October 14 – Disjoint Sets and Iterators
G Carl Evans
Disjoint Sets

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
Disjoint Sets – Smart Union

**Union by height**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>6</strong></td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>-4</td>
<td>10</td>
<td>7</td>
<td>-3</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

**Union by size**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>6</strong></td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>-8</td>
<td>10</td>
<td>7</td>
<td>-4</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

**Idea:** Keep the height of the tree as small as possible.

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

We will show the height of the tree is: \( \log(n) \).
Union by Size

To show that every tree in a disjoint set data structure using union by size has a height of at most $O(\log n)$ we will show that the inverse.

Base Case

Inductive Hypothesis
Union by Size

Case 1
Union by Size

Case 2
Union by Height

Much like before we will show the min(nodes) in a tree with a root of height $k \geq 2^k$

Base Case

IH
Disjoint Sets Find

```cpp
int DisjointSets::find(int i) {
  if ( s[i] < 0 ) { return i; }
  else { return find( s[i] ); }
}
```
Path Compression
Union by Height - Rank

Base

New UpTrees have Rank =

When you join two UpTrees
Union by Rank

1. For all non-root nodes $x$, $\text{rank}(x) < \text{rank}(\text{parent}(x))$

2. Rank only changes for roots and only up
Disjoint Sets Analysis

The **iterated log** function:

*The number of times you can take a log of a number.*

\[
\log^*(n) =
\begin{align*}
0, & \quad n \leq 1 \\
1 + \log^*(\log(n)), & \quad n > 1
\end{align*}
\]

What is \(\log^*(2^{65536})\)?
Disjoint Sets Analysis

In an Disjoint Sets implemented with smart **unions** and path compression on **find**: 

Any sequence of **m union** and **find** operations result in the worse case running time of \( O(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) \),

where \( n \) is the number of items in the Disjoint Sets.