14.2 Quick Find

Keeping track of set membership...



Professor Hug's explanation on Quick Find

List of Sets

Intuitively, we might first consider representing Disjoint Sets as a list of sets, e.g, List<Set<Integer>>.

For instance, if we have N=6 elements and nothing has been connected yet, our list of sets looks like: $[\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}]]$. Looks good. However, consider how to complete an operation like connect(5, 6). We'd have to iterate through up to N sets to find 5 and N sets to find 6. Our runtime becomes O(N). And, if you were to try and implement this, the code would be quite complex.

The lesson to take away is that **initial design decisions determine our code complexity** and runtime.

Quick Find

Let's consider another approach using a single array of integers.

- The **indices of the array** represent the elements of our set.
- The value at an index is the set number it belongs to.

For example, we represent {0, 1, 2, 4}, {3, 5}, {6} as:

Set 4: {0, 1, 2, 4} | Set 5: {3, 5} | Set 6: {6}

The array indices (0...6) are the elements. The value at <code>id[i]</code> is the set it belongs to. *The specific set number doesn't matter as long as all elements in the same set share the same id.*

connect(x, y)

Let's see how the connect operation would work. Right now, id[2] = 4 and id[3] = 5. After calling connect(2, 3), all the elements with id 4 and 5 should have the same id. Let's assign them all the value 5 for now:

Set 5: {0, 1, 2, 3, 4, 5} | Set 6: {6}

```
isConnected(x, y)
```

To check isConnected(x, y), we simply check if id[x] == id[y]. Note this is a constant time operation!

We call this implementation "Quick Find" because finding if elements are connected takes constant time.

Code & Runtimes

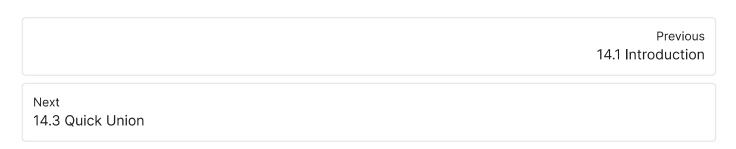
```
public class QuickFindDS implements DisjointSets {
    private int[] id;
    /* \Theta(N) */
    public QuickFindDS(int N){
        id = new int[N];
        for (int i = 0; i < N; i++){
            id[i] = i;
        }
    }
    /* need to iterate through the array => \Theta(N) */
    public void connect(int p, int q){
        int pid = id[p];
        int qid = id[q];
        for (int i = 0; i < id.length; i++){</pre>
            if (id[i] == pid){
                 id[i] = qid;
            3
        }
    }
    /* \Theta(1) */
    public boolean isConnected(int p, int q){
        return (id[p] == id[q]);
    }
}
```

N = number of elements in our DisjointSets data structure

Implementation	Constructor	connect	isConnected
ListOfSets	Θ(N) __ 1	O(N)	O(N)
QuickFind	Θ(N)	Θ(N)	Θ(1)

Note

1. We didn't discuss this but you can reason that having to create N distinct sets initially is $\Theta(N)$



Last updated 1 year ago

