CS3230
Tutorial 7

1. Consider the following undirected graph.

   $G = (V, E)$, where the set of vertices is $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$, and the edges and their weights are given as follows:

   $wt(1, 2) = 3$, $wt(1, 5) = 2$, $wt(1, 4) = 10$, $wt(2, 3) = 4$, $wt(2, 5) = 9$, $wt(3, 5) = 6$,

   $wt(3, 6) = 5$, $wt(4, 5) = 4$, $wt(4, 7) = 4$, $wt(5, 6) = 3$, $wt(5, 7) = 6$, $wt(5, 8) = 2$,

   $wt(5, 9) = 6$, $wt(6, 9) = 6$, $wt(7, 8) = 8$, $wt(7, 10) = 3$, $wt(8, 9) = 8$, $wt(8, 10) = 3$,

   $wt(8, 11) = 5$, $wt(8, 12) = 7$, $wt(9, 12) = 4$, $wt(10, 11) = 5$, $wt(11, 12) = 2$

   (a) Use Prim's algorithm to find a minimal spanning tree for the graph. Start with node 1.

   Ans: The edges are picked in the following order by the Prim's algorithm. (In some cases, there are choices to be made due to equal weights of the edges/distance for the vertices; ties were broken arbitrarily in the following).

   $(1, 5)$, $(5, 8)$, $(1, 2)$, $(5, 6)$, $(8, 10)$, $(7, 10)$, $(2, 3)$, $(4, 5)$, $(8, 11)$, $(11, 12)$, $(9, 12)$ are the edges chosen.

   I am omitting the table for D and Rem.

   (b) Use Kruskal's algorithm to find a minimal spanning tree for the graph.

   Ans: The edges are picked in the following order by the Kruskal's algorithm. (In some cases, there are choices to be made due to equal weights of the edges; ties were broken arbitrarily in the following).

   $(1, 5)$, $(5, 8)$, $(11, 12)$, $(1, 2)$, $(5, 6)$, $(7, 10)$, $(8, 10)$, $(2, 3)$, $(4, 7)$, $(9, 12)$, $(10, 11)$ are the edges chosen.

2. Does Dijkstra's algorithm work if the weights can be negative?

   Either give an argument that it works, or give a counterexample that it does not work.

   Ans: No. Counterexample:

   $G = (V, E)$,

   where $V = \{1, 2, 3, 4, 5\}$

   starting vertex is 1

   $wt(1, 2) = 5$,

   $wt(2, 3) = 3$

$wt(1,4) = 9$

$wt(4,5) = -7$

$wt(5,2) = 1$

Dijkstra's algorithm will give the distance to vertex 2 as 5, which is not the shortest distance.

3. Consider the following divide and conquer algorithm for finding a minimal spanning tree.

   (a) Divide the set of vertices into two halves.
   Note: It is assumed that the divison into two halves might be arbitrary.
   (b) Find the minimal spanning tree for each half.
   (c) Find the minimal weight edge which has two endpoints in two different halves above.
   (d) Use the edge found in (c) above to connect the two spanning trees found in (b) above.

   Is the above algorithm correct? If so, give an argument for it. If not, give a counterexample.

   Ans: No. Consider the following counterexample:

   $G = (V, E)$, where $V = \{1, 2, 3, 4\}$ and $wt(1, 2) = 9$, $wt(3, 4) = 1$, $wt(1, 3) = 1$, $wt(2, 4) = 2$.

   And, the initial division is $\{1, 2\}$ and $\{3, 4\}$.

   Then optimal MST is by using the edges $(1, 3), (2, 4), (3, 4)$.

   However, the algorithm above will give the spanning tree, $(1, 2), (3, 4), (1, 3)$

4. Prove or disprove:

   (a) Suppose the graph contains only one edge of minimal weight. Then show that every minimal spanning tree must contain that edge.

   Ans. Suppose some MST T does not contain the minimal weight edge e. Then consider the graph $T \cup \{e\}$. Then, this graph contains a cycle. Delete an edge from the cycle different from e. Then the resulting graph is a tree with weight less than T. A contradiction.

   (b) Suppose a graph may contain several edges of minimal weight. Then does the minimal spanning tree contain all the edges of minimal weight?

   Ans: No. Consider the graph:

   $V = \{1, 2, 3, 4\}$

   $wt(1, 2) = 1$, $wt(1, 3) = 1$, $wt(2, 3) = 1$, $wt(1, 4) = 2$

   Then, the MST contains two of $(1, 2), (2, 3), (1, 3)$

   AND $(1, 4)$.

5. Suppose we are given n sorted lists.

   Suppose the only further operations allowed to us is to merge two lists (which may be among the above lists or obtained in turn by merging some of the lists).

   Suppose further that merging two lists of size k and ℓ requires time $k + \ell$.

   Give an algorithm to determine the order in which the n lists (having $m_1, m_2, \ldots, m_n$ numbers respectively), should be merged so that the merging operations take the least amount of time.

   Note that the aim above is to minimize the merging time of lists (and not the time taken by your algorithm).

   Ans: This problem is essentially similar to Huffman coding. Just as in Huffman coding, we merge the two lists with least weights first, to form a combined list (which is then again put back in the pool of lists). We continue in a manner similar to Huffman coding. Details omitted.

6. Suppose we modify the continuous knapsack problem done in class to the discrete version as follows:

   > Either take all of item i or none of item i (that is, we cannot take only a part of item i as done for the continuous knapsack problem done in class).

   Appropriately modify the greedy algorithm for continuous knapsack problem to the discrete knapsack problem.

   Is the modified greedy algorithm still optimal? If so, give an argument. If not, give a counterexample.

   Ans: No. Consider the case of

   $W_1 = 50, V_1 = 49.$

   $W_2 = 1$, $V_2 = 1$

   Capacity$= 50$