Μέλη Ομάδας:

Σαρκίρη Φωτεινή

Συκιώτη Κυριακή

Ερώτημα Α

Για την αναπαράσταση του προβλήματος γράψαμε τις κλάσεις:

* Order, η οποία αναπαριστά την κάθε μία από τις 100 παραγγελίες με μεταβλητές στιγμιοτύπου
* ID
* Quantity
* Dark
* isPlaced: αρχικά είναι false γιατί καμία παραγγελία δεν έχει τοποθετηθεί στην αντίστοιχη μηχανή και κάθε φορά που τοποθετείται μία παραγγελία η τιμή της γίνεται true
* Machine, η οποία αναπαριστά την κάθε μία από τις 5 μηχανές με μεταβλητές στιγμιοτύπου
* Machineorders: μία λίστα από αντικείμενα Order που δείχνει ποιες παραγγελίες και με ποια σειρά έχουν τοποθετηθεί στην συγκεκριμένη μηχανή
* Time: ο συνολικός χρόνος που απαιτείται για να ολοκληρωθούν όλες οι παραγγελίες της μηχανής
* Solution
* Machines: μία λίστα από αντικείμενα Machine
* Totaltime: ο μέγιστος αριθμός μεταξύ των time των 5 μηχανών, καθώς μετά από αυτόν τον χρόνο θα είναι ελεύθερες όλες οι μηχανές για την επόμενη παρτίδα
* Maxmachine: το αντικείμενο της μηχανής με τον μεγαλύτερο χρόνο

Για την γέννηση των input χρησιμοποιήθηκε η μέθοδος CreateOrders και CreateTimes που βρίσκονται στην κλάση Greedy2 και δημιουργούν μία λίστα(orders) από τις 100 παραγγελίες και έναν πίνακα(transitionTime) από τον χρόνο που απαιτείται για την εναλλαγή σε μία άλλη παραγγελία αντίστοιχα.

Ερώτημα Β

Η συνάρτηση CalculateObjectiveFunction δέχεται μία λύση τους προβλήματος και επιστρέφει την τιμή της αντικειμενικής συνάρτησης όπου είναι ο μεγαλύτερος χρόνος μεταξύ των μηχανών, και δηλαδή ο ελάχιστος χρόνος που απαιτείται για την έναρξη της επόμενης παρτίδας.

Στην μέθοδο αυτή, για κάθε μηχανή υπολογίζει τον συνολικό χρόνο για να ολοκληρωθούν οι παραγγελίες της. Υπολογίζει το συνολικό χρόνο της εκτέλεσης των παραγγελιών της μηχανής (quantity\*6/60(Μετατροπή second σε λεπτά)+χρόνος καθαρισμού μηχανής).

Ερώτημα Γ

Μορφή Λύσης: 5 διατάξεις παραγγελιών σε μηχανές

Στοιχείο Λύσης: μία σύνδεση 2 χρωμάτων

Κριτήριο Επιλογής: Σε κάθε επανάληψη προστίθεται η παραγγελίας που οδηγεί στην μικρότερη αύξηση του χρόνου.

-Αρχικά βρίσκουμε το μικρότερο αριθμό από τον πίνακα finalTimes και τοποθετούμε την σύνδεση των 2 αυτών χρωμάτων στην πρώτη μηχανή.

-αν υπάρχει έστω και μια κενή μηχανή, βρίσκουμε το μικρότερο χρόνο του πίνακα λαμβάνοντας υπ’ όψη τα isplaced και τους μικρότερους χρόνους από τον τελευταίο κόμβο της κάθε μηχανής, τα συγκρίνουμε και προσθέτουμε την μικρότερη σύνδεση από αυτές.

- Αν καμία μηχανή δεν είναι κενή, συγκρίνουμε μόνο τις τιμές από τους τελευταίους κόμβους της κάθε μηχανής και προσθέτουμε τον μικρότερο γείτονα.

Δημιουργήσαμε τον πίνακα finalTimes, στον οποίο αναπαρίστανται οι τελικοί χρόνοι. (Προσθέσαμε +15 λεπτά στην διαδικασία καθαρισμού όπου χρειαζόταν 🡪 από dark σε ανοιχτό)

CalculateObjectiveFunction: δέχεται την αρχική λύση που δεν περιέχει τίποτα και δημιουργεί την τελική λύση που αντιστοιχεί στον μικρότερο χρόνο.

Ερώτημα Δ:

Επιλέξαμε σαν τύπο κίνησης το 1-1 Exchange.

Για κάθε μηχανή, βρίσκουμε την καλύτερη αντιμετάθεση των παραγγελιών της, υπολογίζουμε τον χρόνο και τον συγκρίνουμε με τον προηγούμενο της. Όσο αυτός ο χρόνος είναι μικρότερος, συνεχίζουμε την επανάληψη και εκτελούμε τις ίδιες εντολές.

Όταν τελειώσουν όλες οι επαναλήψεις, έχουμε βρει τους βελτιωμένους χρόνους της κάθε μηχανής και από αυτούς τους χρόνους λαμβάνουμε υπό όψη τον μεγαλύτερο. Ο χρόνος αυτός αποτελεί την νέα αντικειμενική μας συνάρτηση.