Worth: 11% **Due:** Monday March 29

For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing.

1. Consider a list of cities $c_1, c_2, \ldots, c_n$. Assume we have a relation R such that, for any i, j, $R(c_i, c_j)$ is 1 if cities c_i and c_j are in the same province, and 0 otherwise.

(a) If R is stored as a table, how much space does it require?

(b) Using a Disjoint Sets ADT , write pseudo-code for an algorithm that puts each city in a set such that c_i and c_j are in the same set if and only if they are in the same province. Justify coorectness and running time of your algorithm.

(c) When the cities are stored in the Disjoint Sets ADT, if you are given two cities c_i and c_j, how do you check if they are in the same province?

(d) If we use linked-lists with union-by-weight to implement the union-find ADT, how much space do we use to store the cities?

(e) If we use trees with union-by-rank, what is the worst-case running time of the algorithm from (b) (Hint: the unions from your algorithm probably have a special form). Explain.

(f) If we use trees without union-by-rank, what is the worst-case running time of the algorithm from (b). Explain. Are there more worst-case scenarios than in (e)?

2. Give a sequence of m Make-Set, Union, and Find-Set operations, n of which are Make-Set, that takes $\Omega(m \log n)$ time when we use union by rank only.

3. Consider an implementation of the disjoint-sets ADT using rooted trees, where each node contains one member and each tree represents one set. As discussed in class, each node points only to its parent. Moreover, the root of a tree is the set representative and is its own parent. MAKE-SET(x) simply creates a tree with one node for x. FIND-SET(x) follows parent pointers until the root of the tree containing x is found. UNION(x, y) first finds the roots r_x and r_y of xs and ys trees, respectively, and then sets parent(r_y) = r_x.

a Assume we have performed n MAKE-SET operations and no other operations. Give a sequence of $n1$ UNION operations that cumulatively take time $\Omega(n^2)$. Justify your answer.

b Assume we have performed n MAKE-SET operations and no other operations. Give a sequence consisting rst of $n1$ UNION operations and then n unique FIND-SET operations (i.e., each element is searched for exactly once) such that (i) the UNION operations cumulatively take time $O(n)$ and (ii) the FIND-SET operations cumulatively take time $\Omega(n)^2)$. Justify your answer.

4. A bipartite graph $G = (V, E)$ is a graph where V can be partitioned into two sets V_1 and V_2 (one of which may be empty) such that there are no edges between any two nodes in V_1 or between any two nodes in V_2.

1. Show that any graph that contains an odd cycle is not bipartite.
2. Show how to modify DFS so that, given any graph, it creates a partition like the one described above (that is, it assigns to each node either a 1 or a 2) when the graph is bipartite, and it outputs "not bipartite" when the graph is not bipartite.
3. Is it true that any graph that does not contain an odd cycle is bipartite?