UNIVERSITY OF MALAYA

WIA1002 Data Structure

‘Always On Time’ Delivery (Technical Report)

Group Name: BEES

Tutorial Group: 4

Lecturer’s Name: Dr. Mohd Hairul Nizam Bin Md Nasir

Members:

No.	Name	Matric Number
1.	Aisyah binti Shaharol Munir	17206198/2
2.	Hanan binti Muhammad Nizam	17206018/2
3.	Nursyazwani Binti Mohammad	17206746/2
4.	Nurul Filzah Binti Abdul Hadi	17205112/2

Allocation of Tasks

Basic Requirements:

1. TourRoute, Vehicle, DepotInfo and MyCustomer class
2. Read data from file
3. Basic Simulation - Depth-first Search traversal implementation
4. Greedy Simulation - Greedy Search implementation
5. MCTS Simulation - Monte Carlo Tree Search algorithms

Extra Features:

1. Extra Algorithm Implementation - Best First Search

Requirement of the tasks

Basic Simulation

Simulation to find the best tour for a given case with small number of customers (N) and maximum capacity of all vehicles, C.

Greedy Simulation

Simulation that implements a Greedy Search that is able to find a good solution given a case with small or large N. Greedy Search means we simply look for the best option in the next move when we travel through the whole graph. For illustration, in this context the vehicle will always look for the shortest distance between current location and all next possible locations to select the next location to go.

MCTS Simulation

Simulation to search for the best tour that could find the best way in a limited computation time. This means that if we are given enough time, then we will provide the best tour, else we will just provide the best tour we could find if we are given a limited time with a large N. Thus, we are going to implement another searching algorithm, which is known as Monte Carlo Tree Search.

Best First Search Simulation

Simulation that uses a heuristic function to help search for the best tour where the estimated time to reach the goal is the heuristic function which allows planning the route easily. This search picks the best node as the next node to expand according to some rules called a heuristic and it uses Priority Queue to search the next node (delivery place).

Approach Taken to Solve Tasks Given

Netbeans

- Main platform for project development.

Github

- Reading guidelines in the NeverOnTimeSdnBhd repository.
- Store the sample of inputs and outputs.

Extra reading

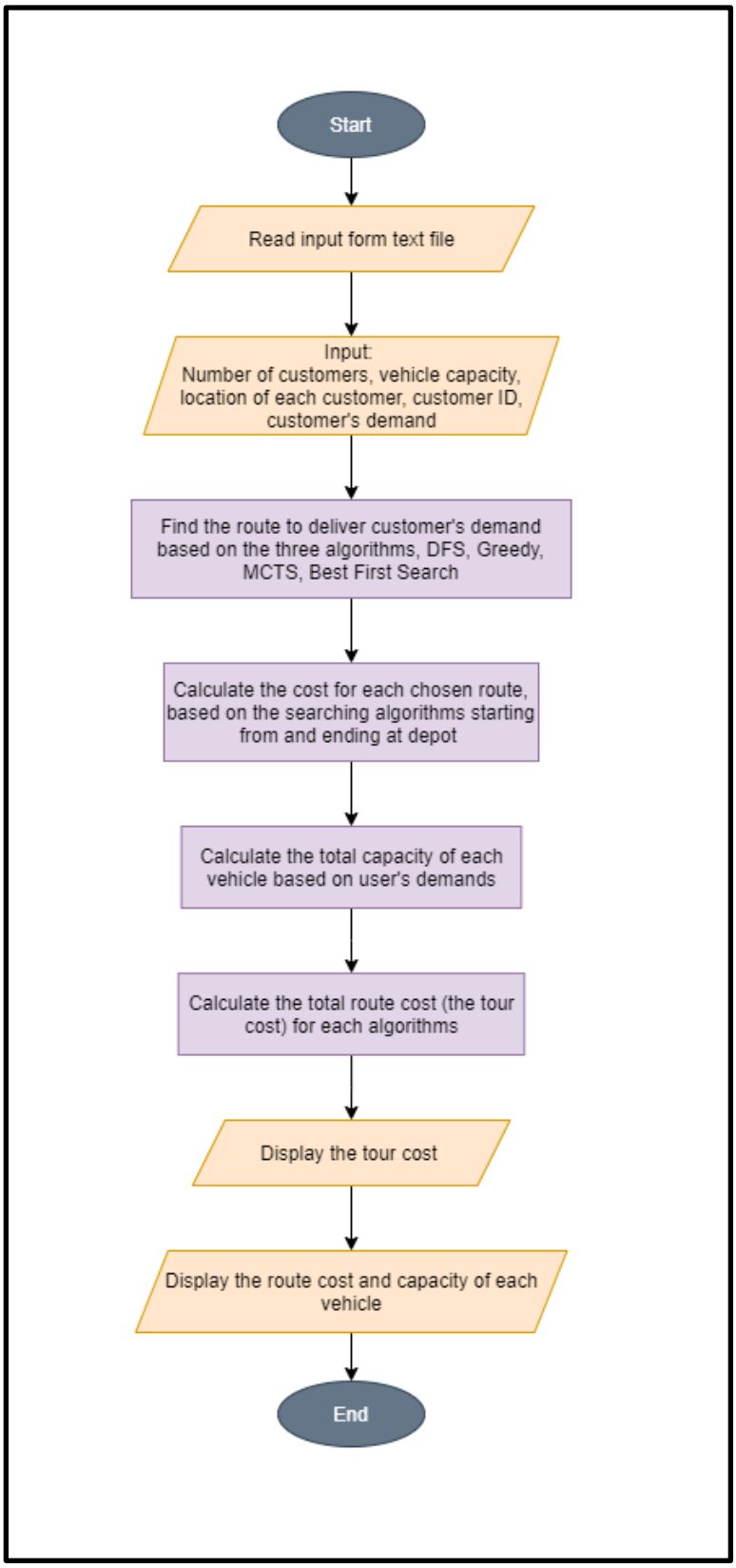
- Lecture Note
- YouTube
- Trusted websites on Google

Solution

IPO Table

Input	Process	Output
These data will be read from text file : ● Number of customer,N ● Maximum capacity of all vehicles,C ● X-coordinate ● Y-coordinate ● Demand size	Step 1 : Create class DataFromTextFile to accept input from file Step 2: Create class of Vehicle,DepotInfo, MyCustomer,TourRoute Step 3: Create class for graph simulation ● Basic Simulation ● GreedySearchSimulation ● MonteCarloTreeSearch Step 4: Add extra features ● BestFirstSearch Step 5: Test program	● Basic Simulation Output ● Greedy Search Simulation Output ● MonteCarlo Tree Search Output

Flow Chart


Start
Read input form text file
Input:
Number of customers, vehicle capacity, location of each customer, customer ID, customer's demand
Find the route to deliver customer's demand based on the three algorithms, DFS, Greedy, MCTS, Best First Search
Calculate the cost for each chosen route, based on the searching algorithms starting from and ending at depot
Calculate the total capacity of each vehicle based on user's demands
Calculate the total route cost (the tour cost) for each algorithms
Display the tour cost
Display the route cost and capacity of each vehicle
End

Sample Output

1. The details of customers and depot.

```
Number of Customer(s): 9

Customer and their demand
Depot (22, 190) : 0
Customer 1 (23, 101) : 14
Customer 2 (54, 34) : 15
Customer 3 (55, 62) : 5
Customer 4 (28, 101) : 32
Customer 5 (137, 8) : 27
Customer 6 (148, 152) : 42
Customer 7 (113, 131) : 24
Customer 8 (175, 178) : 39
Customer 9 (144, 22) : 34
Capacity of each vehicles: 43
```

2. Basic simulation output

```
Basic Simulation
Tour
Tour Cost: 1963.3626027611429
Vehicle 1
0 -> 5 -> 2 -> 0
Capacity: 42
Cost: 461.51342131561523
Vehicle 2
0 -> 1 -> 3 -> 7 -> 0
Capacity: 43
Cost: 338.0451474000387
Vehicle 3
0 -> 4 -> 0
Capacity: 32
Cost: 178.4040358287895
Vehicle 4
0 -> 6 -> 0
Capacity: 42
Cost: 263.21094202179364
Vehicle 5
0 -> 8 -> 0
Capacity: 39
Cost: 306.9397334982879
Vehicle 6
0 -> 9 -> 0
Capacity: 34
Cost: 415.24932269661804
Vehicle 7
0 -> 0
Capacity: 0
Cost: 0.0
```

3. Greedy simulation Output

```
Greedy Simulation
Tour
Tour Cost: 2138.005597521426
Vehicle 1
0 -> 1 -> 3 -> 2 -> 0
Capacity: 34
Cost: 326.7196961741431
Vehicle 2
0 -> 4 -> 0
Capacity: 32
Cost: 178.4040358287895
Vehicle 3
0 -> 7 -> 0
Capacity: 24
Cost: 216.90550938138938
Vehicle 4
0 -> 6 -> 0
Capacity: 42
Cost: 263.21094202179364
Vehicle 5
0 -> 8 -> 0
Capacity: 39
Cost: 306.9397334982879
Vehicle 6
0 -> 9 -> 0
Capacity: 34
Cost: 415.24932269661804
Vehicle 7
0 -> 5 -> 0
Capacity: 27
Cost: 430.5763579204042
```

4. MCTS simulation output

```
MCTS Simulation
Tour Cost: 1939.7805837686142
Vehicle 1
0 -> 9 -> 3 -> 0
Capacity: 39
Cost: 437.3857488081559
Vehicle 2
0 -> 7 -> 1 -> 0
Capacity: 38
Cost: 292.32670229597255
Vehicle 3
0 -> 2 -> 5 -> 0
Capacity: 42
Cost: 461.51342131561523
Vehicle 4
0 -> 4 -> 0
Capacity: 32
Cost: 178.4040358287895
Vehicle 5
0 -> 6 -> 0
Capacity: 42
Cost: 263.21094202179364
Vehicle 6
0 -> 8 -> 0
Capacity: 39
Cost: 306.9397334982879
```

5. Best First Search simulation output

```
Best First Search Simulation
Tour
Tour Cost: 2086.8729759954713
vehicle 1
Capacity: 42
0 -> 6 -> 0
Cost: 263.21094202179364
vehicle 2
Capacity: 39
0 -> 8 -> 0
Cost: 306.9397334982879
vehicle 3
Capacity: 37
0 -> 4 -> 3 -> 0
Cost: 268.82165856974285
vehicle 4
Capacity: 34
0 -> 9 -> 0
Cost: 415.24932269661804
vehicle 5
Capacity: 39
0 -> 7 -> 2 -> 0
Cost: 381.2351244257295
vehicle 6
Capacity: 41
0 -> 5 -> 1 -> 0
Cost: 451.4161947832993
```