Assignment 4 Solutions

1. Describe an efficient algorithm that, given an undirected connected graph $G=(V,E)$ determines a spanning tree of G whose largest edge weight is as small as possible. Thus, we are not trying to minimize the total weight of all edges of the spanning tree, but just the largest weight of an edge in the spanning tree.

We show by induction on the number of vertices in the graph that the minimum spanning tree also has the smallest value of the longest edge in the tree. Assume the statement is true for minimal spanning trees for graphs with at most n vertices. If G has $n+1$ vertices, take out any vertex and find a minimal spanning tree for the rest of the graph. By induction, such minimum spanning tree has least weight of its maximal weight edge. To get a minimal spanning tree we now must add the edge connecting $n+1^{th}$ vertex with the rest of the graph that has the smallest weight. Clearly, such obtained tree must have the smallest possible weight of its largest weight edge, because this is true for the sub-tree for n vertices, and also for the entire tree because of the way we added the last edge for the tree.

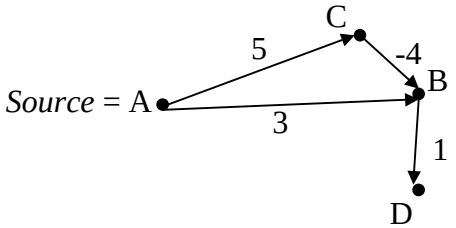
Clearly, this problem can also be solved by showing that Kruskal's (or Prim's) algorithm produce a tree with smallest maximal weight edge.

2. Consider all spanning trees ordered by their weight. Show that the second best minimal spanning tree can be obtained from the minimal spanning tree by dropping only one edge and adding only one new edge instead.

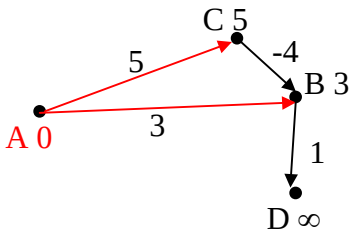
Assume one needs two replacements. If one of the replacements decreases the weight then doing this replacement first would produce a tree lighter than the minimal spanning tree. If both replacements increase weight, then there would be a tree with weight strictly between the best and the second best tree, by using one replacement only.

3. Give a simple example of a directed graph with negative-weight edges for which Dijkstra's algorithm for single source shortest paths problem produces incorrect answers.

Dijksta's algorithm does not work if the graph has negative weight edges, i.e., it might return incorrect result, as the following example shows:

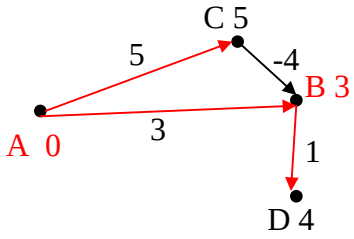


Lets see how the algorithm would run on this graph; first the source A is popped out of the queue and edges (A, C) and (A, B) are relaxed.

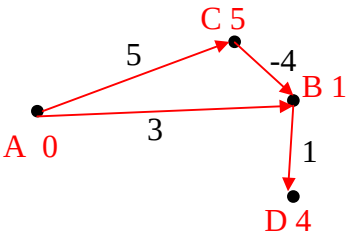



```
Dijkstra(G,s)
for each v ∈ V
    d[v] ← ∞
d[s] ← 0
S ← ∅
Q ← V
while Q ≠ ∅
      u ←ExtractMin(Q)
      S ←S U {u}
      for each v ∈ Adj[u]
          if d[v] > d[u]+w(u,v)
                    DecreaseKey(v, d[u]+ w(u,v))
```

Now B is on the top of min-heap queue Q. We now pop B and relax the only edge out of B, i.e., (B, D) giving D distance 3+1 = 4:



Now D is on the top of the min-heap, and we remove D. There are no edges going out of D to relax. The only vertex now left in the queue is C. We remove C and relax the edge (C, B). Since 5 - 4 = 1 < 3, distance to B must be updated to 1:



The algorithm now terminates because the queue is empty. However, the path A→C→B→D has total length 5 + (-4) + 1 = 2 < 4; thus the length of the shortest path from the source A to D was not correctly evaluated.