Seonhye Yang (sy3420) Homework 7 CS-GI 1170

Problem 1

a. All edge weights of undirected connected graph G are 1. Find minimum spanning tree (MST) of G.

- Since all edge weights are equal weights of 1, any spanning tree of graph is MST. We can find BF tree that will create a tree which has all vertices. This tree is MST of graph G. For unweighted or equal weighted edges, any spanning tree is the minimum spanning tree. In conclusion, we would use Breadth First - Traversal algorithm.

The idea here is to store the predecessor of given vertex while doing BFS.

1. First, make an array $A[0,1,\dots,V-1]$ such that $A[i]$ stores the distance of the $i^{th}$ vertex from the current vertex and array $B[0,1,\dots,V-1]$ such that $B[i]$ is the immediate predecessor of the $i^{th}$ vertex in the BFS search.

2. Then we get the length of the path from source to any other vertex.

3. Since this algorithm is spent doing BFS, it has running time of $O(|V|+|E|)$.

b. All edges are equal to 1, except for single edge $e_0=\{u_0,v_0\}$ whose weight is $w_0$.

- In order to modify the algorithm above, let's look at all cases.

case 1 $w_0$ is larger than 1.
case 2 $w_0$ is smaller than 1.

- Case 1: $e_0$ might or might not be in MST. So we should remove $e_0$ and execute the algorithm to see if BFS generates all vertices. If it does so, then do not implement $e_0$ to MST. If it doesn't generate all vertices, then implement $e_0$ to MST.

- Case 2: Since $e_0$ is smaller than 1, $e_0$ must be included in MST. Then run this algorithm to find MST of G.

The run-time of this algorithm is $O(|V|+|E|)$. The run-time is the same as part a.

## Problem 2

We are given an undirected graph $G$ with weighted edges and MST $T$ of $G$. We change the weight of one edge, $e$ of $G$, and obtain a new undirected weighted graph $G'$.

a. The weight of one edge $e \in T$ is decreased.

$T$ will remain in MST.

- Let $T$ be MST and $W$ be weight of $T$.
- Let $e=(u,v)$.
- Let $d$ be decrease in weight.

⇒ So new $T$ becomes $T = W-d$

- Let $T_u$ and $T_v$ be subtrees when edge $e$ is removed.

⇒ If $T$ does not remain as MST, then any new MST must contain edge $e$. Since this is not the case ($w(T') < W-d < W = w(T)$), this contradicts the idea that $T$ was a MST with original weights.

⇒ However, if $T'$ has edge $e$, $T'$ can is smaller than $W-d$ only by connecting nodes of $T_u$ with lower weights than $w(T_u)$ or by connecting the nodes of $T_v$ with weigts less than $w(T_v)$.

⇒ $T_u$ and $T_v$ are both MST for their node sets.

b. The weight of one edge $e \notin T$ is increased.

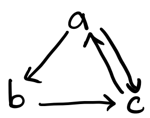
⇒ Run Kruskal's algorithm which produce $T$. All of the same decisions would be made when $e$ has a larger weight, so the same tree will be produced. However, the weights are unique.

Problem 3

Topological sort does not always minimize the bad edges. If we start from different starting points, topological ordering can give different sequences.

Counter-example:

- Let G be graph with vertices a, b, c, d. If vertices is generated by topological ordering, the sequence for directed edges d to a are "bad" edges. But a to d is not "bad" edges.




1. lets say we start from a, then sequence a, b, c has one "bad" edge which is from c to a.

2. lets start from point b, then sequence b, c, a would have 2 bad edges: a to c and a to b.

## Problem 4

a.

- Find strongly connected component in G. SCC of a directed graph is a maximal strongly connected subgraph.

- Replace strongly connected component in G with a vertex so G becomes DAG. In DAG, every vertex is its own strongly connected component.

- Then linear ordering of its vertices so that for every directed edge uv from vertex u to v, u comes before v in the ordering.
- A directed graph G is semi-connected if for all pairs of vertices u and v in G, we have path from u to v or v to u (or both).

  - For example, if ther's an edge between $V[i+1]$ & $V[i+2]$, then it's semi-connected.
  - If the vertices form a linear chain, then G is semi-connected.
    $\rightarrow (V_1, V_2), (V_2, V_3), (V_3, V_4) \cdots (V_{x-1}, V_x)$

Graph of SCC

$$(0,1,2) \rightarrow (3) \rightarrow (4)$$

b. If G is semi-connected, then there exists a path for $(V_{x-1}, V_x)$ such that $V_{x-1} \longrightarrow V_x$.

If $V_{x-1}$, $V_x$ are SCC, there's a path from $V_{x-1} \rightarrow V_x$ AND $V_x \rightarrow V_{x-1}$, reverse graph, since the nodes are strongly connected.

If the algorithm runs, then for the nodes $V_{x-1}, V_x$, $V_{x-1}, V_x$ are in SCC. Thus, there exists a path from $V_{x-1}$ to $V_x$.