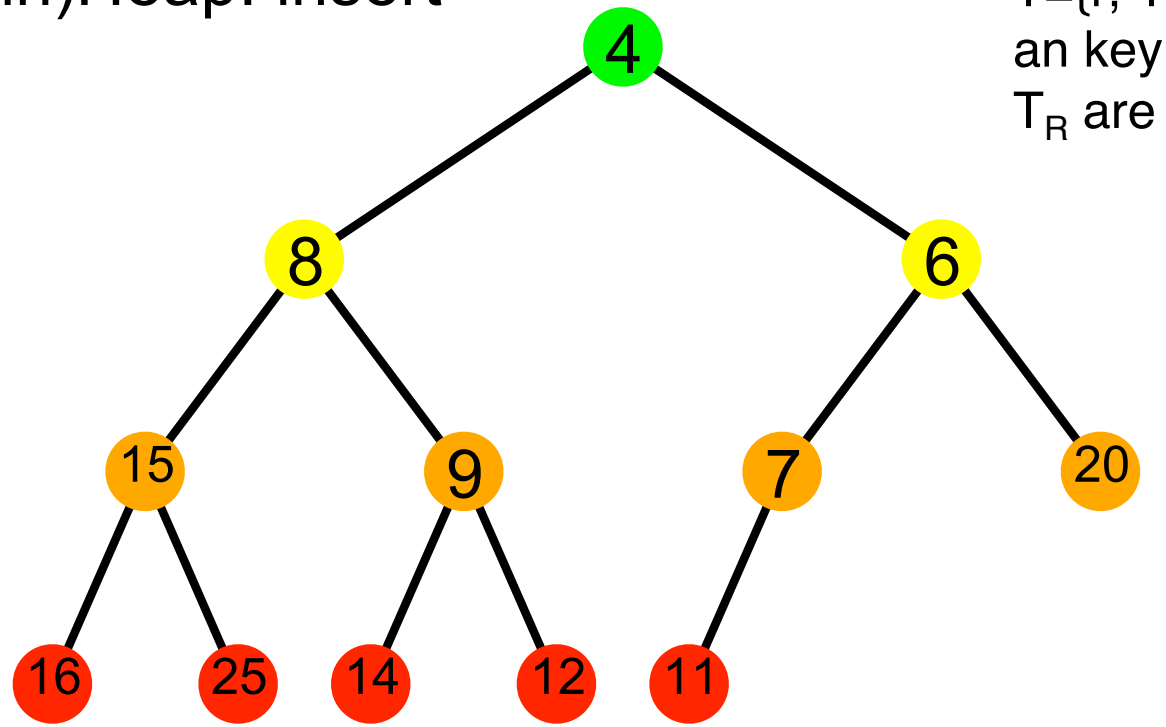
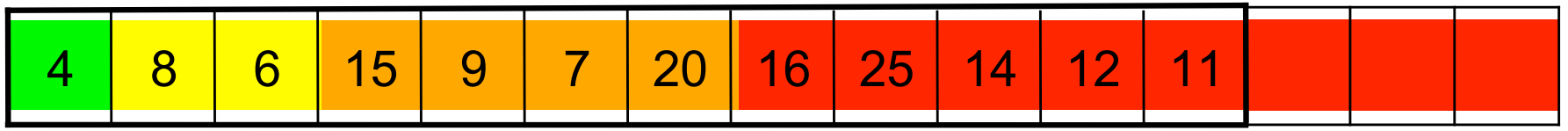
Today's announcements:

MP6 available, due 4/14, 11:59p.

(min)Heap: insert

A complete binary tree T is a heap if
- T is empty OR
- T={r} OR
- T={r, TL, TR} and key of r is less th an key of root of {T_LIT_R} and T_L and T_R are heaps.

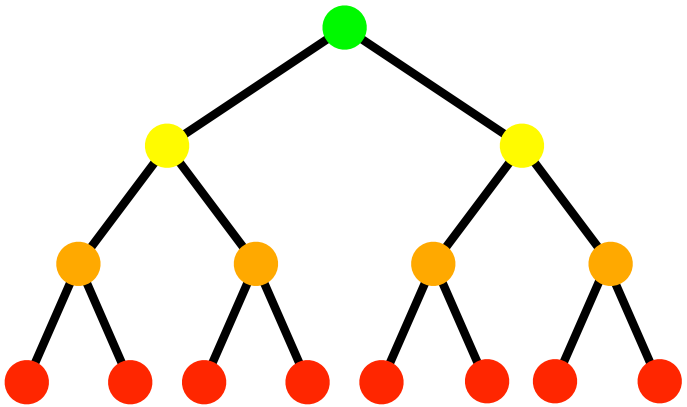




Code:

```
template <class T>
void Heap<T>::insert(const T & key){

   if (size==capacity) growArray();
   size++;
   items[size] = key;
   heapifyUp(size);

}
```

growArray()

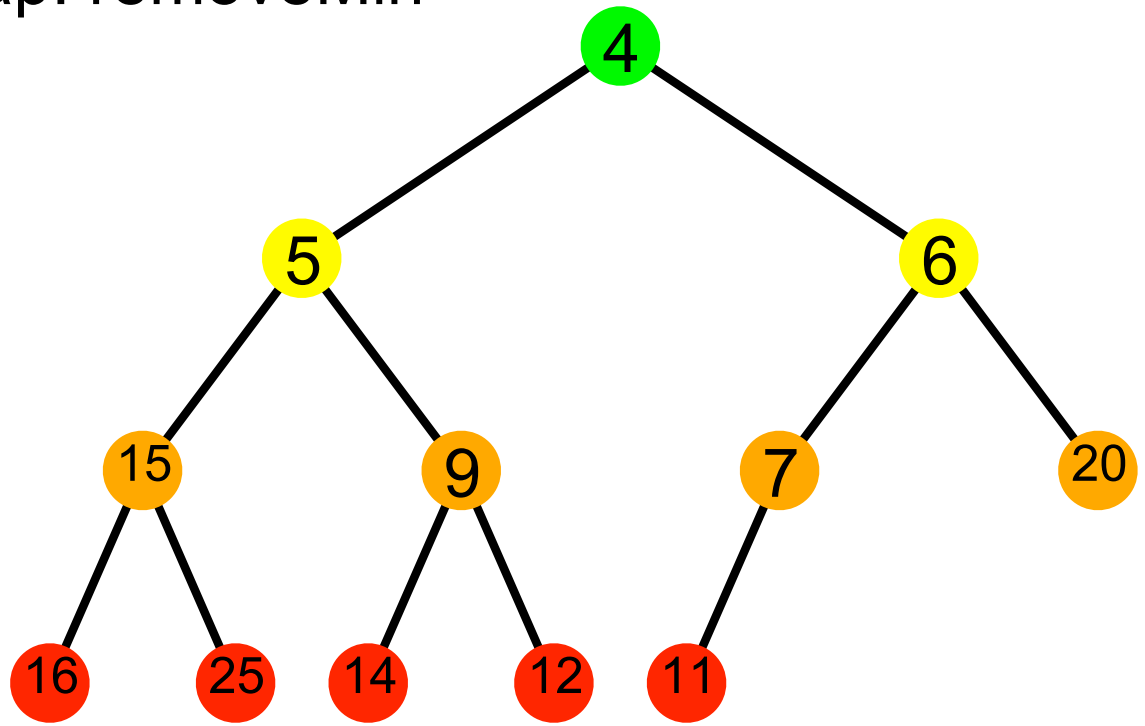


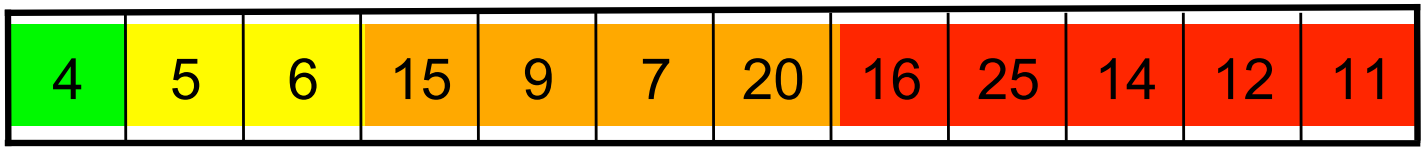
Code:

```
template <class T>
void Heap<T>::insert(const T & key){

   if (size==capacity) growArray();
   size++;
   items[size] = key;
   heapifyUp(size);

}
```

```
template <class T>
void Heap<T>::heapifyUp(int cIndex){
   if (cIndex > ___){
      if (items[cIndex] ___ items[parent(cIndex)]{
         swap(_________________,_________________);
         heapifyUp(_________________);
      }
   }
}
```

(min)Heap: removeMin



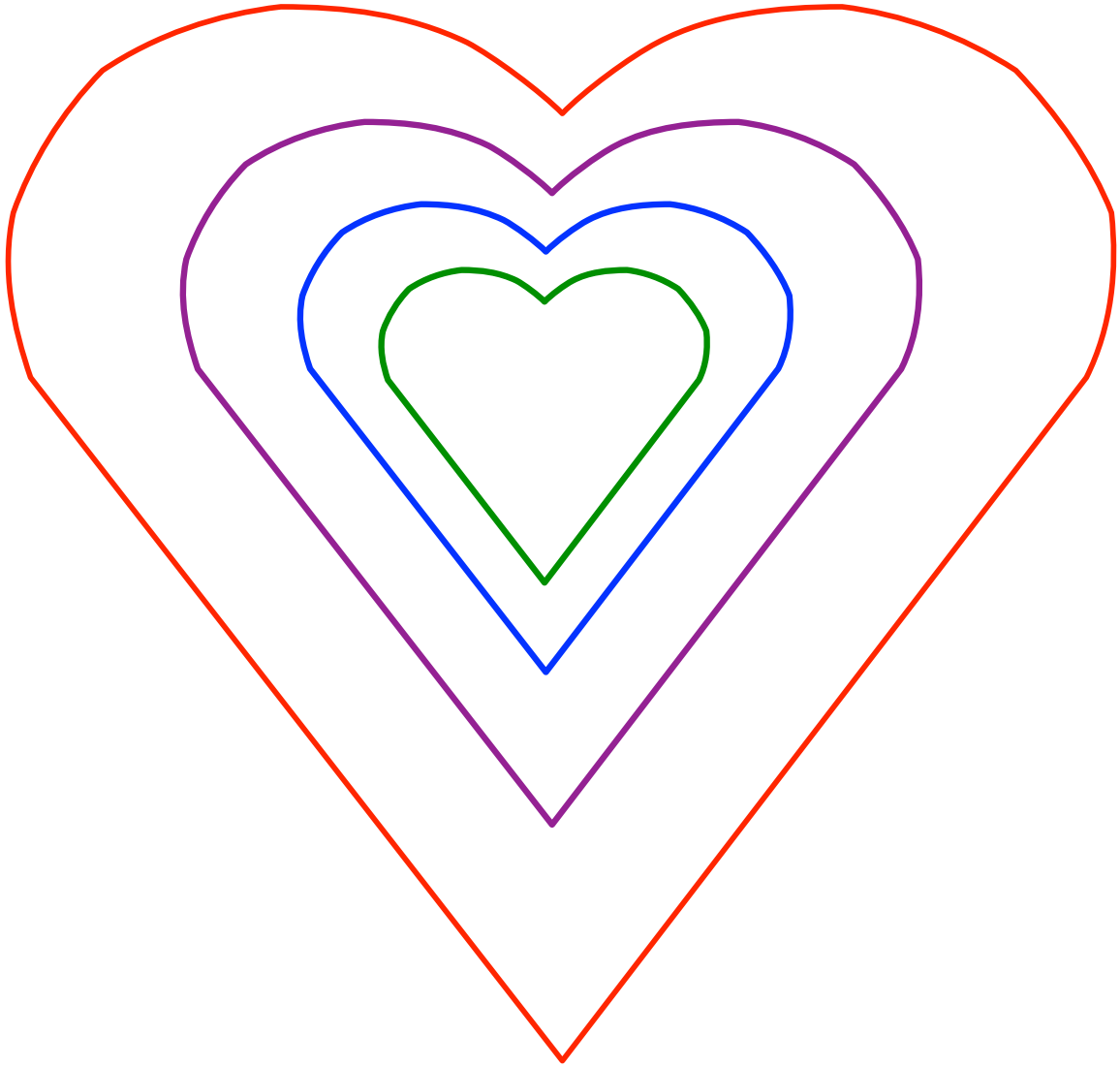


Code:

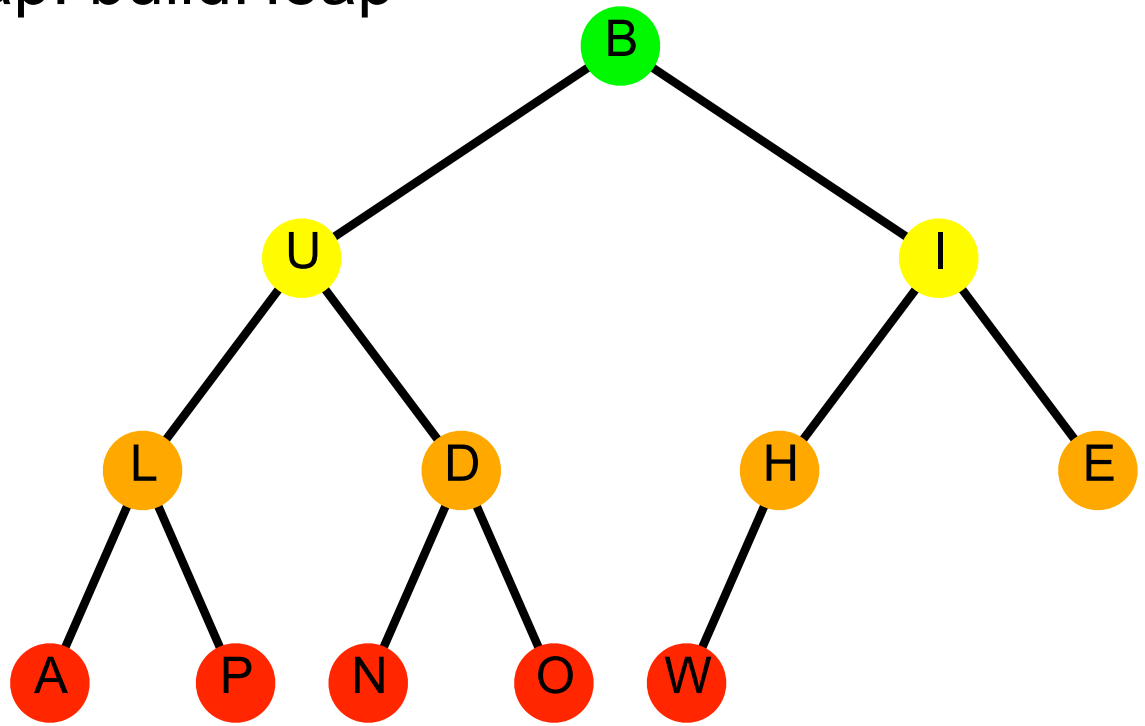
```
template <class T>
T Heap<T>::removeMin(){
   T minVal = items[1];
   items[1] = items[size];
   size--;
   heapifyDown(1);
   return minVal;
}
```

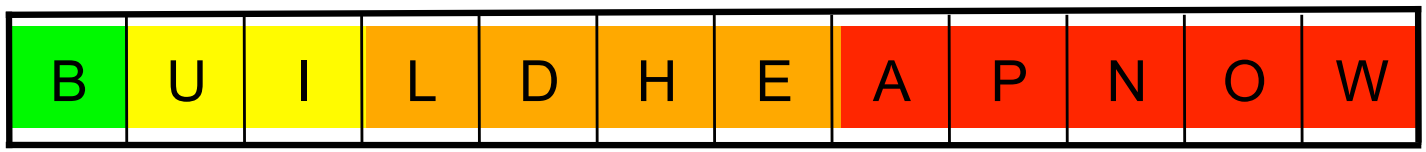
```
template <class T>
void Heap<T>::heapifyDown(int cIndex){
   if (hasAChild(cIndex)){
      minChildIndex = minChild(cIndex);
      if (items[cIndex] ____ items[minChildIndex]{
         swap(_____________,_____________);
         ________________________;
   }
}
```

What have we done?

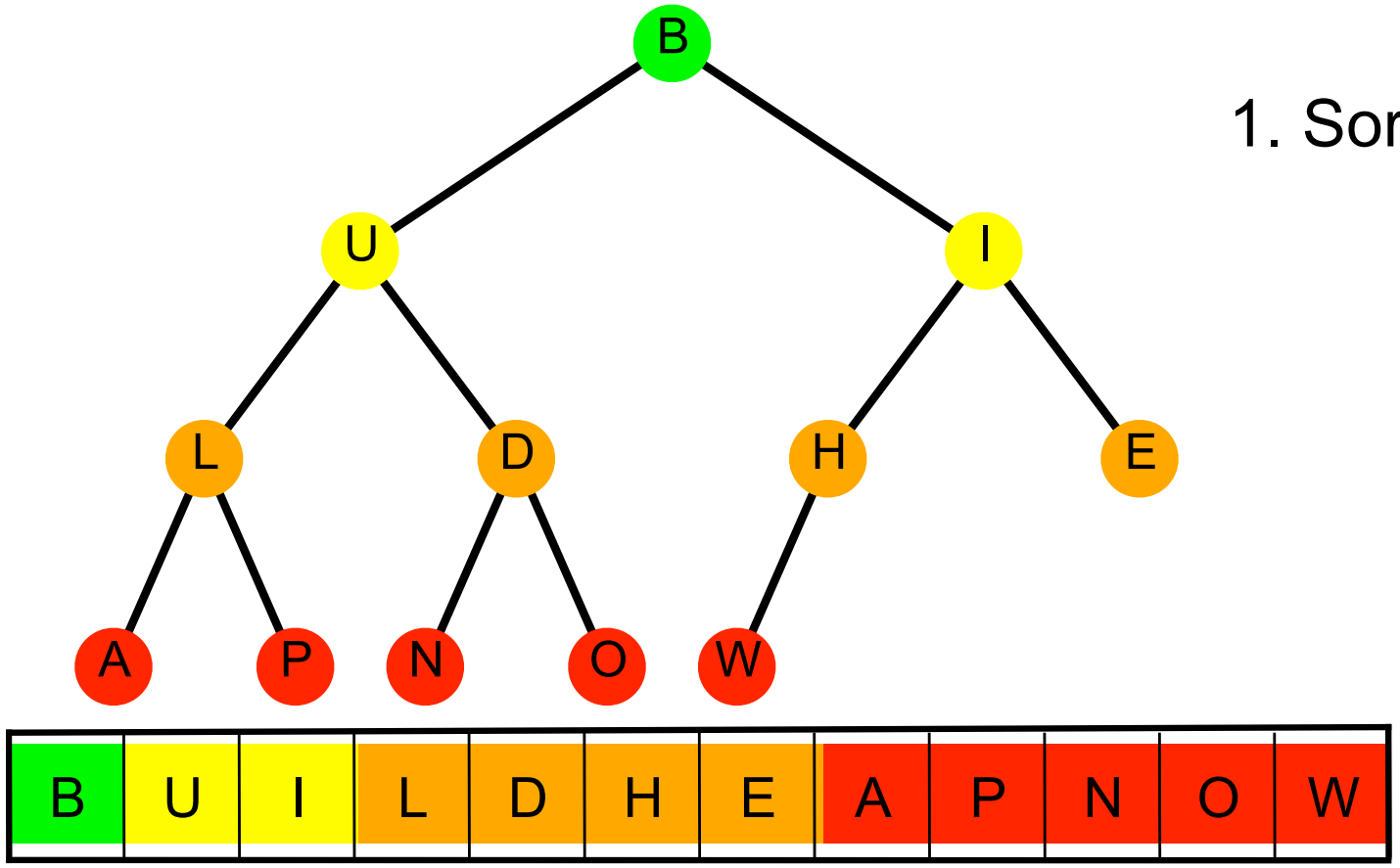


(min)Heap: buildHeap





(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)
}
```