05.04.2015

Koc University
COMP 202 Data Structures and Algorithms
Spring 2015

Project 5

Deadline : May 22nd, 2015 (by 8:30am)

By submitting this assignment, you agree to fully comply with Koç University Student Code of Conduct, and accept any punishment as directed on the course syllabus in case of failure to comply. If you do not submit this assignment on time, you will receive no credit.

The Minimum Cost Spanning Tree

In this assignment, you are given an adjacency matrix of an undirected, weighted, connected graph. The problem is to find the minimum cost spanning tree which connects all the nodes of the graph by the use of Prim algorithm.

Input Format

You are given a file named "input.txt" which contains the adjacency matrix of a weighted undirected graph. The first line of the file contains the number of vertices in the graph. As an example, the input file may contain the following data (You can find the corresponding graph in the Figure1):

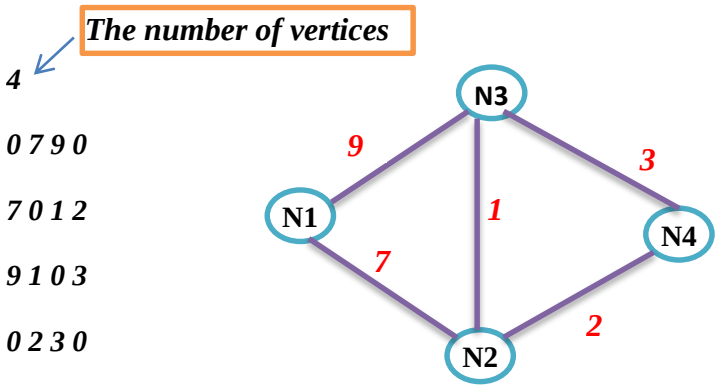

Figure 1

In the example above, the graph has 4 vertices. The values of each element of matrix represent the weight of the edge connecting corresponding nodes. Value 7, which is highlighted with green color in the following table, means that there is an edge weighted 7 connecting N2 and N1.

	N1	**N2**	**N3**	**N4**
N1	0	7	9	0
N2	7	0	1	2
N3	9	1	0	3
N4	0	2	3	0

Finding minimum spanning tree

You should follow prim algorithm mentioned below (it is the same as book's pseudocode).

Algorithm PrimJarnik(G):
  Input: An undirected, weighted, connected graph G with n vertices and m edges
  Output: A minimum spanning tree T for G
Pick any vertex s of G
$D[s] = 0$
for each vertex $v \neq s$ **do**
  $D[v] = \infty$
Initialize $T = \emptyset$.
Initialize a priority queue Q with an entry $(D[v], v)$ for each vertex v.
For each vertex v, maintain connect(v) as the edge achieving $D[v]$ (if any).
while Q is not empty **do**
  Let u be the value of the entry returned by Q.removeMin().
  Connect vertex u to T using edge connect(e).
  for each edge $e' = (u, v)$ such that v is in Q **do**
    {check if edge (u, v) better connects v to T}
    if $w(u, v) < D[v]$ **then**
      $D[v] = w(u, v)$
      connect(v) $= e'$.
      Change the key of vertex v in Q to $D[v]$.
return the tree T

For the priority queue, you can use Java built-in objects or methods. But for the rest (node, graph, etc.), you must have your own objects and implementations.

The sample output file for the Figure 1 is as following (You can find the corresponding MST with bolded lines in the Figure2):

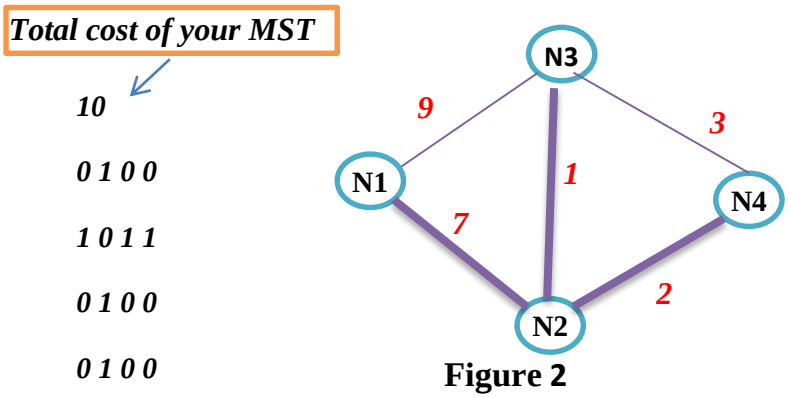

Figure 2

Requirements

- Your program must be able to read the "input.txt" file which should be located in the root directory of your project (It means that your codes should not need any changes to be able to read the "input.txt" file if they are to be run on the different systems, as long as the file is located in the root directory of your project)
- As your outputs, you should write the MST you found in the output.txt file as the adjacency matrix (The file must be located in the root directory of your project). In the first line of the output file, report the total cost of your MST as in the example given above.
- The name of the class that contains your *main* function should be named **Project5.java** . We will use the following commands to compile and run your application:

  >javac Project5.java

  >java Project5
- Put your files under the directory **COMP202/HOMEWORK/Project5**. Your files should be named with your student ID. If your directory name is not your student ID, or if the file name is different, or if your program does not compile, you will obtain zero grade.

Submission

- Along with your java source code (whole project's folder), you should send a text file named README.TXT.
- Your README should include a brief summary of your methods, results and any known problems/bugs in your code.