CS 581 Homework 7

Due on 02/29/2018

Problem 1.
In this problem, we give pseudocode for three different algorithms. Each one takes a connected graph and a weight function as input and returns a set of edges T. For each algorithm, either prove that T is a minimum spanning tree or prove that T is not a minimum spanning tree. Also describe the most efficient implementation of each algorithm, whether or not it computes a minimum spanning tree.

a. MAYBE-MST-A(G, w)

```
sort the edges into nonincreasing order of edge weights $w$
$T = E$
for each edge $e$, taken in nonincreasing order by weight
    if $T - \{e\}$ is a connected graph
        $T = T - \{e\}$
return $T$
```

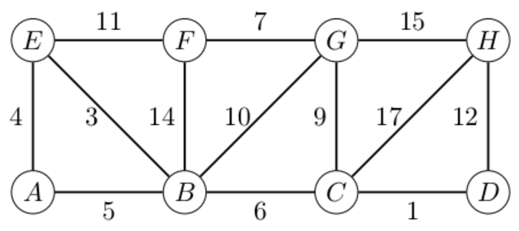
b. MAYBE-MST-B(G, w)

```
$T = \emptyset$
for each edge $e$, taken in arbitrary order
    if $T \cup \{e\}$ has no cycles
        $T = T \cup \{e\}$
return $T$
```

c. MAYBE-MST-C(G, w)

```
$T = \emptyset$
for each edge $e$, taken in arbitrary order
    $T = T \cup \{e\}$
    if $T$ has a cycle $c$
        let $e'$ be a maximum-weight edge on $c$
        $T = T - \{e'\}$
return $T$
```

Problem 2.
Run Dijkstra's algorithm on the weighted graph below, using vertex A as the source. Write the vertices in alphabetically order and compute all distances at each step.



Problem 3.

a) Give a counter example where Dijkstra's algorithm fails to generate shortest paths in graphs with negative edge weights, but no negative weight cycle. Show your graph, source node and show how Dijkstra's algorithm fails to find the shortest paths.

b) Why may Dijkstra's algorithm not find the shortest path when negative-weight edge is present?

c) A man on the street proposes to increment the weight on each edge by a same value s to make all weights non-negative, if there are negative edges, and then run Dijkstra's algorithm. When he finds the shortest paths, he would minus $s * l$ from a path where l is the number of edges on this path to get the final distance. Will this method fix the negative edge problem? Justify your answer.

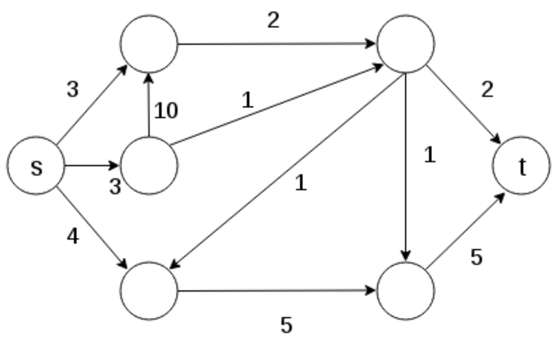
d) If you believe c) won't work, suggest a method yourself to modify Dijkstra's algorithm so that it works with negative edges (no need to handle negative cycles) and analyze the running time. Your running time should be $O(V^{3-\epsilon})$ where ϵ is a small positive number (polynomially smaller than $O(V^3)$).

Problem 4.

Another way to reconstruct shortest paths in the Floyd-Warshall algorithm uses values $\phi_{ij}^{(k)}$ for $i, j, k = 1, 2, \cdots, n$ where $\phi_{ij}^{(k)}$ is the highest-numbered intermediate vertex of a shortest path from i to j in which all intermediate vertices are in the set $1, 2, \cdots, k$. Give a recursive formulation for $\phi_{ij}^{(k)}$, modify the FLOYD-WARSHALL procedure to compute the $\phi_{ij}^{(k)}$ values, and rewrite the PRINT-ALL-PAIRS-SHORTEST-PATH procedure to take the matrix $\Phi = (\phi_{ij}^{(k)})$ as an input.

Problem 5.

Use the Ford-Fulkerson algorithm to maximize flow on the graph shown below. Specify the algorithm of your choice to find augmenting paths. At each iteration, you should (1) draw the residual graph, (2) find an augmenting path in that graph, and (3) draw the new flow graph resulting from the augmenting path.



Problem 6.
For vector $r = \langle 1, 2, 3 \rangle, c = \langle 2, 1, 2, 1 \rangle$,

a) Find a binary matrix for which the sum of the ith row is r_i and the sum of the jth column is c_j, or prove that such a matrix does not exist.

b) If the matrix does exist, determine whether it is unique.