CS 225

Priority Queues: ADT and Heaps

Priority Queue ADT <K, I>

```
void  Insert(int x);      // insert value x
                          // into collection
int   DeleteMin();        // remove & return
                          // most important item
int   FindMin();          // return most important
                          // item w/out removing it
bool  isEmpty();          // true if empty
```

others:

Search and Remove (int x)

Increase Priority (int x)

Decrease Priority (int x)

DeleteMax

FindMax

↓

min-heap

smaller values

↳ more important

race

winner: 1st place

runner up: 2nd place

⋮

max-heap

larger values

↳ more important

video game

winner: 10,962,000

runnerup: 9,861,263

unsorted List : insert : $O(1)$ wc ac

Fm/Dm : $O(n)$ wc ac

insert and delete 1 item : $O(n)$ wc ac

sorted List : insert: $O(n)$

↳ n wc

↳ n/2 ac

Fm/Dm : $O(1)$ wc ac

insert and delete 1 item : $O(n)$ wc ac

BST : insert : $O(n)$ wc

$O(\lg n)$ ac

: DM : $O(n)$ wc

$O(\lg n)$ ac

insert & delete 1 item

↳ $O(n)$ wc

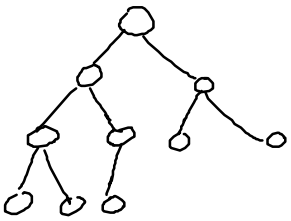
↳ $O(\lg n)$ ac

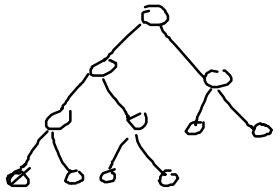
if tree is balanced, we would have $O(\lg n)$ guaranteed

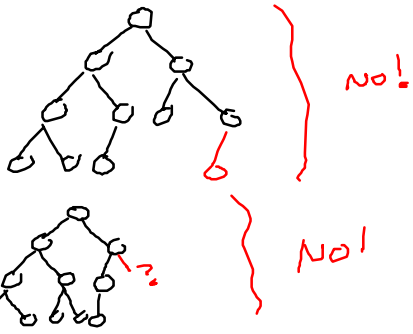
complete binary tree

depth k tree which is
1) a perfect tree of depth k-1 with 2) one or more children added at depth k, from left to right

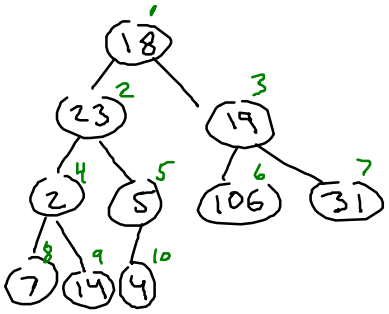
(level order traversal on a binary tree w/ every possible node & infinite depth)






NO!

?
NO!



index i

Left Child $(i) = 2i$

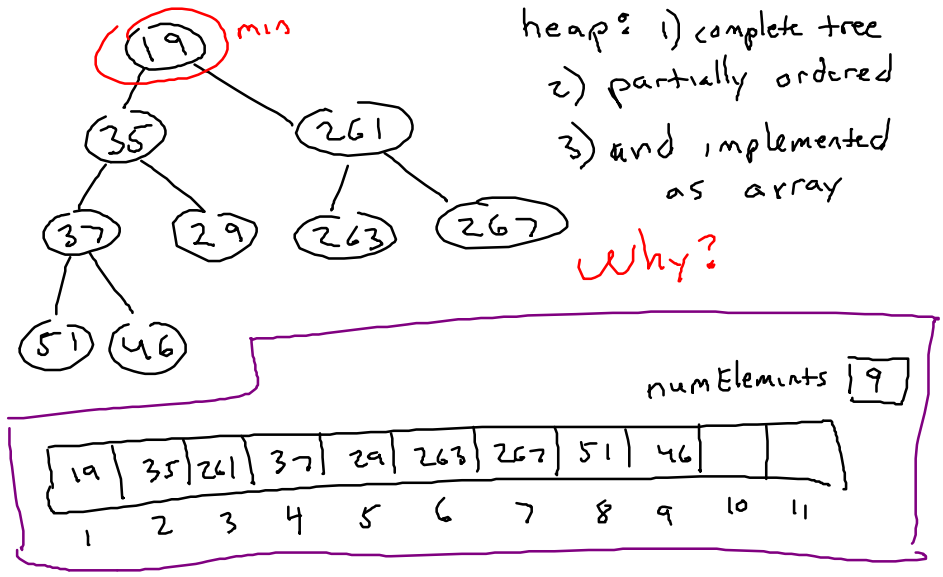
Right Child $(i) = 2i+1$

Parent $(i) = \lfloor i/2 \rfloor$

18	23	19	2	5	106	31	7	14	4
1	2	3	4	5	6	7	8	9	10

Partial ordering

↳ every node in complete tree will hold a value less than values in that node's children



heap: 1) complete tree
2) partially ordered
3) and implemented as array

why?