ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

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

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Διδάσκουσα: Κ. Παπακωνσταντινοπούλου

Χειμερινό εξάμηνο 2022

Εργασία 2
Ταξινόμηση και Ουρές Προτεραιότητας
Προθεσμία υποβολής: 8/1/2023, 23:59

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 προβλήματα βελτιστοποίησης σχετικά με την ανάθεση πόρων μνήμης για αποθήκευση δεδομένων μεγάλου όγκου. Θα εξετάσουμε την εξής απλοϊκή εκδοχή. Έστω ότι θέλουμε να τοποθετήσουμε όλα τα περιεχόμενα από ένα σύνολο N φακέλων με αρχεία σε σκληρούς δίσκους χωρητικότητας 1 TB ο καθένας (σκεφτείτε π.χ. ως εφαρμογή την δημιουργία αντιγράφων backup σε ένα σύστημα διαχείρισης αρχείων). Υποθέτουμε ότι όλοι οι φάκελοι είναι με μέγεθος ανάμεσα σε 0 και 1.000.000 MB (1 TB). Έχουμε τον περιορισμό ότι κάθε φάκελος πρέπει να αποθηκεύεται εξ' ολοκλήρου σε έναν δίσκο. Ιδανικά, η βέλτιστη λύση θα ήταν να χρησιμοποιήσουμε τον ελάχιστο δυνατό αριθμό από σκληρούς δίσκους. Το πρόβλημα αυτό είναι παράδειγμα του γνωστού προβλήματος «Bin Packing», για το οποίο δεν γνωρίζουμε μέχρι σήμερα αν υπάρχει αποδοτικός αλγόριθμος που να μπορεί να βρίσκει πάντα τη βέλτιστη λύση. Μπορούμε όμως να σχεδιάσουμε αποδοτικές μεθόδους προσέγγισης της βέλτιστης λύσης.

Μέρος Α (15 μονάδες)

Ας δούμε πρώτα τους ΑΤΔ που θα χρειαστείτε.

ΑΤΔ δίσκου: Για να υλοποιήσετε τον αλγόριθμο θα πρέπει πρώτα να υλοποιήστε έναν τύπο δεδομένων που αναπαριστά έναν δίσκο μεγέθους 1TB. Ονομάστε την τάξη αυτή Disk. Αντικείμενα της τάξης Disk πρέπει

- να έχουν μοναδικά id που ανατίθενται όταν δημιουργείται ένας νέος δίσκος (χρήσιμο για debugging).
- να περιέχουν μια λίστα με όνομα folders, που αρχικοποιείται ως κενή, και στην οποία θα μπαίνουν οι φάκελοι που αποθηκεύονται σε αυτόν τον δίσκο κατά τη διάρκεια του αλγορίθμου αποθήκευσης.
- να διαθέτουν τη μέθοδο getFreeSpace() η οποία όταν καλείται επιστρέφει τον ελεύθερο χώρο του δίσκου σε MB.

Στην κλάση Disk μπορείτε να προσθέσετε και ό,τι άλλο πεδίο θεωρείτε εσείς χρήσιμο. Τέλος, η Disk θα πρέπει να υλοποιεί το interface Comparable<Disk> έτσι ώστε να μπορέσετε να την χρησιμοποιήσετε σε μία ουρά προτεραιότητας. Για την υλοποίηση του interface Comparable<Disk> (και συνεπώς της συνάρτησης compareTo) μπορείτε απλά να συγκρίνετε τον ελεύθερο χώρο κάθε δίσκου: αν ο δίσκος A έχει περισσότερο ελεύθερο χώρο από τον δίσκο B τότε θεωρούμε ότι $A > B$ και η πράξη

A.compareTo(B)

πρέπει να επιστρέφει 1. Αντίστοιχα, όταν $A < B$ η συνάρτηση επιστρέφει -1 και όταν $A == B$ η συνάρτηση επιστρέφει 0.

ΑΤΔ ουράς προτεραιότητας: Για την εργασία θα χρειαστείτε μια αποδοτική δομή δεδομένων για ουρά προτεραιότητας. Μπορείτε είτε να χρησιμοποιήσετε την ουρά του φροντιστηρίου είτε να φτιάξετε τη δική σας ουρά, βασισμένοι στις διαφάνειες των διαλέξεων. Σε κάθε περίπτωση, η ουρά σας θα πρέπει να διαθέτει οπωσδήποτε τις λειτουργίες insert και getmax (απομάκρυνση στοιχείου με μέγιστο κλειδί), όπως είδαμε και στο μάθημα. Ονομάστε την κλάση αυτή MaxPQ.

Για τη λίστα folders της κλάσης Disk, δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις δομών τύπου λίστας από την βιβλιοθήκη της Java (π.χ. ArrayList, LinkedList, κτλ). Όπως και στην Εργασία 1, μπορείτε να χρησιμοποιήσετε είτε τη λίστα του φροντιστηρίου, είτε να φτιάξετε τη δική σας.

Μέρος Β (40 μονάδες)

Υλοποιήστε τον παρακάτω αλγόριθμο αποθήκευσης. Ονομάστε το πρόγραμμά σας Greedy.java.

Αλγόριθμος 1 - Greedy: Επεξεργάσου τους φακέλους έναν έναν με τη σειρά που εμφανίζονται. Αν ένας φάκελος χωράει σε κάποιον από τους δίσκους που έχουμε ήδη χρησιμοποιήσει ως τώρα, τον αποθηκεύουμε στον δίσκο με τον περισσότερο ελεύθερο χώρο. Διαφορετικά, αν δεν χωράει σε κανέναν, χρησιμοποιούμε καινούργιο δίσκο και αποθηκεύουμε το αρχείο εκεί.

Παράδειγμα: Έστω ότι ο αλγόριθμος βλέπει διαδοχικά 5 φακέλους με μεγέθη 200.000, 150.000, 700.000, 800.000, 100.000 (σε MB). Το αποτέλεσμα του αλγορίθμου θα είναι να χρησιμοποιήσει 3 δίσκους όπου θα αποθηκεύσει τους φακέλους 1, 2 και 5 με μεγέθη 200.000, 150.000, 100.000 στον 1ο δίσκο, τον φάκελο 3 με μέγεθος 700.000 στον 2ο (διότι την στιγμή που επεξεργάστηκε τον φάκελο αυτό, δεν χωρούσε στον 1ο δίσκο), και τον φάκελο 4 με μέγεθος 800.000 στον 3ο δίσκο. **Προσέξτε** ότι στο συγκεκριμένο παράδειγμα, η βέλτιστη λύση θα ήταν.να χρησιμοποιούσαμε 2 δίσκους αντί για 3 (φάκελοι 1, 4 στον 1ο, και φάκελοι 2, 3, 5 στον 2ο). Παρά το γεγονός ότι ο Αλγόριθμος 1 δεν είναι πάντα βέλτιστος ως προς τον αριθμό των χρησιμοποιούμενων δίσκων, είναι ένας αρκετά γρήγορος αλγόριθμος που σε πολλές περιπτώσεις μπορεί να προσεγγίσει την βέλτιστη λύση.

Input και output

Input. Το πρόγραμμα Greedy.java θα περιέχει μία μέθοδο main η οποία αρχικά θα διαβάζει τα μεγέθη των φακέλων από ένα txt αρχείο για να μπορέσει να εκτελέσει τον αλγόριθμο. Κάθε γραμμή του αρχείου θα έχει το μέγεθος ενός φακέλου προς αποθήκευση σε MB, άρα θα είναι ένας ακέραιος μεταξύ 0 και 1000000. Στο παράδειγμα που αναφέρθηκε παραπάνω, το αρχείο εισόδου θα έχει την παρακάτω μορφή:

```
200000
150000
700000
800000
100000
```

Αν κάποιος φάκελος δεν έχει μέγεθος μεταξύ 0 και 1.000.000 θα πρέπει να τυπώνετε μήνυμα λάθους και να τερματίζετε το πρόγραμμα.

Output. Το πρόγραμμα θα πρέπει να υπολογίζει και να τυπώνει τον αριθμό των δίσκων που χρησιμοποιήθηκαν από τον αλγόριθμο, καθώς και το άθροισμα των μεγεθών όλων των φακέλων σε TB (παρατηρήστε ότι αυτό είναι ένα κάτω όριο στον ελάχιστο αριθμό των δίσκων που απαιτούνται). Επίσης αν ο αριθμός των φακέλων προς αποθήκευση δεν είναι μεγαλύτερος του 100, να τυπώνετε και τα περιεχόμενα των δίσκων με σειρά φθίνουσα σε σχέση με τον άδειο χώρο του κάθε δίσκου. Για κάθε δίσκο τυπώστε το id του, το μέγεθος του ελεύθερου χώρου του και στην συνέχεια το μέγεθος κάθε φακέλου που έχει αποθηκευτεί στο δίσκο. Ακολουθεί ένα παράδειγμα για τη μορφή της εξόδου:

```
% java Greedy input.txt
Sum of all folders = 6.580996 TB
Total number of disks used = 8
id 5 325754: 347661 326585
id 0 227744: 420713 351543
id 7 224009: 383972 392019
id 4 190811: 324387 484802
id 6 142051: 340190 263485 254274
id 3 116563: 347560 204065 331812
id 2 109806: 396295 230973 262926
id 1 82266: 311011 286309 320414
```

Το παραπάνω παράδειγμα δείχνει ότι χρησιμοποιήθηκαν 8 δίσκοι για να αποθηκευτούν δεδομένα με συνολικό μέγεθος 6,580996 TB, και π.χ. ο δίσκος με id 5 έχει 325754 MB ελεύθερο χώρο και περιέχει 2 φακέλους με μέγεθος 347661 και 326585 αντίστοιχα.

Μέρος Γ (15 μονάδες)

Στο παράδειγμα της 2ης σελίδας είδαμε ότι η λύση του αλγορίθμου 1 δεν ήταν βέλτιστη. Μία αιτία για αυτό ήταν ότι ο αλγόριθμος επεξεργάστηκε πρώτα 2 μικρούς φακέλους με αποτέλεσμα οι μεγαλύτεροι φάκελοι να μην χωράνε μαζί με τους μικρούς και να χρειάζονται καινούριους δίσκους. Με αφορμή αυτό, μια ιδέα για βελτίωση του αλγορίθμου είναι να κοιτάζουμε τους φακέλους σε φθίνουσα σειρά από τον μεγαλύτερο σε μέγεθος προς τον μικρότερο. Στο συγκεκριμένο παράδειγμα αν το κάναμε αυτό, θα βλέπαμε ότι τελικά θα βάζαμε στον 1ο δίσκο τους φακελους με μεγέθη 800 και 150 GB, και στον 2ο δίσκο τους υπόλοιπους. Επομένως θα φτάναμε σε μια καλύτερη λύση με χρήση μόνο 2 δίσκων. Ο νέος αλγόριθμος λοιπόν είναι ο εξής:

Αλγόριθμος 2 (Greedy-decreasing) Διάταξε τους φακέλους με σειρά από τον μεγαλύτερο σε μέγεθος προς τον μικρότερο και μετά εφάρμοσε τον Αλγόριθμο 1.

Για την υλοποίηση του Αλγορίθμου 2, το μόνο που χρειάζεται είναι να τρέξετε πρώτα έναν αλγόριθμο ταξινόμησης ακεραίων σε φθίνουσα σειρά. Συνεπώς το ζητούμενο του Μέρους Γ είναι να γράψετε ένα πρόγραμμα με όνομα Sort.java, το οποίο θα εκτελεί μία από τις μεθόδους Mergesort, Quicksort ή Heapsort. Είστε ελεύθεροι να διαλέξετε όποια από τις 3 μεθόδους θέλετε, όπως επίσης είστε ελεύθεροι να αποφασίσετε αν θα κάνετε ταξινόμηση πάνω σε πίνακα ή σε λίστα. Δεν απαιτείται να υπάρχει μέθοδος main μέσα στο Sort.java.

ΠΡΟΣΟΧΗ: Απαγορεύεται να χρησιμοποιήσετε έτοιμες συναρτήσεις ταξινόμησης της βιβλιοθήκης της Java. Για την εργασία θα πρέπει να υλοποιήσετε τη δική σας μέθοδο.

Μέρος Δ (20 μονάδες)

Στο μέρος αυτό θα κάνετε μία μικρή πειραματική αξιολόγηση για να διαπιστώσετε ποιος αλγόριθμος είναι καλύτερος στην πράξη. Χρησιμοποιήστε κατάλληλα την κλάση Random του πακέτου java.util και δημιουργήστε τυχαία δεδομένα εισόδου για τουλάχιστον 3 διαφορετικές τιμές του πλήθους των φακέλων. Ενδεικτικά, αν N είναι το πλήθος των φακέλων, μπορείτε να χρησιμοποιήσετε $N = 100, 500, 1.000$. Για κάθε τιμή του N, δημιουργήστε 10 διαφορετικά txt αρχεία εισόδου για τους αλγορίθμους 1 και 2. Συνολικά δηλαδή θα δημιουργήσετε τουλάχιστον 30 αρχεία τυχαίων δοκιμαστικών δεδομένων. Στα αρχεία αυτά, κάθε γραμμή πρέπει να είναι ένας τυχαίος ακέραιος στο διάστημα $[0, 1000000]$.

Στη συνέχεια γράψτε ένα πρόγραμμα που θα εκτελεί και θα συγκρίνει τους δύο αλγόριθμους. Για κάθε αρχείο εισόδου καταγράψτε πόσους σκληρούς δίσκους απαιτούν οι δύο αλγόριθμοι. Υπολογίστε για κάθε τιμή του N, πόσους δίσκους απαιτούν κατά μέσο όρο οι δύο αλγόριθμοι, με βάση τα τυχαία σας δεδομένα. Παραδοτέα του Μέρους Δ είναι ο κώδικας που γράψατε για να παράξετε τα αρχεία εισόδου και για να συγκρίνετε τους 2 αλγορίθμους. Θα πρέπει όμως να εξηγήσετε και στην αναφορά παράδοσης πώς κάνατε αυτά τα βήματα (βλ. Μέρος Ε). Δεν υπάρχει περιορισμος για το πώς θα ονομάσετε τα αρχεία java που θα γράψετε στο Μέρος Δ.

Προαιρετικά: Μπορείτε να χρησιμοποιήσετε περισσότερες τιμές για το N, και περισσότερα από 10 αρχεία για κάθε τιμή του N, και να περάσετε τα αποτελέσματα σε ένα λογιστικό φύλλο (Microsoft Excel, OpenOffice, κλπ). Χρησιμοποιήστε τις δυνατότητες του λογιστικού φύλλου για να σχεδιάσετε ένα διάγραμμα όπου θα απεικονίζεται η σχέση «Πλήθος δίσκων ως συνάρτηση του πλήθους των φακέλων (N)». Στο ίδιο διάγραμμα θα απεικονίζονται δύο καμπύλες: μία για κάθε αλγόριθμο.

Hints και πρόσθετες οδηγίες υλοποίησης για τα μέρη Β, Γ και Δ:

- Ακολουθήστε αυστηρά την ονοματολογία για τις κλάσεις σας, δηλαδή ονομάστε Disk.java, MaxPQ.java τους 2 ΑΤΔ, και Greedy.java, Sort.java τα 2 προγράμματα που ζητούνται.
- Η υλοποίησή του Αλγορίθμου 1 **πρέπει** να κάνει χρήση της ουράς προτεραιότητας MaxPQ. Το πρόβλημα μπορεί να λυθεί και χωρίς την ουρά, αλλά με χειρότερη πολυπλοκότητα. Η ουσία της εργασίας είναι να εκμεταλλευτείτε την χρήση της ουράς προτεραιότητας.
- Αν θέλετε, μπορείτε να χρησιμοποιήσετε ουρά προτεραιότητας με generics.
- Για την καλύτερη ανάπτυξη του κώδικά σας, προσπαθήστε να χωρίσετε τα προγράμματα σας σε modules ανάλογα με την λειτουργία τους. Ενδεικτικά:

  - γράψτε ξεχωριστή μέθοδο που διαβάζει το αρχείο εισόδου και το αποθηκεύει σε κάποιο πίνακα ή λιστα (αν χρησιμοποιήσετε πίνακες αντί για λίστες, μπορείτε πρώτα να μετρήσετε τον αριθμό των γραμμών του αρχείου εισόδου, έτσι ώστε να ξέρετε τι μέγεθος απαιτείται για να αποθηκεύσετε τα δεδομένα).
  - Μπορείτε να υλοποιήσετε τον Αλγόριθμο 1 σαν μια μέθοδο που παίρνει είσοδο τον πίνακα ή τη λίστα που διαβάσατε, έτσι ώστε να μην ασχοληθείτε περαιτέρω με το txt αρχείο εισόδου.
  - Με αυτόν τον τρόπο, η υλοποίηση του Αλγορίθμου 2 απαιτεί μόνο την ταξινόμηση του πίνακα/λίστας και μπορείτε να επαναχρησιμοποιήσετε τα προηγούμενα κομμάτια του κώδικά σας από το Μέρος Β.

- Για το διάβασμα του αρχείου εισόδου, μπορείτε να χρησιμοποιήσετε έτοιμες μεθόδους της Java για διάβασμα αρχείων. Επιπλέον, μπορείτε στον κώδικά σας να χρησιμοποιήσετε τις δομές που έχουμε δει μέχρι στιγμής στο μάθημα, τα εργαστήρια ή την 1η εργασία σας. **Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις δομών τύπου λίστας, στοίβας, ουράς, από την βιβλιοθήκη της Java.**
- Πρέπει να δίνετε ως όρισμα ολο το μονοπάτι για το .txt αρχείο εισόδου στο Μέρος Β, π.χ. αν το τρέξετε από τη γραμμή εντολών, η εκτέλεση θα είναι ως εξής:

  $>$ java Greedy path_to_file/filename.txt

Μέρος Ε - Αναφορά παράδοσης (10 μονάδες)

Ετοιμάστε μία αναφορά σε pdf αρχείο (μην παραδώσετε Word ή txt αρχεία!) με όνομα project2-report.pdf. Στην αναφορά περιμένουμε να σχολιάσετε τα εξής (και ό,τι άλλο θεωρήσετε σκόπιμο εσείς):

1. Μέρος Α: Εξηγήστε συνοπτικά πώς φτιάξατε την ουρά προτεραιότητας (αν φτιάξατε δική σας, αν προσαρμόσατε του φροντιστηρίου, αν χρησιμοποιήσατε generics, κτλ). Άνω όριο: μισή σελίδα.
2. Μέρος Β: Σχολιάστε πώς χρησιμοποιήσατε την ουρά προτεραιότητας για να υλοποιήσετε τον Αλγόριθμο 1. Σχολιάστε επίσης οποιοδήποτε άλλο σημείο του κώδικα σας κρίνετε χρήσιμο (π.χ. που μπορεί να βοηθούσε τον διορθωτή της εργασίας). Άνω όριο: 1 σελίδα.
3. Μέρος Γ: Εξηγήστε ποιον αλγόριθμο ταξινόμησης υλοποιήσατε. Άνω όριο: μισή σελίδα.
4. Μέρος Δ: Εξηγήστε πώς έγινε η παραγωγή των δοκιμαστικών αρχείων εισόδου με τυχαία δεδομένα (πώς χρησιμοποιήσατε την κλάση Random). Ομοίως, εξηγήστε τι κάνατε για να τρέξετε τους 2 αλγορίθμους πάνω στα ίδια αρχεία εισόδου. Στη συνέχεια, σχολιάστε τα ευρήματά σας και δείξτε τους μέσους όρους για τον αριθμό των δίσκων που χρειάστηκαν για κάθε διαφορετική τιμή του N σε κάθε αλγόριθμο. Να αναφέρετε τα συμπεράσματα που βγάζετε για την απόδοση των 2 αλγορίθμων. Άνω όριο 4 σελίδες.

Το συνολικό μέγεθος της αναφοράς θα πρέπει να είναι τουλάχιστον 3 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. Χρησιμοποιήστε τα ονόματα των κλάσεων όπως ακριβώς δίνονται στην εκφώνηση. Επιπλέον, φροντίστε να συμπεριλάβετε όποια άλλα αρχεία πηγαίου κώδικα απαιτούνται για να μεταγλωττίζεται η εργασία σας. Φροντίστε επίσης να προσθέσετε επεξηγηματικά σχόλια όπου κρίνετε απαραίτητο στον κώδικά σας.
2. Τα δοκιμαστικά δεδομένα που φτιάξατε. Τοποθετήστε τα σε ένα subdirectory με όνομα **data**.
3. 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 υποβάλετε το zip αρχείο σας στην περιοχή του μαθήματος «Εργασίες» στο e-class. Δεν χρειάζεται υποβολή και από τους 2 συμμετέχοντες μιας ομάδας.