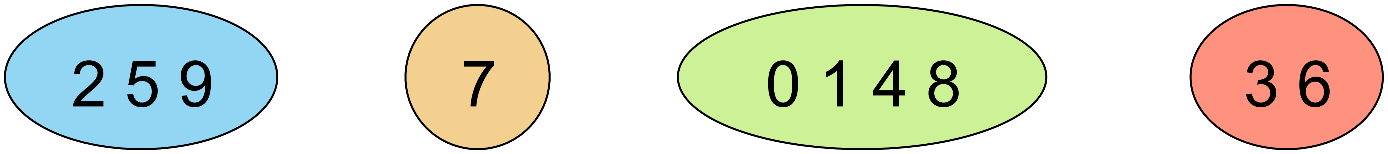
Today's announcements:

MP7 available, due 4/30, 11:59p. EC due 4/19.

Code Challenge #4, 4/17, 9p, Siebel 0224.

A Disjoint Sets example:

Let *R* be an equivalence relation on the set of students in this room, where *(s,t)* ∈ *R* if *s* and *t* have the same favorite among {AB, FN, DJ, ZH, PvZ}.



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

1. Find(4)
2. Find(4)==Find(8)
3. If (!(Find(7)==Find(2)) then Union(Find(7),Find(2))

A better data structure for Disjoint Sets:

```
int DS::Find(int i) {
   if (s[i] < 0) return i;
   else return Find(s[i]);
}
```

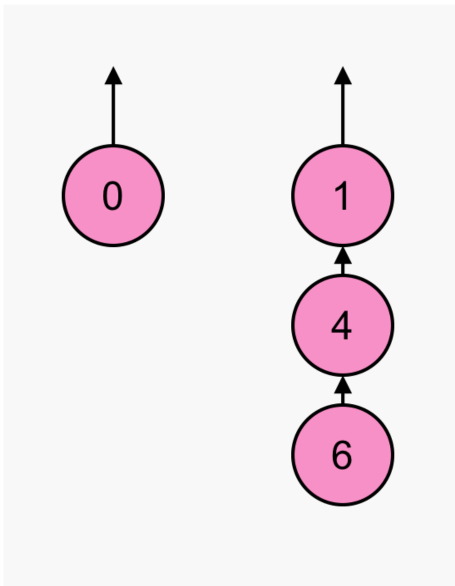
Running time depends on ____________.

Worst case?

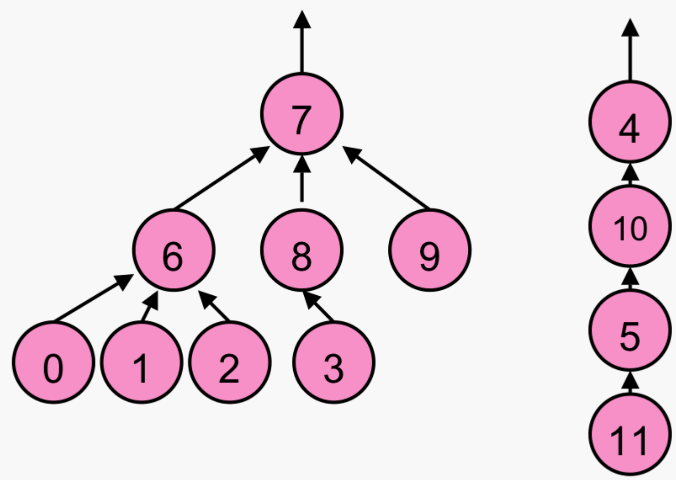
What's an ideal tree?

```
void DS::Union(int root1, int root2) {
   ________________________;
}
```

something to consider…



Smart unions:



Union by height:

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

Keeps overall height of tree as small as possible.

Union by size:

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

Increases distance to root for fewest nodes.

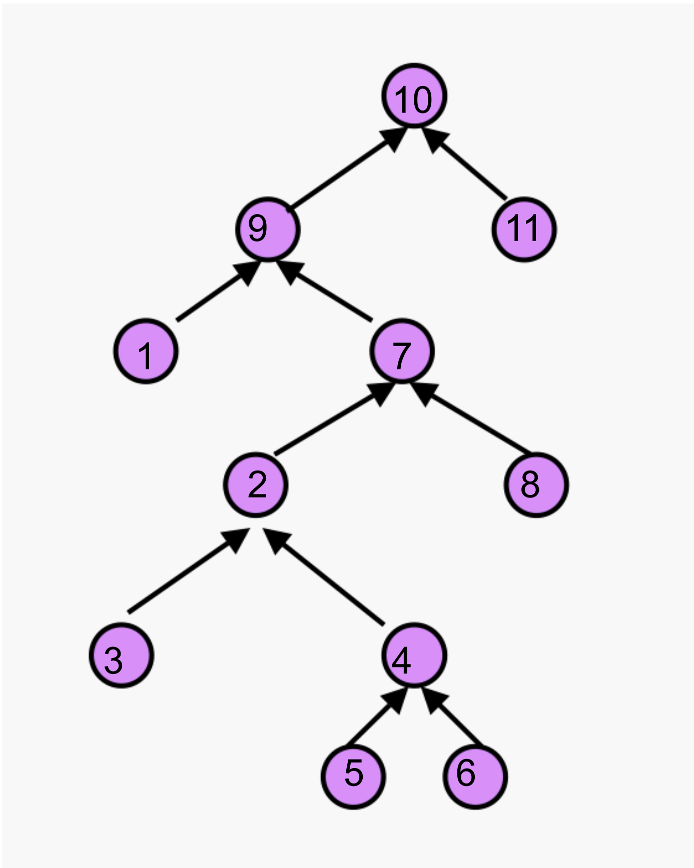
Both of these schemes for Union guarantee the height of the tree is ____________.

Smart unions:

```
int DS::Find(int i) {
   if (s[i] < 0) return i;
   else return Find(s[i]);
}
```

```
void DS::UnionBySize(int root1, int root2) {
   int newSize = s[root1]+s[root2];
   if (isBigger(root1,root2)) {
      s[root2]= root1;
      s[root1]= newSize;
   }
   else {
      s[root1] = root2;
      s[root2]= newSize;
   }
}
```

Path Compression:



Path Compression:

```
int DS::Find(int i) {
   if (s[i] < 0) return i;
   else return      Find(s[i]);
}
```

```
void DS::UnionBySize(int root1, int root2) {
   int newSize = s[root1]+s[root2];
   if (isBigger(root1,root2)) {
      s[root2]= root1;
      s[root1]= newSize;
   }
   else {
      s[root1] = root2;
      s[root2]= newSize;
   }
}
```

Analysis:

$$\log^* n := \begin{cases} 0 & \text{if } n \leq 1; \\ 1 + \log^*(\log n) & \text{if } n > 1 \end{cases}$$

Example:

2^{65536}

Relevant result:

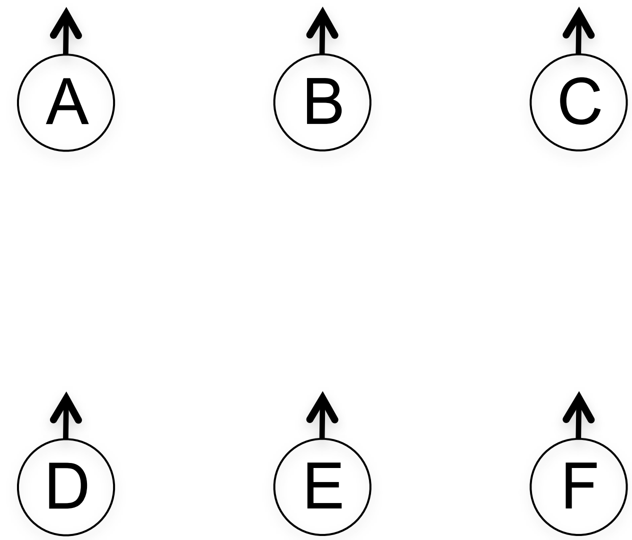
In an upTree implementation of Disjoint Sets using smart `union` and path compression upon `find`...

any sequence of *m* `union` and `find` operations results in worst case

running time of O(____________), where *n* is the number of items.

http://research.cs.vt.edu/AVresearch/UF/

```
If (Find(A) != Find(B)
        Union(Find(A),Find(B));
If (Find(D) != Find(E)
        Union(Find(D),Find(E));
If (Find(A) != Find(C)
        Union(Find(A),Find(C));
If (Find(C) != Find(B)
        Union(Find(C),Find(B));
If (Find(B) != Find(F))
        Union(Find(B),Find(F));
If (Find(D) != Find(F))
        Union(Find(D),Find(F));
```



What's the tree height of the final tree?

Name the last 4 data structures we've discussed:

Which of those 4 is/are dictionaries?

Give 2 applications of a Heap:

What's the buildHeap algorithm and how fast is it?