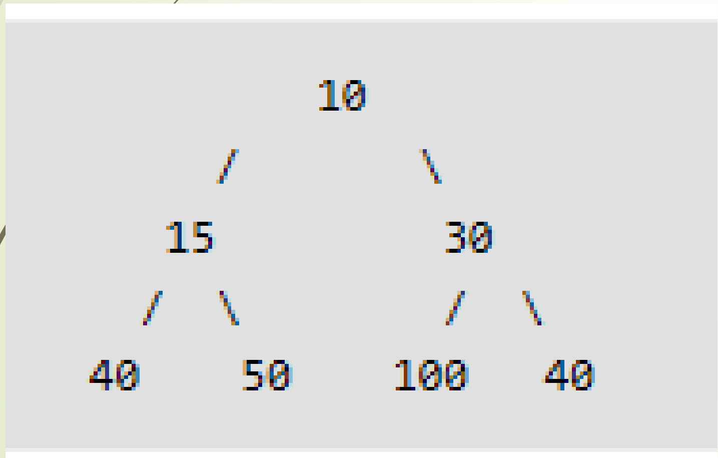
Binary Heap

Pittsford Sutherland Programming Club

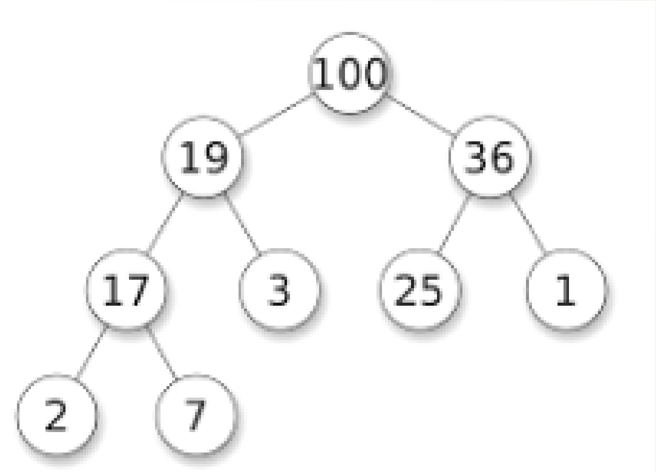
By Yizuo Chen, Yiyou Chen

Two Types of Binary Heap

- Max Heap: root’s value is greater than its sons
- Min Heap: root’s value is less than its sons



Min Heap



Max Heap

Properties of Binary Heaps

```
        10
      /     \
    15        30
   /  \      /  \
 40    50  100   40
       5    6    7
```

For each node with rank i, its left child's rank is 2i, and its right child's rank is 2i +1; its parent's rank is [i / 2].

Rank	1	2	3	4	5	6	7
Value	10	15	30	40	50	100	40

Build a heap

- **a[i]:** We can use an array to save the elements of heap. For each value a[i], a[i / 2] is its parent, a[i * 2] is its left son, and a[i * 2 + 1] is its right son.
- **size:** the total number of elements in the array

Insert an element

Rank	1	2	3	4	5	6	7
Value	10	15	30	40	50	5	40

When Insert 10 into the heap, there's no elements added before it. So create a new node and make it the root.

Insert an element

Rank	1	2	3	4	5	6	7
a[rank]	10	15	30	40	50	5	40

when insert 15 into it, compare 15 (a[2]) with 10 (a[1]); since 15 is greater than 10, we leave them there.

15

10

Insert an element -- shift up

Rank	1	2	3	4	5	6	7
a[rank]	10	15	30	40	50	5	40

The special case: when we are adding 6th element 5 into the heap, it's less than its parent 30, so we swap 30 and 5.

Rank	1	2	3	4	5	6	7
a[rank]	10	15	5	40	50	30	40

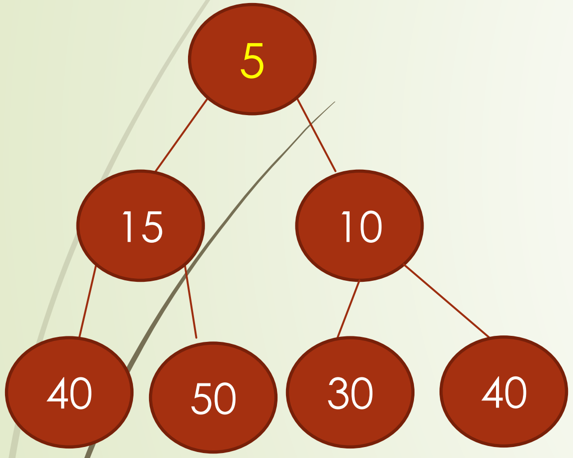
Then compare a[3] 5 with a[1] 10, a[3] < a[1], so we swap 5 and 10 again.

Rank	1	2	3	4	5	6	7
a[rank]	5	15	10	40	50	30	40

Now, it is a Min heap again.

Remove the root

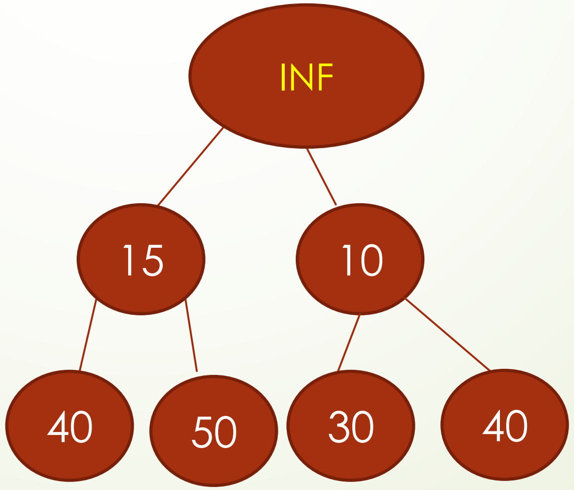
Rank	1	2	3	4	5	6	7
a[rank]	5	15	10	40	50	30	40



The Heap we got before

Since we want to remove the root, we want to make 5 disappear.

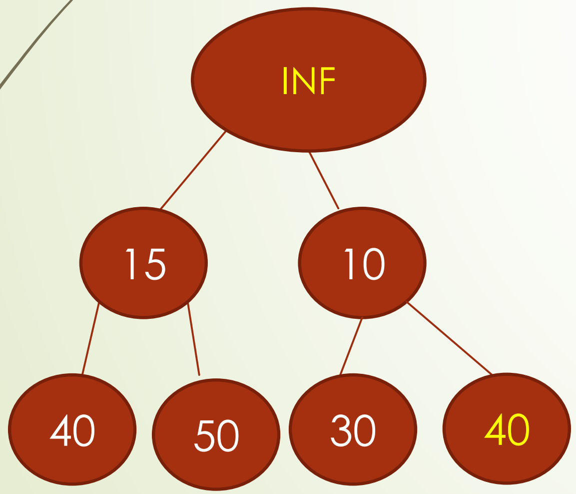
Step1: change the root value to infinite.

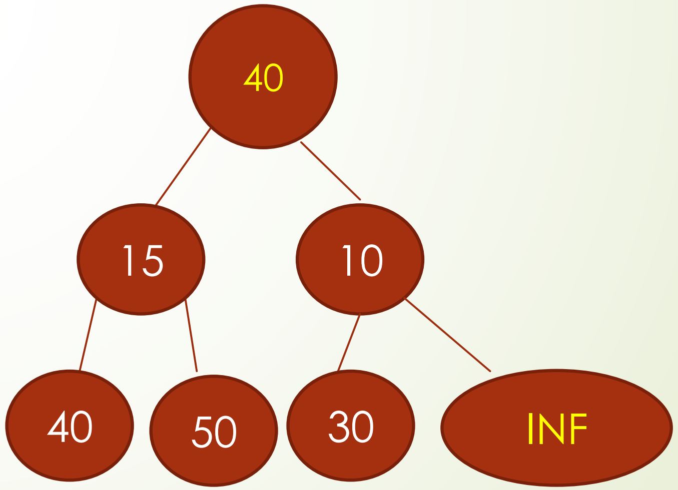


Remove the root

Rank	1	2	3	4	5	6	7
a[rank]	INF	15	10	40	50	30	40

Step2: Swap root element with the last element

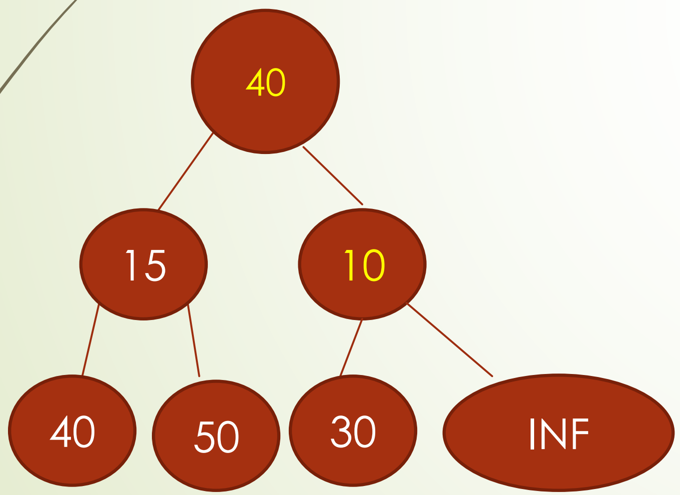


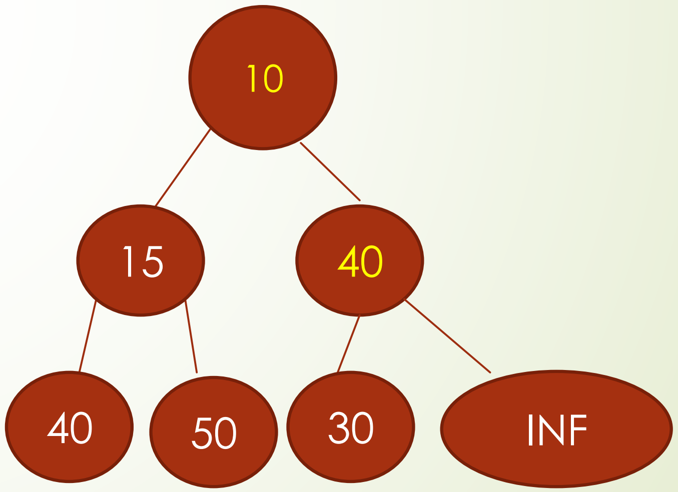


Remove the root

Rank	1	2	3	4	5	6	7
a[rank]	40	15	10	40	50	30	INF

Step3: Shift down, swap the new root value with its smallest child.

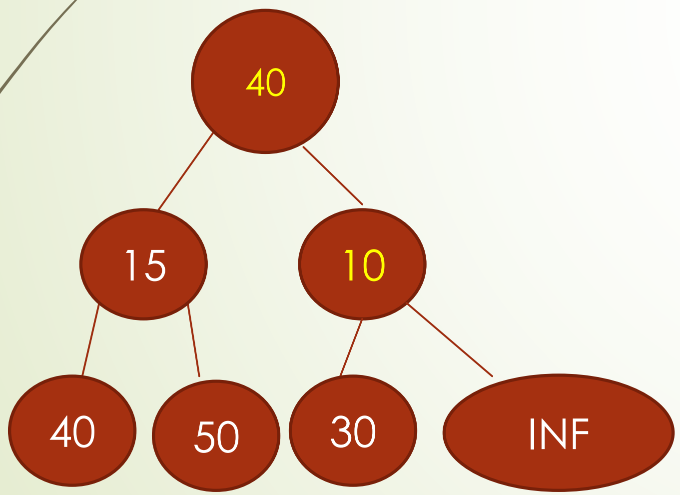


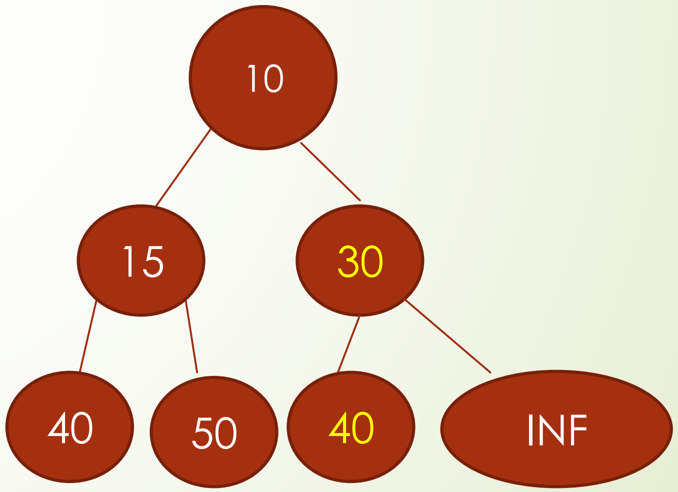


Remove the root

Rank	1	2	3	4	5	6	7
a[rank]	10	15	40	40	50	30	INF

Step4: Shift down, swap again, until it becomes a heap

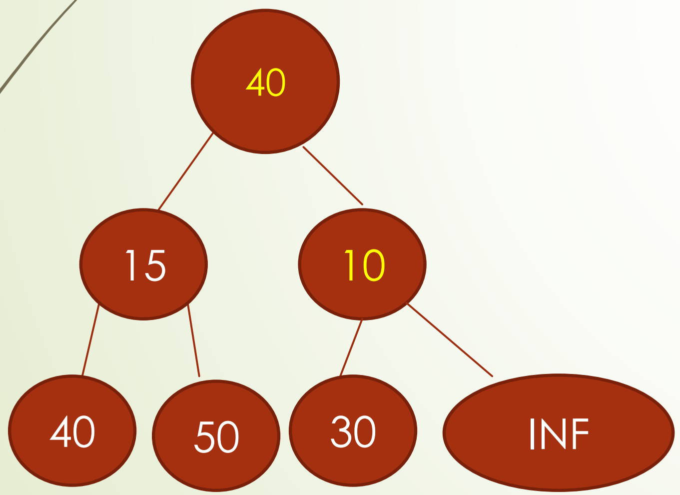


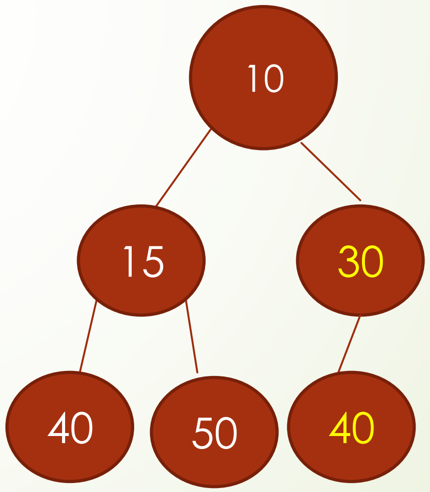


Remove the root

Rank	1	2	3	4	5	6
a[rank]	10	15	30	40	50	40

Step5: decrease the heap's size by one.





Done!

Time complexity of Binary Heaps

- Either “shift up” or “shift down” goes through the height of the heap.
- So the time complexity of insertion and deletion is O(log n). (Very efficient)

Applications: HeapSort

- If we have a series of numbers and we need to sort them in order, we can actually use heap as a tool to make the sort very efficient.
- How?

1. Add all the elements to a heap. (Build a heap).
2. Output the root.
3. Remove the root.
4. Output the root.
5. Remove the root.
6. Output the root.
7. Remove the root, etc.

Time Complexity: Average case = Worst case = O(n log(n))