Project 6 Graph

1. Single Source Shortest Path (Dijkstra's algorithm, Textbook page 643)

Please implement the single source shortest path algorithm on a weighted, **directed graph.** Print out the shortest-path weight and the path if the source and destination nodes are connected. All weight will be positive and there is no self-link in the graph. If there is multi-path between source and destination, just print one of them.

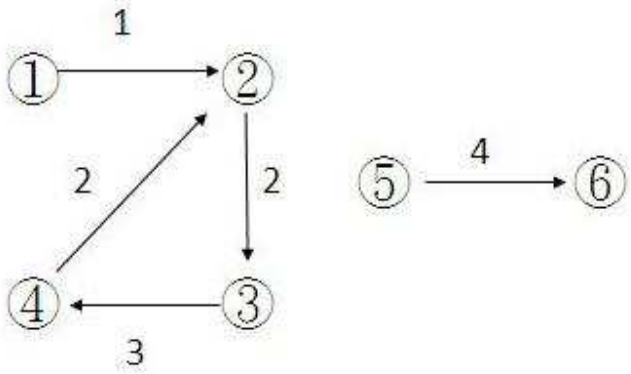
Input will be a file with two parts, separate by a * line. The first part is the graph, including nodes number and weight of the directed edge. The second part is testing data, groups of source node and destination node.

Print out the shortest-path weight and the path if the source and destination nodes are connected. Otherwise, print "no path".

Sample input file:

```
6           // nodes number;    node 1 -- 6
1 2 1       // Edge from node 1 to node 2, with weight 1
2 3 2       // Edge from node 2 to node 3, with weight 2
3 4 3       // Edge from node 3 to node 4, with weight 3
4 2 2       // Edge from node 4 to node 3, with weight 2
5 6 4       // Edge from node 5 to node 6, with weight 4
*           // Test data
1 3
2 4
3 1
5 6
4 3
```

Graph View:



Output:

path weight: 3
path : 1-2-3

path weight: 5
path : 2-3-4

no path from 3 to 1

path weight: 4
path : 5-6

path weight: 4
path : 4-2-3

2. All-Pairs Shortest Path (Floyd-Warshall algorithm, Textbook page 693)

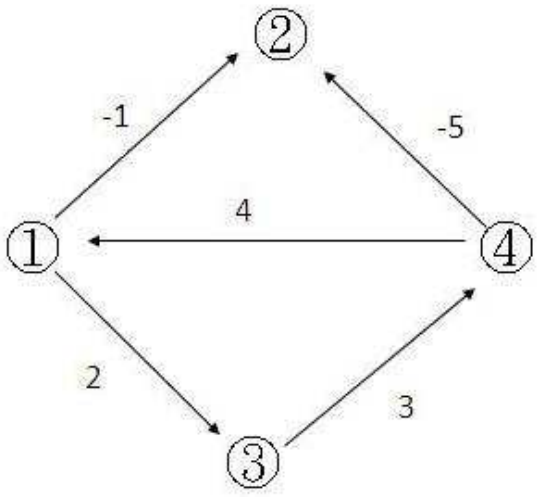
Please implement the all-pairs shortest path algorithm on a weighted, **directed graph**. Print out the matrix of shortest-path weight (D). The weight could be negative, but there will be no negative-weight cycle in the graph and all self-link weight is 0.

Input will be a file, including nodes number and weight of the directed edge.
Output should be a file, print out the path-path weight matrix. If no path exist, print "n".

Sample Input (File):

```
4          // nodes number;    node 1 -- 4
1 2 -1     // Edge from node 1 to 2, with weight -1
1 3 2      // Edge from node 1 to 3, with weight 2
3 4 3      // Edge from node 3 to 4, with weight 3
4 1 4      // Edge from node 4 to 1, with weight 4
4 2 -5     // Edge from node 4 to 2, with weight –5
```

Graph View :



Output (File) :

```
0   -1  2   5
n   0   n   n
7   -2  0   3
4   -5  6   0
```

Restrictions:

- This project will be personal, do not copy or modify by the others, or you will get nothing.
- Use c/c++ only.

Deadline:

- Due time: 6/9 13:00
- Demo time: 6/9 16:30-18:30 at EC 324
- Upload to e3

Please let me know if you have any problem
email: fd3srxs.cs99g@nctu.edu.tw