Traveling Salesperson Report

Alexander Holmes

CS 4310

Dr. Dedonker

The main goal for this assignment was to implement a dynamic method for implementing the traveling salesperson algorithm. The traveling salesperson problem is the idea of finding the shortest path or shortest route for a salesperson to take on a business trip. We are given a starting point and a number of cities (nodes). The program will output the number of cities and the cost adjacency matrix. As well as output the g-functions, minimum cost and associated tour, and execution time of the matrix using TSP.

A cost matrix is given, representing the cost weight associated with each node in the problem. Each number represents the (cost, or time associated with getting to the next node/city).

How to read an adjacency matrix:

```
[0, 10, 15, 20]
[5, 0,  9,  10]
[6, 13, 0,  12]
[8, 8,  9,  0 ]
```

The first row, reads there is 0 cost to get to our starting node from our starting node

The second row, where it has a 0 reads there is 0 cost to get to node 2 from node 2

The third row, where it has a 0 reads there is 0 cost to get to node 3 from node 3.

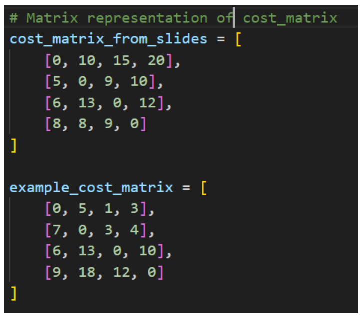
The same holds with node 4 on row 4.

Then, the left over numbers in the row are the cost to get to the other nodes. Using columns to designate nodes. So the first row reads "It costs 0 to get from node 1 to node 1, it costs 10 to get to node 2 from node 1, it costs 15 to get to node 3 from node 1,... "

Essentially each line is giving the cost to travel to each other's neighboring nodes.

If the node wasn't reachable from the other, we would initialize it with infinity 'inf()' in order to let the computer know we didn't have a connection in our graph with those two nodes.

In my program I use a provided example from the slides in class (as asked), as well as an example matrix.



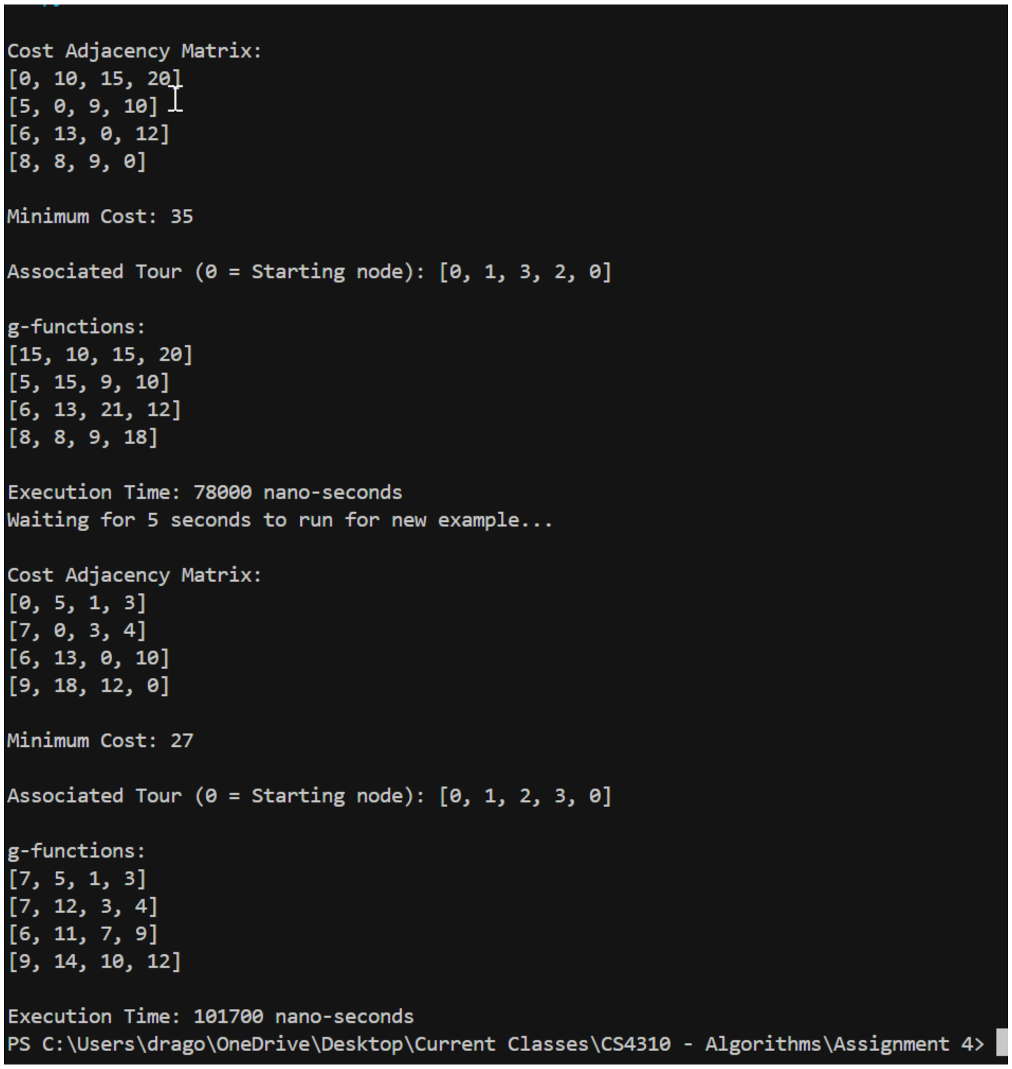
```
# Matrix representation of cost_matrix
cost_matrix_from_slides = [
    [0, 10, 15, 20],
    [5, 0, 9, 10],
    [6, 13, 0, 12],
    [8, 8, 9, 0]
]

example_cost_matrix = [
    [0, 5, 1, 3],
    [7, 0, 3, 4],
    [6, 13, 0, 10],
    [9, 18, 12, 0]
]
```

Using these two matrices my program

- Shows the cost matrix,
- Calculates and provides its minimum cost to get to the start node to the end node
- Displays the associated tour taken to get that minimum cost path (ending back at the starting node)
- Calculates and shows the g-functions for implementing TSP
- The execution time it took to get the solution tour AND the g-functions.

Below is a provided example output.



```
Cost Adjacency Matrix:
[0, 10, 15, 20]
[5, 0, 9, 10]
[6, 13, 0, 12]
[8, 8, 9, 0]

Minimum Cost: 35

Associated Tour (0 = Starting node): [0, 1, 3, 2, 0]

g-functions:
[15, 10, 15, 20]
[5, 15, 9, 10]
[6, 13, 21, 12]
[8, 8, 9, 18]

Execution Time: 78000 nano-seconds
Waiting for 5 seconds to run for new example...

Cost Adjacency Matrix:
[0, 5, 1, 3]
[7, 0, 3, 4]
[6, 13, 0, 10]
[9, 18, 12, 0]

Minimum Cost: 27

Associated Tour (0 = Starting node): [0, 1, 2, 3, 0]

g-functions:
[7, 5, 1, 3]
[7, 12, 3, 4]
[6, 11, 7, 9]
[9, 14, 10, 12]

Execution Time: 101700 nano-seconds
PS C:\Users\drago\OneDrive\Desktop\Current Classes\CS4310 - Algorithms\Assignment 4>
```