Disjoint Sets
Let $R$ be an equivalence relation. We represent $R$ as several disjoint sets. Two key ideas from Monday:
- Each element exists in exactly one set.
- Every set is an equitant representation.
  - Mathematically: $4 \in [0]_R \Rightarrow 8 \in [0]_R$
  - Programmatically: $\text{find}(4) == \text{find}(8)$

Building Disjoint Sets:
- Maintain a collection $S = \{s_0, s_1, \ldots s_k\}$
- Each set has a representative member.
- ADT:
  
  ```cpp
  void makeSet(const T & t);
  void union(const T & k1, const T & k2);
  T & find(const T & k);
  ```

<table>
<thead>
<tr>
<th>Operation: find(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation: union(k1, k2)</td>
</tr>
</tbody>
</table>

Implementation #2:
- Continue to use an array where the index is the key
- The value of the array is:
  - -1, if we have found the representative element
  - The index of the parent, if we haven’t found the rep. element

Impl #2 (continued):

Example:

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>[1]</td>
<td>[2]</td>
<td>[3]</td>
</tr>
</tbody>
</table>
```

...where is the error in this table?

Implementation – DisjointSets::find

```
DisjointSets.cpp (partial)
1 int DisjointSets::find(int i) {
2     if ( s[i] < 0 ) { return i; }
3     else { return find( s[i] ); } }
```

What is the running time of find?

What is the ideal UpTree?
Implementation – DisjointSets::union

```cpp
void DisjointSets::union(int r1, int r2) {
// Implementation of union function
}
```

How do we want to union the two UpTrees?

Building a Smart Union Function

```
[Diagram of UpTrees]
```

The implementation of this visual model is the following:

```
[DisjointSets.cpp (partial)]
```

What are possible strategies to employ when building a “smart union”?

Smart Union Strategy #1: ____________________

Idea: Keep the height of the tree as small as possible!

Metadata at Root:

```
6 6 6 8 10 7 7 7 4 5
```

Smart Union Strategy #2: ____________________

Idea: Minimize the number of nodes that increase in height.

(Observe that the tree we union have all their nodes gain in height.)

Metadata at Root:

```
6 6 6 8 10 7 7 7 4 5
```

Smart Union Implementation:

```
[DisjointSets.cpp (partial)]
```

CS 225 – Things To Be Doing:

1. Theory Exam 3 starts Thursday; Practice Exam Available!
2. MP6 released; MP6 is due April 15
3. Lab: lab_heaps this week!
4. Daily POTDs are ongoing!