

CSE 202 Homework 3: Solutions
Graph Algorithms

1. **Second-best minimum spanning tree: CLR 24-1**

Solution

Part a) Given a spanning tree S that is different from the minimum spanning tree T , there exists an edge $(a, b) \in S, \notin T$, and an edge (u, v) on the path in T from a to b that is not in S (otherwise S has a cycle). Clearly $w(u, v) \leq w(a, b)$ since T is a MST. Consider the cut of S induced by removing (a, b) ; u and v can be on the same side of the cut or opposite sides. If u and v are on opposite sides of the cut then $S' = S - (a, b) + (u, v)$ is a lower cost spanning tree. If S is not of the form $S = T - (u, v) + (a, b)$ then $S' \neq T$ and S cannot be the second best MST. If u and v are on the same side of the cut, then some edge $(c, d) \neq (a, b)$ on the path (WLOG) from u to a in T is not in S and therefore S is not of the form $S = T - (u, v) + (a, b)$.

Part b) An algorithm is as follows: Select the maximum edge, (u, v) , in the tree and find the partition of the graph, $A \cup B = V$ that it induces. For every entry in $A \times B$ write the value $w(u, v)$. Recurse on A and B .

Proof: Inductively, for a given tree, T , and a maximum edge, (u, v) , the maximum weight edge on paths from any element of A to B is clearly $w(u, v)$. Further that edge is not on the path between $x, y \in A$ or $x, y \in B$ and so cannot be responsible for the maximum weight on a path x to y .

Time: Inductively, finding the partition may take time $O(|V|)$ by depth first search and may be repeated $O(|V|)$ times. Independently, each element of the matrix $V \times V$ is written once and there are $O(|V|^2)$ entries. These two times are additive and the total time is $O(|V|^2)$.

Part c) Augment the above algorithm: maintain the overall maximum each time an entry $a, b \in V \times V$ is written of $w(a, b) - w(u, v)$ where (u, v) is the edge that partitioned a and b . This maximum at the end of the algorithm is the difference between the MST and 2nd-MST. The edge that is added and the edge removed can be updated when the maximum is updated. (Note: $w(a, b) = \infty$ for $(a, b) \notin E$)

Proof: This algorithm tries adding each edge in the graph to T and removes the maximum edge on the cycle induced. By (a) we know that the second best MST has only one edge not in T , (a, b) , and one edge missing, (u, v) . Clearly (u, v) must come from the path a to b otherwise there is a cycle in T' . Further, it is the maximum on this path. If not we could add the maximum instead and get a smaller tree T'' and T' is not the 2nd-MST.

Time: Only constant work added to each step in (b); $O(|V|^2)$

2. **Minimum max-weight path**

Solution

To solve this problem, we use Dijkstra's algorithm with slight modification. Instead of maintaining shortest path length to a node, we maintain the bottleneck (i.e. the minimum over paths of the maximum edge weight in the path). This can be done by changing the relaxation function to be:

Relax(u, v, w)

- (a) if $d[v] > \max(d[u], w(u, v))$
- (b) then $d[v] \leftarrow \max(d[u], w(u, v))$
- (c) $\pi[v] \leftarrow u$

Proof: (Let's just look at the induction step... the relaxation lemmas and other parts are very similar to Dijkstra's) Inductively assume all vertices in S satisfy $d[u] = \delta(s, u)$ and have $\delta(s, u) \leq \delta(s, v)$ for $v \notin S$. When we add a vertex we want this to still hold (base case $S = \{s\}$ it is clear). Assume it does not; we have two conditions that could fail. First, assume $d[v] \neq \delta(s, v)$. Clearly we have found a path as good as $d[v]$, so $\delta(s, v)$ must be smaller than $d[v]$. At some point, $x \neq v$, this path must first exit S . When it does it must use an edge $(u, x) : u \in S$. By the selection of v this means that $\delta(s, u) \geq d[v]$. Further, since the *bottleneck* of a path is monotonically decreasing as the path grows, the value of this path when it reaches v is at least $d[v]$ and so $\delta(u, v) \geq d[v]$, contradicting our assumption. The other possible failure is if the δ -inequality fails. This would mean that there is an $x \notin S$ with $\delta(s, x) < \delta(s, v)$, since v was the only change to S . Again, a path that connects s and x must leave S at some point, but by the same reasoning as above (on paths to x instead of v) we have that $\delta(s, x) \geq \delta(s, v)$ contradicting our assumption. Therefore the two invariants of S hold through the algorithm.