Homework 7

Aleck Zhao

November 2, 2017

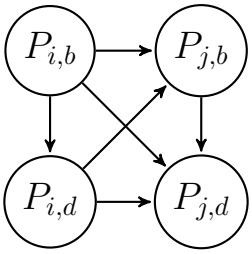
1 Consistent Lifetimes (34 points)

Suppose that you are an historian, and you are trying to figure out possible dates for when various historical figures may have lived. In particular, there are n people $P_1, P_2, \ldots, P_n$ who you are studying. Based on your research, you have discovered a collection of m facts about when these people lived relative to each other. Each of these facts has one of the following two forms:

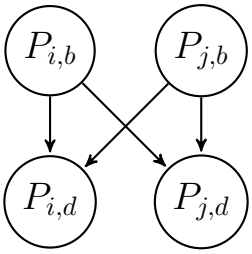
- For some i and j, person P_i died before person P_j was born; or
- For some i and j, the lifetimes of P_i and P_j overlapped.

Unfortunately, since the historical record is never fully trustworthy, it's possible that some of these facts are incorrect. Design an $O(n+m)$ time algorithm to at least check whether they are *internally consistent*, i.e., whether there is a birth and a death date for each person so that all of the facts are true. As always, prove running time and correctness (i.e., prove that if there is a possible set of dates then your algorithm will return yes, and if no such dates are possible then your algorithm will return no).

Solution. For each person P_i, create vertices $P_{i,b}$ and $P_{i,d}$, representing their birth and death dates, respectively. For each fact, if it states that person P_i died before person P_j, then add the following edges:



where an edge from v to u indicates that v happened chronologically before u. Similarly, if the fact states that the lifetimes of P_i and P_j overlapped, then add the following edges:



If there exists a directed cycle in the graph, say $(v_1, v_2, \cdots, v_n, v_1)$, then v_1 occurred earlier than v_n which occurred earlier than v_1, which is an inconsistency. On the other hand, this is the only way for an inconsistency to happen, so directed cycles are equivalent to inconsistencies.

Constructing the graph, we add $2n$ vertices and at most $6m$ edges, which is $O(n+m)$. Then we perform a DFS on this graph, being sure to perform it on every connected component, which takes $O(n+m)$ if we keep track of which vertices have been visited already. If there is a back edge, then there is a directed cycle and therefore an inconsistency, and if not, then the data is consistent. □

2 Minimum Spanning Trees (33 points)

(a) Suppose that we are given a connected graph $G = (V, E)$ where all edge weights are distinct. Prove that there is a *unique* minimum spanning tree.

Proof. Suppose there are two distinct minimum spanning trees T_1 and T_2. Then there are edges that are either in T_1 and not T_2 or T_2 but not in T_1. Take the edge e of minimum weight in this set, and WLOG $e \in T_2$. Suppose $e = \{x, y\}$. Then consider the path from x to y in T_1. There must exist an edge e' on this path not in T_2, since if they were all in T_2, there would be a cycle. Since $e \notin T_1$, it follows that $T' = T_1 \cup \{e\} \setminus \{e'\}$ is also a spanning tree. Now, $\{a, b\}$ is in only one of T_1 and T_2, so

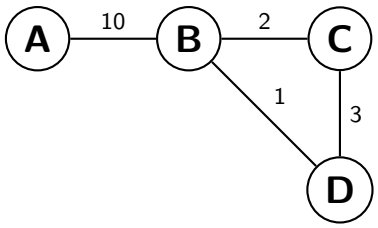
$$w(\{a, b\}) > w(e) \implies w(T') = w(T_1) + w(e) - w(\{a, b\}) < w(T_1)$$

which is a contradiction, since T_1 was assumed to be an MST. Thus, the MST is unique. □

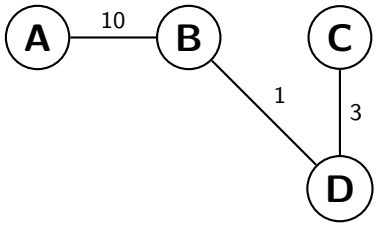
As in the previous part, suppose that we are given a connected graph $G = (V, E)$ where all edge weights are distinct. But instead of trying to construct an MST, we are trying to construct a something else. Given a spanning tree $T = (V, E')$ of G, the *bottleneck edge* of T is the edge in E' of maximum weight. A spanning tree T is a *minimum-bottleneck spanning tree* if there is no spanning tree T' of G with a lower weight bottleneck edge.

(b) Is every minimum-bottleneck spanning tree of G also a minimum spanning tree of G? Prove or give a counterexample (and explanation).

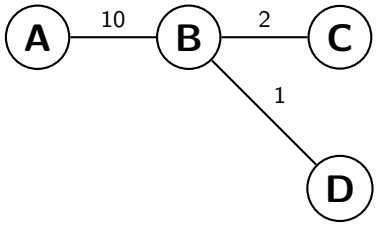
Solution. This is not true. Consider the following graph:



An MBST is given by



but the unique MST is different, as shown below:



□

(c) Is every minimum spanning tree of G a minimum-bottleneck spanning tree of G? Prove or give a counterexample (and explanation).

Proof. Let T_1 be a minimum-bottleneck spanning tree of G, and let T_2 be the unique minimum spanning tree of G. Let $e_1 \in T_1$ and $e_2 \in T_2$ be the edges of maximum weight in their respective trees. Suppose T_2 is not an MBST. Then $e_2 \notin T_1$ and $w(e_2) > w(e_1) \geq w(e)$ for any $e \in T_1$. Suppose $e_2 = \{x, y\}$, and consider any $e \in T_1$ on the path from x to y. Then $T' = T_2 \cup \{e\} \setminus \{e_2\}$ is a spanning tree where

$$w(T') = w(T_2) + w(e) - w(e_2) < w(T_2)$$

which is a contradiction since T_2 was assumed to be an MST. Thus, T_2 is also an MBST. □

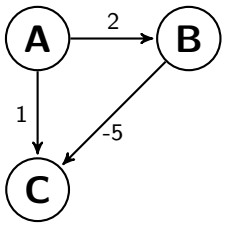
3 Shortest Paths (33 points)

(a) We saw in class that Dijkstra's algorithm for computing shortest paths in a directed graph does not work if there are negative edge lengths. Consider the following idea to fix this.

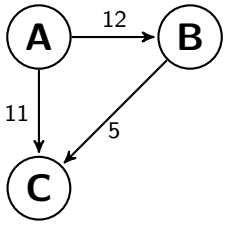
> First add a large enough positive constant to every edge weight so that the "revised" edge lengths are all positive (for example, you could add the value $1 + \max_{v \in V} |len(v)|$). To find the shortest path between two nodes s and t in the original graph, run Dijkstra's algorithm using the revised edge lengths to find the shortest path between s and t using the revised lengths, and return the path which it finds.

Does this work? That is, will this always return the shortest path under the original edge lengths? If so, give a proof. If not, give a counterexample (and explain it).

Solution. This does not work. Consider the following graph:



The shortest path from A to C is $A \to B \to C$. If we add a large constant, say 10 to each edge, then the revised graph becomes



Now, using Dijkstra's algorithm, the shortest path from A to C is just $A \to C$. □

(b) Now suppose that instead of adding the same value to every length, we instead *multiply* every length by the same value $\alpha > 0$ (note that this preserves the sign of each edge length). Let P be a shortest path from s to t under the original edge lengths. Is P still a shortest path from s to t under the new edge lengths? If yes, give a proof. If no, give a counterexample (and explain it).

Proof. This does work. Suppose $P = (s, v_1, v_2, \cdots, v_n, t)$ is a shortest path in the original graph G, and $P' = (s, u_1, u_2, \cdots, u_m, t)$ is a shortest path in the modified graph G'. Then since P is a path in G' and P' is a path in G, we have the following:

$$\begin{aligned}
len_G(P) &\leq len_G(P') \\
len_{G'}(P') &= \alpha\left[len(s,u_1) + len(u_1,u_2) + \cdots + len(u_m,t)\right] \\
&\leq len_{G'}(P) = \alpha\left[len(s,v_1) + len(v_1,v_2) + \cdots + len(v_n,t)\right] \\
\implies len(s,u_1) + len(u_1,u_2) + \cdots + len(u_m,t) &\leq len(s,v_1) + len(v_1,v_2) + \cdots + len(v_n,t) \\
\implies len_G(P') &\leq len_G(P) \\
\implies len_G(P) &= len_G(P') \\
\implies len_{G'}(P) = \alpha \cdot len_G(P) &= \alpha \cdot len_G(P') = len_{G'}(P')
\end{aligned}$$

so P is also a shortest path in G'. □