# CS21003 - Practice Problems on Graph

## Solution Sketch

1. Suppose that you are given a graph $G = (V, E)$ and its minimum spanning tree $T$. Suppose that we delete from $G$, one of the edges $(u, v) \in T$ and let $G'$ denotes this new graph.

   (a) Is $G'$ guaranteed to have a minimum spanning tree? $\Rightarrow$ NO. It is possible that $G$ was already a tree, removing an edge makes it undirected, and it may not have a spanning tree.

   (b) Assuming that $G'$ has a minimum spanning tree $T'$. TRUE or FALSE: the number of edges in $T'$ is no greater than the number of edges in $T$. Explain your answer in one sentence. $\Rightarrow$ TRUE, spanning tree always has $|V| - 1$ vertices.

   (c) Assuming that $G'$ has a minimum spanning tree $T'$, describe an algorithm for finding $T'$. What is the complexity of your algorithm? $\Rightarrow$ Since $(u, v) \in T$ has been deleted, $T$ has 2 connected components. Idea is to find the minimum weight edge between the two components. To do that, we need to first find the two components. Use BFS from $u$ or $v$ to find one of the component. Other consists of all other vertices.

2. Let $G$ be an undirected connected weighted graph. Suppose the graph has at least one cycle (choose one). For that chosen cycle, let edge $e$ be an edge that has strictly greater cost than all other edges in the cycle. (Such an edge might not exist, e.g., there might be two edges that have the same greatest cost). Show that $e$ does not belong to any MST of $G$.

   $\Rightarrow$ Assume that $e$ belongs to an MST $T$ of $G$. Consider the disconnected subgraph with $T$
   $e$ edges. It has two connected components. Now, since $G$ has a cycle containing $e$, to complete the cycle, there must be another edge $e' \notin T$ that connects these two components. Since the weight of $e'$ is lower than $e$, we can replace $e$ by $e'$ to get an MST of lower cost.

3. Consider a "reversed" Kruskal's algorithm for computing an MST. Initialize $T$ to be the set of all edges in the graph. Now, consider edges from largest to smallest cost. For each edge, delete it from $T$ if that edge belongs to a cycle in $T$. Assuming all the edge costs are distinct, does this new algorithm correctly compute an MST?

   $\Rightarrow$ Yes. Using the previous question, every deleted edge is of maximum value in a cycle in $T$ and thus, cannot belong to the MST, and also can't disconnect the graph. Thus, the algorithm terminates with $T$ not having a cycle and not being disconnected, thus we get a spanning tree. Finally, only the edges that could belong to an MST remain, thus the resulting spanning tree is an MST.

4. A *maximum spanning tree* of $G$ is a spanning tree $T$ of $G$ such that the sum of costs of the edges in $T$ is as large as possible. Modify Kruskal's algorithm to compute a maximum spanning tree of $G$. What is the running time of your algorithm?

   $\Rightarrow$ Use Kruskal's algorithm with negative edge weights (multiply by -1 for each edge weight).

5. We are given a directed graph $G(V, E)$ on which each edge $(u, v)$ has an associated value $r(u, v)$, which is a real number in the range $[0, 1]$ that represents the reliability of a communication channel from vertex $u$ to vertex $v$. We interpret $r(u, v)$ as the probability that the channel from $u$ to $v$ will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.

   $\Rightarrow$ The most reliable path from $u$ to $v$ will correspond to the path with the maximum probability of not failing

For a path $u, v_1, v_2, \ldots, v_k, v$, we need to maximize

$r(u, v_1) \cdot r(v_1, v_2) \ldots r(v_k, v)$

$= [logr(u, v_1) + logr(v_1, v_2) + \ldots + logr(v_k, v)]$

Which is equivalent to minimizing

$[-logr(u, v_1) - logr(v_1, v_2) \ldots - logr(v_k, v)]$ We can thus use $-logr(u, v)$ as the edge weight and run Dijkstra's algorithm.

6. Let $G = (V, E)$ be a weighted graph and $T$ be the array containing the parents (previous vertices) corresponding to the shortest distance from source s to all the other vertices. Assume all weights in $G$ are increased by the same amount $c$. Is $T$ still the parent array (from source s) of the modified graph? If yes, prove the statement. Otherwise, give a counter example.

   $\Rightarrow$ No. Easy to find a counter-example shen the shortest path changes after increasing edge weights by a constant amount.

7. Let $G = (V, E)$ be a weighted undirected graph. Let $s, t \in V$ and $s \neq t$. Design an $O(ElogV)$ algorithm to find all vertices $v$ such that $v$ lies on at least one of the shortest paths between $s$ and $t$.

   $\Rightarrow$ Compute shortest distances from $s$ and $t$ to all other vertices [2 pass of Dijkstra's algorithm]. It takes $O(ElogV)$. Now a vertex $v$ lies on the shortest path if

$$mindist[s, v] + mindist[t, v] = mindist[s, t]$$

8. Given a directed graph $G = (V, E)$ with vertex set $V = \{1, 2, \ldots, n\}$, we define the *transitive closure* of $G$ as the graph $G^* = (V, E^*)$, where

$$E^* = \{(i, j) : \text{there is a path from vertex } i \text{ to vertex } j \text{ in } G\}$$

   Can you propose an $O(n^3)$ algorithm to compute the transitive closure of the graph?

   $\Rightarrow$ Apply Floyd-Warshall algorithm. After this, if the entry in $D^{(n)}(i, j)$ is still $\infty$, it implied that there is no path from $i$ to $j$, otherwise, there is some path. Thus $E^*(i, j) = 1$ if $D^{(n)}(i, j) \neq \infty$, 0 otherwise.

9. Describe an algorithm to find the length of the shortest cycle in a weighted directed graph in $O(n^3)$ time. Assume that all the edge weights are positive.

   $\Rightarrow$ Use Floyd Warshall's algorithm but also update the diagonal entries now. At the end, the smallest diagonal entry will correspond to the shortest cycle in the graph.

10. Let $D$ be the shortest path matrix of an undirected weighted graph $G$. Thus $D(u, v)$ is the length of the shortest path from vertex $u$ to vertex $v$, for every two vertices $u$ and $v$. Graph $G$ and matrix $D$ are given. Assume the weight of an edge $e = (a, b)$ is decreased from $w_e$ to $w'_e$. Design an algorithm to update matrix $D$ with respect to this change. The running time of your algorithm should be $O(n^2)$. Describe all details and write a pseudo-code for your algorithm.

    $\Rightarrow$ Weight for edge $e = (a, b)$ is decreased. For $(u, v)$, the shortest path may change if i). the current shortest path does not include $e$ but there is another path including $e$ and distance has decreased for that, or ii). $e$ is already in the shortest path.

    Since it is an undirected graph,

$$D(u, v) = min[D(u, v), D(u, a) + w'_e + D(b, v), D(u, b) + w'_e + D(a, v)$$

    which is doable in $O(n^2)$.