξεων. Η βασική λειτουργία του μετρητή είναι ότι θα μπορεί να "φορτώνει" ένα αρχείο αγγλικού κειμένου και θα μετράει πόσες φορές εμφανίζεται η κάθε λέξη. Για παράδειγμα, ας δούμε το παρακάτω κείμενο.

- Hello, how are you?
- Very well, thank you. I study for the exams. How about you?
- Fine, thank you. How many exams will you have?
- Too many…

Η λέξη «you» εμφανίζεται 5 φορές, η λέξη «how» 3 φορές, οι λέξεις «thank», «many», «exams» από 2 φορές, ενώ από 1 φορά εμφανίζονται οι λέξεις hello, are, very, well, I, study, about, fine, will, have. Ο Μετρητής Λέξεων θα πρέπει να μην λαμβάνει υπόψη τα σημεία στίξης, π.χ., τα , . ? ; ! - :, καθώς και παρενθέσεις, αγκύλες, σύμβολα πράξεων, ή οποιονδήποτε άλλο χαρακτήρα που δεν είναι γράμμα του αγγλικού αλφαβήτου. Από το κείμενο παραπάνω, όταν διαβάσει π.χ. το string "well," θα πρέπει να βγάλει το κόμμα για να μείνει μόνο η λέξη well. Αν υπήρχε φράση σε παρενθέσεις, π.χ. "(Maria asked me)", θα πρέπει να θεωρήσει ότι υπάρχουν μόνο οι λέξεις Maria, asked και me. Επιτρέπεται μόνο η μονή απόστροφος μέσα σε λέξη, π.χ. το don't θα το θεωρούμε σαν 1 λέξη. Επίσης θα πρέπει να αγνοούνται πλήρως όλα τα strings που περιέχουν αριθμούς π.χ. 17:25 ή 1980's. Τέλος θα πρέπει να δίνεται η δυνατότητα στο χρήστη της βιβλιοθήκης να ορίζει ειδικές λέξεις (stop words) που θέλει να αγνοούνται πλήρως, π.χ. σε σχετικές εφαρμογές αγνοούμε συνήθως τα άρθρα, όπως τα "a", "an" και "the". Το πρόγραμμα θα είναι case insensitive. Για παράδειγμα, οι λέξεις Hello και hello θεωρούνται ίδιες.

Μέρος Α [10 μονάδες]. Αρχικά θα πρέπει πρώτα να ορίσετε τις εξής 2 κλάσεις:

Κλάση WordFrequency. Για κάθε διαφορετική λέξη που διαβάζετε, θα πρέπει να δημιουργείτε ένα αντικείμενο με κλειδί τη λέξη αυτή. Συγκεκριμένα, σε κάθε λέξη θα αντιστοιχεί ένα αντικείμενο τύπου `WordFrequency`. Η κλάση αυτή περιέχει (τουλάχιστον) 2 πεδία: την ίδια τη λέξη (`private String`) και το πλήθος εμφανίσεων (`private int`). Μπορείτε εδώ να υπερφορτώσετε κατάλληλα την μέθοδο `toString` για να σας χρησιμεύσει για εκτύπωση αποτελεσμάτων. Η `WordFrequency` πρέπει να περιέχει και μια μέθοδο `key()` που θα επιστρέφει το κλειδί, δηλαδή τη λέξη.

Κλάση TreeNode. Η κλάση αυτή θα είναι ιδιωτική μέσα στον πίνακα συμβόλων (δείτε το Μέρος Β) και αντικείμενα αυτής της κλάσης αντιστοιχούν στους κόμβους του δέντρου δυαδικής αναζήτησης που θα χρησιμοποιήσετε. Κάθε κόμβος του δέντρου είναι ένα αντικείμενο `TreeNode`, και πρέπει να περιέχει ένα αντικείμενο τύπου `WordFrequency`, όπως περιγράφεται παραπάνω. Επιπλέον, πρέπει να περιέχει τους δείκτες προς το αριστερό και το δεξιό υποδέντρο, και ένα πεδίο που θα δηλώνει πόσους κόμβους έχει το υποδέντρο που ξεκινά από αυτόν τον κόμβο. Επομένως στην κλάση `TreeNode` πρέπει να υπάρχουν τουλάχιστον τα εξής πεδία (και ενδεχομένως ό,τι άλλο θέλετε εσείς να προσθέσετε):

```
private class TreeNode {
   WordFrequency item
   TreeNode left   // pointer to left subtree
   TreeNode right  // pointer to right subtree
   int subtreeSize //number of nodes in subtree starting at this node
   ...
}
```

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()`, όπως και στις διαφάνειες του μαθήματος.

Μέρος Β [80 μονάδες]. Η κλάση του πίνακα συμβόλων/ΔΔΑ. Η δομή σας για την υλοποίηση του πίνακα συμβόλων θα είναι ένα Δέντρο Δυαδικής Αναζήτησης (Binary Search Tree). Η κλάση που θα ορίσετε θα ονομάζεται `BST.java` και θα ακολουθεί το παρακάτω υπόδειγμα:

```
public class BST implements WordCounter {
   private class TreeNode {
      ...
   };
   private TreeNode head; //root of the tree
   private List stopWords; // list of stopwords
   ...
}
```

Το interface `WordCounter` περιέχει τις εξής 13 μεθόδους (ακολουθούν επεξηγήσεις):

```
public interface WordCounter {
void insert(String w);
WordFrequency search(String w);
void remove(String w)
void load(String filename);
int getNumTotalWords();
int getNumDistinctWords();
```

```
int getFrequency(String w);
WordFrequency getMaxFrequency();
double getMeanFrequency();
void addStopWord(String w);
void removeStopWord(String w);
void printTreeByWord(PrintStream stream);
void printTreeByFrequency(PrintStream stream);
}
```

Συνοπτική περιγραφή των απαιτούμενων μεθόδων:

- `void insert(String w)`: ψάχνει να βρει αν υπάρχει ήδη στο δέντρο κόμβος με κλειδί w. Αν ναι, τότε αυξάνει τη συχνότητά του κατά 1. Αν όχι, τότε εισάγει ένα νέο κόμβο στο δέντρο (χρησιμοποιώντας την απλή εισαγωγή ως φύλλο), με αυτό το κλειδί και με συχνότητα ίση με 1.
- `WordFrequency search(String w)`: Αναζητά στο δέντρο αν υπάρχει η λέξη w, και επιστρέφει null αν δεν υπάρχει. Η μέθοδος search θα δουλεύει αρχικά όπως και η αντίστοιχη μέθοδος που έχουμε δει στο μάθημα με την εξής όμως σημαντική διαφοροποίηση: όταν βρίσκει τη λέξη w μέσα στο δέντρο, αν η συχνότητα της w είναι μεγαλύτερη της μέσης συχνότητας (που υπολογίζεται από την `getMeanFrequency()`), τότε με χρήση περιστροφών θα φέρνει τη λέξη αυτή στη ρίζα του δέντρου. Ένας τρόπος να γίνει αυτό (αλλά όχι ο μοναδικός), είναι να καλείτε πρώτα τη `remove` για να βγάλει τον κόμβο αυτό από το ΔΔΑ και στη συνέχεια να κάνετε εισαγωγή στη ρίζα για τον συγκεκριμένο κόμβο. Το σκεπτικό είναι ότι λέξεις με μεγάλη συχνότητα είναι λέξεις για τις οποίες μπορεί να γίνουν πολλές αναζητήσεις στο σύντομο μέλλον και επομένως θέλουμε να τις έχουμε όσο το δυνατόν πιο ψηλά στο δέντρο.
- `void remove(String w)`: αφαιρεί τον κόμβο με κλειδί w (αν υπάρχει τέτοιος κόμβος). Μπορείτε να χρησιμοποιήσετε τη μέθοδο αφαίρεσης κόμβου που έχουμε δει στο μάθημα.
- `void load(String filename)`: ξεκινώντας από το τρέχον ΔΔΑ (που μπορεί να είναι και κενό), διαβάζει το αρχείο με όνομα `filename`, με κείμενο Αγγλικής γλώσσας, και ενημερώνει το δέντρο με τις λέξεις που διάβασε ώστε να φτιαχτεί το τελικό ΔΔΑ. Θα πρέπει να τηρούνται οι προϋποθέσεις που αναφέρθηκαν στην πρώτη σελίδα της εκφώνησης σχετικά με τα σημεία στίξης, την ύπαρξη αριθμών, την αγνόηση των λέξεων που έχουν δοθεί ως stop words, κτλ. Δεν απαιτείται να ελέγξετε αν το κείμενο είναι όντως Αγγλικά, ούτε και αν οι λέξεις είναι έγκυρες (π.χ. αν κατά λάθος υπάρχει η λέξη strutures αντί για το σωστό structures, θα την θεωρήσει σαν μια κανονική νέα λέξη).
- `int getNumTotalWords()`: επιστρέφει το συνολικό πλήθος λέξεων του κειμένου που έχει φορτωθεί στο ΔΔΑ, λαμβάνοντας υπόψη τη συχνότητα κάθε λέξης (έχοντας δηλαδή αγνοήσει ήδη stopwords, αριθμούς, κτλ). Μπορεί να γίνει με μια απλή διάσχιση δέντρου (όποια διάσχιση θέλετε από αυτές που είδαμε στο μάθημα).
- `int getNumDistinctWords()`: επιστρέφει το πλήθος διαφορετικών λέξεων του δέντρου. Η συνάρτηση αυτή **πρέπει να εκτελείται σε O(1).**
- `int getFrequency(String w)`: επιστρέφει το πλήθος εμφανίσεων της λέξης w (αν η λέξη δεν υπάρχει στο δέντρο, επιστρέφει 0).
- `WordFrequency getMaxFrequency()`: επιστρέφει ένα αντικείμενο `WordFrequency` που περιέχει τη λέξη με τις περισσότερες εμφανίσεις (δεν χρειάζεται να κάνετε ταξινόμηση για να λύσετε το πρόβλημα αυτό). Σε ισοβαθμίες μπορείτε να επιλέξετε αυθαίρετα ποια λέξη θα επιστρέψετε.
- `double getMeanFrequency()`: υπολογίζει και επιστρέφει την μέση συχνότητα. Ο μέσος όρος παράγεται από τις συχνότητες όλων των διαφορετικών λέξεων μέσα στο κείμενο.
- `void addStopWord(String w)`: προσθέτει στη λίστα `stopWords` τη λέξη `w`. Η συνάρτηση αυτή **πρέπει να εκτελείται σε O(1).**
- `void removeStopWord(String w)`: αφαιρεί τη λέξη `w` από τη λίστα `stopWords`.
- `printTreeByWord(PrintStream stream)`: εκτυπώνει τις λέξεις του δέντρου και το πλήθος εμφανίσεων κάθε λέξης, με αλφαβητική σειρά. Πρέπει να υλοποιηθεί χρησιμοποιώντας κάποια μέθοδο διάσχισης του δέντρου από αυτές που είδαμε στο μάθημα.

- `printTreeByFrequency(PrintStream stream)`: εκτυπώνει τις λέξεις και το πλήθος εμφανίσεών τους ταξινομημένες σε αύξουσα σειρά ως προς το πλήθος εμφανίσεων.

Σχόλια και πρόσθετες οδηγίες υλοποίησης:

- Μπορείτε να χρησιμοποιήσετε τον κώδικα που υπάρχει στις διαφάνειες του μαθήματος και αυτόν που γράψατε στα εργαστήρια για ΔΔΑ.
- Χρησιμοποιήστε τις μεθόδους της βασικής βιβλιοθήκης της Java για την μετατροπή κεφαλαίων σε μικρούς χαρακτήρες, την αφαίρεση των σημείων στίξης και όποια άλλη επεξεργασία κάνετε για τις λέξεις. Όπως και στις Εργασίες 1 και 2, υπάρχουν διάφορες μέθοδοι που μπορείτε να χρησιμοποιήσετε για να διαβάσετε και να επεξεργαστείτε κείμενο.
- Δεν θα θεωρήσετε κάποια stop word ως δεδομένη. Η λίστα με τα stopwords θα δημιουργείται μέσα από κλήσεις της `addStopWord`. Για τη λίστα stopwords υλοποιήστε όποιο τύπο λίστας θέλετε (μονής ή διπλής σύνδεσης), χωρίς την χρήση έτοιμων υλοποιήσεων της Java.
- Το κλειδί των αντικειμένων που αποθηκεύουμε στο δέντρο είναι τύπου String. Επομένως σύγκριση κλειδιών εδώ σημαίνει σύγκριση μεταξύ strings. Επίσης, κάθε κλειδί θα εμφανίζεται το πολύ σε έναν κόμβο του δέντρου.
- Αρκετές μέθοδοι χρειάζονται διάσχιση του δέντρου. Θα πρέπει λοιπόν να υλοποιήσετε μέσα στον πίνακα συμβόλων και κάποια ή κάποιες μεθόδους διάσχισης.
- Προσέξτε ώστε να γίνεται σωστή ενημέρωση του πεδίου `subtreeSize`, της κλάσης `TreeNode`.
- Η υλοποίηση της `printTreeByFrequency` μπορεί να γίνει είτε με συνδυασμό κάποιας διάσχισης και μετέπειτα κάποιας μεθόδου ταξινόμησης (χρησιμοποιώντας `Quicksort`, `Mergersort` ή `Heapsort`), είτε με άλλους τρόπους. Εναλλακτικά, π.χ. μπορείτε (χωρίς να είναι απαραίτητο, με τον πρώτο τρόπο λύνεται πιο απλά), όταν καλείται η `printTreeByFrequency` να δημιουργείτε επί τόπου ένα προσωρινό ΔΔΑ με τον κατάλληλο τύπο κλειδιού και έναν κατάλληλο `Comparator` (σκεφτείτε τι πρέπει να συγκρίνετε) για να διατρέξετε το δέντρο με βάση το πλήθος εμφανίσεων της κάθε λέξης.
- **Πολυπλοκότητα μεθόδων:** Δεν υπάρχει αυστηρή απαίτηση να πετύχετε την βέλτιστη πολυπλοκότητα όλων των μεθόδων, παρά μόνο για τις μεθόδους που αναφέρεται κάτι ρητά στην επεξήγηση τους στην προηγούμενη σελίδα. Μπορείτε να έχετε ως γνώμονα ό,τι έχουμε πει και στο μάθημα. Αν όμως κάνετε κάτι υπερβολικά χρονοβόρο, π.χ. αν υλοποιήσετε την `printAlphabetically` σε χρόνο $O(N^2)$, τότε δεν θα πάρετε όλες τις μονάδες που αντιστοιχούν στη μέθοδο αυτή. Γενικά θα πρέπει να αποφύγετε το $O(N^2)$ για όλες τις μεθόδους εκτός της `load`.
- **Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις δομών δεδομένων για λίστες, ουρές, δέντρα, κτλ, από την βιβλιοθήκη της Java** (π.χ. Vector, ArrayList κτλ).

Προαιρετικά: Όποιος θέλει, μπορεί αντί για ΔΔΑ να χρησιμοποιήσει κάποια από τις δομές του Κεφαλαίου 13, που έχουν ως αποτέλεσμα το δέντρο να είναι πιο ισοζυγισμένο, π.χ. τυχαιοποιημένα ΔΔΑ ή δέντρα κόκκινου-μαύρου. Επίσης, μπορείτε αν θέλετε να έχετε και μια μέθοδο `main` η οποία να κάνει κάποιο ενδεικτικό τρέξιμο, π.χ. να προσθέτει κάποια stopwords, μετά να καλεί τη μέθοδο `load`, και μετά να τυπώνει τη μέση συχνότητα, ή ακόμα και να εμφανίζει κάποιο μενού διαχείρισης (αυτό είναι καλό να το κάνετε ούτως ή άλλως για να ελέγξετε τις μεθόδους σας).

Μέρος Γ - 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 το πολύ 4-5 γραμμές για κάθε μέθοδο).
b. Σχολιάστε την πολυπλοκότητα κάθε μεθόδου.

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

Οδηγίες Υποβολής

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 τους. Η διόρθωση των εργασιών από τους βοηθούς του μαθήματος αφορά την αξιολόγηση μεταγλωττίσιμων υλοποιήσεων και όχι την εύρεση λαθών μεταγλώττισης.**

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. 3230056_3230066.zip. Στη συνέχεια, θα υποβάλετε το zip αρχείο σας στην περιοχή του μαθήματος «Εργασίες» στο e-class. Δεν χρειάζεται υποβολή και από τα 2 μέλη μιας ομάδας. Φροντίστε να υπάρχουν οι αριθμοί μητρώου και των δύο μελών της ομάδας στο όνομα του zip αρχείου, καθώς επίσης να υπάρχουν τα ονοματεπώνυμα και οι αριθμοί μητρώου και των δύο μελών της ομάδας στην αναφορά. Αν δεν υπάρχει το όνομά σας, δε μπορείτε να διεκδικήσετε βαθμό στην εργασία αυτή.