**Όνομα:** Αναστάσιος Παπαπαναγιώτου
**ΑΜ:** 3200143

## Ερώτημα Α

Μέσα στην ουρά προτεραιότητας έμπαιναν οι επεξεργαστές οι οποίοι είχαν τα δικά τους tasks και αποθηκεύουν μια μεταβλητή που κρατάει τον συνολικό χρόνο (άθροισμα χρόνων όλων των tasks που έχουν) και με βάση τον συνολικό χρόνο κατανεμεται κάθε επεξεργαστής στην ουρά προτεραιότητας. Τα task του κάθε επεξεργαστή αποθηκεύονται μέσα σε μια κανονική ουρά και αυτό γιατί τα tasks που μπαίνουν πρώτα στον επεξεργαστή πρέπει να τελειώνουν και πρώτα. Το αρχείο διαβάζει και αποθηκεύει τα δεδομένα σε ένα νέο Object που ονομάζεται FileContents και έχει όλα τα στοιχεία που διαβάζονται από το αρχείο για να χρησιμοποιηθούν μετά, συγκεκριμένα έχει: τα id και time κάθε task σε ξεχωριστά arrays, τον αριθμό των επεξεργαστών και των tasks, αν διαβάστηκε σωστά το αρχείο (δεν διαβάζεται σωστά αν έχει δοθεί λάθος path ή έχει περισσότερα/λιγότερα tasks). Για την υλοποίηση της PQ, χρησιμοποιείται η ίδια λογική με τις max heaps αλλά μεγαλύτερη προτεραιότητα έχουν οι επεξεργαστές με το μικρότερο active time δηλαδή max θα είναι πάντα το στοιχείο με το λιγότερο active time και αποθηκεύουμε ως μεταβλητή το makespan που είναι το μέγιστο active time όλων των επεξεργαστών. Η υλοποίηση της κανονικής ουράς γίνεται κανονικά, αποθηκεύει αντικείμενα τύπου Task, αλλά με κάθε enqueue προστίθεται ο χρόνος αυτού του task στο συνολικό χρόνο των task που είναι μέσα στην ουρά και ομοίως αφαιρείται με το dequeue και δύο τέτοιες ουρές είναι ίδιες αν κάθε task έχει την ίδια σειρά και το ίδιο id & time.

## Ερώτημα Β

Επέλεξα τον αλγόριθμο της quicksort και για την υλοποίησή του χρησιμοποίησα πίνακα που έχει στοιχεία τύπου Task και γίνονται sorting με βάση τον χρόνο τους (task.time) και επειδή θέλουμε να είναι σε decreasing order, ελέγχουμε κάθε φορά αν ο τωρινός χρόνος είναι μεγαλύτερος από το pivot για να φτιάξουμε τα partitions.

## Ερώτημα Γ

| | N = 100 | N = 250 | N = 500 |
|---|---|---|---|
| Makespan First Algorithm | 553.4 | 2207.9 | 5940.8 |
| Makespan Second Algorithm | 517.3 | 2085.5 | 5700.5 |

Παρατηρώ ότι σε για όλα τα N ο δεύτερος αλγόριθμος έχει λιγότερο makespan και αυτό γιατί οι αρχικοί επεξεργαστές θα γεμίσουν πρώτα με τα task που έχουν τον περισσότερο χρόνο ενώ πριν μπορεί να είχαμε δύο σχεδόν μεγάλα task μαζί. Μπορεί να υπάρχουν όμως και περιπτώσεις όπου ο πρώτος είναι καλύτερος (π.χ. TestingFiles/data2.txt).

## Ερώτημα Δ

Το TestingFiles περιέχει τα αρχεία που χρειάζονται για να γίνει έλεγχος της σωστής λειτουργίας. Επίσης το src περιέχει δύο .jar αρχεία που χρησιμοποιούνται για να τρέξω JUnit Tests από cmd και τα test περιέχονται στα αρχεία: AlgorithmTest.java, MaxPQTest.java, SortTest.java, τα οποία είναι test cases και τρέχουν στο cmd με τις εντολές (πρέπει να το τρέξω από το directory που είναι το src π.χ. ../…/…/src/):

- javac -cp junit-4.13.1.jar;. AlgorithmTest.java MaxPQTest.java SortTest.java
- java -cp junit-4.13.1.jar;hamcrest-core-1.3.jar;. org.junit.runner.JUnitCore AlgorithmTest MaxPQTest SortTest

Τα αρχεία που χρειάζονται όμως δουλεύουν κανονικά και με το cmd χωρίς JUnit. Π.χ. για το Greedy

- javac Greedy.java
- java Greedy “C:/Users/dir/dir/dir/file.txt”