Problem Set 4: Union Find

1. (EASY) Show that the Union by Rank algorithm requires $O(m \log n)$ time for a sequence of m union and find operations.

2. (EASY) Consider the following Union-find algorithm. MAKESINGLETON(u) make a node u. FIND(u) returns the root of the tree containg u. UNION((u, v) finds the root of the tree containing u and v, say r_u and r_v. It then makes r_v the child of r_u. Show that there is a sequence of:
   (a) $n - 1$ union operations that take $\Omega(n^2)$ time.
   (b) $n - 1$ union operations that take $O(n)$ time and n unique find operations that take $\Omega(n^2)$ time.

3. (EASY) You are given a list of n cities numbered from 1 to n. You need to define a function INSAMESTATE(i, j) with the following properties. INSAMESTATE(i, j) returns 1 if city i and j are in the same state. Else it returns 0.
   (a) Design a data-structure of size $O(n^2)$ which will be used in writing the function INSAMESTATE(i, j). The running time of INSAMESTATE(i, j) should be $O(1)$.
   (b) Design a data-structure of size $O(n)$ which will be used in writing INSAMESTATE(i, j). The running time of INSAMESTATE(i, j) should be $O(1)$.

   Write each and every detail of your data-structure. For this question, please do not use hashing.

4. (MEDIUM) Consider the following implementation of Union-Find algorithm. For each element u, we will store a pair (I_u, N_u) where I_u is the unique identifier of the tree containing u and N_u is the number of nodes in the tree containing u.

```
def MAKESINGLETON(u):
    label(u) ← (u, 1)
```

```
def FIND(u):
    return label(u)[0]
```

```
def UNION(u, v):
    if FIND(u) ≠ FIND(v)
        _____;
        _____;
        _____;
        _____;
```

   In the class, we performed a BFS from u and v to find the number of nodes in the tree of u and v. But here, we can find N_u and N_v using $label(u)$ and $label(v)$ respectively. Nevertheless, show that the running time of UNION(u, v) can still be $O(n)$ in the worst case. Write a good implementation of UNION such taht $n - 1$ union operations take $O(n \log n)$ time.

5. (MEDIUM) You are given a grid of size $n \times n$. Each cell in the grid is either white of black, that is $grid(i, j) = 1$ if cell (i, j) is white else it is 1. You want to find the largest connected component of the grid that contains only black cells.

(a) Show how you will use union-find algorithm to solve this problem.

(b) Using union-find algorithm may not be the best strategy to solve this problem. Design an algorithm (from scratch) that can solve this problem in $O(n^2)$ time.

6. (HARD) s In the offline minimum problem, you maintain a set of n numbers from 1 to n under the following m operations.

   1. Insert(i): Insert the number i in the set.
   2. Extract-Min(): Remove the minimum element from the set.

   The two operations may appear in any order. Normally, you use a heap to solve the problem. But in our case, the problem is offline not online. In other words, the output of Extract-min() needs to be provided at the end of the m operations, not in an online fashion.

   Show how you will use union-find data structure to solve this problem efficiently.