1 Disjoint-set data structure

1.1 Operations

Disjoint-set data structure enable us to efficiently perform operations such as placing elements into sets, querying whether two elements are in the same set, and merging two sets together. (Thus they’re very useful for simulating graph connectivity.) Must implement the following operations:

- **MAKESET**(x): create a new set containing the single element x.
- **UNION**(x, y): replace sets containing x and y by their union.
- **FIND**(x): return name of set containing x.

We add for convenience the function **LINK**(x, y) where x, y are roots: LINK changes the parent pointer of one of the roots to be the other root. In particular, **UNION**(x, y) = **LINK**(**FIND**(x), **FIND**(y)), so the main problem is to make the **FIND** operations efficient.

1.2 Optimization Heuristics

We have two main methods of optimization for disjoint-set data structures:

- **Union by rank.** When performing a **UNION** operation, we prefer to merge the shallower tree into the deeper tree.
- **Path compression.** After performing a **FIND** operation, we can simply attach all the nodes touched directly onto the root of the tree.

**Exercise 1** (1.0). When using neither union by rank nor path compression, what is the asymptotic runtime of **FIND**(x)? **UNION**(x, y)?

**Exercise 2.** Draw how the disjoint set data structure changes after each of the following operations using a particular heuristic:

(a) **UNION**(x, y) with union by rank.
Exercise 3. When using the union by rank optimization only, what is the asymptotic runtime of the operation FIND($x$)?

Exercise 4. Suppose that you are given an undirected friendship map for everyone at Harvard. (Vertices are students, and there is an edge between two students if they are friends.) Define a clique as a strongly connected component of this graph (note for those familiar with graph theory, this is not the typical definition of a clique on a graph).

(a) How could you (not using the union-find datastructure) determine the number of cliques?

(b) How could you solve this same problem using the union-find datastructure?
(c) How do the runtimes between the two approaches differ?

(d) What if now you are at Visitas and people are constantly making friends. You would like to know how many cliques there are at any given point in time with the knowledge that more edges are constantly being drawn on the graph. How do both approaches generalize?

(e) What about if people are also allowed to have falling outs and unfriend each other? Is there any way to modify union-find to work in this situation as well.

2 Minimum Spanning Trees

2.1 Definition and Motivation

A spanning tree is a sub-graph with all the vertices of the initial graph that is also a tree (recall that a tree is characterized by being connected and acyclic). The minimum spanning tree of a (possible weighted) graph is a spanning tree whose sum of the weights on edges is minimal.

Exercise 5. True or false?

(a) All undirected graphs have a minimum spanning tree.

(b) All connected undirected graphs have a unique minimum spanning tree.

(c) All trees have a unique minimum spanning tree.
A minimum spanning tree is generally helpful when trying to connect \( n \) objects of interest in an optimized way. For instance, if vertices represent cities and edges represent the cost of building a railroad between any two cities, then a MST gives the least possible cost of connecting all the cities. Analogous problems can be constructed for internet connectivity, electric grids, etc.

## 3 Exchange Property

In class, we saw the cut property, upon which the correctness of both Prim’s and Kruskal’s algorithms depend. We can also make an even stronger claim: the exchange property.

The exchange property states that if \( T \) and \( T' \) are spanning trees in \( G(V, E) \), then given any \( e' \in T' - T \), there exists an edge \( e \in T - T' \) such that \((T - \{e\}) \cup \{e'\}\) is also a spanning tree.

**Exercise 6.** Prove the exchange property.

**Exercise 7.** Let \( G \) be a graph. Let \( G' \) be a directed graph whose vertices are spanning trees of \( G \). Draw an edge from one vertex of \( G' \) to another if the first vertex (spanning tree) can be transformed into the other by replacing one of its edges with another edge.

(a) Show that this is effectively an undirected graph.

(b) Is this graph connected?

(c) Suppose that \( v \) is an MST. Prove that there exists a path in \( G' \) from any \( v' \) to \( v \).