

# Δομές Δεδομένων 2019-20

## **Project 2:**

**Δημήτριος Δεληκούρας 3170034**

**Ευάγγελος-Ανδρέας Βενετσάνος 3180019**

### **Μέρος A:**

Υλοποίηση Ουράς Προτεραιότητας μέσω MaxPQ

Η ουρά προτεραιότητας υλοποιείται με χρήση σωρού.

**Υλοποίηση Constructor:** Αρχικοποίηση ενός πίνακα τύπου Disk με μέγεθος ίσο με τον μέγιστο αριθμό δίσκων που μπορεί να αποθηκεύσουμε στην ουρά προτεραιότητας.

**Υλοποίηση less:** Σύγκριση 2 δίσκων.

**Υλοποίηση insert:** Εισαγωγή ενός δίσκου στην ουρά και κλήση της μεθόδου swim.

**Υλοποίηση getmax:** Επιστροφή του δίσκου με τη μέγιστη προτεραιότητα και απομάκρυνσή του από την ουρά προτεραιότητας.

**Υλοποίηση swap:** Ανταλλαγή της θέσης 2 δίσκων.

**Υλοποίηση sink:** Κατάδυση του δίσκου στο σωρό αν είναι μικρότερος από τουλάχιστον ένα “παιδί” του.

**Υλοποίηση swim:** Ανάδυση του δίσκου στο σωρό αν είναι μεγαλύτερος από τον “πατέρα” του

**Υλοποίηση isEmpty:** Σύγκριση του αριθμού των αντικειμένων της ουράς προτεραιότητας με το 0.

**Υλοποίηση size:** Επιστροφή του πλήθους των στοιχείων της ουράς προτεραιότητας

### **Μέρος B:**

Υλοποίηση Του Αλγόριθμου Greedy μεσώ Greedy

Με την κατασκευή ενός αντικειμένου τύπου Greedy καλείται η μέθοδος readFolders, η οποία δημιουργεί την λίστα folder\_list με τα μεγέθη των φακέλων που θέλουμε να αποθηκεύσουμε, ενώ στην συνέχεια καλείται ο αλγόριθμος.

Ο αλγόριθμος παίρνει ως ορίσματα την λίστα folder\_list και την ουρά προτεραιότητας disk\_heap η οποία αρχικά δεν περιέχει κανένα δίσκο.

Όσο διαβάζονται μεγέθη φακέλων γίνεται σύγκριση τους με το αποτέλεσμα της μεθόδου getFreeSpace του δίσκου με τον μέγιστο ελεύθερο χώρο. Αρχικά ο σωρός είναι κενός οπότε δημιουργείται ένας

άδειος δίσκος ενώ στην συνέχεια για να βρούμε αυτόν με το max free space χρησιμοποιούμε την μέθοδο getmax της ουράς προτεραιότητας. Αν ο φάκελος που διαβάζεται χωράει στον δίσκο με τον μέγιστο ελεύθερο χώρο το τοποθετούμε σε αυτόν αλλιώς δημιουργούμε έναν καινούργιο δίσκο.

## Μέρος Γ:

### Υλοποίηση Του Αλγόριθμου Greedy-decreasing μέσω Greedy\_decreasing

Χρησιμοποιήθηκε η μέθοδος ταξινόμησης Quicksort για την οργάνωση των φακέλων.

**Υλοποίηση quicksort:** Δημιουργούμε έναν πίνακα στον οποίο τοποθετούμε τα μεγέθη των φακέλων που έχουν διαβαστεί από την μέθοδο readFolders του αρχείου Greedy\_decreasing.java. Έπειτα, καλούμε τη μέθοδο sort που κάνει ταξινόμηση στον folder\_array με τη μέθοδο Quicksort και επιστρέφουμε τον ταξινομημένο folder\_array.

Στην συνέχεια εκτελείται ο αλγόριθμος Greedy του Μέρους Β χρησιμοποιώντας την ταξινομημένη λίστα.

## Μέρος Δ:

Για την σύγκριση των αλγορίθμων 1 και 2 δημιουργούμε 10 αρχεία txt για 3 διαφορετικά πλήθη φακέλων: 100, 500 και 1000. Αυτά τα txt τοποθετούνται στον φάκελο data ο οποίος βρίσκεται στο ίδιο directory με τα αρχεία java.

Τα παραπάνω αρχεία έχουν την ακόλουθη ονομασία:

Path\_to\_folder/data/N\_rep.txt όπου N= 100, 500, 1000 και rep = 0, 1, ..., 9.

Και στο καθένα από αυτά προσθέτουμε “μεγέθη φακέλων” με την χρήση της κλάσης Random, ενώ στην συνέχεια εκτελούμαι τους Αλγορίθμους 1 και 2 με την δημιουργία των αντικειμένων greedy και greedy\_decreasing.

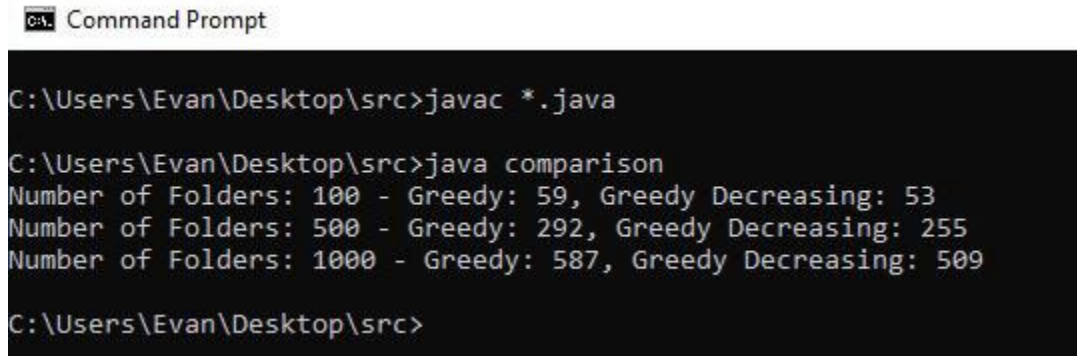
Παράλληλα με ένα switch statement βρίσκουμε σε ποιο “stage” ανήκει το κάθε αρχείο που μόλις δημιουργήθηκε και κρατώντας ένα ξεχωριστό μετρητή για τον συγκεκριμένο αλγόριθμο και πλήθος φακέλων (συνολικά θα υπάρχουν  $2 \times 3 = 6$  μετρητές), προσθέτουμε κάθε φορά το πλήθος των δίσκων που χρειάστηκαν για να αποθηκευτούν όλοι οι φάκελοι.

Οι μετρητές αυτοί πλέον, αν διαιρεθούν με το 10, μπορούν να μας δώσουν το μέσο πλήθος δίσκων που χρειάστηκε για κάθε εκτέλεση των δύο αλγορίθμων, αφού δημιουργήσαμε 10 αρχεία κάθε πλήθος φακέλων.

Εισάγουμε έπειτα τις τιμές αυτές σε μία λίστα με όνομα result list, προκειμένου να μπορέσουμε να εμφανίσουμε το αποτέλεσμα της σύγκρισης με περισσότερη ευκολία.

## Συμπέρασμα

Παρακάτω φαίνεται ένα ενδεικτικό αποτέλεσμα της σύγκρισης των δυο αλγορίθμων από το αρχείο comparison.java με  $N=100, 500, 1000$ .



```
C:\> Command Prompt

C:\Users\Evan\Desktop\src>javac *.java

C:\Users\Evan\Desktop\src>java comparison
Number of Folders: 100 - Greedy: 59, Greedy Decreasing: 53
Number of Folders: 500 - Greedy: 292, Greedy Decreasing: 255
Number of Folders: 1000 - Greedy: 587, Greedy Decreasing: 509

C:\Users\Evan\Desktop\src>
```

Βλέπουμε πως ο μέσος αριθμός δίσκων που χρειάζεται από την Greedy decreasing ανά πλήθος δίσκων, είναι χαμηλότερος από αυτόν της Greedy.

Όσο αυξάνουμε το πλήθος των φακέλων τόσο μεγαλύτερη είναι η διαφορά ανάμεσα στο μέσο πλήθος δίσκων που χρειάζονται οι δύο αλγόριθμοι.

Αυτό συμβαίνει λόγω της φθίνουσας ταξινόμησης των φακέλων πριν την ένταξη τους στους δίσκους. Έτσι αυξάνεται σημαντικά η απόδοση, δηλαδή η οικονομία δίσκων, για μεγάλα πλήθη φακέλων.