CS 222 Lab Exam

IIT Goa
27-03-23

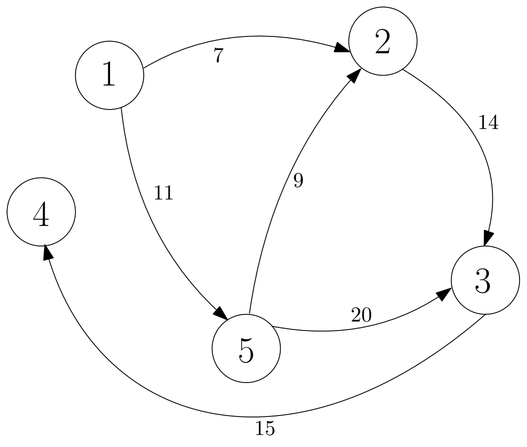
03:15PM to 04:30PM

Set A

Question

In today's problem, you are supposed to read the *positive weighted* adjacency matrix of a **directed graph** G from the input file provided with. The first line contains the number of vertices n and the next n lines correspond to the rows in the adjacency matrix (the vertices are labelled 1 to n). You are provided with a wrapper program which reads the name of the file as a command line argument and loads the details to the adjacency matrix. You are supposed to implement the following member functions in the class **graph** (*refer the cpp file provided with*). For those subproblems where the adjacency matrix has to be expanded, you may assume that the space allotted is sufficiently large.

Let the input adjacency matrix and hence the graph be as follows.



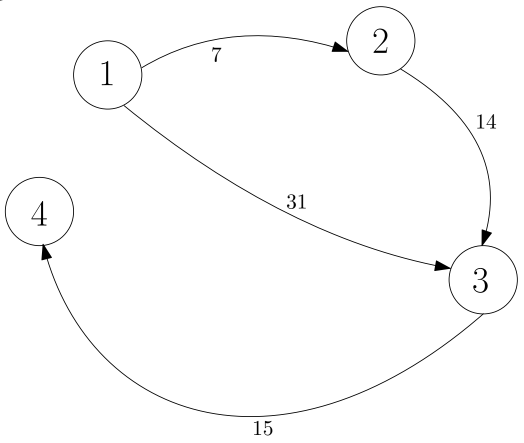
$$\begin{bmatrix} 0 & 7 & 0 & 0 & 11 \\ 0 & 0 & 14 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 20 & 0 & 0 \end{bmatrix}$$

***See the respective problem illustrations*

1. **findInDegree(u):** Receives a *vertex name* (which is an integer) and outputs the in-degree (number of incoming edges) of the vertex.
   - **findOutDegree(3):** Returns 2 as there are two incoming edges ($2- > 3$ and $5- > 3$) to vertex 3.

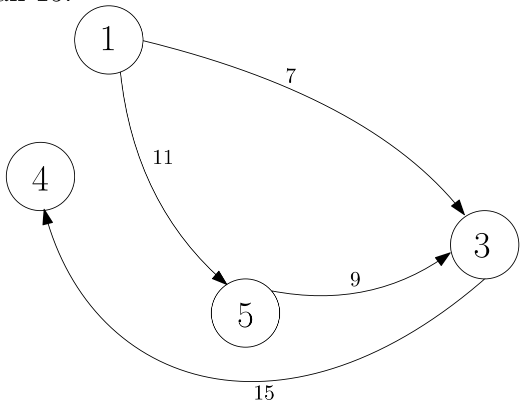
2. **findPathWeight($vertexList[], pathLength$):** Reads the list of vertices in a path and prints the total weight of the path. If the given path does not exist, then your function should print INF(infinity).
   - **findPathWeight($vertexList[], 4$):** Returns 38 for $vertexList[] = [5, 2, 3, 4]$ (sum of weights of the edges $5- > 2$, $2- > 3$, $3- > 4$ is 11+9+15=38).

3. **deleteVertex(u):** Deletes the vertex u from the graph and add/update the weights of the edges in the graph such that the length of the shortest path between any pair of vertices x and y ($x \neq y \neq u$) remains the same as in the original graph. Note that we don't modify the original adjacency matrix but work on a copy of it and prints the same.
   - **deleteVertex(5):** Deletes vertex 5 from the graph and we add edge $1- > 3$ of weight 11+20 ($1- > 5- > 3$). Further, we don't update the weight of the edge $1- > 2$ as current weight of $1- > 2$ is smaller than $1- > 5- > 2$ (11+9).



$$\begin{bmatrix} 0 & 7 & 31 & 0 \\ 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

4. **contractEdge(u, v):** Contracts the edge $u- > v$ (you may assume it exists) to a single vertex w_{uv} such that all the incoming edges to both the vertices u and v are now directed to w_{uv} and all the edges leaving both u and v are now directed away from w_{uv}. Furthermore, for every pair of multi edges (edges with the same pair of endvertices but different in weight) resultling from Contraction, retain the *lightest* one. Note that we don't modify the original adjacency matrix but work on a copy of it and prints the same.
   - **contractEdge($2, 3$):** Contraction of the edge $2- > 3$ results in the following graph (adjacency matrix). We merge vertices 2 and 3 to a single vertex. For the modified graph, you may assume any vertex naming (row/column index in the adjacency matrix) convention of your choice. Note that the vertex labels in the figure are different from the corresponding row/column indices in the adjacency matrix. For pair of multi edges of weights 9 and 20, we choose 9 as it is *lighter than* 20. Similarly, "if there was" an edge $2- > 4$ of weight 13, we would have replaced weight of edge $3- > 4$ with 13, as 13 is smaller than 15.



$$\begin{bmatrix} 0 & 7 & 0 & 11 \\ 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \end{bmatrix}$$

*******END*******