Χρήστος Ξυδέας (Αρ. Μητρώου: 3160124)
Δικαία Σωτηροπούλου (Αρ. Μητρώου: 3160172)

- Ουρά Προτεραιότητας

Δημιουργούμε έναν πίνακα τύπου *Processor* με όνομα *listOfPro* και μέγεθος ίσο με αυτό των επεξεργαστών. Στη συνέχεια δημιουργούμε μια ουρά προτεραιότητας τύπου *MaxPQ* με όνομα *listOfProcessors* και μέγεθος ίσο με τους επεξεργαστές. Επειδή η ουρά αυτού του τύπου έχει σαν ρίζα της το μέγιστο στοιχείο που ανήκει στην ουρά, την αντιστρέφουμε. Γεμίζω τον πίνακα *listOfPro* με τα πρώτα *txt.processors* στοιχεία με τα δεδομένα από τον πίνακα *txt.time,* του οποίου τα στοιχεία έχω πάρει από το διάβασμα του αρχείου txt. Μέσα στην ουρά *listOfProcessors* προσθέτουμε τον ενεργό χρόνο των επεξεργαστών που παίρνουμε από τον πίνακα *listOfPro*. Έπειτα, σαρώνουμε τον πίνακα των επεξεργαστών, και μόλις βρούμε αυτόν με το μικρότερο ενεργό χρόνο (ο οποίος είναι ίσος με το *listOfProcessors.getMax()*), τον προσθέτουμε στην *listOfProcessors*, ενημερώνοντας κατάλληλα και τον πίνακα *listOfPro*.

- Διάβασμα Αρχείου (readTxt.java)

Ορίζουμε τη μέθοδο *openFile(Scanner file)* για να ανοίξουμε και να διαβάσουμε τα δεδομένα του txt αρχείου *file*. Η μεταβλητή *lines* είναι ένας δείκτης για να μας δείχνει κάθε φορά σε ποια σειρά του αρχείου βρισκόμαστε (τη χρειαζόμαστε κυρίως για να πάρουμε τα δεδομένα των επεξεργαστών (*processors*) και των διεργασιών (*procedures*)). Η μεταβλητή τύπου *wrongFormat* θα πάρει την τιμή true όταν το αρχείο δεν έχει τη μορφή που θέλουμε, αν δηλαδή υπάρχει ένα αλφαριθμητικό δεδομένο κάπου στο αρχείο κειμένου μας. Στη συνέχεια, ελέγχουμε τα δεδομένα του αρχείου γραμμή προς γραμμή. Η δομή επανάληψης *while* σταματάει όταν πλέον δεν υπάρχουν άλλα στοιχεία στο txt. Αποθηκεύω τα στοιχεία της εκάστοτε γραμμής σε μια μεταβλητή με όνομα *nextData*, τα οποία και ελέγχουμε με τη μέθοδο *check(String s)* για να δούμε αν τα δεδομένα αυτά είναι τύπου int. Αν δεν γίνει επιτυχώς η μετατροπή από String σε integer, τότε η μεταβλητή wrongFormat γίνεται true και το πρόγραμμα τερματίζεται. Αν η μετατροπή γίνει με επιτυχία, αποθηκεύουμε τα δεδομένα στις μεταβλητές *processors* (δεδομένα 1ης σειράς), *procedures* (δεδομένα 2ης σειράς) και στον πίνακα *time* (όλα τα υπόλοιπα δεδομένα).

Χρήστος Ξυδέας (Αρ. Μητρώου: 3160124)
Δικαία Σωτηροπούλου (Αρ. Μητρώου: 3160172)

- Ταξινόμηση

Επιλέξαμε να υλοποιήσουμε τον αλγόριθμο *HeapSort*, καθώς μπορέσαμε να κάνουμε χρήση της ουράς προτεραιότητας και η καταφέραμε να κρατήσουμε την πολυπλοκότητα του αλγορίθμου σε σχετικά χαμηλά επίπεδα.
Αρχικά, διαβάζουμε το αρχείο για να αποθηκεύσουμε τα δεδομένα. Δημιουργούμε έναν πίνακα τύπου int με όνομα *sortedList* μεγέθους ίσο με *txt.procedures* μια ουρά προτεραιότητας με όνομα *list* και μέγεθος ίσο με *txt.procedures*. Αποθηκεύουμε στην ουρά προτεραιότητας τους χρόνους των διεργασιών (από τον πίνακα *time*). Γνωρίζουμε ότι η ρίζα των ουρών με τύπο MaxPQ μας δίνει το μέγιστο στοιχείο και ότι η ταξινόμηση που θέλουμε είναι φθίνουσα. Άρα, στον πίνακα αποθηκεύουμε το *list.getMax()* στοιχείο της *sortedList* στην i θέση του πίνακα. Με αυτό τον τρόπο μπορούμε να ταξινομήσουμε τους χρόνους των διεργασιών.

- Μέρος Δ (Comparisons.java)

Στο τελευταίο μέρος της εργασίας, συγκρίνουμε τους αλγορίθμους *Greedy* (μέρος Β) και *Greedy-decreasing*, ο οποίος είναι ο αλγόριθμος του μέρους Β με ταξινομημένο τον πίνακα με τους χρόνους διεργασιών σε φθίνουσα σειρά. Η ταξινόμηση αυτή έγινε με τη χρήση του αλγορίθμου ταξινόμησης που φτιάξαμε στο μέρος Γ.
Δημιουργούμε 2 μεταβλητές τύπου int με τα ονόματα *momakespanwithsort* και *momakespanwithoutsort*, κάθε μια απ' τις οποίες είναι το άθροισμα των makespan με και χωρίς την ταξινόμηση αντίστοιχα. Οι μεταβλητές *mosort* και *monosort* είναι ο μέσος όρος των makespan για κάθε αριθμό διεργάσιων (για 100, 500 και 1000).

Παραθέτουμε το πινακάκι με τα αποτελέσματα των μέσων όρων των δύο αλγορίθμων.

N	**makespan without sort**	**makespan with sort**
100	5259	486
500	16722	1174
1000	32722	1604

Χρήστος Ξυδέας (Αρ. Μητρώου: 3160124)
Δικαία Σωτηροπούλου (Αρ. Μητρώου: 3160172)

Το διάγραμμα με τα δεδομένα του παραπάνω πίνακα