1 Disjoint Sets, a.k.a. Union Find

In lecture, we discussed the Disjoint Sets ADT. Some authors call this the Union Find ADT. Today, we will use union find terminology so that you have seen both.

(a) What are the last two improvements (out of four) that we made to our naive implementation of the Union Find ADT during lecture 14 (Monday’s lecture)?

1. Improvement 1: __________________________
2. Improvement 2: __________________________

(b) Assume we have nine items, represented by integers 0 through 8. All items are initially unconnected to each other. Draw the union find tree, draw its array representation after the series of connect() and find() operations, and write down the result of find() operations using WeightedQuickUnion. Break ties by choosing the smaller integer to be the root.

Note: find(x) returns the root of the tree for item x.

```java
connect(2, 3);
connect(1, 2);
connect(5, 7);
connect(8, 4);
connect(7, 2);
find(3);
connect(0, 6);
connect(6, 4);
connect(6, 3);
find(8);
find(6);
```

(c) Repeat the above part, using WeightedQuickUnion with Path Compression.
2 Asymptotics

(a) Order the following big-\(O\) runtimes from smallest to largest.

\[
O(\log n), O(1), O(n^3), O(n \log n), O(n!), O(2^n), O(n^2 \log n)
\]

(b) Are the statements in the right column true or false? If false, correct the asymptotic notation (\(\Omega(\cdot), \Theta(\cdot), O(\cdot)\)). Be sure to give the tightest bound. 

\[
\begin{align*}
\Omega(\cdot) & \text{ is the opposite of } O(\cdot), \quad \text{i.e. } f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n)). \\
\end{align*}
\]

(c) Give the worst case and best case runtime in terms of \(M\) and \(N\). Assume \texttt{ping} is in \(\Theta(1)\) and returns an \texttt{int}.

```
1 int j = 0;
2 for (int i = N; i > 0; i--) {
3     for (; j <= M; j++) {
4         if (ping(i, j) > 64) break;
5     }
6 }
```

(d) Assume \texttt{mrpoolsort(array)} is in \(\Theta(N \log N)\) and returns \texttt{array} sorted.

```
1 public static boolean mystery(int[] array) {
2     array = mrpoolsort(array);
3     int N = array.length;
4     for (int i = 0; i < N; i += 1) {
5         boolean x = false;
6         for (int j = 0; j < N; j += 1) {
7             if (i != j && array[i] == array[j]) x = true;
8         }
9         if (!x) return false;
10     }
11     return true;
12 }
```

1. Give the worst case and best case runtime where \(N =\texttt{array.length}\). 
   What is \texttt{mystery()} doing?
2. Now that you know what mystery() is doing, try to come up with a way to implement mystery() that runs in Θ(NlogN) time. Can we get any faster?