ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Δομές Δεδομένων - Εργασία 2

Τμήμα Πληροφορικής

Φθινοπωρινό Εξάμηνο 2018-2019

Διδάσκων: Ε. Μαρκάκης

Ταξινόμηση και Ουρές Προτεραιότητας

Σκοπός της 2ης εργασίας είναι η εξοικείωση με τους αλγορίθμους ταξινόμησης και με τη χρήση ουράς προτεραιότητας, μία από τις πιο βασικές δομές δεδομένων στη σχεδίαση αλγορίθμων. Η εργασία αφορά ερωτήματα που συναντώνται σε απλά recommendation systems, όπως συστήματα που προτείνουν ταινίες, σειρές, ή τραγούδια σε συνδρομητές ή απλούς χρήστες. Για παράδειγμα, το Netflix διατηρεί πάντα μια (προσωποποιημένη) λίστα από προτεινόμενες σειρές προς κάθε συνδρομητή του, ενώ το YouTube επίσης επιλέγει με κάποιον αλγόριθμο τα επόμενα τραγούδια μετά την πρώτη επιλογή του χρήστη.

Έστω ότι σε μια εφαρμογή τύπου YouTube, θέλουμε να προτείνουμε στον χρήστη τα k καλύτερα τραγούδια (είτε γενικά είτε από μια κατηγορία μουσικής) για κάποια παράμετρο k, με κριτήριο τα likes που έχουν πάρει σε κάποιο κοινωνικό δίκτυο ή άλλο μέσο. Για την υλοποίηση θα χρησιμοποιήσετε τον εξής ΑΤΔ:

ΑΤΔ Song: Είναι ο ΑΤΔ που αναπαριστά ένα τραγούδι και θα είναι στο αρχείο Song.java. Τα αντικείμενα αυτού του τύπου θα πρέπει να περιέχουν:

- ένα πεδίο τύπου int με όνομα id. Ο αριθμός αυτός είναι μοναδικός για κάθε τραγούδι. Για απλότητα, θεωρούμε ότι το id παίρνει τιμές από το 1 ως το 9999.
- ένα πεδίο τύπου String με όνομα title, για το όνομα του τραγουδιού. Θα θεωρήσουμε ότι ο τίτλος δεν υπερβαίνει τους 80 χαρακτήρες (μαζί με τυχόν κενά που θα έχει).
- ένα πεδίο τύπου int με όνομα likes. Είναι τα likes που παίρνει κάθε τραγούδι,
- επιτρέπεται να προσθέσετε και άλλα πεδία στην κλάση αυτή.

Μέρος Α [20 μονάδες]. Υλοποιήστε ένα πρόγραμμα το οποίο θα βρίσκει και θα τυπώνει τα k πιο δημοφιλή τραγούδια με χρήση κάποιου αλγορίθμου ταξινόμησης, και σύμφωνα με τις οδηγίες που ακολουθούν.

Δεδομένα εισόδου: Η είσοδος θα είναι ένα txt αρχείο, το όνομα του οποίου θα δίνεται από τον χρήστη, και το περιεχόμενό του θα είναι όπως στο παρακάτω παράδειγμα:

```
114 Fuego 51
313 Despacito 63
22 Bitter Sweet Symphony 71
812 The Passenger 63
9 Daddy Cool 78
```

Το format αυτό έχει την εξής ερμηνεία: το πρώτο πεδίο είναι το id κάθε τραγουδιού, στη συνέχεια ακολουθεί ο τίτλος του τραγουδιού και το τελευταίο πεδίο είναι τα likes που έχει πάρει. Έτσι, το τραγούδι The Passenger έχει id ίσο με 812 και έχει 63 likes.

Για να συγκρίνετε τραγούδια μεταξύ τους, θα χρησιμοποιήσετε τα πεδία likes και title. Όταν ένα τραγούδι έχει περισσότερα likes από ένα άλλο θεωρείται καλύτερο. Σε περίπτωση που το πεδίο likes είναι ίδιο σε 2 ή παραπάνω τραγούδια, θα κάνετε σύγκριση αλφαβητικά με βάση τα ονόματα (δηλαδή προηγείται το τραγούδι με λεξικογραφικά μικρότερο όνομα). Επομένως, η κλάση Song θα πρέπει να υλοποιεί το interface Comparable<Song> (και συνεπώς της συνάρτησης compareTo), έτσι ώστε να μπορέσετε να την χρησιμοποιήσετε για σύγκριση ή ταξινόμηση.

Για το Μέρος Α, η υλοποίηση θα πρέπει να γίνει χρησιμοποιώντας κάποιον αλγόριθμο ταξινόμησης για όλα τα τραγούδια του αρχείου (και επιλέγοντας μετά τα k καλύτερα, αφού ταξινομήσετε). Μπορείτε να υλοποιήσετε οποιαδήποτε από τις μεθόδους Mergesort, Quicksort ή Heapsort. Είστε ελεύθεροι να διαλέξετε όποια από τις 3 μεθόδους θέλετε, απαγορεύεται όμως να χρησιμοποιήσετε έτοιμες υλοποιήσεις που παρέχονται από την Java. Θα πρέπει να υλοποιήσετε τη δική σας μέθοδο είτε με πίνακα είτε με λίστα.

Τέλος, η παράμετρος k θα δίνεται επίσης από τον χρήστη, και συνήθως θα είναι μια μικρή σταθερά (μικρότερη από τον συνολικό αριθμό τραγουδιών του αρχείου). Αν το k είναι μεγαλύτερο από τον συνολικό αριθμό τραγουδιών του αρχείου θα τυπώνετε μήνυμα λάθους και το πρόγραμμα θα τερματίζει. Με k=3, στο παράδειγμα που φαίνεται παραπάνω η εκτύπωση του προγράμματος θα πρέπει να είναι στη μορφή:

```
The top k songs are:
Daddy Cool
Bitter Sweet Symphony
Despacito
```

Υπόψιν ότι επιλέχθηκε ως 3^{ο} τραγούδι το Despacito καθώς ενώ έχει το ίδιο σκορ με το 4^{ο} τραγούδι, προηγείται αλφαβητικά.

Μέρος Β [15 μονάδες]. Ένα μειονέκτημα της παραπάνω προτεινόμενης υλοποίησης είναι ότι πρέπει να αποθηκεύσετε πρώτα όλα τα τραγούδια για να αποφασίσετε ποια είναι τα k καλύτερα. Ιδανικά θα θέλαμε μετά το διάβασμα κάθε γραμμής του αρχείου να ξέραμε ποια είναι τα k καλύτερα μέχρι εκείνη τη στιγμή, και να ενημερώνουμε δυναμικά αυτή την πληροφορία χωρίς να χρειάζεται να αποθηκεύσουμε κάπου όλα τα τραγούδια του αρχείου. Επίσης, εφόσον στο Μέρος Α στηρίζεστε σε αλγόριθμο ταξινόμησης, θα έχετε πολυπλοκότητα τουλάχιστον O(nlogn) αν έχετε n τραγούδια. Όταν όμως η παράμετρος k είναι πολύ μικρή σε σχέση με το n, τότε αξίζει να δούμε πώς θα μπορούσαμε να έχουμε μια καλύτερη υλοποίηση. Για να γίνει αυτό, ένας ενδεδειγμένος τρόπος σε τέτοια προβλήματα είναι η χρήση ουράς προτεραιότητας.

ΑΤΔ ουράς προτεραιότητας. Θα χρειαστείτε μια ουρά προτεραιότητας, όπου θα αποθηκεύετε αντικείμενα τύπου Song. Η υλοποίηση της ουράς θα γίνει με χρήση σωρού, όπως έχουμε δει στο μάθημα και στο εργαστήριο. Μπορείτε να βασιστείτε στην ουρά προτεραιότητας του

Εργαστηρίου 5 ή σε αυτή του βιβλίου και των διαφανειών, και να τις τροποποιήσετε κατάλληλα, ή μπορείτε να φτιάξετε εξ ολοκλήρου τη δική σας ουρά. Η ουρά σας θα πρέπει να διαθέτει, εκτός από τις λειτουργίες *insert* και *getmax*, που είδαμε και στο μάθημα, και κάποιες επιπλέον μεθόδους. Συγκεκριμένα, οι μέθοδοι που απαιτούνται είναι οι εξής (μπορείτε να έχετε κι άλλες μεθόδους μέσα στην υλοποίησή σας):

- `boolean isEmpty()`: έλεγχος για το αν είναι άδεια η ουρά.
- `int size()`: επιστρέφει τον αριθμό ενεργών στοιχείων στην ουρά.
- `void insert(Song x)`: εισαγωγή αντικειμένου στην ουρά. Η εισαγωγή θα πρέπει να καλεί και μια μέθοδο resize που θα κάνει διπλασιασμό του πίνακα της ουράς όταν η ουρά έχει γεμίσει κατά το 75%.
- `Song max()`: επιστρέφει το στοιχείο με τη μέγιστη προτεραιότητα χωρίς να το αφαιρεί από την ουρά.
- `Song getmax()`: Αφαιρεί και επιστρέφει το αντικείμενο με τη μέγιστη προτεραιότητα.
- `Song remove(int id)`: Αφαιρεί και επιστρέφει το αντικείμενο με το συγκεκριμένο id. Πρέπει μετά την αφαίρεση να αποκατασταθεί η ιδιότητα του σωρού για να μην καταστραφεί η ουρά.

Η πολυπλοκότητα που πρέπει να έχει η κάθε μέθοδος, σε μια ουρά με n αντικείμενα δίνεται στον παρακάτω πίνακα.

Μέθοδος	Πολυπλοκότητα
`size, isEmpty`	O(1)
`insert`	O(logn) (*δειτε σχόλια*)
`max`	O(1)
`getmax`	O(logn)
`remove`	O(logn) (*δειτε σχόλια*)

Σχόλια:

- Για την insert, η απαίτηση για O(logn) είναι μόνο όταν δεν χρειάζεται να καλέσει τη resize για την επέκταση του πίνακα
- Για να υλοποιήσετε την remove σε O(logn), ίσως χρειαστεί να κάνετε κάποιες προσθήκες στα χαρακτηριστικά της ουράς και στον τρόπο λειτουργίας των insert και getmax. Οι τροποποιήσεις αυτές δεν θα πρέπει να επιβαρύνουν τις insert και getmax παραπάνω από ένα σταθερό παράγοντα, ώστε να παραμένει λογαριθμική η πολυπλοκότητά τους.
- Αν δυσκολεύεστε να κάνετε την remove σε O(logn), υπάρχει ένας πολύ πιο απλοϊκός τρόπος να την κάνετε σε O(n). Μπορείτε να κάνετε αυτή την υλοποίηση για τις μισές μονάδες από αυτές που αντιστοιχούν στην remove.

Μέρος Γ [30 μονάδες]. Αυτό είναι το πιο ενδιαφέρον κομμάτι της εργασίας. Χρησιμοποιώντας την ουρά, θέλουμε τώρα να διαβάζουμε το αρχείο εισόδου γραμμή προς γραμμή, και μετά την ανάγνωση κάθε γραμμής, να διατηρούμε τα k καλύτερα τραγούδια μέχρι εκείνη τη στιγμή. Το πρόγραμμα και πάλι θα τυπώνει μόνο τα k καλύτερα στο τέλος της ανάγνωσης όλου του αρχείου όπως και στο Μέρος Α. Για το πρόγραμμά σας, επιτρέπεται να χρησιμοποιήσετε μια ουρά προτεραιότητας **που θα περιέχει το πολύ k στοιχεία**. Δεν επιτρέπεται να χρησιμοποιήσετε δομή που να αποθηκεύει όλα τα τραγουδια του αρχείου εισόδου (θεωρούμε πάντα ότι το k είναι μικρότερο από τις γραμμές του αρχείου). Μπορείτε να αποθηκεύσετε τραγούδια μόνο στην ουρά προτεραιότητας αλλά κατά τη διάρκεια εκτέλεσης του προγράμματος, η ουρά δεν πρέπει ποτέ να έχει παραπάνω από k τραγούδια. Τονίζουμε ότι το μέγεθος του πίνακα της ουράς μπορεί να είναι παραπάνω από k λόγω της resize στις εισαγωγές, αυτό που θέλουμε

είναι το πλήθος των ενεργών στοιχείων στην ουρά να μην είναι ποτέ πάνω απο k (μπορείτε αν θέλετε να αρχικοποιήσετε την ουρά με μέγεθος 2k για να μην χρειαστεί να γίνει ποτέ η resize). Σκεφτείτε πώς αρκεί μια τέτοια ουρά για να λύσετε το πρόβλημα της επιλογής των k καλύτερων τραγουδιών. Η απαίτηση για την πολυπλοκότητα του προγράμματος είναι ότι σίγουρα δεν πρέπει να είναι χειρότερη από την πολυπλοκότητα του Μέρους Α.

Hint: Η ουρά που θα χρησιμοποιήσετε είναι μεγιστοστρεφής, δηλαδή η getmax φέρνει πάντα το στοιχείο με την μέγιστη προτεραιότητα. Αν για την επίλυση του προβλήματος χρειαστεί να υποστηρίξετε κι άλλες λειτουργίες, αυτό δεν πρέπει να σας οδηγήσει σε εσωτερικές αλλαγές στον τρόπο λειτουργίας ενός μεγιστοστρεφούς σωρού (ο σωρός πρέπει να δουλεύει όπως είδαμε στο μάθημα), αλλά σε ορθή χρήση της δομής. Σκεφτείτε προσεκτικά πως θα ορίσετε την προτεραιότητα ενός τραγουδιού για τις ανάγκες του συγκεκριμένου προβλήματος και κατασκευάστε τον κατάλληλο τρόπο σύγκρισης για την ουρά.

Πρόσθετες οδηγίες υλοποίησης για τα μέρη Α, Β και Γ:

- Το πρόγραμμα σας για το μέρος Α **πρέπει να λέγεται** Top_k.java.
- Η υλοποίηση της ουράς προτεραιότητας θα πρέπει να είναι στο αρχείο PQ.java.
- Το πρόγραμμα για το μέρος Γ **πρέπει να λέγεται** Top_k_withPQ.java.
- Αν θέλετε, μπορείτε να υλοποιήσετε την ουρά προτεραιότητας με generics με bonus 10% στον βαθμό της εργασίας.
- Για το διάβασμα του αρχείου εισόδου, μπορείτε να χρησιμοποιήσετε έτοιμες μεθόδους της Java. Μπορείτε επίσης να χρησιμοποιήσετε έτοιμες μεθόδους για να επεξεργαστείτε μεταβλητές τύπου String. Επιπλέον, μπορείτε να χρησιμοποιήσετε τις δομές που έχουμε δει μέχρι στιγμής στο μάθημα, τα εργαστήρια ή την $1^{η}$ εργασία σας: συνδεδεμένη λίστα (μονή ή διπλή), ουρές, κτλ. Μπορείτε να χρησιμοποιήσετε είτε δικές σας υλοποιήσεις ή τις έτοιμες υλοποιήσεις του εργαστήριου. **Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις δομών τύπου λίστας, στοίβας, ουράς, από την βιβλιοθήκη της Java** (π.χ. Vector, ArrayList, κτλ).
- Δεν χρειάζεται να ασχοληθείτε με ανίχνευση λαθών στο format του αρχείου εισόδου, παρά μόνο ότι αφορά τους περιορισμούς για το id και τον τίτλο.
- Το μονοπάτι για το προς εξέταση αρχείο θα δίνεται ως όρισμα όπως και το k, π.χ. η εκτέλεση της main του Μέρους Α από τη γραμμή εντολών θα γίνεται ως εξής:

```
java Top_k k path_to_file
```

Μέρος Δ [25 μονάδες]. Έστω ότι διαβάζουμε και πάλι ένα αρχείο εισόδου με το ίδιο format, και μας ενδιαφέρει να απαντάμε σε ερωτήσεις της μορφής: «Πόσα likes χρειάζεται να πάρει ένα τραγούδι για να είναι στο top 50% των τραγουδιών που εξετάζουμε;». Αυτο σημαίνει ότι πρέπει να υπολογίζουμε το median από τα likes που βλέπουμε. Ο median μιας ακολουθίας ακεραίων είναι ο αριθμός που θα βρίσκεται στη μέση της ακολουθίας αν την ταξινομήσουμε. Π.χ. στην ακολουθία 13, 17, 11, 24, 19, 8, 14, ο median είναι ο 14. Αν η ακολουθία έχει άρτιο αριθμό ακεραίων, τότε θα κάνουμε τη σύμβαση να παίρνουμε ως median τον μεγαλύτερο από τους 2 μεσαίους, δηλαδή στην ακολουθία 13, 17, 11, 24, 19, 8, θεωρούμε ως median τον 17.

Θέλουμε να υλοποιήσουμε μια δομή, η οποία να μπορεί να διατηρεί ενημερωμένο τον median δυναμικά, όπως επεξεργάζεται κάθε νέο τραγούδι από το αρχείο εισόδου. Θέλουμε δηλαδή ένα πρόγραμμα, το οποίο μετά την ανάγνωση κάθε γραμμής του αρχείου εισόδου, να μπορεί εύκολα να υπολογίσει τον median αριθμό από likes, με βάση τα τραγούδια που έχουμε επεξεργαστεί μέχρι εκείνη τη χρονική στιγμή (γενικεύοντας, η δυναμική ενημέρωση του median σε ενα stream από δεδομένα είναι σημαντικό πρόβλημα σε αρκετές άλλες εφαρμογές).

Υπάρχουν φυσικά πολλοί τρόποι για να λυθεί ένα τέτοιο πρόβλημα. Σκοπός του μέρους Δ είναι η αποδοτική επίλυση του προβλήματος με τη χρήση 2 ουρών προτεραιότητας. Θα πρέπει λοιπόν να χρησιμοποιήσετε κατάλληλα 2 ουρές προτεραιότητας έτσι ώστε η επεξεργασία που θα κάνετε μετά την ανάγνωση κάθε γραμμής του αρχείου να επιτρέπει την εύκολη ενημέρωση του median.

Οδηγίες υλοποίησης:

- Το πρόγραμμα σας **πρέπει να λέγεται** Dynamic_Median.java.
- Για να μην είναι πολύ μεγάλη η εκτύπωση της εξόδου, απαιτείται να τυπώνετε τον median μόνο μετά από κάθε 5 γραμμές του αρχείου εισόδου. Το πρόγραμμά σας θα πρέπει με την ανάγνωση κάθε γραμμής εισόδου να κάνει κάποια επεξεργασία, αλλά ο υπολογισμός και η εκτύπωση του median θα γίνεται μόνο όταν (i%5 == 0). Π.χ. στο παράδειγμα που υπάρχει στο Μέρος Α, θα τυπωνόταν μόνο 1 φορά το εξής:
  `Median = 63 likes, achieved by Song: Despacito`
- Για απλότητα, δεν θα χρησιμοποιήσουμε αρχεία εισόδου άνω των 500 γραμμών.

Μέρος Ε - Αναφορά παράδοσης [10 μονάδες]. Ετοιμάστε μία σύντομη αναφορά σε pdf αρχείο (μην παραδώσετε Word ή txt αρχεία!) με όνομα project2-report.pdf, στην οποία θα αναφερθείτε στα εξής:

a. Εξηγήστε συνοπτικά ποιον αλγόριθμο ταξινόμησης υλοποιήσατε στο μέρος Α (άνω όριο 1 σελίδα).
b. Για τα Μέρη Β και Γ, σχολιάστε αρχικά πώς υλοποιήσατε την remove του Μέρους Β. Μετέπειτα, εξηγήστε αναλυτικά την ιδέα για την υλοποίηση του προγραμματος Top_k_withPQ στο Μέρος Γ. Περιγράψτε τι πολυπλοκότητα αναμένεται να έχει το Μέρος Γ και αν είναι τελικά συμφέρον να χρησιμοποιήσουμε το Top_k_withPQ σε περιπτώσεις όπου το k είναι αρκετά μικρότερο του συνολικού αριθμού τραγουδιών (άνω όριο 3 σελίδες).
c. Ομοίως για το μέρος Γ. Σχολιάστε την πολυπλοκότητα του προγράμματός σας. Π.χ. όταν διαβάζετε μια γραμμή του αρχείου εισόδου, τι πολυπλοκότητα έχει η εισαγωγή του τραγουδιού στη δομή σας; Επίσης, σε κάθε γραμμή i, με (i%5 ==0), όπου πρέπει να τυπώσουμε τον median, τι πολυπλοκότητα έχει ο υπολογισμός του median; (άνω όριο 2 σελίδες)

Το συνολικό μέγεθος της αναφοράς θα πρέπει να είναι τουλάχιστον 2 σελίδες και το πολύ 6 σελίδες.

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. Ενδεικτικά θα πρέπει να περιέχει (χρησιμοποιήστε τα όνοματα των κλάσεων όπως ακριβώς δίνονται στην εκφώνηση):
   - Τα προγράμματα Top_k.java, Top_k_withPQ.java.
   - Την υλοποίηση της ουράς προτεραιότητας PQ.java.
   - Το πρόγραμμα Dynamic_Median.java.

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. 3030056_3030066.zip ή 3030056.zip (αν δεν είστε σε ομάδα). Στη συνέχεια, θα υποβάλλετε 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 Παρασκευή, 14 Δεκεμβρίου 2018 και ώρα 23:59.