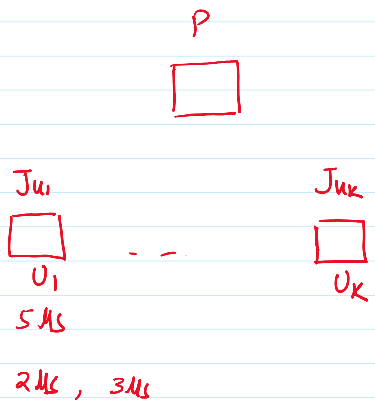
12) Lecture 29-30.5 | Priority Queue + Heap Sort |

15 October 2024 08:19

Motivation from Scheduling Problem:

- In a multi-user computer system, multiple users submit jobs to run on a single processor.
- We assume that the time required by each job is known in advance. Further, jobs can be preempted (stopped and resumed later)
- One policy which minimizes the average waiting time is SRPT (shortest remaining processing time).
- The processor schedules the job with the smallest remaining processing time. If while a job is running a new job arrives with processing time less than the remaining time of current job, the current job is preempted



① Need to maintain remaining processing time of all unfinished job, at any point of time

② Need to find job with the shortest remaining processing time.

③ When a job finishes, need to remove it from the list

④ " new job arrives need to add it to the list.

Priority Queue maintains a $\overset{S}{\text{set}}$ of elements each with some associated priority

Supports the following operations.

- Insert (X) : insert an element X in the set S $S \leftarrow S \cup \{X\}$
- Minimum(): return element with smallest priority
- Delete_min(): return & remove elements S with smallest priority.

Naive Implementation

① Maintain a sorted array

Insert (X) : Finding $\overset{O(\log n)}{\text{correct location}}$ + $\overset{O(n)}{\text{Shifting}}$:= O(n)

Minimum () O(1)

Insert (X) : Finding correct location + Shifting := O(n)

Minimum () O(1)

Delete-min () O(1)

② Maintain Unsorted array

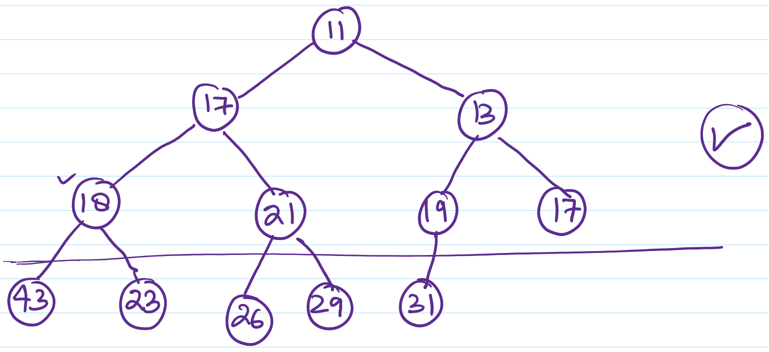
Insert(X) O(1)

Minimum () O(n)

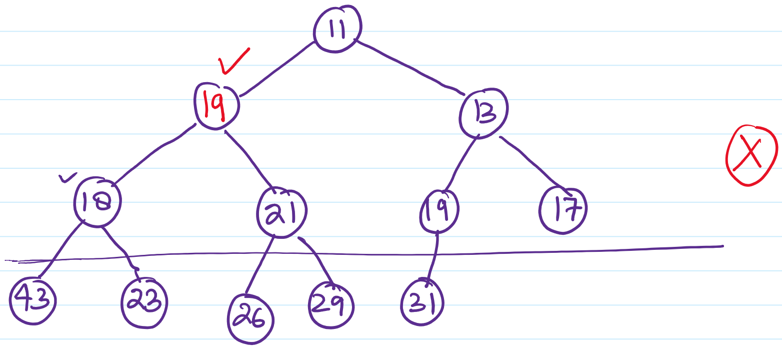
Delete-min () O(n)

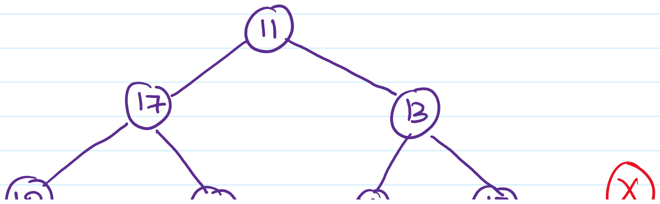
Clever Approach: Heap data structure:

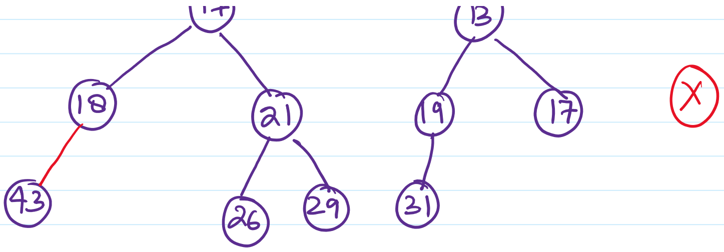
- A binary tree that stores priority at nodes. ✓
- All levels except last are full. ✓
- Last level is left filled ✓
- Priority of a node should be smaller than its children.



Tree is called "Heap"







Finding the mininm element

The element with smallest priority always sits at the root of heap.

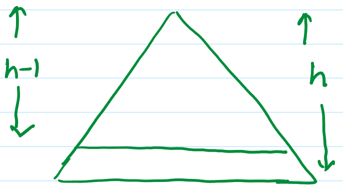
(If it was any where else it would have a parent with larger priority which will violate the heap property)

Compute the minimum in O(1) time

Height of Heap

Suppose a heap over n node has height h

$$(\underbrace{2^{h}-1}_{\substack{\downarrow \\ \#\text{ of nodes till level }(h-1)}})+1 \leqslant n \leqslant 2^{h+1}-1$$



$$2^{h} \leqslant n \leqslant 2^{h+1}-1$$

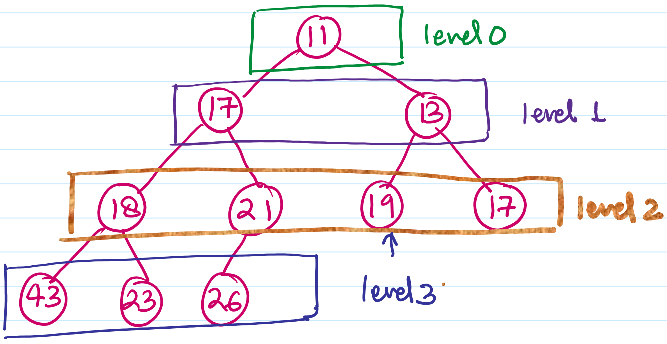
$$\Rightarrow h = O(\log_2 n)$$

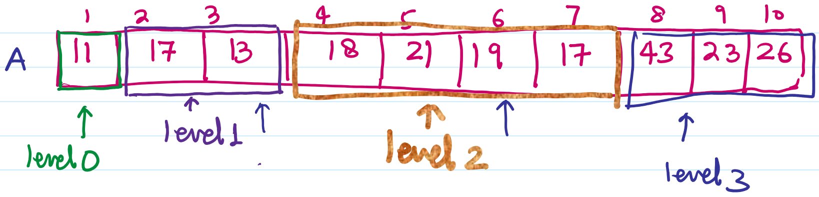
Implementing Heaps

Parent (A[i])
    return A[i/2]

Left (A[i])
    return A[2i]

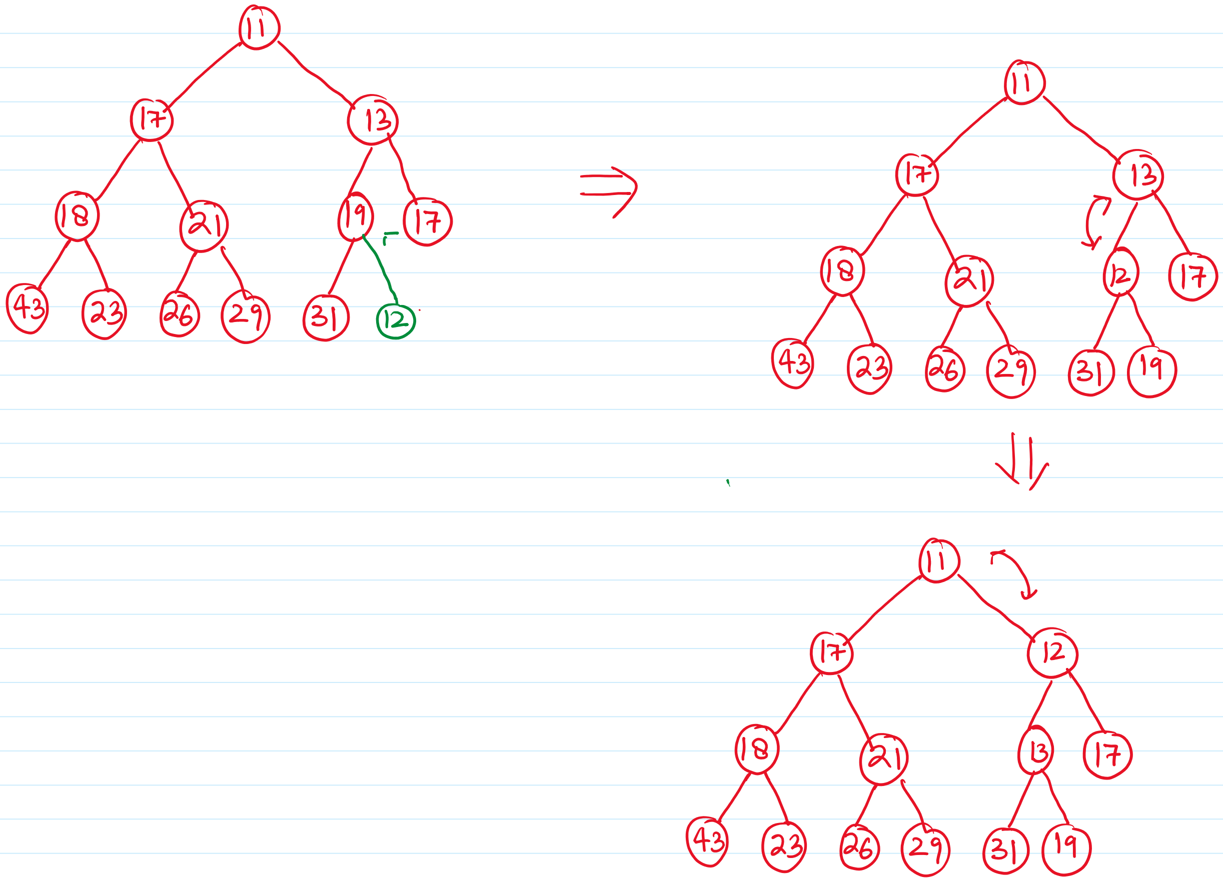
Right (A[i])
    return A[2i+1]





Insertion in a Heap:

Insert 12



Proof of correctness:

- The only nodes whose content changes are on the path from newly inserted node to root

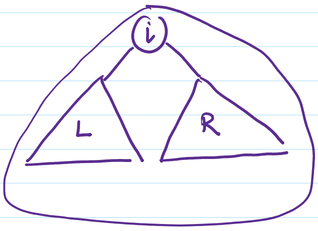
- We swap nodes if required on this path

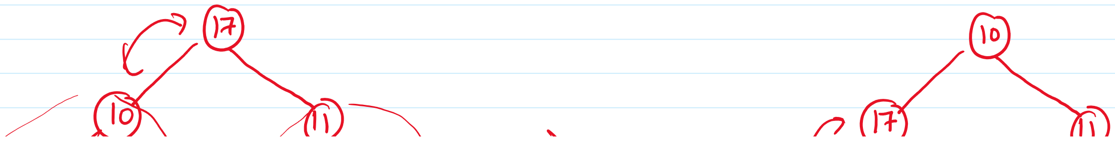
⇒ Heap property is not violated.

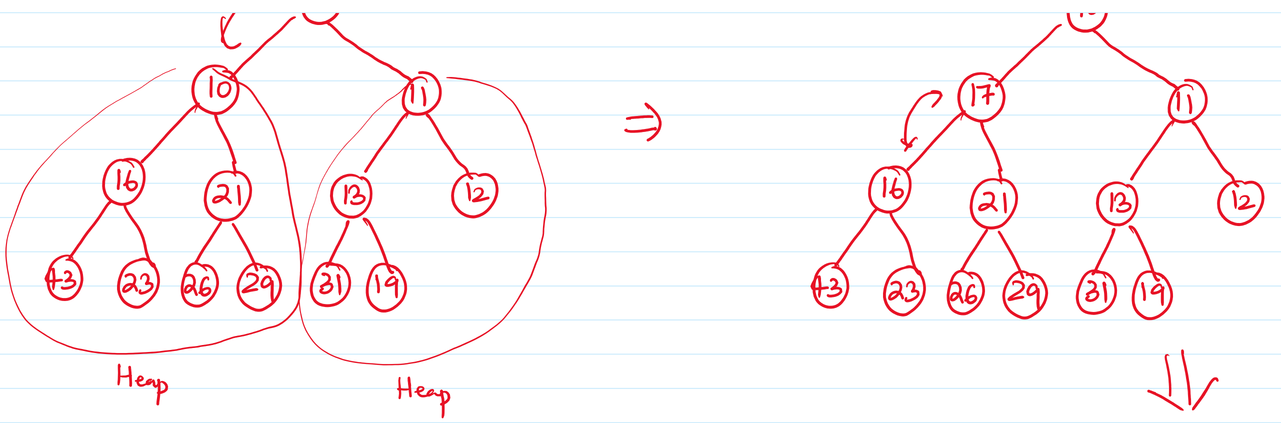
Time complexity: $O(\log n)$

Heapify:

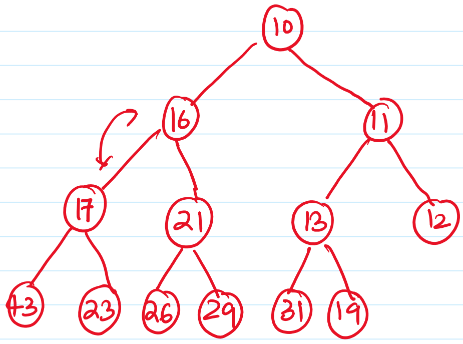
- i be the index into array A
- Binary tree rooted at left(i) and right(i) are heaps (L, R are heaps)
- A[i] may be bigger than it children violating heap property







Running time O(logn)


10
16
11
17
21
13
12
43
23
26
29
31
19