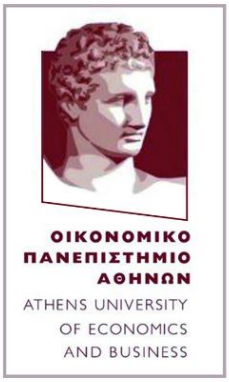
Δομές Δεδομένων
Χειμερινό εξάμηνο 2022-2023
Εργασία 2
Ταξινόμηση και Ουρές Προτεραιότητας



Ονοματεπώνυμο:	Αριθμός Μητρώου:
Γεώργιος Βεάζογλου	3190347
Βασιλική Δρόσου	3200044

Μέρος Α:

Η ουρά προτεραιότητας της εργασίας, αποτελεί μία τροποποιημένη εκδοχή της δοθείσας ουράς από το φροντιστήριο του μαθήματος με generics. Πιο συγκεκριμένα, έγινε αλλαγή σε access modifiers μεθόδων, με σκοπό να χρησιμοποιηθούν στους αλγορίθμους Greedy και Sort. Επίσης προστέθηκαν οι μέθοδοι:

boolean isEmpty(): Επιστρέφει true εάν η ουρά προτεραιότητας είναι άδεια.

int getSize(): Επιστρέφει το μέγεθος της ουράς.

Μέρος Β:

Αρχικά, δημιουργήθηκε η μέθοδος addFolderToDisk(int folder, MaxPQ<Disk> disk). Η παράμετρος folder ορίζει το μέγεθος του φακέλου προς αποθήκευση και η παράμετρος disk ορίζει την ουρά προτεραιότητας στην οποία βρίσκονται οι δίσκοι.

Εάν η ουρά είναι κενή, δημιουργείται ο πρώτος δίσκος, στη συνέχεια αποθηκεύεται σε αυτόν ο φάκελος και, τέλος, αποθηκεύεται ο δίσκος στην ουρά.

Εάν η ουρά δεν είναι κενή, με τη χρήση της μεθόδου peek() ελέγχουμε αν ο υπάρχων δίσκος έχει αρκετό αποθηκευτικό χώρο για το νέο φάκελο. Εφόσον ο ελεύθερος χώρος επαρκεί, ολοκληρώνεται η νέα αποθήκευση και χρησιμοποιούμε τη μέθοδο sink με σκοπό να

τοποθετήσουμε το δέντρο χαμηλά στη σωρό, εφόσον δεν αποτελεί πλέον τον δίσκο με τον μεγαλύτερο ελεύθερο χώρο.

Εάν η ουρά δεν είναι κενή, αλλά στον δίσκο που βρίσκεται στο root (δίσκος με τον περισσότερο ελεύθερο χώρο) δεν μπορεί να αποθηκευτεί ο νέος φάκελος, δημιουργούμε νέο δίσκο και τον τοποθετούμε στο root της ουράς.

Στη main μέθοδο του αλγορίθμου Greedy, διαβάζεται το δοσμένο txt έγγραφο με τη μέθοδο input και γίνεται χρήση της μεθόδου distributeFolders που μοιράζει τα αρχεία στους δίσκους. Έπειτα, στη μέθοδο output που τυπώνει το επιθυμητό αποτέλεσμα, γίνεται χρήση της μεθόδου getSize που επιστρέφει τον αριθμό των δίσκων που χρησιμοποιήθηκαν και της μεθόδου getMax που, σε συνδυασμό με την diskToString, θα επιστρέψει τους αποθηκευμένους δίσκους της ουράς σε φθίνουσα σειρά με κριτήριο τον ελεύθερο αποθηκευτικό χώρο που διαθέτουν.

Μέρος Γ:

Ο αλγόριθμος που υλοποιήθηκε είναι ο Mergesort. Αρχικά, με τη μέθοδο addFolders προσθέτουμε τα αρχεία σε ένα int array. Μετά, καλούμε τη μέθοδο mergeSort με παράμετρο το array των αρχείων. Η mergeSort, χωρίζει το array στα δύο. Έπειτα χρησιμοποιώντας αναδρομικά τη μέθοδο, διαιρούμε τα μέρη του array μέχρι να απομονώσουμε όλα τα στοιχεία. Όταν αυτό επιτευχθεί, καλούμε τη μέθοδο merge, η οποία θα ενώσει εκ νέου τα απομονωμένα στοιχεία, τοποθετώντας τα σε ένα νέο array σε φθίνουσα σειρά.

Μέρος Δ:

Η πειραματική αξιολόγηση της απόδοσης των δύο αλγορίθμων έγινε στην κλάση Test. Με τη χρήση της for τρεις διαδοχικές φορές δημιουργούμε συνολικά 30 αρχεία txt. Τα πρώτα 10, περιέχουν 100 τυχαίους αριθμούς, τα υπόλοιπα 10 περιέχουν 500 τυχαίους αριθμούς και τα 10 τελευταία περιέχουν 1.000 τυχαίους αριθμούς.

Η παραγωγή των δοκιμαστικών αρχείων έγινε χρησιμοποιώντας την κλάση Random και την κλάση FileWriter. Με μία for από 0 μέχρι N(100, 500, 1.000) δημιουργούμε τον ανάλογο αριθμό τυχαίων ακεραίων με τη μέθοδο rand.nextInt(1000001), που παράγει νούμερα από 0 μέχρι 1.000.000. Ύστερα, αποθηκεύουμε τους τυχαίους ακεραίους σε ένα αρχείο με τη μέθοδο write. Το αρχείο όπου θα αποθηκευτεί η κάθε ομάδα ακεραίων καθορίζεται κατά το definition της μεταβλητής writer.

Έπειτα, ορίζουμε 6 μεταβλητές στις οποίες θα αποθηκευτεί το πλήθος των δίσκων που χρησιμοποίησε ο κάθε αλγόριθμος κατά την αξιολόγηση του. Πιο συγκεκριμένα, οι 3 μεταβλητές greedyDisksUsed100, greedyDisksUsed500 και greedyDisksUsed 1000 θα αποθηκεύσουν τους δίσκους που χρειάστηκε ο αλγόριθμος Greedy σε κάθε στάδιο αξιολόγησης και, αναλόγως, οι 3 μεταβλητές sortDisksUsed100, sortDisksUsed500 και sortDisksUsed1000 θα αποθηκεύσουν τους δίσκους που χρειάστηκε ο αλγόριθμος Sort.

Στη συνέχεια χρησιμοποιούμε εκ νέου 3 for μέσα στις οποίες θα κληθούν οι μέθοδοι greedyTest και sortTest:

Η μέθοδος int greedyTest(String path) υλοποιείται στην κλάση Greedy και χρησιμοποιείται στην κλάση Test αποκλειστικά. Παίρνοντας σαν όρισμα το επιθυμητό αρχείο εισόδου, εκτελεί τον αλγόριθμο Greedy σε αυτό και επιστρέφει το μέγεθος της ουράς προτεραιότητας στην οποία αποθηκεύσαμε τους δίσκους που δημιουργήθηκαν κατά την εκτέλεσή του.

Ομοίως, η μέθοδος int sortTest(String path) υλοποιείται στην κλάση Sort και χρησιμοποιείται αποκλειστικά στην κλαση Test. Εκτελεί τον αλγόριθμο Sort στο επιθυμητό αρχείο που περνάμε ως όρισμα και επιστρέφει το μέγεθος της ουράς προτεραιότητας στην οποία αποθηκεύτηκαν οι δίσκοι κατά την εκτέλεσή του.

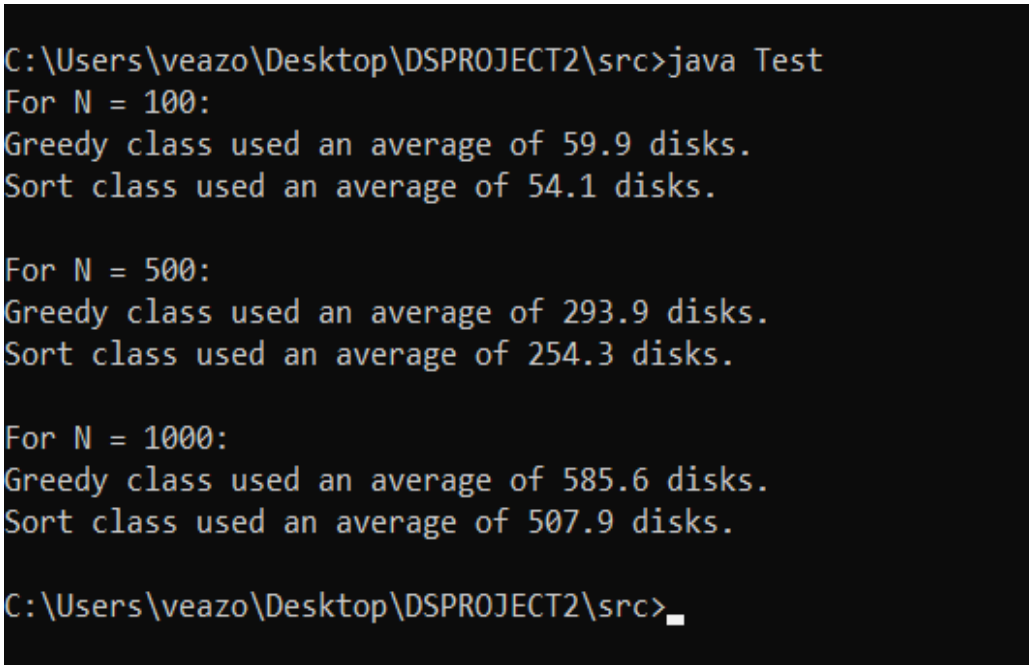
Στην πρώτη for, περνάμε στις μεθόδους τα δέκα αρχεία “folderi_100.txt” όπου το i παίρνει τιμές από 0 – 9. Έτσι, αποθηκεύουμε στη μεταβλητή greedyDisksUsed100 τους δίσκους που δημιούργησε ο αλγόριθμος Greedy και στη μεταβλητή sortDisksUsed100 τους δίσκους που δημιούργησε ο αλγόριθμος Sort.

Ακολουθείται η ίδια διαδικασία στη δεύτερη for για N = 500 και στην τρίτη για N = 1000.

Στο τέλος, τυπώνονται τα επιθυμητά μηνύματα και υπολογίζεται ο μέσος όρος των δίσκων που χρειάστηκε κάθε αλγόριθμος. Ο υπολογισμός του μέσου όρου γίνεται εάν διαιρεθεί το πλήθος των δίσκων που χρησιμοποίησε ο κάθε αλγόριθμος σε κάθε στάδιο με το πλήθος των ανάλογων αρχείων(10).

Παραθέτουμε τρία παραδείγματα εκτέλεσης πειραματικών αξιολογήσεων:

```
C:\Users\veazo\Desktop\DSPROJECT2\src>java Test
For N = 100:
Greedy class used an average of 58.9 disks.
Sort class used an average of 52.5 disks.

For N = 500:
Greedy class used an average of 295.5 disks.
Sort class used an average of 258.5 disks.

For N = 1000:
Greedy class used an average of 588.3 disks.
Sort class used an average of 509.0 disks.

C:\Users\veazo\Desktop\DSPROJECT2\src>java Test
For N = 100:
Greedy class used an average of 60.7 disks.
Sort class used an average of 54.9 disks.

For N = 500:
Greedy class used an average of 287.6 disks.
Sort class used an average of 252.4 disks.

For N = 1000:
Greedy class used an average of 582.6 disks.
Sort class used an average of 507.2 disks.

C:\Users\veazo\Desktop\DSPROJECT2\src>
```



Από τα αποτελέσματα της πειραματικής αξιολόγησης, συμπεραίνουμε ότι ο αλγόριθμος Sort χρησιμοποιεί σημαντικά λιγότερους δίσκους σε σύγκριση με τον αλγόριθμο Greedy. Αυτό συμβαίνει επειδή ο αλγόριθμος Greedy επεξεργάζεται τα δοσμένα αρχεία και τα αποθηκεύει σε δίσκους άτακτα. Από την αντίθετη πλευρά, ο αλγόριθμος Sort υλοποιεί ταξινόμηση των δοσμένων αρχείων σε φθίνουσα σειρά με βάση το μέγεθος και ύστερα τα κατανέμει σε δίσκους.

Αναπαράσταση των αποτελεσμάτων προς ευκολότερη μελέτη:

	Αλγόριθμος Greedy	Αλγόριθμος Sort
Πείραμα πρώτο		
N = 100	58.9	52.5
N = 500	295.5	258.5
N = 1000	588.3	509.0
Πείραμα δεύτερο		
N = 100	60.7	54.9
N = 500	287.6	252.4
N = 1000	582.6	507.2
Πείραμα τρίτο		
N = 100	59.9	54.1
N = 500	293.9	254.3
N = 1000	585.6	507.9