Coding Assignment 4: CS2233

4th November, 2023

Max Marks 70

Instructions: Strictly follow the input and output format for each problem. **Your code should run in an infinite loop expecting for query number**.

Input format

- The first line of input consists of three space-separated integers, N, M and s, denoting the number of vertices in the graph, the number of edges in the graph and the source vertex s (only s will be not there for the problems 1.c, 2) respectively.
- Following M lines consist of three space-separated each a, b, and w (only w will be not there for the problem 1), denoting there is an edge between vertex a and vertex b, followed by the weight of that edge. Assume all vertex is denoted in a numeric fashion (assume it's 0 indexed, which means, for n vertices, vertices numbering would be $0, 1 \ldots n-1$).

Output format

- Your output should contain n lines. Each line will contain the path from s to a vertex v, followed by the length of that path.
- For problem 1.c, just print YES if the input graph is bipartite o.w print NO.
- For problem 2, your output should contain n lines, where the first $n-1$ lines will contain the $n-1$ edges of the MST followed by the total weight of the MST in the last line.
- For problem 3, your output should contain *n-1* lines. Each line will contain the shortest path from s to a vertex v, followed by the length of that path. In case there is any negative cycle in 3.b, then print YES and print the edges of the negative cycle (If there is more than one such cycle, then print anyone).

Example:
Input:
4 4 3

```
0 2
1 2
1 3
2 3
```

Output:

1.a

```
3 1 1
3 1 2 2
3 1 2 0 3
```

1.b

```
3 1 1
3 2 1
3 2 0 2
```

1.c

```
NO
```

Note:- Code should run in an infinite loop expecting the query.

1 Coding Problems:

In the following problem suppose the graph is given as input in the adjacency list format.

1. **Graph Traversal:**

   (a) Suppose $G = (V, E)$ - an undirected graph, and a vertex $s \in V$ are given as input. Given the non-recursive implementation of the DFS traversal algorithm. $\forall v \in V$, you need to output both - the length of the path from s, and the list of vertices in the path. **10 Marks**

   (b) Suppose $G = (V, E)$ - an undirected graph, and a vertex $s \in V$ are given as input. Implement the BFS traversal algorithm. $\forall v \in V$, you need to output both - the length of the path from s, and the list of vertices in the path. **10 Marks**

   (c) Given a graph as input, write an algorithm to check whether it is bi-partite. Recall that a graph is called bi-partite if its vertices can be partitioned into two sets such that there are no edges within the same partition and edges exist only across the partition. **10 Marks**

2. **Minimum Spanning Tree (MST):**

   (a) Suppose $G = (V, E)$ an undirected weighted graph is given as input. Compute the MST of this graph using Prim's algorithm. You need to output the list of edges that comprises MST. **10 Marks**

(b) Suppose $G = (V, E)$ an undirected weighted graph is given as input. Compute the MST of this graph using Kruskal's algorithm. You need to output the list of edges that comprises MST. **10 Marks**

3. **Single Source Shortest Path:**

   (a) Suppose $G = (V, E)$ - an undirected weighted graph with non-negative edge weights and a vertex $s \in V$, are given as input. Implement Dijkstra's algorithm for computing the shortest path of $\forall v \in V/s$ from the vertex s. $\forall v \in V/s$, you need to output both - the length of the shortest path of v from s, and the list of vertices in the path. **10 Marks**

   (b) Suppose $G = (V, E)$ - an undirected weighted graph (negative edge weights are also allowed), and a vertex $s \in V$, are given as input. Write a code to check if the graph has any cycle of negative length. If it has one, then report it. If the graph doesn't have any negative cycle, then compute the shortest path of all the vertices from s. $\forall v \in V/s$, you need to output both - the length of the shortest path of v from s, and the list of vertices in the path. **10 Marks**