CS 222 Lab Exam

IIT Goa
27-03-23

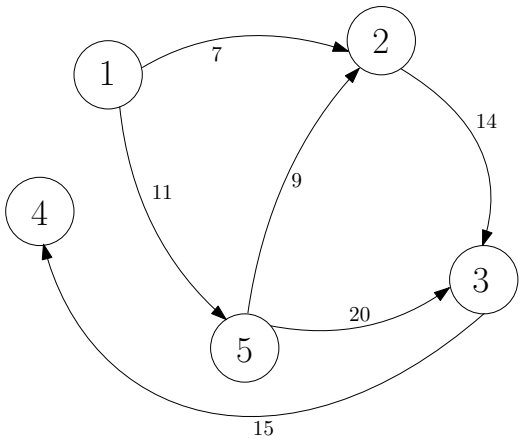
03:15PM to 04:30PM

Set B

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.



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

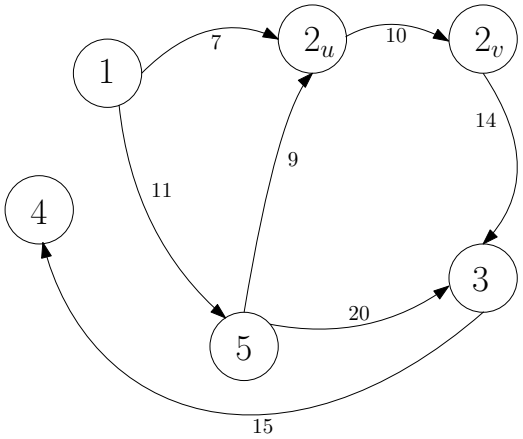
***See the respective problem illustrations*

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

2. **checkPath($vertexList[], pathLength$):** Reads the sequence of vertices ($pathLength$ is the number of vertices in the path) in a path and prints the missing edges if any. Otherwise, your function may print "Path is present in graph".

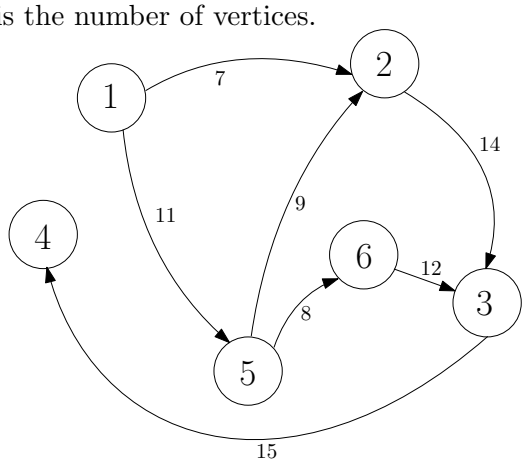
– **checkPath(**$vertexList[], 5$**):** For $vertexList[] = [4, 1, 2, 5, 3]$, prints $4- > 1$ and $2- > 5$ as they are the missing edges in the path $4- > 1- > 2- > 5- > 3$.

3. **expandVertex(**w**):** Expands the vertex w to a pair of vertices w_u and w_v and all the incoming edges to w are now directed to w_u and all the edges leaving w are now directed away from w_v. Further, add an edge $w_u- > w_v$ to the graph having **weight 10**.
   – **expandVertex(**2**):** Modifies the graph (the adjacency matrix) as follows. Note that you may use your own naming convention for the newly added vertices. In the adjacency matrix, 2_v is captured by index 2 and 2_u is captured by index 6.



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 11 & 7 \\ 0 & 0 & 14 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 9 \\ 0 & 10 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4. **replaceEdge(**u, v**):** Replaces the edge $u- > v$ (you may assume that $u- > v$ is present in the graph) with a vertex w. This means we delete edge $u- > v$, add the vertex w, and add/update edges such that all the shortest paths in the graph are retained as such.
   – **replaceEdge(**$5, 3$**):** Replaces the edge $5- > 3$ with a vertex 6 and adds edges so as to retain all those shortest paths. You may assume the name of the new vertex to be $n + 1$ where n is the number of vertices.



$$\begin{bmatrix} 0 & 7 & 0 & 0 & 11 & 0 \\ 0 & 0 & 14 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 & 8 \\ 0 & 0 & 12 & 0 & 0 & 0 \end{bmatrix}$$

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