UNIVERSITY OF MINNESOTA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
4041: ALGORITHMS AND DATA STRUCTURES
FALL 2017

PROGRAMMING ASSIGNMENT 2, PART 1 :
Assigned: 11/19/17 Due: 12/03/17 at 11:55pm (submit via moodle)

Make sure you apply your code to different inputs to test the results. Copying code from others or the internet constitutes cheating and the University policies will be followed. For each problem hand in these three things (after putting them in a single .zip file **without any folders**):

1. The code you create.
2. A "readme.txt" file explaining any how to run your code and any assumptions that you made. If you were not able to complete a problem, describe your approach here.
3. A "run.sh" file that will both compile (if necessary) and run your program on the input text file. **You must ensure your code works with the run.sh on a cselabs machine**.

We will use files to both send input into your program and to read the output of your program. We should be able to run your program by saying "./run.sh input.txt" for some input text file. Your run.sh has to open the string after it as the input file (i.e. not always "input.txt"). In each part, your code should create a text file named "output.txt" with your solution. When printing multiple things, separate them by spaces (except after the last thing). To get your run.sh to work, you might need to type: chmod 700 run.sh

Please ensure your code is easily readable and well structured. If the code is obfuscated, has poorly named variables/functions, not well documented, etc., then points will be taken off. You may choose to code in any of these languages: C/C++, Java, or Python. For each problem the point breakdown will be roughly: 65% correct output, 15% readme.txt and .sh file sufficient, and 20% coding style. Your code must work on a caselabs machine. We will test it on the machine: csel-kh1260-13.cselabs.umn.edu

Problem 4. (20 points)
Implement Johnson's algorithm to fin all pairs shortest paths. The input text file will contain the graph represented as an adjacency matrix. Values of 2 millino will represent "infinity" edge weights (i.. when there is no edge between vertexes). You can assume all pairs of shortest paths will be less than 2 million.

Your output.txt should contain a matrix of all pairs shortest paths (just the distances). If there is a negative cycle present, output.txt should contain just the words (without quotes): "Negative cycle". The matrix for shortest paths should have spaces between the numbers and a single row on one line (similar to the input file). The vertex order must also be the same as the input text file, so the first row in the input text file must correspond to the shortest paths from the first vertex in output.txt.

Sample input file #1 (contains adjacency matrix):

```
0 2000000 4
2 0 7
2000000 3 0
```

Sample output.txt (no trailing spaces after last number):

```
0 7 4
2 0 6
5 3 0
```

Sample input file #2 (contains adjacency matrix):

```
0 2000000 -4
-1 0 7
2000000 -2 0
```

Sample output.txt (just this line and no other words):

```
Negative cycle
```

Problem 5. (20 points)

Assume you ran your problem 4 code on an undirected graph to find all pairs shortest path, but afterwards new vertexes are added to the graph and you want to compute the new all pairs shortest paths. In other words, find an efficient way to find all pairs shortest paths when new vertexes and edges are added for an **undirected** graph. Here "efficient" means that the runtime should be smaller than rerunning your problem 4 on the new graph from scratch. Your runtime should be approximately runtime(# old vertexes) + runtime(# added vertexes) = runtime(# total vertexes), where "old vertexes" and "total vertexes" could be computed directly from problem 4's algorithm.

The first thing in the input file will be the adjacency matrix (same setup as problem 4, except you can assume it will correspond to an undirected graph). After this will the list of new vertexes in the format of their corresponding rows in the adjacency matrix (see below for an example).

Your output.txt should contain two all pairs shortest paths: one using only the original matrix (same as the output would be for problem 4) and one with all the vertexes. (Again, you should not re-run problem 4 from scratch to get the all pairs shortest paths with all vertexes.) If there are no extra vertexes, just simply output the same as problem 4 with a single matrix. For each "all pairs shortest paths", if a negative cycle is present simply output (without quotes): "Negative cycle". If this happens in both the original and new graph, this output should appear twice.

Sample input file #1 (contains adjacency matrix with no "new vertexes"):

```
0 2 1
2 0 4
1 4 0
```

Sample output.txt (no trailing spaces after last number):

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

Sample input file #2 (contains adjacency matrix, with two “new” vertexes):

```
0 2 1
2 0 4
1 4 0
2000000 4 6 0 1
2 1 3 1 0
```

Sample output.txt (two matrices, one 3x3 and one 5x5):

```
0 2 1
2 0 3
1 3 0
0 2 1 3 2
2 0 3 2 1
1 3 0 4 3
3 2 4 0 1
2 1 3 1 0
```

Sample input file #3 (contains adjacency matrix):

```
0 2 1
2 0 4
1 4 0
2000000 4 6 0 1
-2 1 3 1 0
```

Sample output.txt:

```
0 2 1
2 0 3
1 3 0
Negative cycle
```

Sample input file #4 (contains adjacency matrix):

```
0 2 1
2 0 -4
1 -4 0
2000000 4 6 0 1
-2 1 3 1 0
```

Sample output.txt:

```
Negative cycle
Negative cycle
```

Sample input file #5 (contains adjacency matrix):

```
0 2 1
2 0 -4
1 -4 0
```

Sample output.txt (just this line and no other words):

```
Negative cycle
```