Problem Set 3

This problem set is due at **10:00 am** on **Tuesday, February 14th**.

Problem 3- 1: Updating MST after edge addition

Given an undirected, connected, weighted graph $G = (V, E)$, a minimum spanning tree T of G, and $e \in V \times V - E$, we would like to find a minimum spanning tree of the graph $G' = (V, E \cup \{e\})$. Show an algorithm for this task that is as efficient as possible. Analyze the run-time of the algorithm and prove its correctness.

The algorithm is as follows: let the edge to be added be $e = (u, v)$. Using BFS, find the path in T from u to v; let the set of edges in that path be p. Let $e' = (u', v')$ be the edge of maximum weight in $p \cup \{e\}$. Augment T with e and remove e', forming T'. T' is the MST of G'. **Solution:**
The runtime of BFS is $O(|V| + |E'|) = O(|V| + |E| + 1) = O(|V| + |E|)$. All other operations in the algorithm are constant, so total runtime is $O(|V| + |E|)$.

Lemma 1 *The e' chosen by the algorithm is not in the MST of G'.*

Proof. Because T is a spanning tree over V, there must exists a unique path in T that starts at v and ends at u; let that path be p, such that $v \to^p u$. Note that $p \cup \{e\}$ is therefore a cycle because $e = (u, v)$. The heaviest edge in $p \cup \{e\}$ is e'. $p \cup \{e\} \subseteq E'$, so by the cycle lemma of MSTs, e' is not in any MST of G'.

Lemma 2 $w(T') \leq w(T)$

Proof. Because e and e' are in the same cycle, and e' is the heaviest edge in the cycle, $w(e') \geq w(e)$. $w(T') = w(T) - w(e') + w(e) \leq w(T) - w(e) + w(e) = w(T)$.

Lemma 3 *The T' chosen by the algorithm is a spanning tree of G'*

Proof. First, note that because T is a spanning tree of G', T spans all the vertices of G'. If $e = e'$, then $T' = T$ and also spans the vertices of G'. If $e \neq e'$, let U and V be the connected components of $T - e'$ such that $u' \in U$ and $v' \in V$. Note that there are only two connected components in $T - e'$, because only one edge was removed. Because $e = (u, v)$ and $e' = (u', v')$ are in the same cycle, wlog, there must exist a path from u to u' and v to v' that does not include e'. Thus, $u \in U$ and $v \in V$. This implies that e spans the cut between U and V, so $T' = T - e' + e$ is connected.

$|T'| = |T| = |V| - 1$, and T' is connected, so T' is a spanning tree of G'.

Theorem 4 *The T' chosen by the algorithm is a MST of G'.*

Proof. Suppose for the sake of contradiction there exists an MST S' of G' such that $w(S') < w(T')$. Either $e \notin S'$ or $e \in S'$.

Case 1: $e \notin S'$, so S' is a spanning tree of G. By lemma 2, $w(T') \leq w(T)$, so $w(S') < w(T)$, which is a contradiction because T is a MST of G.

Case 2: $e \in S$. By lemma 1, $e' \notin S$, so $S = S' - e + e'$ is a spanning tree of both G' and G. $w(S) = w(S') - w(e) + w(e') < w(T') - w(e) + w(e') = w(T)$, so $w(S') < w(T)$. This is a contradiction because T is a MST of G.

Problem 3- 2: Shortest paths and spanning trees

- Prove or disprove. Given an undirected, connected, weighted graph, if you run an MST algorithm on the graph, then the weight of the shortest path between any two vertices is the weight of the unique path between them in the tree.
  Solution:
  False. Let $G = (V, E)$ and $V = \{v_1, v_2, v_3\}$, $E = V \times V$, $w(v_1, v_2) = 2$, $w(v_2, v_3) = 3$, $w(v_3, v_1) = 4$. The MST of G is $\{(v_1, v_2), (v_2, v_3)\}$. The shortest path from v_3 to v_1 is simply (v_3, v_1), but that is not the path in the MST.
- Prove or disprove. Let $G = (V, E)$ be an undirected, connected, weighted graph. Suppose we run Dijkstra's algorithm on G starting a vertex $u \in V$. For every vertex $v \in V$ let p_v denote the edges of the shortest path from u to v that is found by Dijkstra. Let $E' = \cup_{v \in V} p_v$.
  (a) (V, E') is a spanning tree of G;
  Solution:
  True. First, Dijkstra's algorithm will find a path from u to v for all v because G is connected. Thus, the union of paths will span all v.
  Second, the union of paths will not contain a cycle. Suppose for the sake of contradiction there exists a cycle in the graph. Let u be the vertex in the graph with minimal $d(u)$, and v be the vertex in the graph with maximal $d(v)$. This means Dijkstra found two different paths from u to v. However, if these paths are the different lengths, one will be longer than the other, which is a contradiction. If they have the same length, a deterministic implementation of Dijkstra's algorithm will always choose one over the other, which is a contradiction.
  (b) (V, E') is a minimum spanning tree of G.
  Solution:
  False. Consider the graph in part 1, running Dijkstra's algorithm starting from v_3. The union of paths will be $\{(v_3, v_1), (v_2, v_3)\}$, which is not the MST of G.

Problem 3- 3: Median finding

You are given two *sorted* lists of numbers $l_1 = [a_1, \ldots, a_n], l_2 = [b_1, \ldots, b_m]$. Describe a divide and conquer algorithm that is as efficient as possible to find the median of the numbers $a_1, \ldots, a_n, b_1, \ldots, b_m$. Write down your algorithm and prove that it is correct. Find the recurrence relation for the runtime of the algorithm and solve it.

Solution:

This is the solution for the $m = n$ version. For the $m \neq n$ version you just need to ensure that you delete the same number of elements from either end. The algorithm is as follows:

(a) Calculate the medians m_1 and m_2 of the input lists.

(b) If $m_1 = m_2$ output m_1

(c) If $m_1 > m_2$, then median is present in one of $l_1[n/2 - 1 : n]$ or $l_2[0 : n/2]$.

(d) If $m_1 < m_2$, then the median is present in one of $l_2[0 : n/2]$ or $l_1[n/2 : n]$.

(e) Repeat the above process until size of both the subarrays becomes 3 or less.

(f) Sort the length 6 array and output the median.

To see that this works, observe that the median of $2n$ elements must have at most n elements greater than it and n elements less than it. If $m_1 < m_2$ then m_1 has at least n elements greater than it (everything greater than it in l_1 and then all elements from m_2 to the end of l_2). So the median cannot be less than m_1 similarly the median cannot be greater than m_2. If $m_1 = m_2$ then they satisfy the property of the median and you can simply output those. Every time we make a recursive call we remove half the elements smaller than the median and half the elements larger than the median. This results in a new list of size half the original in which the median remains the same.

The recurrence relation here is given by $T(n) = T(\lceil \frac{n}{2} \rceil) + O(1)$. This can be solved by induction (which should be written down) to get $T(n) = O(\log(n))$. It is not immediately clear that the master theorem will work here, because of the ceiling, however, this is true, and one can get an estimate of this by seeing that the recurrence would correspond to the second case of the master theorem with $k = 0, a = 1, b = 2$.