CS207A: Data Structures and Algorithms (Module #3)

Assignment #2

Max marks:55
Due on:17.00, 9-July-2017 1-July-2017

Note: All questions will be graded using an automated judge so please ensure that your programs follow the input-output instructions exactly.

1. Given a connected graph $G = (V, E)$ with non-negative edge weights here is a different way to construct an MST $T = (V_T, E_T)$ greedily by deleting edges instead of adding edges. The basic idea is as follows:

   $E_T \leftarrow E$
   Repeat

   Pick any cycle in G, find the heaviest edge (that is one with largest weight), say e_{xy}, delete e_{xy} that is $E_T = E_T - \{e_{xy}\}$.

   Until E_T is an MST.

   (a) Convert the above basic idea into a proper algorithm through an implementation. Your program should read all its input from a file called INPUT structured as follows:
   First line: n //number of nodes in the tree. Nodes are labelled 1 to n.
   Remaining lines: Edges in the graph, one per line as a three numbers separated by spaces i j e_{ij}. i, j are node numbers representing an edge and e_{ij} is the edge weight.
   Your output should be in file OUTPUT giving the edges in E_T with one edge per line ordered in increasing order of the first node then second node. For example, if $E_T = \{(2,3),(1,4),(1,5),(4,3)\}$ then the output should be:

   $(1,4)$
   $(1,5)$
   $(2,3)$
   $(3,4)$

   The following parts are not to be submitted but try to answer them.

   (b) Is your algorithm provably optimal? That is can you prove that your algorithm will always find the minimum spanning tree for any connected graph.

   (c) What is the complexity of your algorithm?

[25]

2. Implement changes to Dijkstra's algorithm for the following two variants:

   (a) Find the shortest path given two nodes: the source node and the destination node. This is called the single pair shortest path problem. The input is in file INPUT and is structured as follows:

`First line:` n s d//n is number of nodes. Nodes are labelled 1 to n. s is the source node and d is the destination node.
`Remaining lines`: Edges one per line similar to problem 1 above.

Output should be in file OUTPUT structured as follows:

`First line:` Shortest path length
`Second line:` Actual path as a sequence of nodes starting with source.

(b) The graph is now a directed graph. The input is again in file INPUT structured same as above except now the edges are directed so i j means the direction of the edge is from i to j. The output is again structured as above and is in file OUTPUT.

[15,15=30]