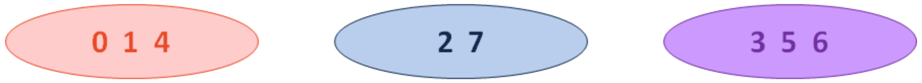
#32: Disjoint Sets Implementation

November 10, 2017 · *Wade Fagen-Ulmschneider*

Implementation #1:



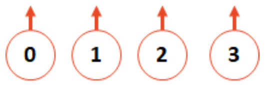
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Operation: `find(k)`

Operation: `union(k1, k2)`

Implementation #2 - UpTrees:

- 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



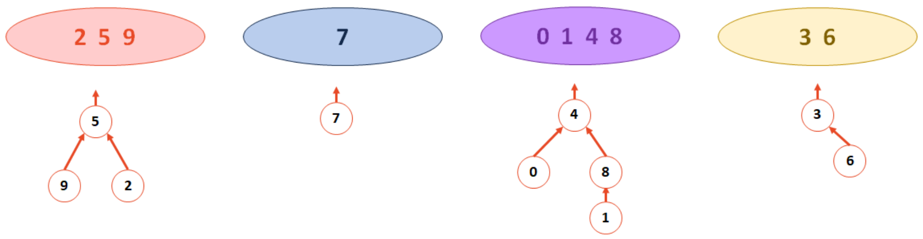
[0]	[1]	[2]	[3]

[0]	[1]	[2]	[3]

[0]	[1]	[2]	[3]

[0]	[1]	[2]	[3]

Example using UpTrees:



4	8	5	6	-1	-1	-1	-1	4	5
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

...what is the error in this table?

Implementation

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

What is the running time of **`find`**?

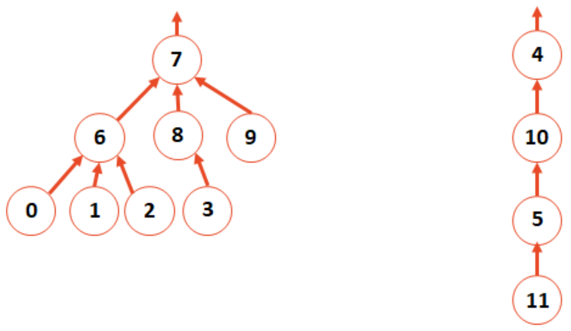
What is the ideal UpTree?

```
DisjointSets.cpp (partial)
void DisjointSets::union(int r1, int r2) {


}
```

How do we want to union the two UpTrees?

Building a Smart Union Function



The implementation of this visual model is the following:

6	6	6	8	-1	10	7	-1	7	7	4	5
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]

Strategy #1: Union by Height

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

Metadata at Root:

After `union( 4, 7 ):`

6	6	6	8		10	7		7	7	4	5
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]

Strategy #2: Union by Size

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:

After `union( 4, 7 ):`

6	6	6	8		10	7		7	7	4	5
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]

Smart Union Implementation:

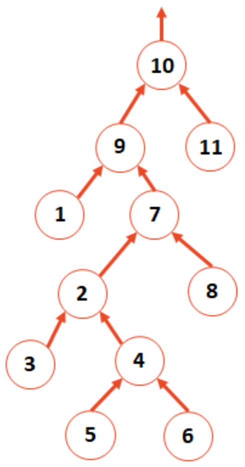
DisjointSets.cpp (partial)

```
void DisjointSets::unionBySize(int root1, int root2) {
  int newSize = arr_[root1] + arr_[root2];

  // If arr_[root1] is less than (more negative), it is the larger
  // set; we union the smaller set, root2, with root1.
  if ( arr_[root1] < arr_[root2] ) {
    arr_[root2] = root1;
    arr_[root1] = newSize;
  }

  // Otherwise, do the opposite:
  else {
    arr_[root1] = root2;
    arr_[root2] = newSize;
  }
}
```

Path Compression:



CS 225 – Things To Be Doing:

1. Exam #10 (programming) starts Monday
2. MP6 due Friday, Nov. 17 (Friday before break starts)
3. lab_heaps due Sunday, Nov. 12
4. Daily POTDs