

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: `find(4) == find(8)`

Building Disjoint Sets:

- Maintain a collection $S = \{s_0, s_1, \dots, s_k\}$
- Each set has a representative member.
- ADT:

```
void makeSet(const T & t);
void union(const T & k1, const T & k2);
T & find(const T & k);
```



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

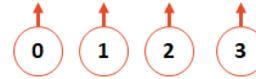
Operation: `find(k)`

Operation: `union(k1, k2)`

Implementation #2:

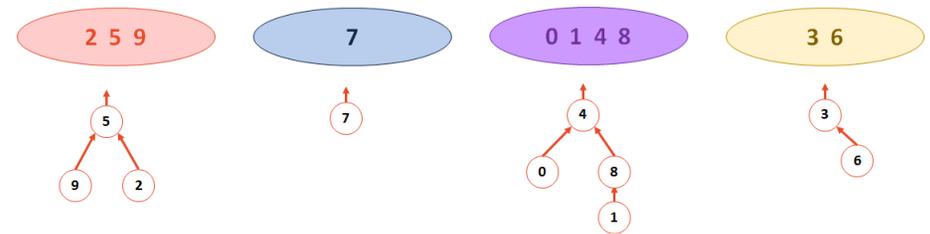
- Continue to use an array where the index is the key
- The value of the array is:
 - **-1**, if we have found the representative element
 - **The index of the parent**, if we haven't found the rep. element

Impl #2 (continued):



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

Example:



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

...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] ); }
4 }
```

What is the running time of `find`?

What is the ideal UpTree?

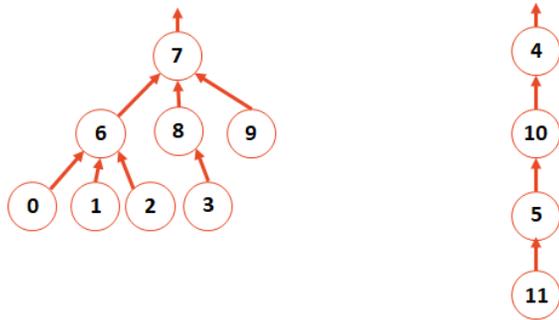
Implementation – DisjointSets::union

```

DisjointSets.cpp (partial)
1 void DisjointSets::union(int r1, int r2) {
2
3
4 }
    
```

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)
1 void DisjointSets::unionBySize(int root1, int root2) {
2     int newSize = arr_[root1] + arr_[root2];
3
4     if ( arr_[root1] < arr_[root2] ) {
5         arr_[root2] = root1; arr_[root1] = newSize;
6     } else {
7         arr_[root1] = root2; arr_[root2] = newSize;
8     }
9 }
    
```

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!