Assignment: NP-Completeness and Heuristic Algorithms

*Note: You will discuss Question 1 as part of the Group Assignment. (Check this week’s Group Assignment on Canvas for details).*

1. **NP-Completeness:** Consider the Travelling Salesperson (TSP) problem that was covered in the exploration.

Problem: Given a graph G with V vertices and E edges, determine if the graph has a TSP solution with a cost of at most k.

Prove that the above stated problem is NP-Complete.

To establish the NP-completeness of the Traveling Salesman Problem (TSP), we need to demonstrate that it falls within the class of NP problems and also that it is NP-hard.

i. TSP is in NP:

We can provide a polynomial-time verifier for TSP as follows:

Inputs: A graph G(V, E), an integer k, and a certificate T (a tour of G).

1. If T does not include all vertices, return false.
2. If the sum of edge weights in tour T is less than k, return true.
3. Otherwise, return false.

The runtime of this verifier is O(n + m), which is polynomial in the size of graph G, where n is the number of vertices and m is the number of edges. Thus, the verifier runs in polynomial time.

ii. TSP is NP-hard:

We can establish the NP-hardness of TSP by reducing it from the well-known NP-hard problem, the Hamiltonian Cycle Problem:

Given a graph G, does G contain a Hamiltonian cycle?

Here's a reduction from Hamiltonian Cycle to TSP:

Input: Graph G(V, E).

1. Create a new graph G', having the same vertices as V, forming a complete graph (an edge between each pair of vertices).
2. For each vertex pair (u, v) in G':
   1. If there's an edge (u, v) in graph G, set its edge weight to 0 in G'.
   2. Otherwise, set its edge weight to 1 in G'.
3. Return the instance <G', 0>.

Step 2 of the reduction takes polynomial time O(n^2), ensuring the reduction is polynomial.

To prove the correctness of the reduction:

a. If G has a Hamiltonian cycle, denoted as H:

* H visits each vertex exactly once in G.
* In G', edges corresponding to H have weight 0 (as per the reduction).
* Thus, H is a valid TSP solution for G' with cost 0.

b. If G' has a TSP solution with cost at most 0:

* Assume G' has a TSP solution T with cost 0 (minimum possible cost).
* Edges in T must have cost 0, indicating they were part of H in G.
* These edges are present in G, and since T visits each vertex once, it forms a Hamiltonian cycle in G.

As both directions are established, the reduction is valid, confirming the NP-hardness of TSP.

Since TSP is shown to be in NP and also NP-hard, we conclude that TSP is NP-complete.

1. **Implement Heuristic Algorithm:** 
   1. Below matrix represents the distance of 5 cities from each other. Represent it in the form of a graph

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 2 | 3 | 20 | 1 |
| B | 2 | 0 | 15 | 2 | 20 |
| C | 3 | 15 | 0 | 20 | 13 |
| D | 20 | 2 | 20 | 0 | 9 |
| E | 1 | 20 | 13 | 9 | 0 |

* 1. Apply Nearest-neighbor heuristic to this matrix and find the approximate solution for this matrix if it were for TSP problem.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 2 | 3 | 20 | 1 |
| B | 2 | 0 | 15 | 2 | 20 |
| C | 3 | 15 | 0 | 20 | 13 |
| D | 20 | 2 | 20 | 0 | 9 |
| E | 1 | 20 | 13 | 9 | 0 |

* + 1. Start at A, let yellow be visited nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 2 | 3 | 20 | 1 |
| B | 2 | 0 | 15 | 2 | 20 |
| C | 3 | 15 | 0 | 20 | 13 |
| D | 20 | 2 | 20 | 0 | 9 |
| E | 1 | 20 | 13 | 9 | 0 |

* + 1. Closest neighbor is E, we move to E, order is A-E

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 2 | 3 | 20 | 1 |
| B | 2 | 0 | 15 | 2 | 20 |
| C | 3 | 15 | 0 | 20 | 13 |
| D | 20 | 2 | 20 | 0 | 9 |
| E | 1 | 20 | 13 | 9 | 0 |

* + 1. From E we move to D, order is A-E-D

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 2 | 3 | 20 | 1 |
| B | 2 | 0 | 15 | 2 | 20 |
| C | 3 | 15 | 0 | 20 | 13 |
| D | 20 | 2 | 20 | 0 | 9 |
| E | 1 | 20 | 13 | 9 | 0 |

* + 1. From D we move to B, order is A-E-D-B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 2 | 3 | 20 | 1 |
| B | 2 | 0 | 15 | 2 | 20 |
| C | 3 | 15 | 0 | 20 | 13 |
| D | 20 | 2 | 20 | 0 | 9 |
| E | 1 | 20 | 13 | 9 | 0 |

* + 1. From B we move to C, order is A-E-D-B-C
  1. What is the approximation ratio of your approximate solution?
     1. AR = Cost of Optimal Solution / Cost of Approximate Solution
     2. AR = 26/28 or about 1.076
     3. This means the neighbor heuristic is about 7.6% worse than the optimal solution.
  2. Implement Travelling Salesman Problem using the nearest-neighbor heuristic.

**Input**: The input Graph is provided in the form of a 2-D matrix (adjacency matrix). Consider the first node as the starting point.

Sample input:

G = [

[0, 2, 3, 20, 1],

[2, 0, 15, 2, 20],

[3, 15, 0, 20, 13],

[20, 2, 20, 0, 9],

[1, 20, 13, 9, 0],

]

**Output**: A list of indices indicating the path taken. You must return the sequence of nodes, the path taken starting from node 0. In this example, G is 5x5, indicating there are 5 nodes in this graph: 0-4. You will always begin with node 0, and your path should include every node exactly once, and only go between nodes with a nonzero edge between them. You path will end at the starting node.

Sample output (For above graph G):

[0, 4, 3, 1, 2, 0]

Note: Not all graphs are fully connected: some rows in G may have more than one 0. These indicate absence of an edge.

Name your function **solve\_tsp(G)**. Name your file **TSP.py**.