Quiz #2
Introduction to Algorithms/Algorithms 1
600.363/463

Thursday, April 3rd, 9:00-10:15am

Solutions

1 Problem 1 (20 points)

Give a definition of minimum spanning tree.

1.1 Solution

A spanning tree of graph $G = (V, E)$ is a subgraph T of G with $T = (V, E')$, with the constraint that T is a tree. That is, a spanning tree is a tree that includes every nodes in G and whose edges are a subset of those of G. A minimum spanning tree of G is the spanning tree with the smallest total edge weight, given by $w(T) = \sum_{e \in T.E} w(e)$.

2 Problem 2 (75 points)

2.1 (25 points– 15 for part (a), 10 for part (b)

Let $G = (V, E)$ be a directed cycle, given by $V = \{v_1, v_2, \ldots, v_n\}$, and $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n), (v_n, v_1)\}$ (that is, for each $i \in \{1, 2, \ldots, n-1\}$, $(v_i, v_{i+1}) \in E$, and $(v_n, v_1) \in E$.

(a) State whether or not the following claim is true and give a brief explanation as to why: No matter what node $v \in V$ we start DFS at, the shortest paths from v to every other node in the DFS tree are exactly the shortest paths from v to every other node in G. You **do not** need to give a formal proof here. Just give a two- or three-sentence sketch of the proof or give a counter-example.

Solution: True. Since the graph in question is a directed cycle, there is exactly one path from any one node to any other node in the graph. Thus, the path discovered by DFS from node v to any other node (which is encoded in the DFS tree as a path of tree edges) must, by necessity, be the shortest path.

(b) Suppose we make a new graph G' from G by reversing a single edge in G, so that our new graph is given by $G' = (V, E')$ where $E' = \big(E \cup \{(v_{i+1}, v_i)\}\big) \setminus \{(v_i, v_{i+1})\}$. State whether or not the following claim is true, and give a brief explanation as to why: No matter what node $v \in V$ we start DFS at, the shortest paths from v to every other node in the DFS tree are exactly the shortest paths from v to every other node in G'. You **do not** need to give a formal proof here. Just give a two- or three-sentence sketch of the proof or give a counter-example.

Solution: False. Number the nodes in the cycle starting at some node v_1 and proceeding clock-wise. Node reverse the edge (v_n, v_1). The resulting graph is such that the shortest path from v_1 to v_n is simply the edge (v_1, v_n), but DFS starting at v_1 will not include this edge in the DFS tree if it follows edge (v_1, v_2) before it follows (v_1, v_n).

2.2 (25 points)

Let us say that the Bellman-Ford algorithm *pseudo-terminates* if after the first iteration is finished, no more values change for the remainder of the algorithm. (i.e., no node v has its $v.d$ or $v.\pi$ attributes updated during iterations 2 through $|V|-1$ of the outer for-loop). That is, Bellman-Ford pseudo-terminates on a graph $G = (V, E)$ if after the first iteration of the for loop, we have already found all of the shortest paths.

(a) Suppose that we have a directed path on n nodes, with unit edge weights. Let v denote the unique node in this path that has no predecessors. If we perform Bellman-Ford, and consider the vertices in G in topological order, will Bellman-Ford pseudo-terminate? Explain. You **do not** need to give a rigorous proof, here. A brief explanation and/or counter-example will suffice.

Note: this question was clarified during the exam to mean that we use the topological ordering of the vertices to determine an order in which to consider the edges in the inner for-loop.

Solution: True. As clafified in the exam, Bellman-Ford here is assumed to consider the edges in topological order– number the nodes in topological order as $v_1, v_2, \ldots, v_n$, then the edges can be ordered as $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$. If Bellman-Ford considers the edges in this order, then at the time we consider edge (v_i, v_{i+1}) during the first iteration of the outer for-loop, we have already considered edges $(v_1, v_2), (v_2, v_3), \ldots, (v_{i-1}, v_i)$, so Bellman-Ford will have already called RELAX on each of these edges, in order. Since this sequence of edges is the shortest path to v_i, when Bellman-Ford calls $\mathrm{RELAX}(v_i, v_{i+1}, w)$, the result will be that we have found the shortest path to v_{i+1}. Thus, after the first time through the edges, we will have already found all of the shortest paths.

(b) As before, suppose that we have a directed path on n nodes, with unit edge weights and let v denote the unique node in this path that has no predecessors. If we perform Bellman-Ford, and v is *not* the first node considered in the outer for-loop, will Bellman-Ford pseudo-terminate? Again, you need not give a rigorous proof. Explain. You **do not** need to give a rigorous proof, here. A brief explanation and/or counter-example will suffice.

Note: this question was clarified during the exam to mean that we use the topological ordering of the vertices to determine an order in which to consider the edges in the inner for-loop.

Solution In the event that we do not consider the edge (v_1, v_2) first, then during the first iteration of the for-loop, the distance between v_1 and at least one other

node remains at $+\infty$ for the entirety of the first iteration– any node v_i for which (v_{i-1}, v_i) is considered before (v_1, v_2) is certain to have this property.

2.3 (25 points)

Let $G = (V, E)$ be a connected, undirected graph with non-negative edge weights. Prove or disprove the following claim: if edge $e \in E$ is the unique smallest-weight edge in graph G, then e must be included in any minimum spanning tree of G.

Solution Suppose that there exists a minimum spanning tree T for graph G with $e \notin T$. Now consider $T \cup \{e\}$. Since T is a tree and $e \notin T$, $T \cup \{e\}$ contains a cycle. Consider some other edge $e' \neq e$ on this cycle. By assumption, $w(e') > w(e)$. Removing e' from T results in a new spanning tree, with weight

$$\begin{aligned} w(T \cup \{e\} - \{e'\}) &= w(T) + w(e) - w(e') \\ &< w(T), \end{aligned}$$

contradicting the assumption that spanning tree T exists.

3 Problem 3 (55 points)

Given a weighted directed graph $G = (V, E)$ with non-negative edge weights, call node $w \in V$ a *critical* node for nodes $u \in V$ and $v \in V$ if for all shortest paths P from u to v in G, P visits node w. Give a polynomial-time algorithm that checks whether or not there is a critical node for nodes u and v in graph G. If one or more critical nodes exist for nodes u and v, your algorithm should return one. If no critical node exists for nodes u and v, return `None`. Prove the correctness of your algorithm and prove that it runs in polynomial time.

Solution It is simple enough to run Dijkstra's algorithm once to find a shortest path from u to v. Call the length of this shortest path δ. Then, for every node t on the shortest path with $t \neq u, t \neq v$, remove t from G and run Dijkstra's algorithm on the resulting graph to find the shortest path from u to v. If the length of this path is larger than δ, then we know that removing t forced us to use a longer shortest path, which implies that all shortest paths in G (i.e., all paths from u to v with length δ) must have used node t. The correctness of this algorithm is more or less immediately from the explanation just given, but it remains to establish that it runs in polynomial time. Polynomial runtime follows from the fact that there can be at most $O(|V|)$ nodes in the shortest path from u to v. Thus, we must run Dijkstra's algorithm at most $O(|V|)$ times in total, for a total runtime of $O\left(|V|(|V|\log|V| + |E|)\right)$, which is indeed polynomial in the size of the graph.