Homework#1 Union-Find

Question 1
Union-find with specific canonical element. Add a method `find()` to the union-find data type so that `find(i)` returns the largest element in the connected component containing `i`. The operations, `union()`, `connected()`, and `find()` should all take logarithmic time or better. For example, if one of the connected components is `{1,2,6,9}`, then the `find()` method should return `9` for each of the four elements in the connected components.

Question 2
Successor with delete. Given a set of N integers $S = \{0, 1, ..., N-1\}$ and a sequence of requests of the following form:

- Remove x from S
- Find the *successor* of x: the smallest y in S such that $y \geq x$.

Design a data type so that all operations (except construction) should take logarithmic time or better.

Question 3
Union-by-height. Develop a union-find implementation that uses the same basic strategy as weighted quick-union but keeps track of tree height and always links the shorter tree to the taller one. Prove a $\log N$ upper bound on the height of the trees for N sites with your algorithm.

Hint:
1.
1) 5 5 5 5 5 5 5 5 5 5
2) 7 7 5 4 1 5 7 2 2 1
3) 7 7 7 4 1 7 7 7 2 1
4) 7 2 5 2 2 5 2 2 2 1

2. Use weighted union, and maintain an array max[i] to record the maximum element in the tree rooted at i. Update max[] when union happens.

3. Initialize: 0-N-1 independent sites,
remove(x), if x = 0, do nothing; otherwise, union(x-1,x), update max[]
succ(x) = max[root of x] + 1

4. proof