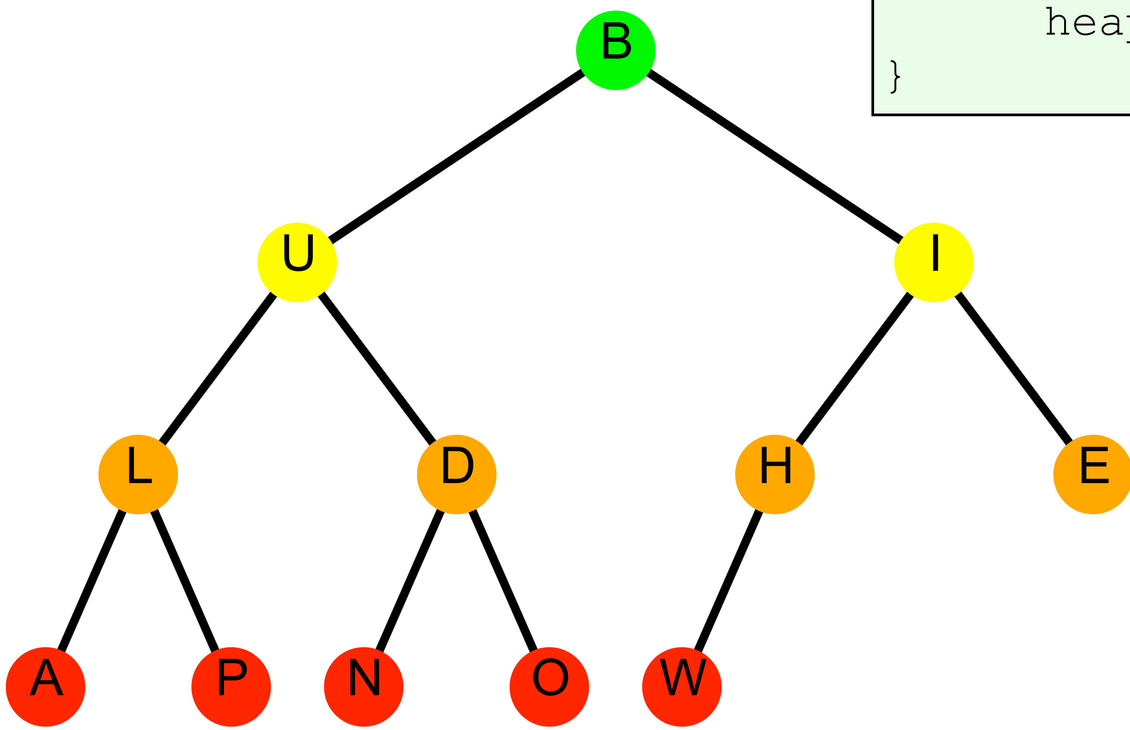
Today's announcements:

MP6 available, due 11/15, 11:59p.

This image reminds us of a ___________, which is one way we can implement ADT __________________, whose functions include ______________ and _________________, with running times ______________________.

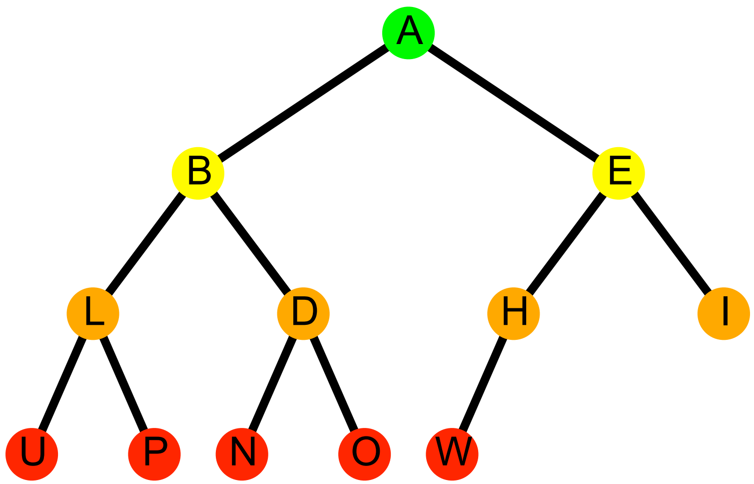
(min)Heap: buildHeap

```
template <class T>
void Heap<T>::buildHeap(){
   for (int i=parent(size);i>0;i--)
      heapifyDown(i)
}
```



B	U	I	L	D	H	E	A	P	N	O	W

(min)Heap: buildHeap



A	B	E	L	D	H	I	U	P	N	O	W

Thm: The running time of buildHeap on an array of size n is _________.

Instead of focussing specifically on running time, we observe that the time is proportional to the sum of the heights of all of the nodes, which we denote by S(h).

S(h) =

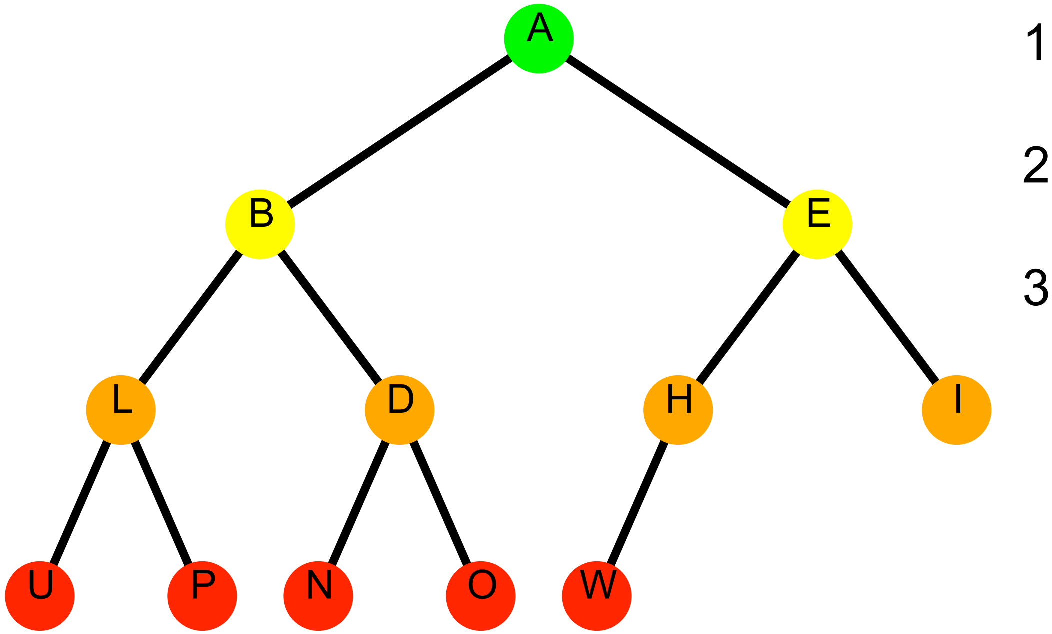
S(0) =

Soln S(h) =

Proof of solution to the recurrence:

But running times are reported in terms of n, the number of nodes...

(min)Heap: heapSort



A	B	E	L	D	H	I	U	P	N	O	W

Running time?

-

Why do we need another sorting algorithm?

-

Remembering CS173…

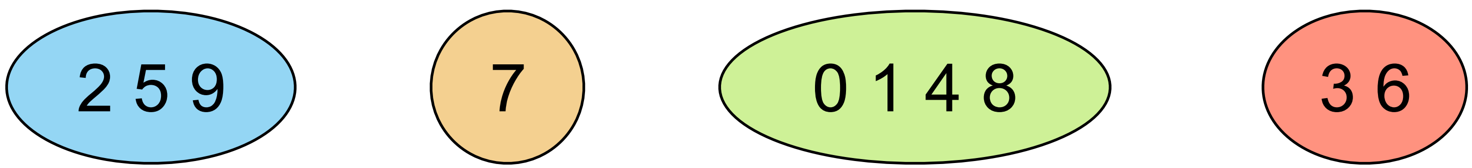
Let R be an equivalence relation on the set of students in this room, where $(s,t) \in R$ if s and t have the same favorite among {A, FB, TR, CC, PMC, ____}.

Notation from math: $[\_\_\_]_R = \{x : xR\_\_\_\}$

One big goal for us: Given s and t we want to determine if sRt.

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 A, FB, TR, CC, PMC, _____}.



1. Find(4)

2. Find(4)==Find(8)

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

Disjoint Sets ADT

We will implement a data structure in support of “Disjoint Sets”:

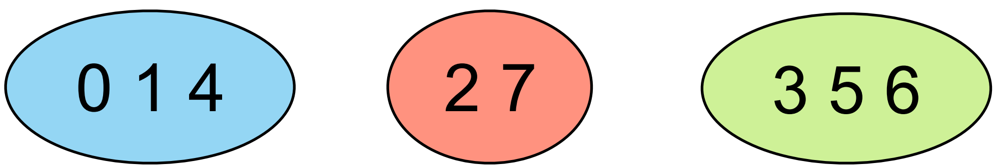
- Maintains a collection $S = \{s_0, s_1, \ldots s_k\}$ of disjoint sets.
- Each set has a representative member.
- Supports functions:

```
void MakeSet(const T & k);

void Union(const T & k1, const T & k2);

T & Find(const T & k);
```

A first data structure for Disjoint Sets:



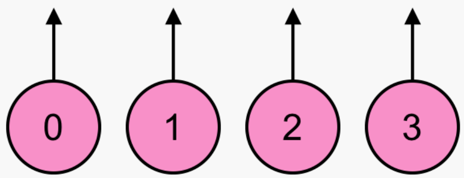
0	1	2	3	4	5	6	7
0	0	2	3	0	3	3	2

Find:

Union:

A better data structure for Disjoint Sets: UpTrees

- if array value is -1, then we’ve found a root, o/w value is index of parent.
- x and y are in the same tree iff they are in the same set.



0	1	2	3
-1	-1	-1	-1

0	1	2	3

0	1	2	3

0	1	2	3