EECS4020 Design and Analysis of Algorithms

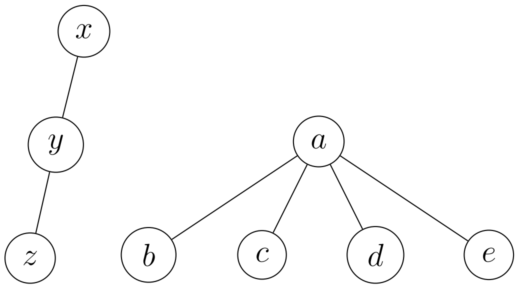
Homework 5

(On a voluntary basis: No need to submit)

You are encouraged to work in groups. Please submit your solution (of any question) to our tutors on or before May 31, 2022, so that they may have a chance to give feedback of your work.

I. Disjoint Sets

1. John has maintained a union-find data structure for a set using trees. The following is the current status of the trees, so that the set is currently partitioned into two subsets:



   (a) Suppose John is using Union-By-Size strategy to perform `union`, and the Path-Compression heuristic to perform `find`. Describe what will happen after (i) `union`(a, x), and (ii) then `find`(y).

   (b) Suppose John is using Union-By-Rank strategy to perform `union`, and the Path-Compression heuristic to perform `find`. Suppose the rank of x is 2 and the rank of a is 1. Describe what will happen after (i) `union`(a, x), and (ii) then `find`(y).

2. John has maintained another union-find data structure for a set using trees. At this moment, the trees contain k edges in total. Show that if John next performs any n `find` operations, together with the path compression heuristic, the total time for these `find` operations is bounded by $O(n + k)$.

II. BFS and DFS

1. Let $T = (V, E)$ be an undirected unweighted tree. For any two vertices u and v, the distance between them, denoted by `dist`(u, v), is the number of edges on the (unique) simple path that connects u and v. The *diameter* of T is defined as:

$$\texttt{diameter}(T) = \max_{u,v \in V} \texttt{dist}(u, v).$$

   Show how to compute `diameter`(T) in $O(|V|)$ time.

   Hint: One way is by BFS + DP. Another (magical) way is to run BFS twice.

2. Let $G = (V, E)$ be a connected, undirected graph. An *articulation point* of G is a vertex whose removal will disconnect G. Suppose we perform DFS on G, and let T be the resulting DFS tree. We are going to find all articulation points of G based on T.

   (a) Prove that the root of T is an articulation point of G if and only if it has at least two children in T.

   (b) When we perform DFS on a connected undirected graph, we can classify the edges into two types:

   - tree edges — those edges used in the DFS tree.
   - back edges — those edges not used in the DFS tree.

   Let v be a non-root vertex of T. Prove that v is an articulation point of G if and only if v has a child s such that there is no back edge from s or any descendant of s to a proper ancestor of v.

   (c) Let $d(x)$ denote the discovery time of a node x in T during the DFS. Let

   $$\texttt{low}[v] = \min d(w)$$

   such that (u, w) is a back edge from some descendant u of v.

   Show how to compute $\texttt{low}[v]$ for all vertices v in $O(|E|)$ time.

   (d) Show how to compute all articulation points in $O(|E|)$ time.

III. Topological Sort and Strongly Connected Components

1. Let G be a directed acyclic graph (DAG).

   In the lecture, we described how to perform topological sort of the vertices of G by running DFS on G. Here, let us examine another way to do so.

   (a) Show that there exists some vertex v of G whose in-degree is 0. That is, there is no directed edge pointing to v.

   (b) Consider the following algorithm, which removes the vertices of G, successively, if the in-degree is 0.

```
0. Initialize an array InDeg[1..n] and an empty queue Q;
1. Compute InDeg[v] of every vertex v;
2. for each vertex v
3.     if InDeg[v] = 0 then Insert v to Q;
4. while Q is not empty
5.     v ← Q.pop();  Output v;
6.     for each neighbor u of v
7.         Decrease InDeg[u] by 1;
8.         Insert u to Q if InDeg[u] is now 0;
```

   - Show that the algorithm correctly performs topological sort on G.
   - Show that the running time is linear.
   - If the input graph G is a directed graph but not a DAG, what will happen? How should we modify the algorithm to detect such a case occurs?

2. A directed graph is said to be *semi-connected* if for all pairs of vertices u and v, we have $u \leadsto v$, or $v \leadsto u$, or both. (The notation $u \leadsto v$ means u can reach v by a directed path.)

   (a) Suppose G is a directed acyclic graph with n vertices, and suppose we have performed a topological sort on G. Let v_i denote the ith vertex in the topological sort order. Show that G is semi-connected if and only if there is an edge (v_i, v_{i+1}) for all $i = 1, 2, \ldots, n-1$.

   (b) Suppose G is a general directed graph (which may contain cycles). Give an $O(|V| + |E|)$-time algorithm to check if G is semi-connected. Show that your algorithm is correct.

   Hint: Find SCC, then topological sort on component graph.