CIS 121
Solutions to Practice Problems for Exam 2
March 20, 2018

---

1. Prove or disprove: You are given a connected undirected graph $G = (V, E)$ with a weight function w defined over its edges. Let $s \in V$ be an arbitrary vertex in G. Starting at vertex s, if you do a depth-first search (DFS) in G such that the edges going out of any vertex are always explored in increasing order of weight, then the resulting DFS tree is also a minimum spanning tree.

Solution. The assertion is **false**. Consider the graph $G = (V, E)$ where $V = \{a, b, c\}$ and $E = \{(a, b), (b, c), (a, c)\}$. The weight function is $w(a, b) = 1, w(a, c) = 2$ and $w(b, c) = 3$. If we perform a DFS from the vertex a using the above rule, the unique DFS tree is given by the edges $\{(a, b), (b, c)\}$, and its weight is 4. On the other hand, the MST is formed by edges $\{(a, b), (a, c)\}$, and has a weight of 3.

2. You are given an input stream which will display n integers, and you only get to view each element once. Design an efficient algorithm which will find the k largest elements in the stream, using at most $O(k)$ space (assume $k << n$).

Solution We create a min-heap H and keep track of how many elements are added. As we receive elements from the stream, while $|H| < k$ we just add the element to the heap. If $|H| = k$, then we compare the next element to the minimum element of the heap. If it is larger, we add it to the heap and then remove the minimum element.

We prove that the algorithm is correct by doing induction on the number of elements we have seen so far, m. We claim that for each m, the heap H contains the k largest elements from the first m elements in the stream. For $m \leq k$ this is clearly true. Suppose for $m = i$ we have that H contains the k largest elements of the first i elements, with $i > k$. The k largest elements for $m = i + 1$ must be a subset of the k elements currently in H plus the $(i + 1)$th element, and we can find this optimum set by comparing the minimum element of H with the $(i + 1)$th element.

The algorithm maintains the invariant $|H| \leq k$, so on each element we spend at most $O(\log k)$ time. Hence the runtime is $O(n \log k)$. Because we are storing only k elements in H, the space used is $O(k)$.

3. Prove that an edge e is contained in every spanning tree for a connected graph G if and only if removal of e disconnects G

Solution. We will first prove that if e is contained in every spanning tree of a connected graph G then removal of e disconnects G. We will prove the claim by proving its contrapositive. Assume that removal of e does not disconnect G. Let $G' = G - e$. Since G' is connected, G' has a spanning tree T'. Observe that T' is a also spanning tree of G since it contains every vertex in G. Moreover, T' does not contain e since G' does not contain e. Thus T' is a spanning tree of G that does not contain e.

We will now prove that if removal of e disconnects G then e must be contained in every spanning tree of G. We will prove the claim by proving its contrapositive. Assume that $e = (u, v)$ is not contained in every spanning tree of G. Let T be a spanning tree of G that does not contain e. Then $T + e$ has a cycle C containing e. This means that e is part of the cycle C in G. Then removing e does not disconnect G since any path that used the edge (u, v) can now use the alternate path from u to v along C.

4. Let $G = (V, E)$ be a strongly connected directed graph and let T be a DFS tree in G. Prove that if all the forward edges in G, with respect to T, are removed from G, the resulting graph is still strongly connected.

Solution. Let $e = (u, v)$ be a forward edge (with respect to the DFS tree T) in G. By definition, there must be a path from u to v in T (this path consists of tree edges only). This means that if we remove a forward edge $e = (u, v)$ from G, the resulting graph still has a path from u to v. Thus, if a path P in G contains a forward edge $e = (u, v)$, we can replace e in P by the path containing tree edges from u to v in T. In other words, even after we remove the forward edges from G, the resulting graph remains strongly connected.

5. Give an example of a weighted connected undirected graph $G = (V, E)$ and a vertex v such that the minimum spanning tree of G is different than the shortest path tree rooted at v. Can the two trees be completely disjoint?

Solution. They can't be completely disjoint as the smallest edge incident on v will be the same in both trees (assuming that the smallest weight edge incident on v is unique). The two trees can be different though, as the following example shows: consider a graph G that is a cycle on n vertices $\{v_0, v_1, v_2, \ldots, v_{n-1}\}$. Let the edge $e = (v_0, v_{n-1})$ have a weight of $n - 2$ and all other edges in $\{(v_0, v_1), (v_1, v_2), \ldots, (v_{n-2}, v_{n-1})\}$ have a weight of 1. The shortest path rooted at v_0 will contain the edge e, whereas the minimum spanning tree will not contain e.