***Αναφορά Παράδοσης (εργασία 2)***

* Α

Για το μέρος Α χρησιμοποιήθηκε η ουρά προτεραιότητας του φροντηστηρίου με μία μικρή αλλαγή : επειδή τα δεδομένα με τα οποία δουλεύουμε δεν ειναι primitive πρεπει κάθε φορά που αλλάζουμε κάπως την ουρά να την ανανεώνουμε βάση του comparator. Γι΄αυτό δημιουργήθηκε μία extra μέθοδος update η οποία απλά βγάζει και ξαναβάζει όλα τα στοιχεία της ουράς.Χρησιμοποιήθηκαν και generics για να καλύπτει πολλαπλούς τύπους δεδομένων.

Ψευδοκώδικας για MaxPQ.update:

Copy this.heap to an array named heap

Copy this.size to an int named named size

While peek() is not null:

GetMax()

For object in heap:

Add(obj)

This.size = size

* Β

Η υλοποίηση του αλγορίθμου 1 είναι πολύ απλή (6 γραμμές κώδικα) χάρη στην ουρά προτεραιότητας.Συγκεκριμένα για να λύσουμε το πρόβλημα αρκεί να τρέξουμε τον παρακάτω ψευδοκώδικα (όπου folders οι φάκελοι προς αποθήκευση και disks η ουρά προτεραιότητας που κρατάει τους δίσκους με τα δεδομένα):

For folder in folders:

If no disk in disks has enough space to hold the folder or disks is empty:

add new disk to disks

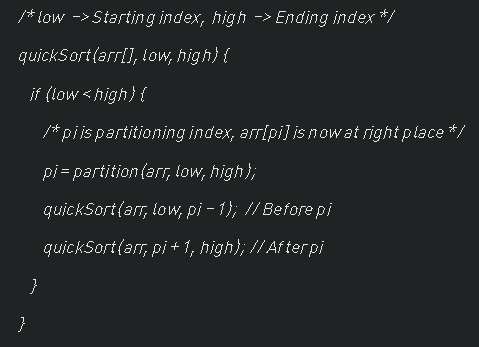
update disks

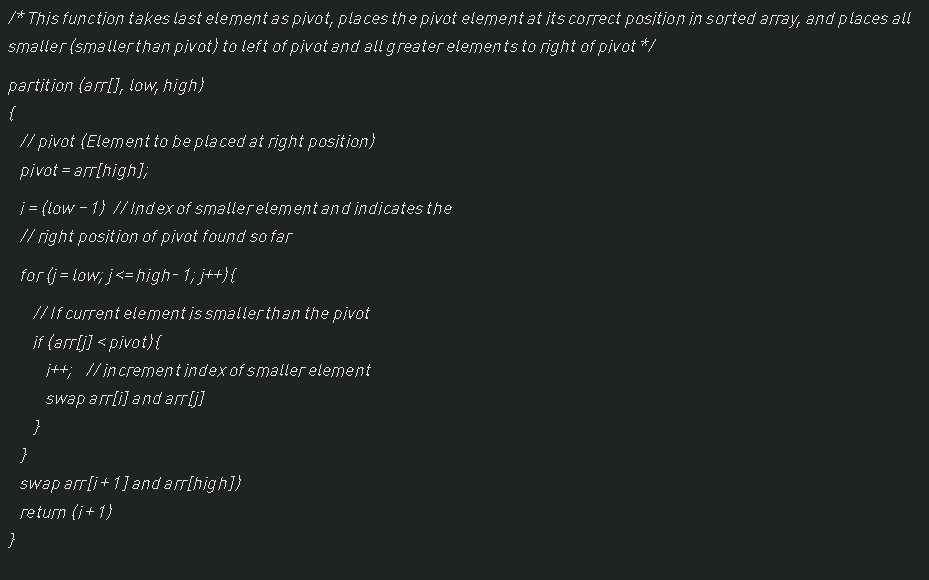
Add folder to the head of the disks priority queue

Update disks

* Γ

Για το μέρος Γ χρησιμοποιήθηκε ο αλγόριθμος quick sort.





* Δ

Η παραγωγή των δοκιμαστικών δεδομένων για το μέρος Δ έγινε με τη χρήση της κλάσης Random του πακέτου java.util. Συγκεκριμένα δοσμένου ενός πίνακα Ν ο οποίος περιέχει τα διάφορα ν για τα οποία θα δημιουργήσουμε δεδομένα (οπου ν ειναι το πλήθος φακέλων, ή αλλιώς το πλήθος των γραμμών σε κάθε αρχείο), καθώς και ενός αριθμου amount ο οποίος μας λέει πόσα αρχεία θα παράγουμε για κάθε ν θα πάρουμε τον παρακάτω ψευδοκώδικα:

Random r = new Random()

for n in N

for I in 0..amount

for j in 0..n

int generated\_number = r.nextInt(0, 1\_000\_001)

file.write(generated\_number + “\n”)

Τα ονόματα για ευκολία δίνονται από τον παρακάτω τύπο:

nNiI όπου Ν ειναι το πλήθος φακέλων-γραμμών για το αρχείο και το Ι είναι ο δείκτης του αρχείου για τη συγκεκριμένη κατηγορία

Αυτό έχει ως αποτέλεσμα στη συνάρτηση review η οποία θα τρέξει το benchmark με ενα nested for loop να μπορούμε να φτιάξουμε τα ονόματα όλων των αρχείων και για το κάθε ενα να καλούμε αντίστοιχα τους αλγορίθμους 1 και 2 απο το μέρος β και γ.

Ψευδοκώδικας:

For I in 0..N.length

For j in 0..amount

filename = “n” + N[i] + “i” + j;

Στον παραπάνω ψευδοκώδικα το Ν είναι ο πίνακας με το μέγεθος των δοκιμαστικών δεδομένων (πχ Ν=100,500,1000 από εκφώνηση) και το amount είναι το πλήθος αρχείων για κάθε ν στο Ν (πχ στην εκφώνηση 10).

Όσον αφορά τα ευρήματα του benchmark για Ν = 100, 500, 1000 και amount 100 παίρνουμε στα 10 runs ενα μέσο όρο απο 60 δίσκους για Ν = 100, 300 δίσκους για Ν = 500 και 600 δίσκους για Ν = 1000 στον αλγόριθμο 1. Για τον αλγόριθμο 2 οι αριθμοί αυτοί γίνονται 50, 250 και 500 αντίστοιχα.







Η απόδοση των 2 αλγορίθμων παρόλο που το πλήθος των δεδομένων είναι σχετικά μικρό, είναι φανερή και υπέρ του αλγορίθμου 2.