Μέλη Ομάδας: Στασινού Ελπίδα – P3200277
Τζοβάνης Δημήτριος Ορέστης – P3200195

ΜΕΡΟΣ Ε

Α)

Ο αλγόριθμος 1 (Greedy) λειτουργεί διαβάζοντας ένα αρχείο μορφής txt. Το αρχείο αυτό διαβάζεται μέσω των args[], όπου το δέχεται filereader,και έπειτα ο bufferedreader για να το διαβάσει.
Έπειτα, η πρώτη σειρά του αρχείου διαβάζεται (δηλαδή ο αριθμός των επεξεργαστών) και αποθηκεύεται σε μια μεταβλητή. Με τον ίδιο τρόπο αποθηκεύεται ο αριθμός των task, και έπειτα τα ιδιά τα id και time του κάθε task. Εδώ πρέπει να επισημανθεί ότι ο αριθμός των task με τα στοιχεία που δόθηκαν για κάθε task συγκρίνονται ώστε να μην υπάρχουν περισσότερα ή λιγότερα. Επίσης, για κάθε σειρά στοιχείων task (id, time) , δημιουργείται ένα instance της κλάσης task και αποθηκεύεται σε ένα πίνακα tasks (Αυτός ο πίνακας μπορεί να ταξινομηθεί από τον αλγόριθμο 2).
Σε αυτό το σημείο του κώδικα δημιουργείται η ουρά προτεραιότητας, όπως και οι επεξεργαστές (έχουν id τον αριθμό σειράς που δημιουργήθηκαν), οι οποίοι εισέρχονται στην ουρά και ταξινομούνται με βάση το id.
Οι επεξεργαστές αφαιρούνται από την ουρά προτεραιότητας ένας ένας, και περνάνε τα tasks από τον πίνακα, μέσα στην λίστα των tasks του εκάστοτε επεξεργαστή (Η λίστα αυτή είναι της μορφής ουράς). Έπειτα ο επεξεργαστής επαναπροστήθεται στην ουρά και ταξινομείται ανάλογα με το id ή το συνολικό χρόνο των task που έχει επεξεργαστεί, ώστε να επεξεργαστεί τα επόμενα.
Τέλος, αν ο αριθμός των επεξεργαστών που υπάρχουν στην ουρά προτεραιότητας είναι μεγαλύτερος του 50, τότε βρίσκεται ο επεξεργαστής που έχει ξοδέψει τον περισσότερο χρόνο εκτελώντας tasks, και αυτός ο χρόνος αποθηκεύεται στην μεταβλητή makespan, η οποία εκτυπώνεται. Αν τώρα ο αριθμός των επεξεργαστών είναι μικρότερος του 50, τότε εκτός του να βρεθεί το makespan, ο αλγόριθμος εκτυπώνει τους επεξεργαστές έναν έναν, από το ελάχιστο συνολικό χρόνο επεξεργασίας που ξόδεψε προς τον μέγιστο (total time), το id, το συνολικό αυτό χρόνο, και το time του κάθε task που επεξεργάστηκε.

*****Φτιάξαμε δυο φορές τον κώδικα greedy, μέσα σε αυτό το αρχείο. Μια όπως ζητείται στην εκφώνηση (μέσα σε μια main), και μια με παραμέτρους ώστε να χρησιμοποιείται από τον τη κλάση comparisons του μέρους Δ (μέσα σε ένα constructor)**

Β)

Ο αλγόριθμος που χρησιμοποιήσαμε είναι η QuickSort, τον οποίο υλοποιήσαμε με χρήση πίνακα. Ο αλγόριθμος δέχεται έναν πίνακα με tasks, μια μεταβλητή που προσδιορίζει την αρχή, και μια το τέλος του. Μια μεταβλητή 'pivot' προσδιορίζεται που αντιστοιχεί σε ένα σημείο του πίνακα, και γίνεται αναδρομή στον πίνακα αυτό, από την αρχή του μέχρι το προηγούμενο του σημείου pivot, και από το επόμενο του pivot, μέχρι το τέλος του πίνακα.
Το pivot βρίσκεται ως εξής: Το τελευταίο στοιχείο του πίνακα επιλέγεται ως pivot (στην προκειμένη περίπτωση ο χρόνος του τελευταίου task), και έπειτα όλα τα στοιχεία του πίνακα διατρέχονται και συγκρίνονται (με βάση το time του task), ώστε τα στοιχεία μικρότερα του pivot να πάνε αριστερά του, ενώ αυτά μεγαλύτερα δεξιά του.

Γ)

Τρέξαμε τον αλγόριθμο Comparisons για N=100, N=250, N=500 με τον αλγόριθμο 1 και 2

Αυτά είναι τα συμπεράσματά μας:

N	100	250	500
Αλγόριθμος 1	5393	8732	11927
Αλγόριθμος 2	5007	8234	11444
Ποσοστό Μείωσης	7,1%	5,7%	4,04%

Φανερά ο αλγόριθμος 2 είναι η καλύτερη επιλογή, αφού ο συνολικός χρόνος είναι πάντα μικρότερος (δηλαδή το makespan) από το συνολικό του αλγορίθμου 1

Όμως όσο μεγαλύτερος είναι ο αριθμός Ν, τόσο μικρότερο είναι το ποσοστό βελτίωσης. Δηλαδή τόσο μικρότερη η διαφορά μεταξύ των δύο αλγορίθμων

Δ)

Το μονοπάτι για το αρχείο εισόδου δίνεται μέσω του args[] και για τον αλγόριθμο greedy όπως και για τον comparisons.
****ΠΡΟΣΟΧΗ** για τον comparisons δίνετε το μονοπάτι εισόδου του φακέλου με τα txt αρχεία στο args[].
Για την δημιουργία αρχείων βάζετε το path στη γραμμή 61.

Επίσης στο comparisons στη γραμμή 6 υπάρχει σε σχόλια η εντολή που δημιουργεί τα ζητούμενα αρχεία.