Binary Heap

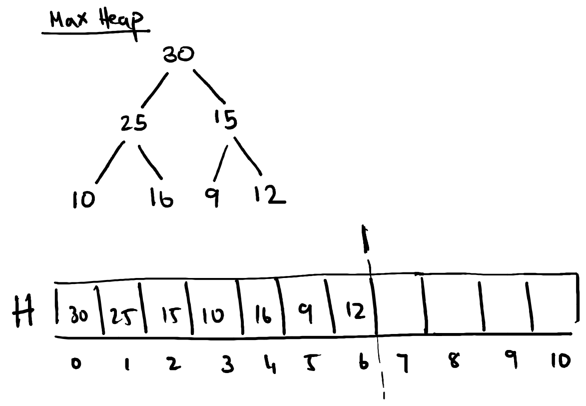
Monday 17 June 2024 6:10 PM

Content

1. What is Heap
2. Insert in a Heap
3. Deleting from Heap
4. Heap sort
5. Heapify
6. Priority Queues

1. Heap
   - A complete binary tree (height always logn)
   - Implemented using array
   - Duplicates allowed
   - every node is greater than or equal to all it descendants. [Max Heap]
   - every node is smaller than or equal to all it descendants [Min Heap]

2. Max Heap



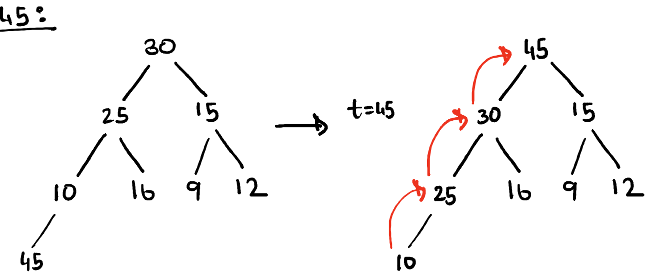
element (i)

lchild (2i+1)

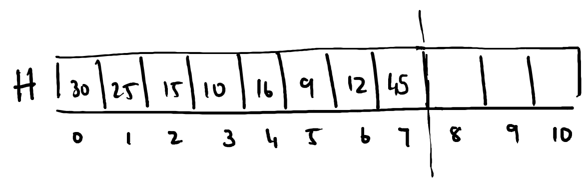
rchild (2i+2)

Parent $\left\lfloor \frac{i-1}{2} \right\rfloor$

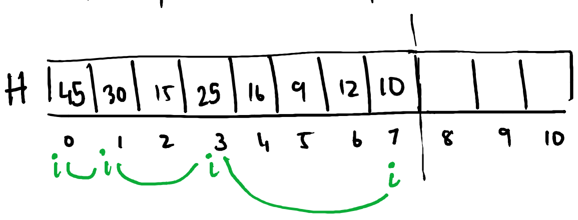
Insert 45:



①



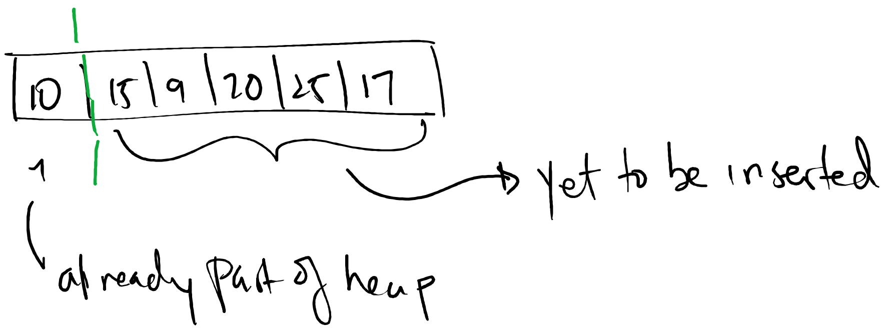
② t=45 , compare 45 with its parent



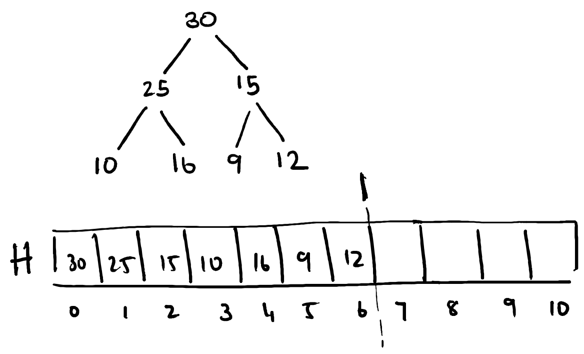
→ we don't require extra array/memory for Heap

we can have an inplace heap

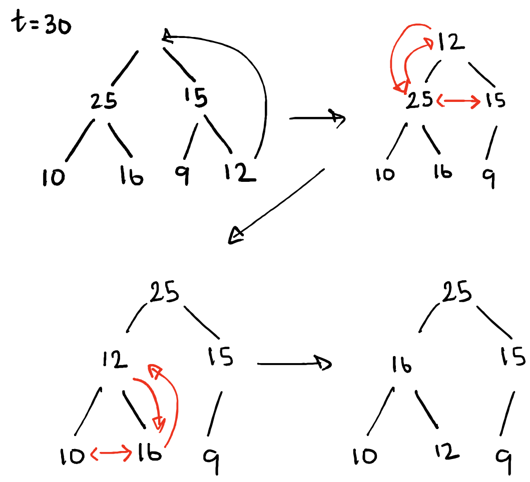
Consider the first element as part of heap & the right side as elements to be inserted.



3. Max Heap



* From Heap we can only delete root element



4. Heap Sort

1. Create Heap of n elements
2. Delete 'n' elements 1 by 1
3. Replace deleted elements into emptspace in reverse fashion

O(nlogn)

5. Heapify

→ Faster method of creating a heap

instead of working out way from left to right
ie. inserting element & comparing with parent

we will start from the end of the array
& compare with CHILDREN
if necessary will will shift elements.

for max heap we will swap will the larger
of the available children.

O(n)

6. Binary Heap as Priority Queue

Insert — O(logn)
Delete — O(logn)

Array Implementation

insertion — O(1)

... O(n)

Delete - O(log n)

Insertion - O(1)
Deletion - search & shift element O(n)