Report CourseWork1:

Self-Marking Sheet:

**Marking scheme**:

|  |  |
| --- | --- |
| **Points** | **Area** |
| 10 | Self Marking Sheet |
| 10 | Solve First Training Problem |
| 10 | Get Optimal Result for All Three Training Problems. |
| 8 | Describe Algorithm(s) Used |
| 8 | Quality of Code |
| 20 | Get Optimal Results for the First Three Tests |
| 20 | Get Optimal Results for First Three Tests in under a minute. |
| NOT ATTEMPTED | Best system on Fourth Test (Path length times time.) |

Algorithm :

Let’s consider a starting city A. we can store the optimal distance from A to all other city in the graph. [ Done using adjacency Matrix for e.g., adjacency Matrix[X][Y] will represent the distance between city X and city Y]. This will solve the TSP problem with exactly 2 City.

Now let’s consider 3 cities, we need to store set Of Visited Cities and last Visited City from sub case of 2 City.

To represent Visited City bit Representation is used. For e.g., let consider 4 city TSP problem if a visited cities are 1 and 4 the representation will be [1001]. And we can store this in table (2d Array)

Now from last visited City the algorithm is expanded to visit all other unvisited City and we consider the path with the least Cost. That City gets added to last visited City. For a N city problem there will be N possible City that can be visited Last that gives 2ⁿ possible subsets of visited Noes

And repeat this until all the Cities are marked visited.

At last, we Reconnect the Tour to the start City

For a TSP for N city. We assign a start City. The algorithm starts by calculating the each set of cities from 2 to N excluding marked as the stating City. we can store the optimal distance from A to all other city in the graph. [ Done using adjacency Matrix for e.g., adjacency Matrix[X][Y] will represent the distance between city X and city Y]. This will solve the TSP problem with exactly 2 City.

Lets consider a 3 city problem with city 1 as start city. The shortest path will be given by route({2},3) thus {1->2->3} the path{1->3->2} will be the same for our problem the distance calculated by Euclidian formula will be same.

Once {\displaystyle S}contains three or more cities, the number of paths through {\displaystyle S}rises quickly, but only a few such paths need to be examined to find the shortest. For instance, if {\displaystyle 1\rightarrow 2\rightarrow 3\rightarrow 4}is shorter than {\displaystyle 1\rightarrow 3\rightarrow 2\rightarrow 4}, then {\displaystyle 1\rightarrow 2\rightarrow 3\rightarrow 4\rightarrow 5}must be shorter than {\displaystyle 1\rightarrow 3\rightarrow 2\rightarrow 4\rightarrow 5}, and the length of {\displaystyle 1\rightarrow 3\rightarrow 2\rightarrow 4\rightarrow 5}is not a possible value of {\displaystyle g(\{2,3,4\},5)}. Similarly, if the shortest path from {\displaystyle 1}through {\displaystyle \{2,3,4\}}to {\displaystyle 5} is {\displaystyle 1\rightarrow 4\rightarrow 3\rightarrow 2\rightarrow 5}, and the shortest path from {\displaystyle 1}through {\displaystyle \{2,3,4,5\}}to {\displaystyle 6}ends with the edge {\displaystyle 5\rightarrow 6}, then the whole path from {\displaystyle 1}to {\displaystyle 6}must be {\displaystyle 1\rightarrow 4\rightarrow 3\rightarrow 2\rightarrow 5\rightarrow 6}, and not any of the other five paths created by visiting {\displaystyle \{2,3,4\}}in a different order.

The algorithm finishes when route({2,….,k},i) is know for all the every city until N.

Problems :

For data over 26 City Java returns a Heap Storage out of Memory Error.

INSTRUCTION :

Please add any new file to a folder name “CityData” inside “Code” folder.

Update the public variable <fileName> and add an “.txt” extension.

OUTPUT:

TRAINNING PROBLEMS :

train 1: (test1tsp.txt)

Tour : -> 2 -> 3 -> 1 -> 4 -> 2

Tour cost: 24.293023070189598

Total time taken (in Nano second):81533000

Train 2: (test2atsp.txt)

Tour : -> 2 -> 4 -> 6 -> 5 -> 8 -> 7 -> 1 -> 3 -> 2

Tour cost: 65.65395780324546

Total time taken (in Nano second):87424800

Train 3: (test3atsp.txt)

Tour : -> 2 -> 9 -> 6 -> 8 -> 3 -> 4 -> 5 -> 7 -> 1 -> 2

Tour cost: 229.50916652583456

Total time taken (in Nano second):86371400

2021 TEST PROBLEMS :

Test1-21 :

Tour : -> 10 -> 3 -> 5 -> 4 -> 2 -> 12 -> 8 -> 9 -> 1 -> 7 -> 11 -> 6 -> 10

Tour cost: 275.5834109126775

Total time taken (in Nano second):88694100

Test2-21:

Tour : -> 10 -> 12 -> 3 -> 13 -> 2 -> 4 -> 1 -> 14 -> 9 -> 11 -> 8 -> 5 -> 7 -> 6 -> 10

Tour cost: 836.7919803287864

Total time taken (in Nano second):99800600

Test3-21:

Tour : -> 10 -> 7 -> 6 -> 2 -> 11 -> 13 -> 8 -> 1 -> 17 -> 3 -> 9 -> 14 -> 12 -> 16 -> 15 -> 4 -> 5 -> 10

Tour cost: 121052.34049190061

Total time taken (in Nano second):206844600

Test4-21:

NOT ATTEMPTED

java.lang.OutOfMemoryError: Java heap space