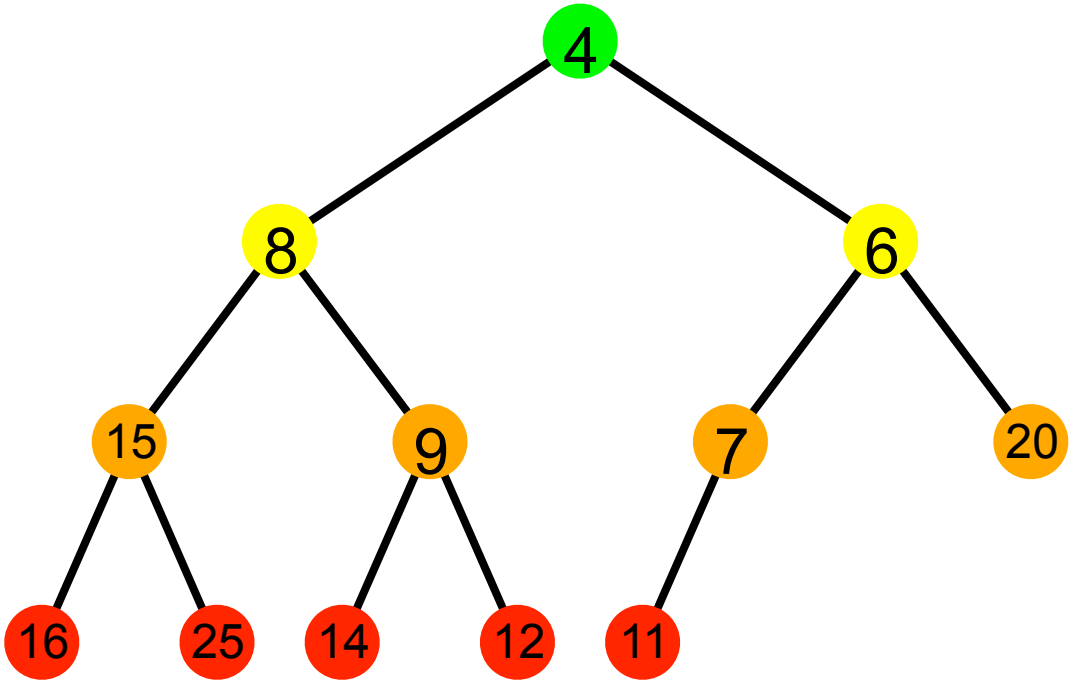
Today's announcements:

MP6 available, due 4/12, 11:59p.

Exam 2: returned in section this week.

This image reminds us of a _____,
which is one way we can implement ADT _____,
whose functions include _____ and _____.

(min)Heap: tell me as many details as you can...

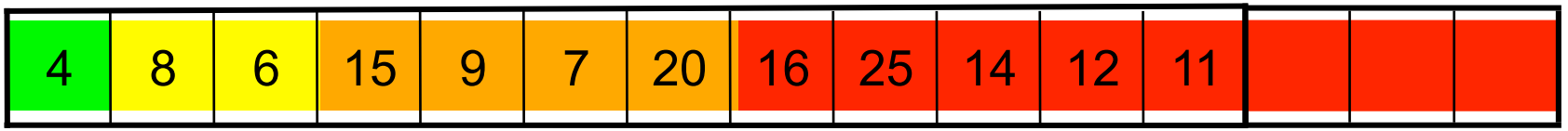


Structure:

-
-

Max ht:

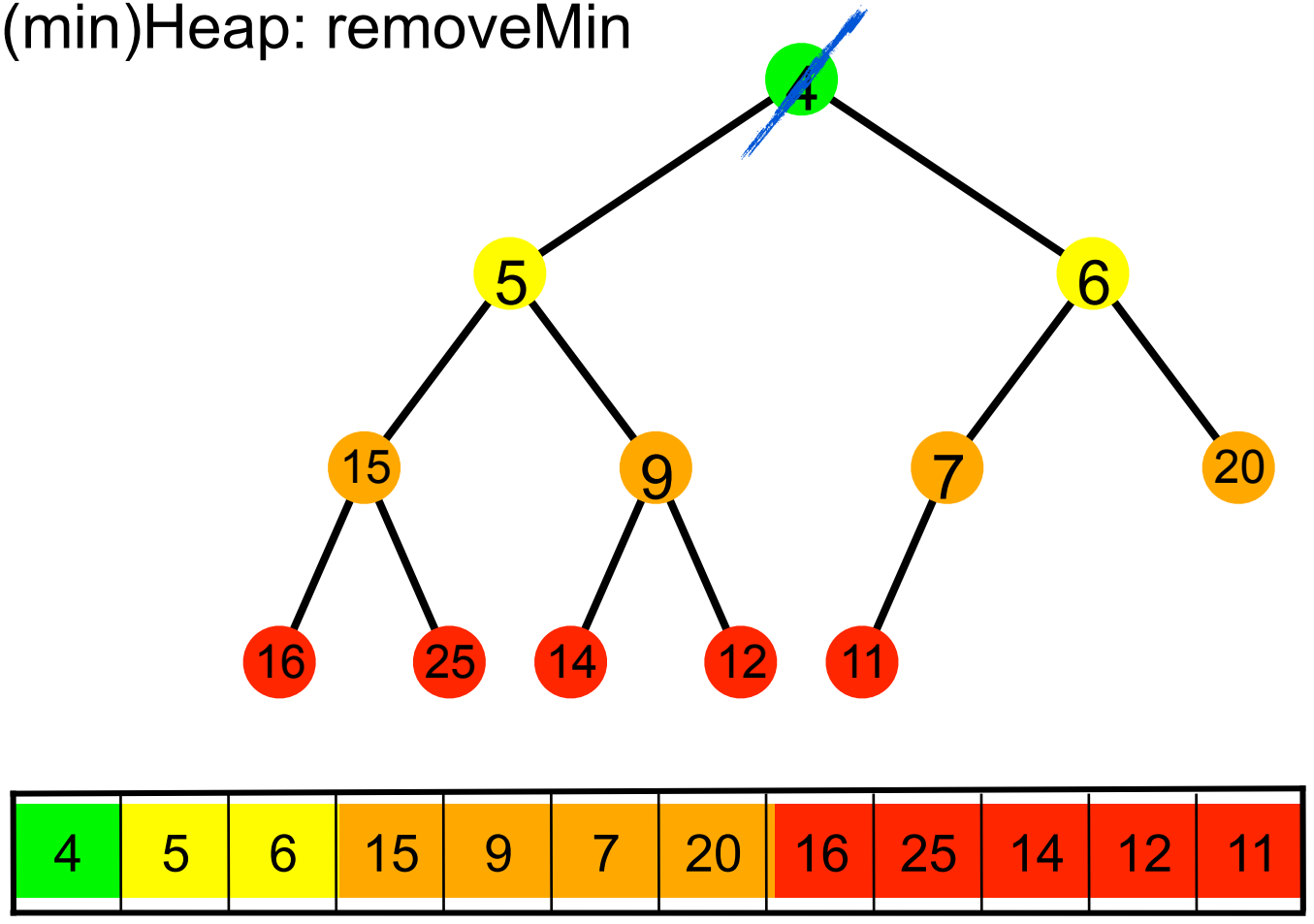
Level order traversal:



Implementation:

- LeftChild(i) =
- RightChild(i) =
- Parent(i) =

(min)Heap: removeMin



Code:

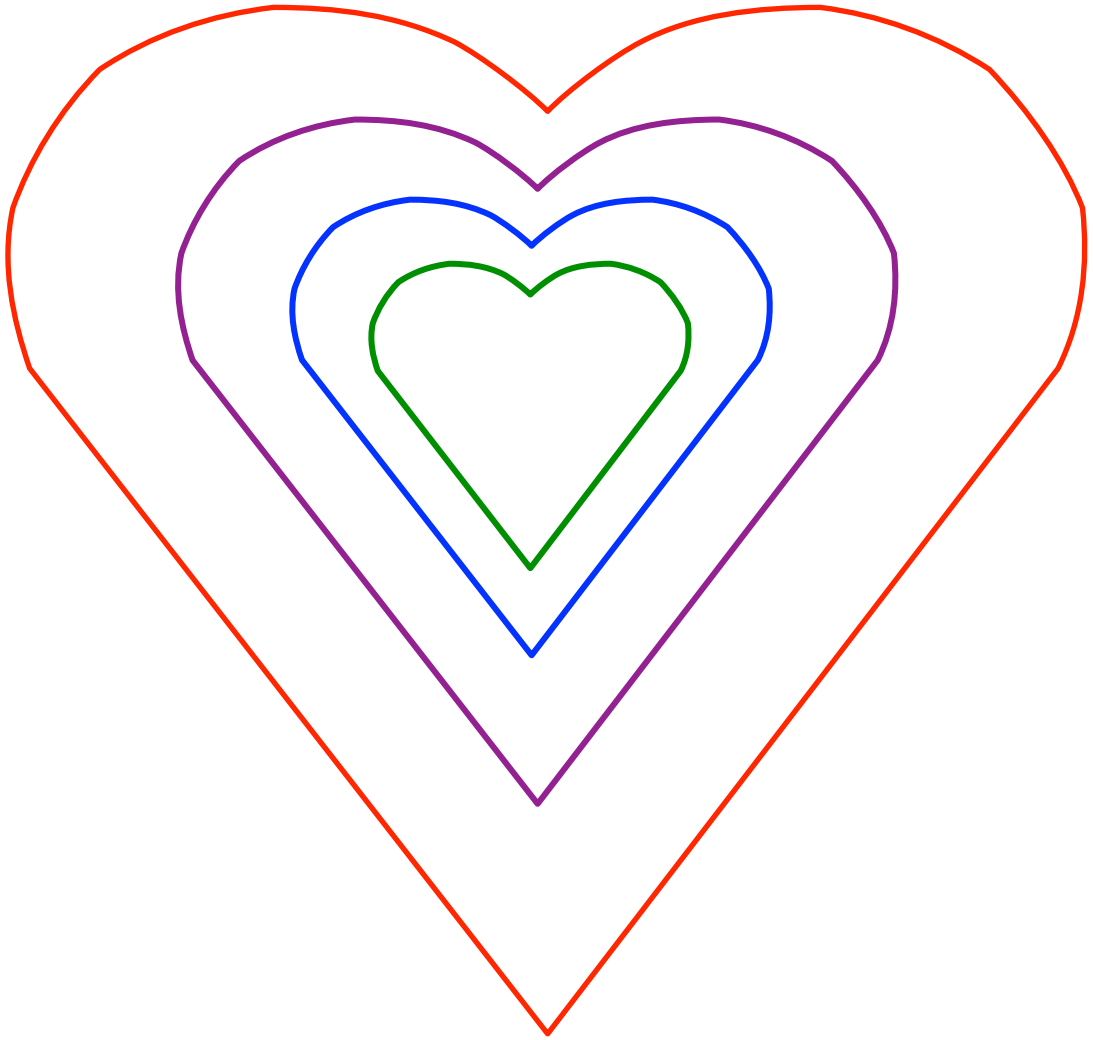
```
template <class T>
T Heap<T>::removeMin(){
   T minVal = items[1];
   items[1] = items[size];
   size--;
   heapifyDown(1);
   return minVal;
}
```

Code:

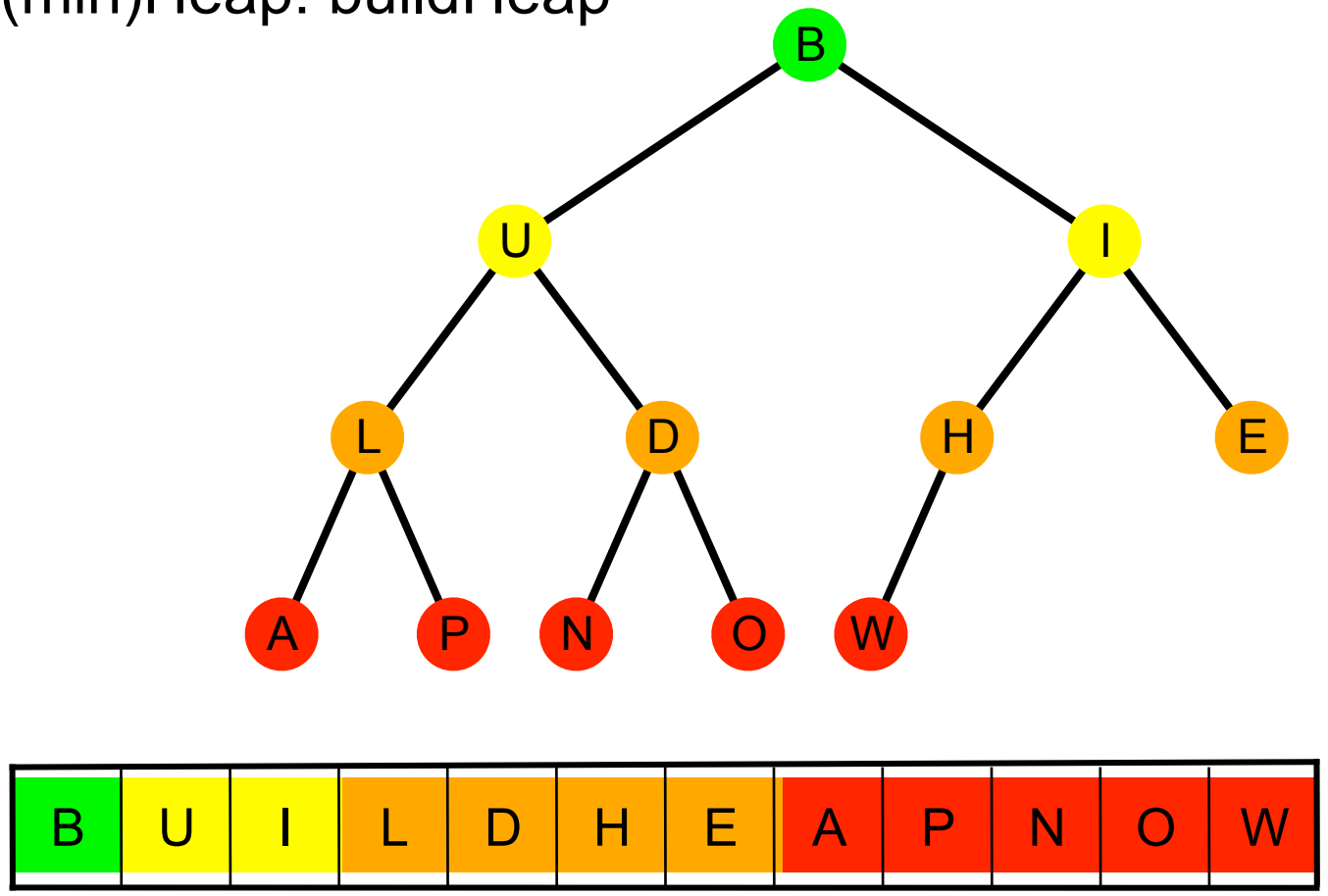
```
template <class T>
T Heap<T>::removeMin(){
   T minVal = items[1];
   items[1] = items[size];
   size--;
   heapifyDown(1);
   return minVal;
}
```

```
template <class T>
void Heap<T>::heapifyDown(int cIndex){
   if (hasAChild(cIndex)){
      minChildIndex = minChild(cIndex);
      if (items[cIndex] ____ items[minChildIndex]{
         swap(_____________,_____________);
         ___________________________;
   }
}
```

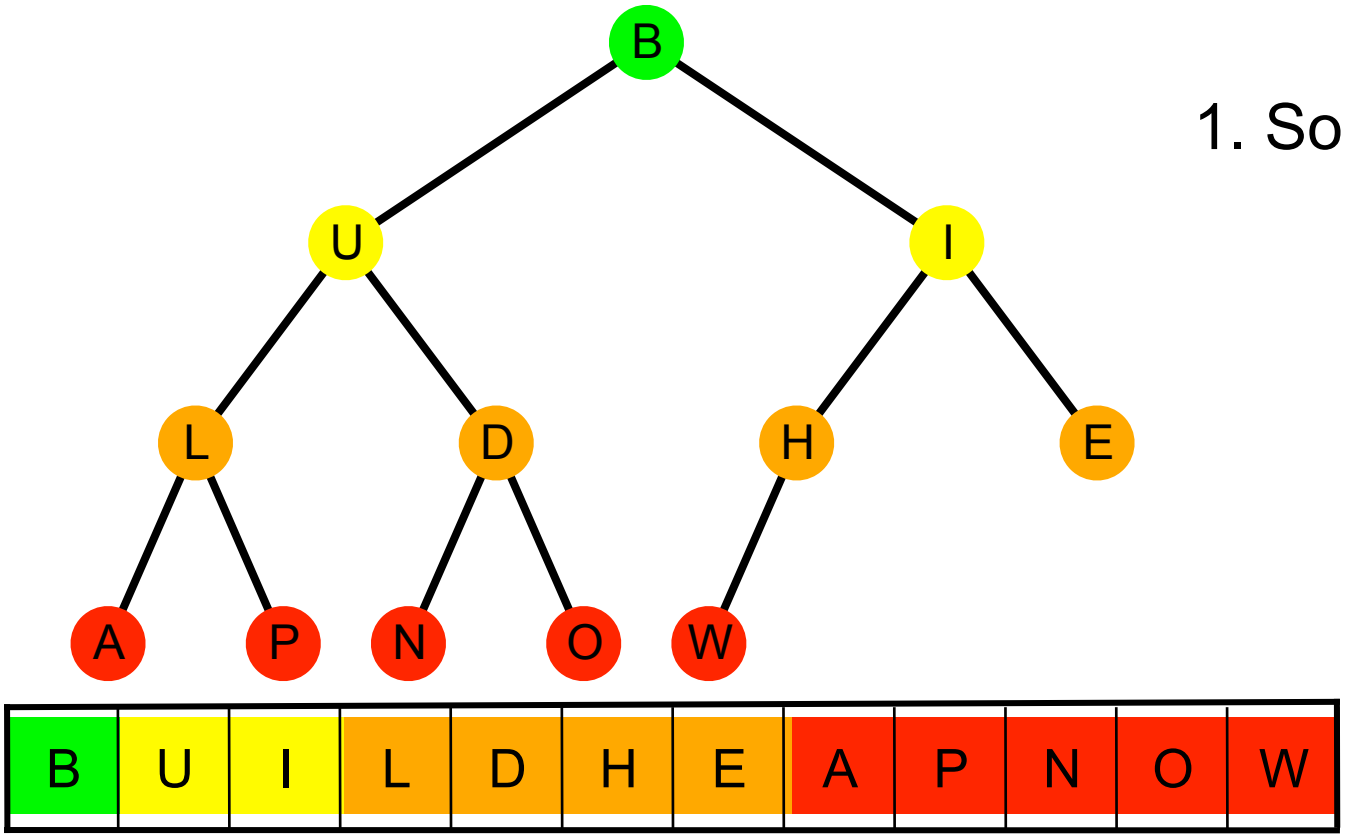
What have we done?



(min)Heap: buildHeap



(min)Heap: buildHeap - 3 alternatives



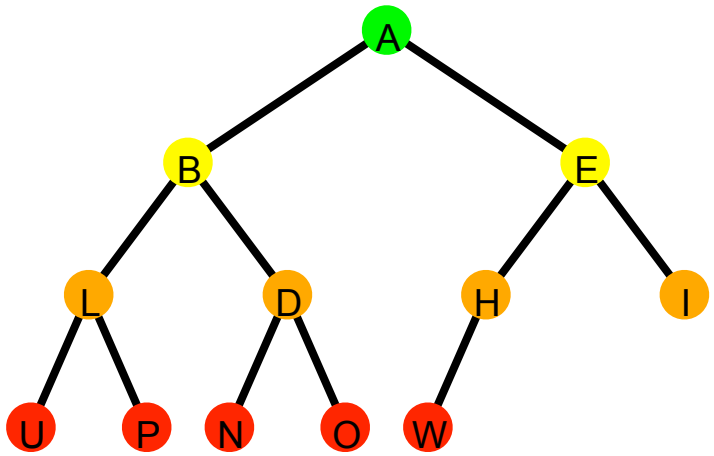
1. Sort the array:

2.
```
template <class T>
void Heap<T>::buildHeap(){
   for (int i=2;i<=size;i++)
      heapifyUp(i)
}
```

3.
```
template <class T>
void Heap<T>::buildHeap(){
   for (int i=parent(size);i>0;i--)
      heapifyDown(i)
}
```

(min)Heap: buildHeap

level height



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) =

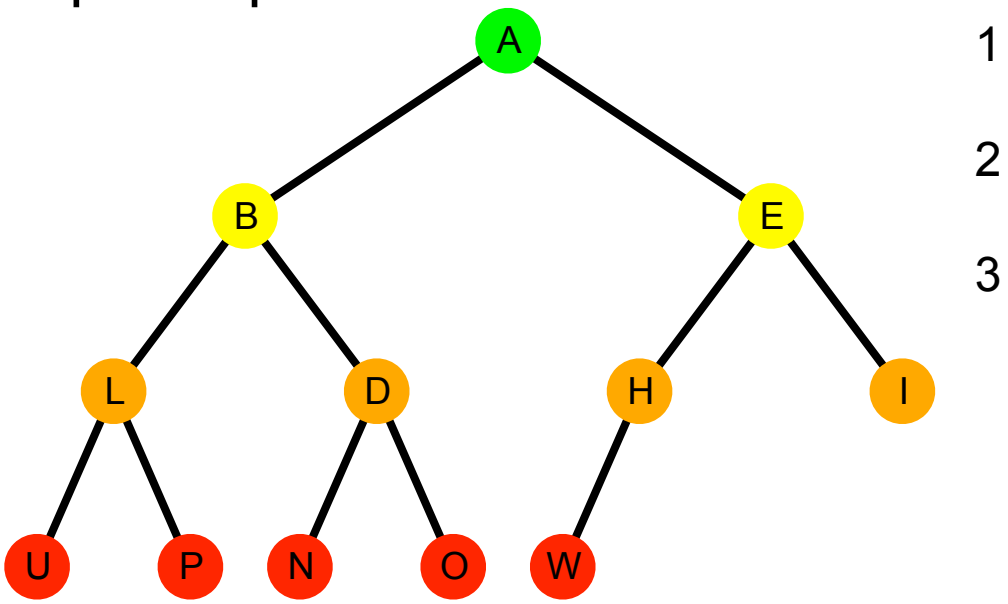
S(0) =

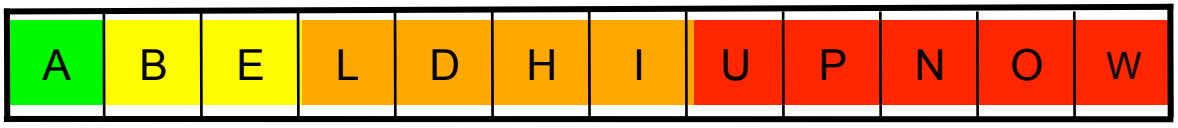
Soln S(h) =

Proof of solution to the recurrence:

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

(min)Heap: heapSort





Running time?

Why do we need another sorting algorithm?