Outline Zach is @ a conference

- Review AVL trees
- Priority Queue ADT
- Consider possible implementations
- Introduce Heaps and complete trees

Review

Q: What's the difference between a BST and an AVL tree?

A: An AVL tree is a BST w/ invariant that for every node in tree the heights of its children differ by at most 1.

Q: In AVL tree, what's max # of rotations needed to fix an unbalanced node?

A: At most 2.

Q: What do we know about the height of AVL trees?

A: height is $O(\lg_2 n)$ where n = # of nodes in tree

Priority Queue ADT (Maximum out first)

A queue sorted by priority

Applications?

- Netflix queue of recommendations
- Prioritizing mail
- Triage at an ER
- Relevance of web searches

Template PQ using

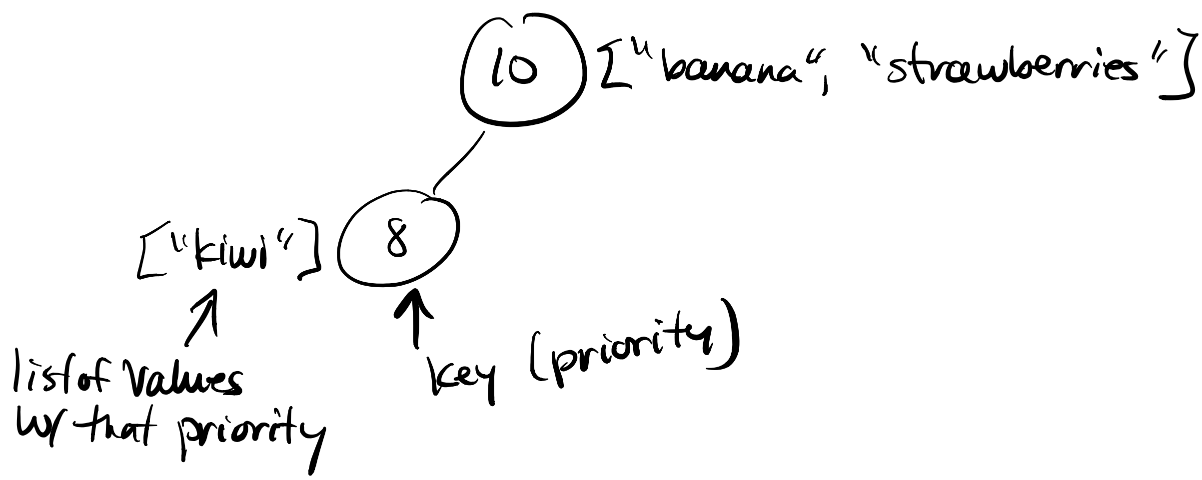
- P Priority
- V Value

ADT

```
V    removeMax()
void insert(P priority, V value)
V    getMax()          // peek at the value associated
                       //   w/ max priority
P    getMaxPriority()  // peek at max priority
bool isEmpty()
int  getSize()
```

How to implement this ADT?

1. AVL tree, keys must be unique
   To represent a PQ, keys would be priorities
   But priorities are <u>not</u> unique
   We can store all values w/ equal priorities in a list / queue



2. Sorted ArrayList
3. Unsorted ArrayList
   Here's an ArrayList sorted by priorities

<8, "kiwi">	<10, "banana">	<10, "strawberries">

PQ ops	Heap	AVL trees	Sorted AL	Unsorted AL
insert	O(lg n)*	O(lg n)	O(n)	O(1)*
removeMax	O(lg n)	O(lg n)	O(1)	O(n)
getMax	O(1)	O(lg n)	O(1)	O(n)
getMaxPri.	O(1)	O(lg n)	O(1)	O(n)

*amortized

new data structure called a Heap will be better than all of these options

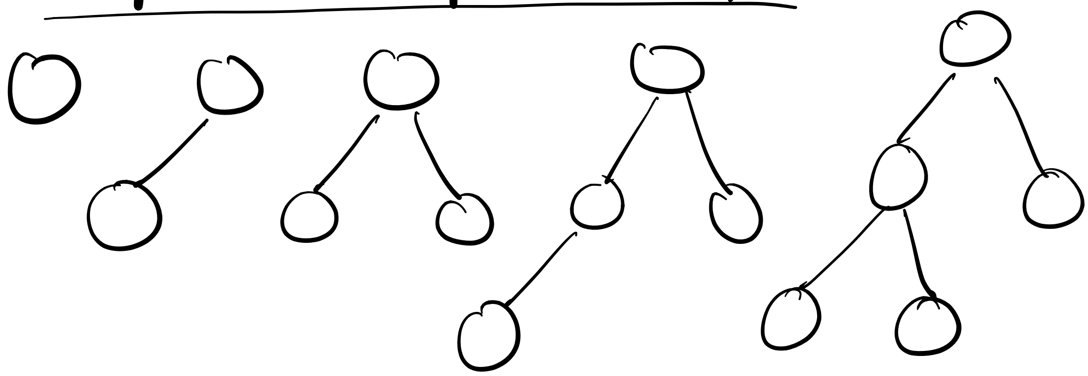
Definition of a Heap

- A complete binary tree (Not BST)
- An invariant: for every node n in the heap priority(n) ≥ priorities of all its descendants

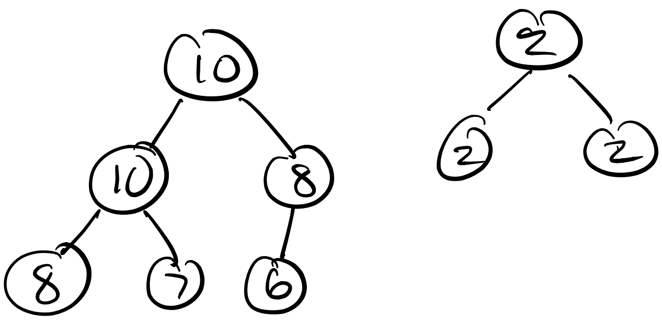
Definition of a complete binary tree

- Binary tree
- Every level is full, except potentially the last one which is left aligned

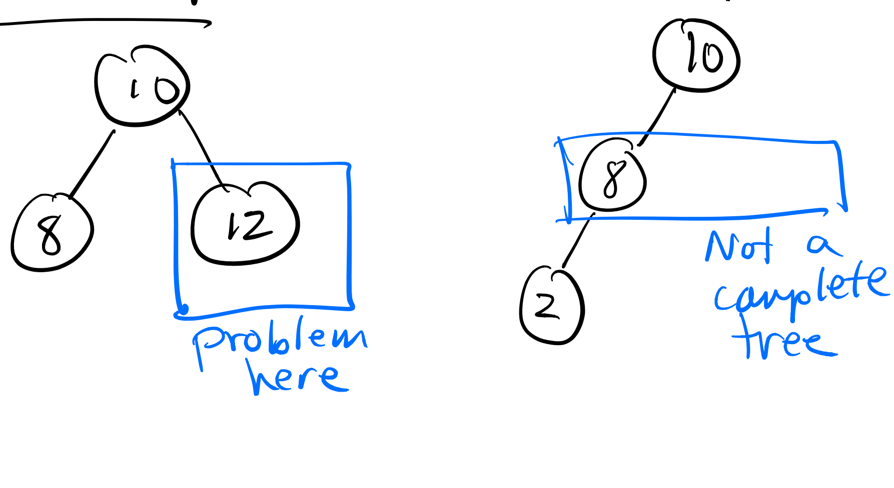
Examples of complete binary trees



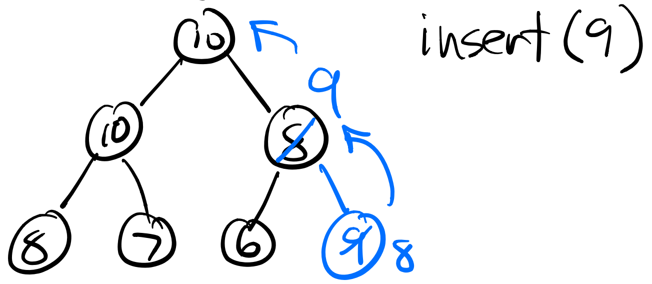
Examples of Heaps

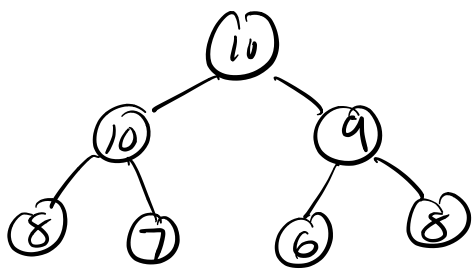


Not Heaps!



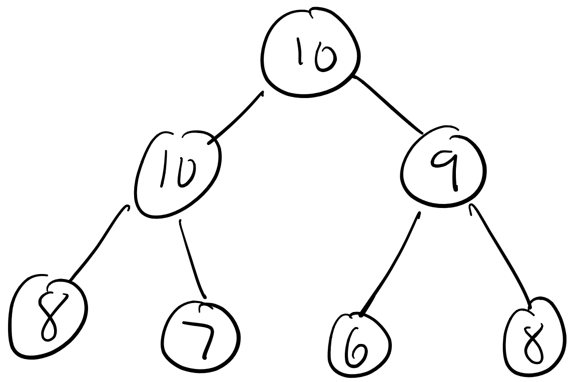
Inserting into a Heap



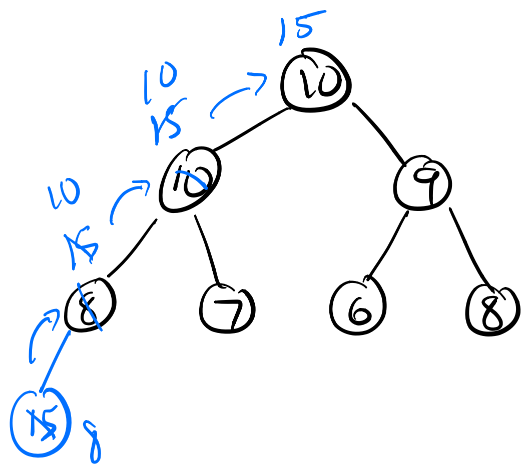


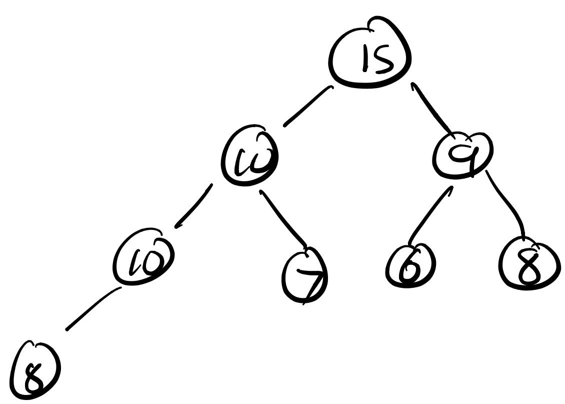
Algorithm

1. Add the new element to the next open spot in the complete binary tree
2. Fix the heap by bubbling up the new value until its priority ≤ its parent's priority or root is reached.



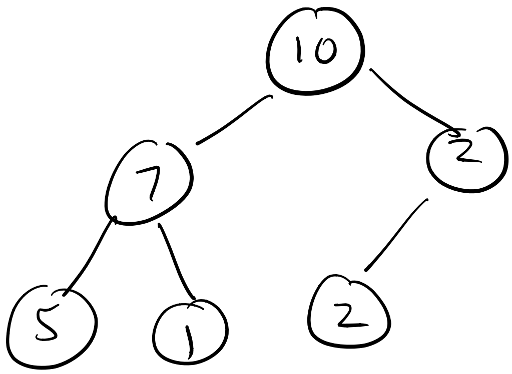
insert (15)





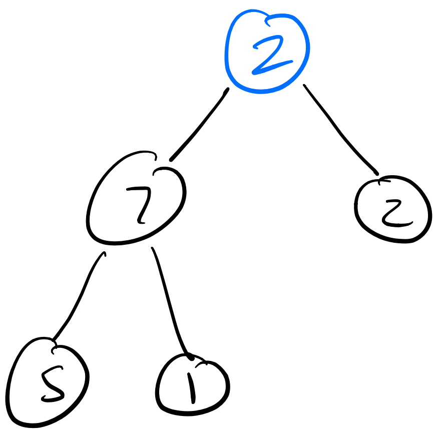
Here is the heap after inserting 15

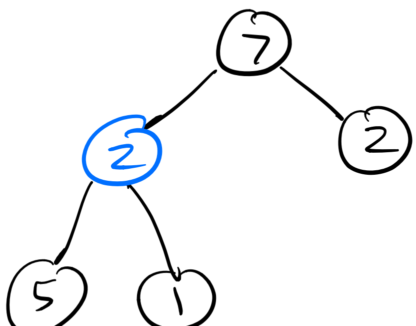
Removing from a heap

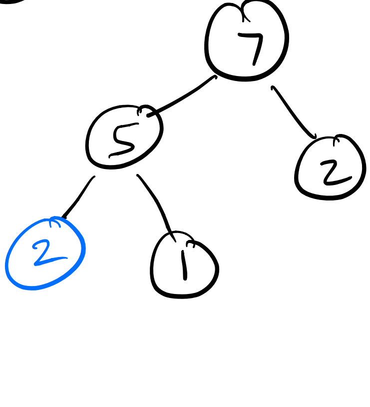


Algorithm

1. save the value at root
2. Replace data @ root with data @ last node in tree
3. Fix the heap invariant by bubbling down until node's priority ≥ the largest priority of its children or leaf is reached





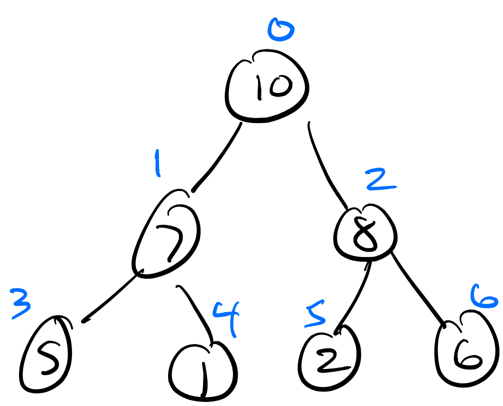


Here is the heap after removing max

Need a new representation for trees

goals:
- to get parent of a node O(1)
- to get children of a node O(1)
- find last node of a tree O(1)
- find next empty spot in complete tree O(1)

Let's use an array



traverse in level order to fill array

no gaps because tree is complete

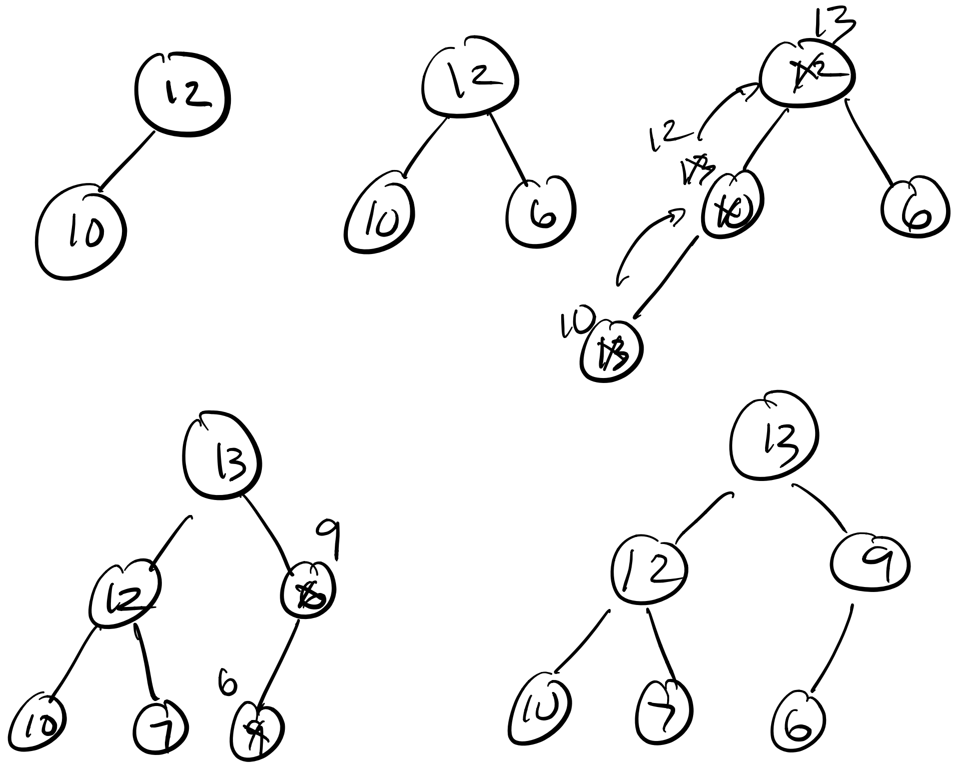
0	1	2	3	4	5	6	7	8	9
10	7	8	5	1	2	6			

left(i) – index of left child $2i+1$
right(i) – index of right child $2i+2$
parent(i) – index of parent $(i-1)/2$
uses int division

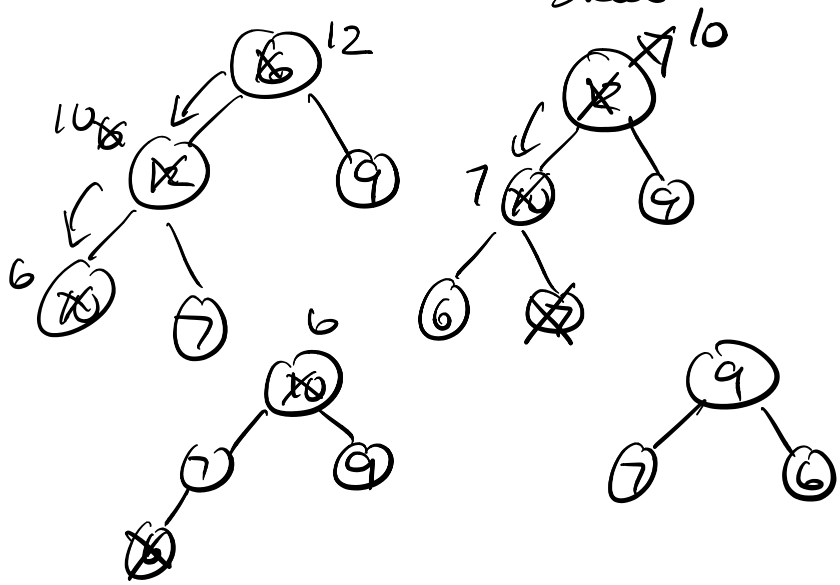
last element of tree – A[size-1]
next empty spot in tree – A[size]

These are all O(1) operations – just doing index arithmetic

Exercise: Insert the following priorities into a Heap: 12, 10, 6, 13, 7, 9



Exercise: Remove 3 times, what priorities are removed? Show how heap changes.



removed	
1.	13
2.	12
3.	10