CSE100 Algorithm Design and Analysis — Lab 15, Fall 2020
Last update: December 1, 2020 — Deadline: Friday, Dec 11, 5:00pm

Graphs: read/print graph; graph ops (BFS)

The C++ file `Graph.cpp` provided contains the definition of a struct `Graph`. The struct stores the information necessary to represent a directed weighted graph. A list called `vertices` stores the vertices, which are indicated as strings. A map called `edges` stores the edges. The key in the map is the edge, defined as a pair of strings, which are possible vertex names. The value is the weight of the edge. You can assume that the weights are all non negative. The class provided cannot be changed; you can only add methods, if needed, and variables, but you cannot change the underlying representation used to store the adjacency list.

Input structure Read from the keyboard the described graph and store it using an instance of the class `Graph`. The input is divided in sections. The first section starts with the vertices' names. There is one name per line. You can assume that names do not contain any space. The list of vertices terminate when the string END is read. Then the section describing edges starts. For each edge you have the starting vertex, the ending vertex, and the weight. The section ends when you read the string END. As usual, the END strings should not be processed. Note that the input may contain the erroneous edges connecting vertices not declared in the former section; these edges should not be inserted. Once the graph has been built, print out its structure using the provided `PrintOut` function.

Then start a cycle where you read operation codes and parameters and perform the following operations:

- *Operation code 0*, no parameters: terminate the program.
- *Operation code 1*, 1 parameter of type string (called A): print 1 if A is a vertex in the given graph, 0 otherwise.
- *Operation code 2*, 2 parameters of type string (called A and B): print the edge cost if the edge (A, B) exists, -1 otherwise.
- *Operation code 3*, 2 parameters of type string (called A and B): print the number of edges from vertex A to B, -1 if unreachable. You have to implement breadth-first search (BFS) to solve this part of the assignment.

Example input

```
Bremen
Hannover
Hamburg
Osnabruck
END
Hannover Bremen 123
Hannover Osnabruck 50
Bremen Hannover 123
Hamburg Hannover 190
END
1 Bremen
1 Brem
2 Osnabruck Hannover
2 Hannover Osnabruck
3 Hannover Hannover
3 Hannover Hamburg
3 Osnabruck Bremen
0
```

Example output

```
Bremen
Hannover
Hamburg
Osnabruck
Bremen Hannover 123
Hamburg Hannover 190
Hannover Bremen 123
Hannover Osnabruck 50
1
0
-1
50
0
2
-1
```