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

**5η Εργαστηριακή Άσκηση**
**Ουρές Προτεραιότητας, Δυαδικά Δένρα και Κώδικας Huffman**
Παράδοση έως Πέμπτη 3/12, 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 lab5results.txt) σε ένα zip αρχείο το οποίο να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.**

Συμπληρώστε το πρόγραμμα Huffman.java ώστε να υλοποιεί την κωδικοποίηση Huffman ενός αρχείου κειμένου. Ο αλγόριθμος Huffman χρησιμοποιεί μια ουρά προτεραιότητας ελάχιστου, στην οποία αποθηκεύει τους χαρακτήρες $\sigma_1, \sigma_2, \dots, \sigma_n$ που πρέπει να κωδικοποιηθούν, μαζί με τις συχνότητες εμφάνισής τους $f_1, f_2, \dots, f_n$. Υπολογίζει μια κωδικοποίηση των $\sigma_1, \sigma_2, \dots, \sigma_n$ με δυαδικές συμβολοσειρές, έτσι ώστε οι χαρακτήρες με τη μεγαλύτερη συχνότητα εμφάνισης να κωδικοποιούνται με λιγότερα δυαδικά ψηφία.

Η κωδικοποίηση μπορεί να αναπαρασταθεί με ένα δυαδικό δένδρο, όπου οι αρχικοί χαρακτήρες αποθηκεύονται στα φύλλα του, και οι ενδιάμεσοι κόμβοι αποθηκεύουν βοηθητικούς (κενούς) χαρακτήρες με συχνότητα ίση με το άθροισμα των συχνοτήτων των παιδιών τους.

## Ουρά προτεραιότητας ελάχιστου

Συμπληρώστε το αρχείο MinPQ.java έτσι ώστε να υλοποιεί μια ουρά προτεραιότητας ελάχιστου. Η ουρά προτεραιότητας θα πρέπει να αποθηκεύει αντικείμενα ενός γενικού τύπου Item, όπου κάθε αντικείμενο έχει ένα κλειδί γενικού τύπου Key.

Για το σκοπό αυτό χρησιμοποιούμε ένα πίνακα $pqK[]$ τύπου Key και ένα πίνακα $pqI[]$ τύπου Item:

- Ο πίνακας $pqK[]$ αποθηκεύει τα κλειδιά των αντικειμένων και είναι διατεταγμένος σε δυαδικό σωρό ελάχιστου. Δηλαδή, για κάθε θέση $2 \le j \le n$ ισχύει ότι το κλειδί $pqK[j]$ είναι μεγαλύτερο από το κλειδί του γονέα του $pqK[j/2]$ (δηλαδή $pqK[j].compareTo(pqK[j/2]) > 0$).
- Ο πίνακας $pqI[]$ αποθηκεύει το αντικείμενο που αντιστοιχεί σε κάθε κλειδί. Δηλαδή, για κάθε θέση $1 \le j \le n$, το αντικείμενο $pqI[j]$ έχει κλειδί $pqK[j]$.

Προσέξτε ότι στους πίνακες $pqK[]$ και $pqI[]$ δε χρησιμοποιούμε τη θέση 0 για να απλοποιήσουμε τον τρόπο υπολογισμού του γονέα και των παιδιών κάθε θέσης. (Ο γονέας της θέσης $j$ είναι ο $j/2$ και τα παιδιά της τα $2j$ και $2j+1$.)

Το αρχείο MinPQ.java περιλαμβάνει έτοιμες τις ακόλουθες μεθόδους:

| | |
|---|---|
| MinPQ(int maxN) | δημιουργεί μια ουρά προτεραιότητας για maxN αντικείμενα |
| int size() | επιστρέφει το πλήθος των αντικειμένων που βρίσκονται στην ουρά προτεραιότητας |
| Item minItem() | επιστρέφει το αντικείμενο της ουράς προτεραιότητας με το ελάχιστο κλειδί |
| Key minKey() | επιστρέφει το ελάχιστο κλειδί που βρίσκεται στην ουρά προτεραιότητας |

***Συμπληρωματικές μέθοδοι***

Συμπληρώστε τις παρακάτω μεθόδους στο πρόγραμμα IndexMinPQ. java.

| | |
|---|---|
| void insert(Item $v$, Key $k$) | εισάγει το αντικείμενο $v$ με κλειδί $k$ |
| void delMin() | διαγράφει το αντικείμενο με το ελάχιστο κλειδί στην ουρά προτεραιότητας (μαζί με το κλειδί του) |

Για την υλοποίηση των παραπάνω μεθόδων θα χρειαστεί να υλοποιήσετε βοηθητικές μεθόδους, όπως οι fixUp και fixDown, με κατάλληλες τροποποιήσεις έτσι ώστε να ενημερώνονται σωστά οι πίνακες $pqK[]$ και $pqI[]$.

***Επαλήθευση αποτελεσμάτων***

Η main() μέθοδος του MinPQ.java εισάγει στην ουρά προτεραιότητας αντικείμενα που είναι οι χαρακτήρες του λατινικού αλφάβητου ($'a' - 'z'$) και αναθέτει σε κάθε αντικείμενο ένα τυχαίο κλειδί τύπου int. Στη συνέχεια διαγράφει διαδοχικά το αντικείμενο με το ελάχιστο κλειδί μέχρι να αδειάσει η ουρά προτεραιότητας. Εκτελέστε το πρόγραμμα MinPQ. java και αποθηκεύστε τα αποτελέσματα της εκτέλεσης στο αρχείο lab5results. txt. Για την εκτέλεση του προγράμματος γράψτε

java MinPQ

Τα αντικείμενα που τυπώνει το πρόγραμμα καθώς εκτελεί την ακολουθία διαγραφών ελάχιστου θα πρέπει να εμφανίζονται **σε αύξουσα σειρά** ως προς τις τιμές των κλειδιών τους.

## Κώδικας Huffman

Αφού έχετε υλοποιήσει την MinPQ.java, συμπληρώστε τη μέθοδο $encode()$ στο αρχείο Huffman.java, ώστε να υλοποιεί την κωδικοποίηση Huffman ενός αρχείου κειμένου. Το πρόγραμμα διαβάζει από την είσοδο ένα αρχείο κειμένου με πεζούς λατινικούς χαρακτήρες και υπολογίζει το πλήθος των εμφανίσεων κάθε γράμματος. Συγκεκριμένα, οι λατινικοί χαρακτήρες αποθηκεύονται σε ένα πίνακα $char[]\ C$, έτσι ώστε $C[1] = 'a', C[2] = 'b', \dots, C[26] = 'z'$, ενώ το πλήθος των εμφανίσεων κάθε γράμματος αποθηκεύεται στον πίνακα $int\ F[]$ στην αντίστοιχη θέση με τον πίνακα $C$, δηλαδή το γράμμα $C[i]$ εμφανίζεται $F[i]$ φορές στο αρχείο εισόδου.

Υλοποιήστε τη μέθοδο $encode()$, κάνοντας χρήση της ουράς προτεραιότητας MinPQ, έτσι ώστε ο δυαδικός κώδικας του κάθε γράμματος να αποθηκευτεί στον πίνακα $String[]\ code$. Η μέθοδος $encode()$ θα πρέπει να κατασκευάζει το δυαδικό δένδρο του κώδικα Huffman, όπου κάθε γράμμα αποθηκεύεται σε ένα κόμβο δυαδικού δένδρου (τύπου Node). Για το σκοπό αυτό, χρησιμοποιούμε την ουρά προτεραιότητας ελάχιστου

MinPQ<TreeNode,Integer> PQ

όπου τα αντικείμενα είναι κόμβοι δυαδικού δένδρου (Node $x$) με κλειδιά το πλήθος εμφανίσεων των συμβόλων αποθηκεύουν οι κόμβοι (char $x.c$).

Μετά την ολοκλήρωση της κατασκευής του δένδρου του κώδικα Huffman, μπορούμε να λάβουμε το δυαδικό κώδικα που αντιστοιχεί σε κάθε γράμμα με μια προδιατεταγμένη διάσχιση του δένδρου.

Έστω $code$ το δυαδικό string που έχουμε σχηματίσει μέχρι τον κόμβο $x$:

- Όταν επισκεπτόμαστε το αριστερό παιδί ($x.left$) προσθέτουμε "0" στο τέλος του $code$ ($code + "0"$).
- Όταν επισκεπτόμαστε το δεξί παιδί ($x.right$) τοποθετούμε "1" στο τέλος του $code$ ($code + "1"$).

### *Εκτέλεση Προγράμματος*

Η main() μέθοδος του προγράμματος Huffman.java χρησιμοποιεί το πρόγραμμα In.java για να διαβάσει από ένα αρχείο εισόδου μία ακολουθία λέξεων. Για κάθε λέξη $s$ που διαβάζει, καλεί τη μέθοδο $processWord(s)$, η οποία μετρά το πλήθος των εμφανίσεων κάθε γράμματος στη λέξη $s$ και ενημερώνει τον πίνακα $F[]$.

Αφού ολοκληρώσει την ανάγνωση των λέξεων από το αρχείο εισόδου, το πρόγραμμα καλεί τη μέθοδο $encode()$ για να υπολογίσει τον κώδικα Huffman των γραμμάτων $C[1] = 'a', C[2] = 'b', \dots, C[26] = 'z'$, με βάση το πλήθος των εμφανίσεων τους $F[1], F[2], \dots, F[26]$. Τέλος, καλεί την έτοιμη μέθοδο $printCode()$ για να τυπώσει τον κώδικα κάθε γράμματος.

Για την εκτέλεση του προγράμματος θα χρησιμοποιήσουμε για αρχεία εισόδου το κείμενο TomSawyerB.txt, καθώς και το λεξικό words.txt.

Για την εκτέλεση των προγραμμάτων γράψτε

java Huffman < TomSawyerB.txt

και

java Huffman < words.txt.

## Παραδοτέα

Ανεβάστε στο eCourse ένα zip αρχείο με τα τελικά προγράμματα σας MinPQ.java και Huffman.java, το έτοιμο πρόγραμμα In.java, καθώς και με το αρχείο των αποτελεσμάτων lab5results.txt. Το zip αρχείο πρέπει να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.