REPORT

Team Members =>
Rishabh Roy (19ME10097)
Neil Prasanna Butala (19ME10100)
Rahul Choudhary (19AE10024)
SALMAAN AHMED ANSARI (18BT10017)

1)OPTIMIZATION METHOD

Depth First Search METHOD

Depth for search algorithm- the algorithm starts at a node of a graph and goes as far as possible in a given path, then backtracks until it finds an unexplored path, and then explores the rest of it. The process is repeatedly followed until all the nodes in the graph are explored.

For optimization, we have implemented DFS to find all possible paths between source and destination of all cars and then compared the max<tr> to get find the optimal path which has minimum max<tr>.The code is explained as follows

1) We first initialize the best_path_time = infinity
2) All possible sets of paths which lead each car from their source to destination are calculated using the function all_paths_combinations.
3) Then we choose a random set of paths from all possible sets.
4) The cars follow their respective path and reach their destination.
5) The tr of each car is noted. If max(tr) is less than max(tr) of best_path_time then the best_path_time is updated to this current path time and best_path is updated to the current set of paths.
6) Step 3-5 are repeated until all sets are explored

When the algorithm is runned more than once it may provide results i.e it may produce different path then from previous run but the algorithm is successful each time in the achieving the objective i.e. minimum of max(tr). The different results are due to the reason that initially a path is chosen randomly from all possible combinations of all possible paths of each car which can be different. Since the objective of algorithm is minimize max(tr) and minimize tr of each other the results obtained on each run of algorithm can be different.

PRUNING METHOD

Now, we know that the time complexity of the of DFS can be very high so to reduce this and also prevent unnecessary computations we also use pruning in DFS

Here pruning happens when the tr of any car exceeds the minimum max(tr) found untill now by algorithm.

Here also one can obtain different path due to reasons stated previously

How the code works=>

1) We first initalize the best_path_time = infinity
2) All possible sets of paths which leads each car from their source to destination is are calculated using function all_paths_combinations.

3) Then we choose a random set of paths from all possible sets.
4) The cars follow the their respective path and reach their destination. But while this happens if tr of any car exceeds the max(tr) of best_path_time then all the set is prunned and algorithms proceed with another set.

5) If all cars successfully each their destinations then the tr of each car is notted. If max(tr) is less than max(tr) of best_path_time then the best_path_time is updated to this current path time and best_path is updated to the current set of paths.
6) Step 3-5 are repeated untill all sets are explored

TEST CASES

TEST CASE 1

FIRST RUN

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 6 70 80 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10 --Pruning 0
By DFS Method:
Best path: [[2, 6, 7], [1, 0, 4, 5], [0, 1, 2, 6]]
Best path Time: [9.0, 6.0, 9.0]
Max Tr: 9.0

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 6 70 80 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10
By Prunning Method:
Best path: [[2, 6, 7], [1, 0, 4, 5], [0, 1, 2, 6]]
Best path Time: [9.0, 6.0, 9.0]
Max Tr: 9.0

SECOND RUN

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 6 70 80 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10 --Pruning 0
By DFS Method:
Best path: [[2, 6, 7], [1, 2, 6, 5], [0, 1, 2, 6]]
Best path Time: [9.0, 7.000000000000001, 9.0]
Max Tr: 9.0

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 6 70 80 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10
By Prunning Method:
Best path: [[2, 6, 7], [1, 2, 6, 5], [0, 1, 2, 6]]
Best path Time: [9.0, 7.000000000000001, 9.0]
Max Tr: 9.0

TEST CASE 2

FIRST RUN

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 8 70 85 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10 --n 2 8 55 --n 6 8 20 --n 3 8 35 --Pruning 0
By DFS Method:
Best path: [[2, 6, 7], [1, 2, 3, 6, 5], [0, 3, 8]]
Best path Time: [9.0, 10.0, 9.333333333333334]
Max Tr: 10.0

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 8 70 85 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10 --n 2 8 55 --n 6 8 20 --n 3 8 35
By Prunning Method:
Best path: [[2, 6, 5, 7], [1, 2, 8, 6, 5], [0, 3, 8]]
Best path Time: [10.0, 9.5, 10.0]
Max Tr: 10.0

SECOND RUN

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 8 70 85 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10 --n 2 8 55 --n 6 8 20 --n 3 8 35 --Pruning 0
By DFS Method:
Best path: [[2, 6, 7], [1, 2, 6, 5], [0, 3, 8]]
Best path Time: [9.0, 10.0, 7.000000000000001]
Max Tr: 10.0

python3 Optimal_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 8 70 85 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10 --n 2 8 55 --n 6 8 20 --n 3 8 35
By Prunning Method:
Best path: [[2, 6, 7], [1, 0, 4, 5], [0, 3, 8]]
Best path Time: [9.0, 10.0, 6.0]
Max Tr: 10.0
We can see that the result path is different in first run and second run this is as explained before if we run the code the algorithm may provide results with different path then from previous run but the algorithm is successful each time in the achieving the objective i.e. minimum of max(tr).
As the number of cars increases the number of paths will increase and also runtime increases so the process will get slower

2)HEURISTIC METHOD

We have used minimum distance as the heuristic.And the code is explained as follows
Using a loop, in each iteration a car is taken from the list and one move is made using the following steps:
(a)if current node is same as destination node, the node is added to path and the car is removed from list of cars
(b)The node which is at minimum distance to the current node is chosen
(c)using enough_battery function , it is determined whether it is possible for the car to go to chosen node or not
(d)If it is not possible, then using check_cars function, it is determined whether there are other cars charging there or not
(e)If there are , using wait function the ng time is added to total time of car
(f)using charging function, the charging time is added to total time of car

TEST CASES

CASE 1
python3 Heuristic_Algorithm.py --num_cars 3 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 6 70 80 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10
By Heuristic Method:
Best path: [['2', 3, 2, 6, 5, 7], ['1', 0, 1, 2, 6, 5], ['0', 1, 0, 4, 5, 7, 6]]
Best path Time: [13.5, 8.333333333333334, 20.0]
Max Tr: 20.0

CASE 2
python3 Heuristic_Algorithm.py --num_cars 4 --c 0 5 80 90 1 2 15 --c 2 7 80 100 1 2 10 --c 1 5 75 80 2 3 15 --c 0 7 70 80 4 2 10 --n 6 7 60 --n 5 7 40 --n 5 6 25 --n 4 5 50 --n 3 6 45 --n 3 4 50 --n 0 3 65 --n 2 3 20 --n 2 6 30 --n 0 4 30 --n 1 2 50 --n 0 1 10
By Heuristic Method:
Best path: [['0', 1, 0, 4, 5], ['1', 0, 1, 2, 6, 5], ['2', 3, 2, 6, 5, 7], ['0', 1, 0, 4, 5, 7]]
Best path Time: [6.666666666666666, 8.333333333333334, 13.5, 14.0]
Max Tr: 14.0