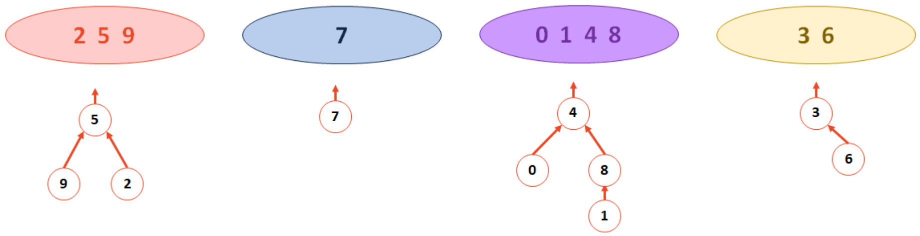
#31: Disjoint Sets Implementation

April 6, 2018 · *Wade Fagen-Ulmschneider*

Disjoint Sets



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

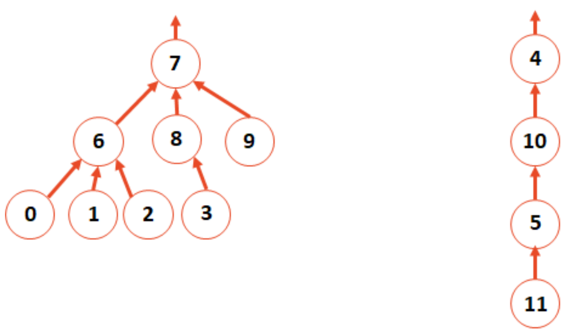
Implementation – DisjointSets::union

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]

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:

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 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:

After `union( 4, 7 ):`

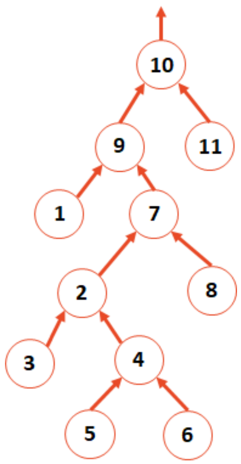
6	6	6	8		10	7		7	7	4	5
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]

Smart Union Implementation:

DisjointSets.cpp (partial)

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

  if ( arr_[root1] < arr_[root2] ) {
    arr_[root2] = root1;  arr_[root1] = newSize;
  } else {
    arr_[root1] = root2;  arr_[root2] = newSize;
  }
}
```

Path Compression:



UpTree Implementation with a smart union function and path compression:

DisjointSets.cpp (partial)

```
int DisjointSets::find(int i) {
  if ( arr_[i] < 0 ) { return i; }
  else { return                    find( arr_[i] ); }
}
```

DisjointSets.cpp (partial)

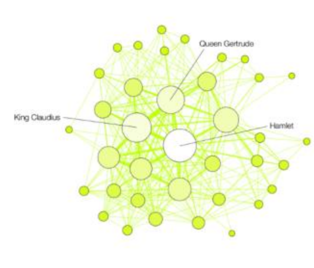
```
void DisjointSets::unionBySize(int root1, int root2) {
  int newSize = arr_[root1] + arr_[root2];

  // 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;
  }
}
```

A Review of Major Data Structures so Far

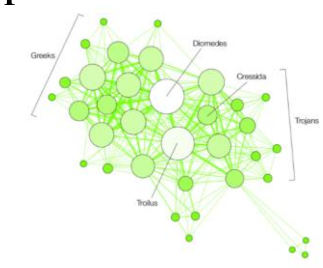
Array-based	List/Pointer-based

- Sorted Array
- Unsorted Array
  - Stacks
  - Queues
  - Hashing
  - Heaps
    - Priority Queues
  - UpTrees
    - Disjoint Sets
	- Singly Linked List
	- Doubly Linked List
	- Skip Lists
	- Trees
	  - BTree
	  - Binary Tree
	    - Huffman Encoding
	    - kd-Tree
	    - AVL Tree

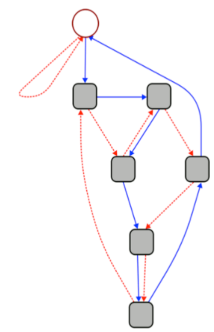
An Introduction to Graphs



HAMLET



TROILUS AND CRESSIDA



CS 225 – Things To Be Doing:

1. Theory Exam 3 final day is **today**
2. lab_heaps due Sunday, April 8th
3. MP6 released; Extra Credit deadline on Monday, April 9th
4. Daily POTDs are ongoing!