CS 170 DIS 04

Released on 2019-02-11

1 Short Answer

For each of the following, either prove the statement is true or give a counterexample to show it is false.

(a) If (u, v) is an edge in an undirected graph and during DFS, $\text{post}(v) < \text{post}(u)$, then u is an ancestor of v in the DFS tree.

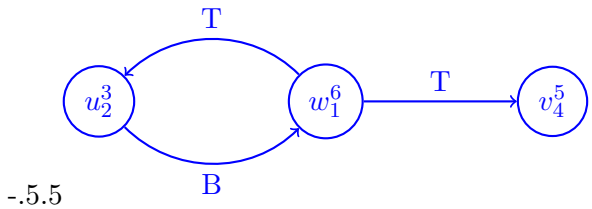
(b) In a directed graph, if there is a path from u to v and $\text{pre}(u) < \text{pre}(v)$ then u is an ancestor of v in the DFS tree.

(c) In any connected undirected graph G there is a vertex whose removal leaves G connected.

Solution:

(a) True. There are two possible cases: $\text{pre}(u) < \text{pre}(v) < \text{post}(v) < \text{post}(u)$ or $\text{pre}(v) < \text{post}(v) < \text{pre}(u) < \text{post}(u)$. In the first case, u is an ancestor of v. In the second case, v was popped off the stack without looking at u. However, since there is an edge between them and we look at all neighbors of v, this cannot happen.

(b) False. Consider the following case:



(c) True. Remove a leaf of a DFS tree of the graph.

2 Midterm Prep: Dijkstra Tiebreaking

We are given a directed graph G with positive weights on its edges. We wish to find a shortest path from s to t, and, among all shortest paths, we want the one in which the longest edge is as short as possible. How would you modify Dijkstra's algorithm to this end?

Solution: Modify Dijkstra's algorithm to keep a map $\ell(v)$ which holds the longest edge on the current shortest path to v. Initially $\ell(s) := 0$ for source s and $\ell(v) := \infty$ for all $v \in V \ \{s\}$. When we consider a vertex u and its neighbour v, if $dist(u) + w(u, v) < dist(v)$, then we set $\ell(v) := \max(\ell(u), w(u, v))$ in addition to the standard Dijkstra's steps. If there is a neighbour v of u such that $dist(v) = dist(u) + w(u, v)$, *and* $\ell(v) > \max(\ell(u), w(u, v))$, we set v's predecessor to u and update $\ell(v) := \max(\ell(u), w(u, v))$.

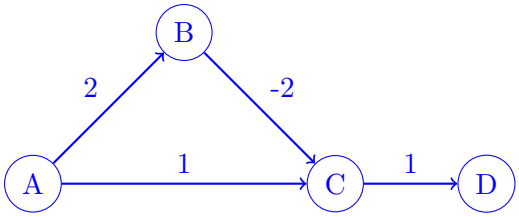
3 Dijkstra's Algorithm Fails on Negative Edges

Draw a graph with five vertices or fewer, and indicate the source where Dijkstra's algorithm will be started from.

1. Draw a graph with no negative cycles for which Dijkstra's algorithm produces the wrong answer.
2. Draw a graph with at least two negative weight edge for which Dijkstra's algorithm produces the correct answer.

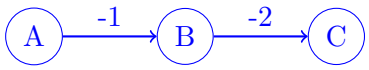
Solution:

1. Here's one example:



Dijkstra's algorithm from source A will give the distance to D as 2 rather than 1, because it visits C before B.

2. Dijkstra's algorithm always works on directed paths. For example:



4 Midterm Prep: Graph Short Answer

Answer each question below *concisely* (one short sentence or a number should suffice). Do not justify your answer. Do not show your work.

(a) Suppose we are given a directed graph $G = (V, E)$ represented in adjacency list format, and we want to test whether G is a dag or not, using a method that is as asymptotically efficient as possible. In a sentence, what approach would you use?

Solution: Either: Use DFS, check for back-edges.

Or: Decompose into strongly connected components, check for a SCC with more than one vertex.

(b) What's the running time of your solution in (a), using $O(\cdot)$ notation?

Solution: $O(|V| + |E|)$.

(c) Let $G = (V, E)$ be a directed graph with $|V| = 1000$ vertices, $|E| = 5000$ edges, and 700 strongly connected components. How many vertices does the metagraph have?

Solution: 700.

Comment: Each vertex in the metagraph corresponds to a strongly connected component in G, so the number of vertices in the metagraph is the same as the number of SCCs in G.

(d) Let $G = (V, E)$ be a dag. Let s be a source vertex in G. Suppose we set the weight of each edge to 1 and run Dijkstra's algorithm to compute the distance from s to each vertex $v \in V$, and then order the vertices in increasing order of their distance from s. Are we guaranteed that this is a valid topological sort of G?

Circle YES or NO.

Solution: No.

(e) Justify your answer to part (d) as follows: If you circled YES, then give one sentence that explains the main idea in a proof of this fact. If you circled NO, then give a small counterexample (a graph with at most 4 vertices) that disproves it.

Solution: $V = \{a, b, c, d\}$, $E = \{(a, b), (b, c), (c, d), (a, d)\}$, $w(e) = 1$ for $e \in E$. **Comment:** Distances are $a : 0, b : 1, c : 2, d : 1$ but $c < d$ in any topological ordering.

(f) Suppose we run Dijkstra's algorithm on a graph with n vertices and $O(n \lg n)$ edges. Assume the graph is represented in adjacency list representation. What's the asymptotic running time of Dijkstra's algorithm, in this case, if we use a binary heap for our priority queue? Express your answer as a function of n, and use $O(\cdot)$ notation.

Solution: $O(n(\lg n)^2)$.

Comment: $|V| = n$, $|E| = O(n \lg n)$, and Dijkstra's runs in $O((|V| + |E|) \lg |V|)$ time, which is $O((n + O(n \lg n)) \lg n) = O((n \lg n) \times \lg n) = O(n(\lg n)^2)$.

5 The Greatest Roads in America

Arguably, one of the best things to do in America is take a great American road trip. And in America there are some amazing roads to drive on (think Pacific Crest Highway, Route 66, etc). An intrepid traveller has chosen to set course across America in search of some amazing driving. What is the length of the shortest path that hits at least k of these amazing roads?

Assume, that the roads in America can be expressed as a directed weighted graph $G = (V, E, d)$ and that our traveller wishes to drive across at least k roads from the subset $R \subset E$ of 'amazing' roads. Furthmore, assume that the traveller starts and ends at her home $h \in V$. Also, you can assume that the traveller is OK with repeating roads from R i.e. the k roads she chooses from R do not need to be unique.

Provide a 4 part solution with runtime in terms of $n = |V|, m = |E|, k$, and $r = |R|$.

Hint: First try out $k = 1$. How can G be modified so that we can use a 'common' algorithm to solve the problem?

Solution: The main idea is that we want to build a new graph G' such that we can apply Dijkstra's algorithm on this new graph. We start by creating $k + 1$ copies of the graph G, where each copy represents how many 'amazing' roads the traveller has crossed. We modify the special edges so that they now cross between the various copies. Then we apply Dijkstra's to find the distance between h in the 0th copy and h in the kth copy.

Pseudocode

Generate $k+1$ copies of the graph G. Call these copies $G_0, \ldots, G_k$, and let $R_0, \ldots, R_k$ be their respective amazing roads. For each edge $r_i = (u_i \to v_i) \in R_i$ for $i = 0, \ldots, k-1$, modify the edge to be between u_i and v_{i+1}. Let the entire graph be G'. Run Dijkstra's algorithm on G' starting from h in G_0 and ending at h in G_k.

Runtime Since G' includes k copies of G, Dijkstra's algorithm will run in time $O((km + kn)\log(kn))$. Note $k \leq m$ and $\log m = O(\log n)$, so the runtime is $O(k(m+n)\log n)$.

Correctness Assume there is a valid shorter path p in G than the one produced by this algorithm. Consider the equivalent path p' in G' formed by modifying the path to go to the next copy of G whenever crossing an edge of R. Since p is valid, p' must go from h in G_0 to h in G_k. But then p' would be a shorter path in G' than the one produced by Dijkstra's, a contradiction.