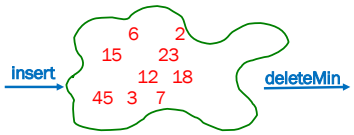
Priority Queue ADT

Priority Queue ADT
State: • Set of comparable elements   • Order based on "priority" Operations: • **insert(element)** • **deleteMin()** – returns the element with the smallest priority, removes it from the collection • **findMin()**



- Assume each item has a "priority"
  - The *lesser* item is the one with the *greater* priority
  - So "priority 1" is more important than "priority 4"
  - Just a convention, could also do a maximum priority

6/29/2022 5

5

Preliminary Implementations of Priority Queue ADT

	insert	deleteMin
Unsorted Array		
Unsorted Linked-List		
Sorted Circular Array		
Sorted Linked-List		
Binary Search Tree (BST)		

Notes: Worst case, Assume arrays have enough space

6/29/2022 9

9

Q: Some More Tree Terminology

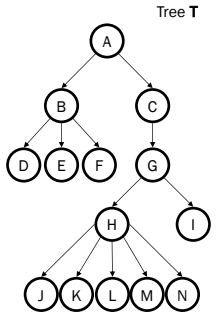
depth(B):

height(G):

height(**T**):

degree(B):

branching factor(**T**):



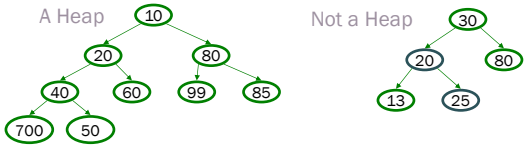
6/29/2022 15

15

Now Formalizing: Binary Min-Heap Datastructure

More commonly known as a binary heap or simply a heap

- Structure Property:
  A complete [binary] tree
- Heap-Order Property:
  Every non-root node has a priority value larger than (or possibly equal to) the priority of its parent



6/29/2022 22

22

Note: Exercises and P2 start counting from 0

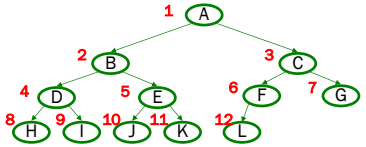
Array Representation of a Binary Heap

From node i:

left child:

right child:

parent:



	A	B	C	D	E	F	G	H	I	J	K	L	
0	1	2	3	4	5	6	7	8	9	10	11	12	13

- We skip index 0 to make the math simpler
- Actually, it can be a good place to store the current size of the heap

36

Note: Exercises and P2 start counting from 0

Example

1. insert: 16, 32, 4, 57, 80, 43, 2
2. deleteMin

0	1	2	3	4	5	6	7

41

Note: Exercises and P2 start counting from 0

Floyd's `buildHeap` Method

percolateDown, **bottom-up**:

- Leaves are already in heap order
- Work up toward the root one level at a time

```
void buildHeap() {
  for(i = size/2; i>0; i--) {
    val  = arr[i];
    hole = percolateDown(i,val);
    arr[hole] = val;
  }
}
```


51