CS 170 Homework 4

Due 2019-09-25, at 10:00 pm

1 Study Group

List the names and SIDs of the members in your study group.

2 Vertex Cut

Let \( G = (V, E) \) be an undirected, unweighted graph with \( n = |V| \) vertices. The distance between two vertices \( u, v \in G \) is the length of the shortest path between them. A vertex cut of \( G \) is a subset \( S \subseteq V \) such that removing the vertices in \( S \) (as well as incident edges) disconnects \( G \).

Show that if there exist \( u, v \in G \) of distance \( d > 1 \) from each other, that there exists a vertex cut of size at most \( \frac{n-2}{d-1} \). Assume \( G \) is connected.

3 True Source

Design an efficient algorithm that given a directed graph \( G \) determines whether there is a vertex \( v \) from which every other vertex can be reached. (Hint: first solve this for directed acyclic graphs. Note that running DFS from every single vertex is not efficient.)

Please give a 3-part solution to this problem.

4 Finding Clusters

We are given a directed graph \( G = (V, E) \), where \( V = \{1, \ldots, n\} \), i.e. the vertices are integers in the range 1 to \( n \). For every vertex \( i \) we would like to compute the value \( m(i) \) defined as follows: \( m(i) \) is the smallest \( j \) such that vertex \( j \) is reachable from vertex \( i \). (As a convention, we assume that \( i \) is reachable from \( i \).) Show that the values \( m(1), \ldots, m(n) \) can be computed in \( O(|V| + |E|) \) time.

Please give a 3-part solution to this problem.

5 Disrupting a Network of Spies

Let \( G = (V, E) \) denote the “social network” of a group of spies. In other words, \( G \) is an undirected graph where each vertex \( v \in V \) corresponds to a spy, and we introduce the edge \( \{u, v\} \) if spies \( u \) and \( v \) have had contact with each other. The police would like to determine which spy they should try to capture, to disrupt the coordination of the group of spies as much as possible. More precisely, the goal is to find a single vertex \( v \in V \) whose removal from the graph splits the graph into as many different connected components as possible. This
problem will walk you through the design of a linear-time algorithm to solve this problem. In other words, the running time will be \( O(|V| + |E|) \).

In the following, let \( f(v) \) denote the number of connected components in the graph obtained after deleting vertex \( v \) from \( G \). Also, assume that the initial graph \( G \) is connected (before any vertex is deleted), has at least two vertices, and is represented in an adjacency list format.

For each part, prove that your answer is correct (some parts are simple enough that the proof can be a brief justification; others will be more involved).

(a) Let \( T \) be a tree produced by running DFS on \( G \) with root \( r \in V \). (In particular, \( T = (V, E_T) \) is a spanning tree of \( G \).) Given \( T \), find an efficient way to calculate \( f(r) \).

(b) Let \( v \in V \) be some vertex that is not the root of \( T \) (i.e., \( v \neq r \)). Suppose further that no descendant of \( v \) in \( T \) has any non-tree edge (i.e. edge in \( E \setminus E_T \)) to any ancestor of \( v \) in \( T \). How could you calculate \( f(v) \) from \( T \) in an efficient way?

(c) For \( w \in V \), let \( D_T(w) \) be the set of descendants of \( w \) in \( T \) including \( w \) itself. For a set \( S \subseteq V \), let \( N_G(S) \) be the set of neighbors of \( S \) in \( G \), i.e. \( N_G(S) = \{ y \in V : \exists x \in S \text{ s.t. } \{x, y\} \in E \} \). We define \( \text{up}_T(w) := \min_{y \in N_G(D_T(w))} \text{depth}_T(y) \), i.e. the smallest depth in \( T \) of any neighbor in \( G \) of any descendant of \( w \) in \( T \).

Now suppose \( v \) is an arbitrary non-root node in \( T \), with children \( w_1, \ldots, w_k \). Describe how to compute \( f(v) \) as a function of \( k, \text{up}_T(w_1), \ldots, \text{up}_T(w_k), \) and \( \text{depth}_T(v) \).

Hint: Think about what happened in part (b); think about what changes when we can have non-tree edges that go up from one of \( v \)'s descendants to one of \( v \)'s ancestors, and think about how you can detect it from the information provided.

(d) Design an algorithm which, on input \( G, T \), computes \( \text{up}_T(v) \) for all vertices \( v \in V \), in linear time.

(e) Given \( G \), describe how to compute \( f(v) \) for all vertices \( v \in V \), in linear time.

6 All Roads Lead to Rome

You are the chief trade minister under Emperor Caesar Augustus with the job of directing trade in the ancient world. The Emperor has proclaimed that all roads lead to (and from) Rome; that is, all trade must go through Rome. In particular, you are given a strongly connected directed graph \( G = (V, E) \) with positive edge weights, and there is a particular node \( v_0 \in V \) (Rome).

(a) Give an efficient algorithm for finding shortest paths between all pairs of nodes, with the one restriction that these paths must all pass through \( v_0 \) (Rome). Make your algorithm as efficient as you can (perhaps as fast as Dijkstra’s algorithm).

Please give a 3-part solution.
(b) Occasionally, Augustus will ask you for the (smallest) distance between two vertices. You want to do this as quickly as possible, so that Augustus does not have your head.

This is called a distance query: Given a pair of vertices \((u, v)\), give the distance of the shortest path from \(u\) to \(v\) that passes through \(v_0\). Describe how you might store the results such that you require \(O(|V|)\) storage, and you can compute the result in \(O(1)\) time. For your answer, a clear description of the data structure and its usage is sufficient.

(c) On the other hand, the traders need to know the paths themselves.

This is called a path query: Given a pair of vertices \((u, v)\), give the shortest path from \(u\) to \(v\) that passes through \(v_0\). Describe how you might store the results such that you require \(O(|V|)\) storage, and you can compute the result in \(O(|V|)\) time. Again, a clear description of the data structure and its usage is sufficient.

7 The Greatest Roads in America

Arguably, one of the best things to do in America is to 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 traveler 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 traveler wishes to drive across at least \(k\) roads from the subset \(R \subseteq E\) of “amazing” roads. Furthermore, assume that the traveler starts and ends at her home \(h \in V\). You may also assume that the traveler is fine with repeating roads from \(R\), i.e. the \(k\) roads chosen from \(R\) need not be unique.

Provide a 3-part solution with runtime in terms of \(n = |V|, m = |E|, k, \text{ and } r = |R|\).

Hint: First consider \(k = 1\). How can \(G\) be modified so that we can use a “common” algorithm to solve the problem?