CSE 101 Summer Session 2 2022
Programming Assignment 4

INSTRUCTIONS: Solutions to this assignment must be submitted through Gradescope by Sunday, August 28th 2022 at 10:00 PM— no late submissions will be accepted.

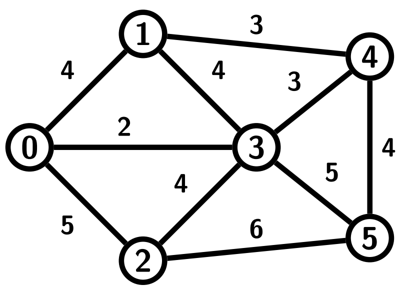
Note about Graph representation: Here and in the following Programming Assignments, the vertices in the graphs will be represented using numbers starting from 0 (0, 1, 2,...) in contrast to how vertices were represented using letters (A, B, C,..) in the graphs from the lecture.

1. **(Prim's Algorithm)** Implement Prim's Algorithm to find a Minimum Spanning Tree of a weighted undirected graph. Write a procedure `myPrim(graph, startV)` which takes as input:

   - *graph*: the adjacency matrix representation of a graph G
   - *startV*: the starting vertex of the algorithm

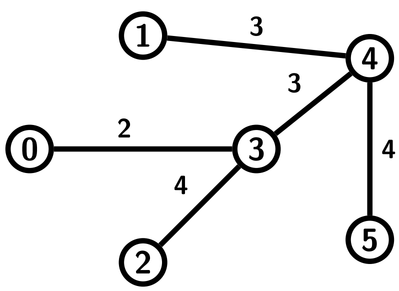
   It should return the adjacency matrix representation of the Minimum Spanning Tree of G.

   For example, consider the following graph G and its adjacency matrix representation:



$$G = \begin{bmatrix} 0 & 4 & 5 & 2 & 0 & 0 \\ 4 & 0 & 0 & 4 & 3 & 0 \\ 5 & 0 & 0 & 4 & 0 & 6 \\ 2 & 4 & 4 & 0 & 3 & 5 \\ 0 & 3 & 0 & 3 & 0 & 4 \\ 0 & 0 & 6 & 5 & 4 & 0 \end{bmatrix}$$

   The minimum spanning tree of G and its adjacency matrix representation is



$$MST = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 4 & 0 & 3 & 0 \\ 0 & 3 & 0 & 3 & 0 & 4 \\ 0 & 0 & 0 & 0 & 4 & 0 \end{bmatrix}$$

   Calling `myPrim(G, 0)` should return:

```
0 0 0 2 0 0
0 0 0 0 3 0
0 0 0 4 0 0
2 0 4 0 3 0
0 3 0 3 0 4
0 0 0 0 4 0
```

2. **SSSP (Dijkstra)**

   Recall the *Single Source Shortest Paths* problem: to compute the shortest paths from a source s to all other vertices in the network.

   Suppose that G is a directed graph with n vertices and m edges. Write a procedure `myDijkstra(vertices, edges)` which takes as input:

   - *vertices* $= [0, 1, 2, ..., n-1]$: a vector of the first n non-negative integers (the n vertices of G)
   - *edges*: a vector of the edges in G, where each index is of the form $[i, j, c(i, j)]$, corresponding to the directed edge from vertex i to vertex j with weight $c(i, j)$.

   and finds a solution to the SSSP problem. For simplicity, we use vertex 0 as the source vertex. Your procedure must return a tuple containing `(shortestPaths, shortestPathLengths)`:

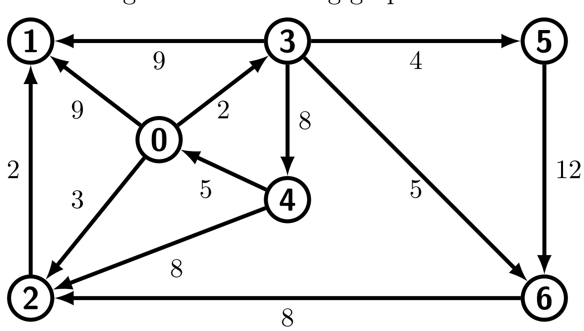
   (a) *shortestPath*: A list containing the shortest paths from source vertex to vertex i at index i. **Note**: If you are using Python, use a list of lists. If you are using cpp, implement *shortestPath* as a vector of vectors.

   (b) *shortestPathLength*: A 1D list that contains the shortest distances (from source vertex to vertex i at index i).

   Note:

   - You will only be tested on graphs where every node are reachable from the start node.
   - All edge weights will be positive.
   - You will be provided with a graph with at least 2 vertices.
   - You might be provided with graphs that contains cycles.

   . For example, suppose that we are given the following graph:



   We would input the vertices and edges of this graph to our procedure as *vertices* $= [0, 1, 2, 3, 4, 5, 6]$ and

$$edges = \begin{bmatrix} 0 & 1 & 9 \\ 0 & 2 & 3 \\ 0 & 3 & 2 \\ 2 & 1 & 2 \\ 3 & 1 & 9 \\ 3 & 4 & 8 \\ 3 & 5 & 4 \\ 3 & 6 & 5 \\ 4 & 0 & 5 \\ 4 & 2 & 8 \\ 5 & 6 & 12 \\ 6 & 2 & 8 \end{bmatrix}.$$

   Calling `myDijkstra(vertices, edges)`, the following is returned:

```
(
  [ [0],
    [0, 2, 1],
    [0, 2],
    [0, 3],
    [0, 3, 4],
    [0, 3, 5],
    [0, 3, 6] ],
  [ 0, 5, 3, 2, 10, 6, 7 ]
)
```

where *shortestPaths* are the shortest paths from vertex 0 to vertex 0, 1, 2, 3, 4, 5 and 6 respectively. The total cost of these shortest paths are returned in *shortestPathLengths*.