CS 61B
Spring 2025

# Disjoint Sets, ADTs, BSTs

Discussion 06: March 03, 2025

## 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) 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** without path compression. **Break ties by choosing the smaller integer to be the root.**

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

```
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);
```

Below is an implementation of the **find** function for a Disjoint Set. Given an integer **val**, **find(val)** returns the root value of the set **val** is in. The helper method **parent(int val)** returns the direct parent of **val** in the Disjoint Set representation. Assume that this implementation only uses **QuickUnion**.

```java
public int find(int val) {
    int p = parent(val);
    if (p == -1) {
        return val;
    } else {
        int root = find(p);
        return root;
    }
}
```

(b) If `N` is the number of nodes in the set, what is the runtime of **find** in the worst case? Draw out the structure of the Disjoint Set representation for this worst case.

(c) Using a function **setParent(int val, int newParent)**, which updates the value of **val**'s parent to **newParent**, modify **find** to achieve a faster runtime using path compression. You may add at most one line to the provided implementation.

(d) *Extra Practice:* Draw out the tree and array representation for the following WeightedQuickUnion with path compression that has 9 elements from 0 to 8. Break ties by choosing the smaller integer to be root.

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

## 2 ADT Matchmaking

Match each task to the best Abstract Data Type for the job and justify your answer (ie. explain why other options would be less ideal). The options are **`List, Map, Queue, Set, and Stack`**. Each ADT will be used once.

1. You want to keep track of all the unique users who have logged on to your system.

2. You are creating a version control system and want to associate each file name with a Blob.

3. We are grading a pile of exams and want to grade starting from the top of the pile (*Hint:* what order do we pile papers in?).

4. We are running a server and want to service clients in the order they arrive.

5. We have a lot of books at our library and we want our website to display them in some sorted order. We have multiple copies of some books and we want each listing to be separate.

## 3 BST Asymptotics

Below we define the **find** method of a BST (Binary Search Tree) as in lecture, which returns the BST rooted at the node with key **sk** in our overall BST. In this setup, assume a **BST** has a **key** (the value of the tree root) and then pointers to two other child BSTs, **left** and **right**.

```java
public static BST find(BST tree, Key sk) {
    if (tree == null) {
        return null;
    }
    if (sk.compareTo(tree.key) == 0) {
        return tree;
    } else if (sk.compareTo(tree.key) < 0) {
        return find(tree.left, sk);
    } else {
        return find(tree.right, sk);
    }
}
```

(a) Assume our BST is perfectly bushy. What is the runtime of a single **find** operation in terms of N, the number of nodes in the tree? Can we generalize the runtime of **find** to a theta bound?

(b) Say we have an empty BST and want to insert the keys `[6, 2, 5, 9, 0, -3]` (in some order). In what order should we insert the keys into the BST such that the runtime of a single **find** operation after all keys are inserted is $O(N)$? Draw out the resulting BST.