Priority Queue

Complexities.

1. add → logn
2. remove → logn
3. peek → O(1)
4. size → O(1)
5. Display → O(n)

Max priority

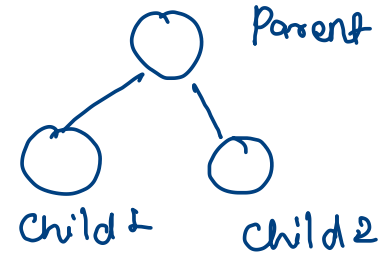
priority(30) > priority(10, 20)

1. Heap order property.
2. Complete Binary Tree.

Priority → Min / Max

Heap order property.

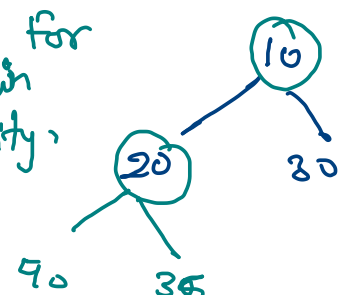
Min Heap →

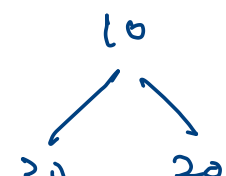


str1 = abc
str2 = yaten
abc < yaten

priority(Parent) > priority(child1, child2)

for min priority,





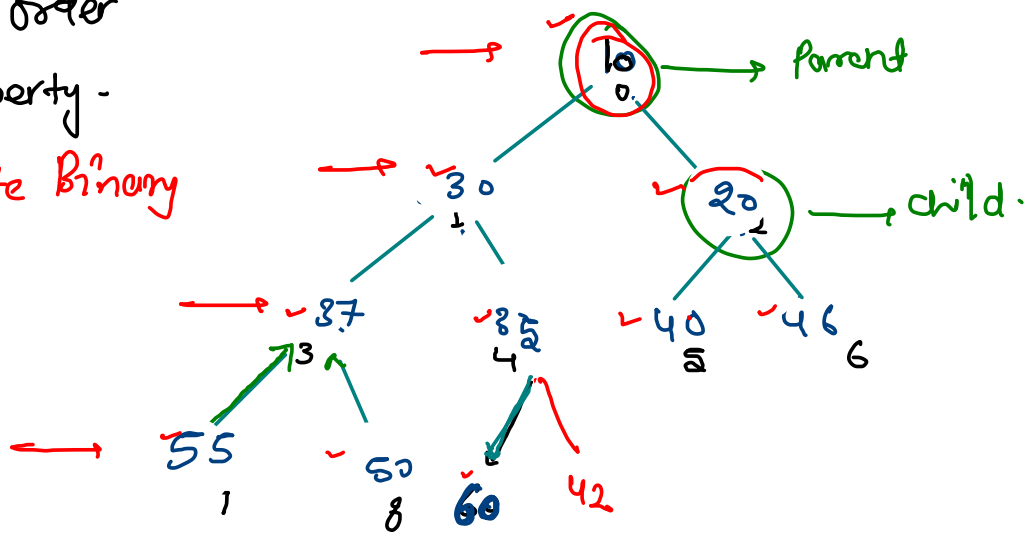
AL →

0	1	2	3	4	5	6	7	8	9	10
10	20	30	35	37	40	46	50	55	60	42

PQ → min.

Heap order property.

Complete Binary Tree



Indirect connection b/w parent and child. →

parent index = pi

left child = 2*pi + 1

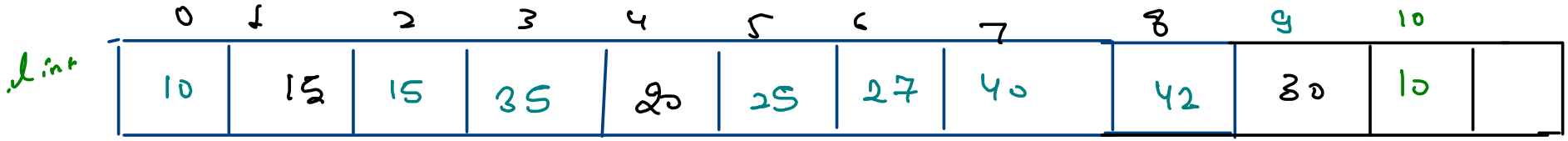
right child = 2*pi + 2

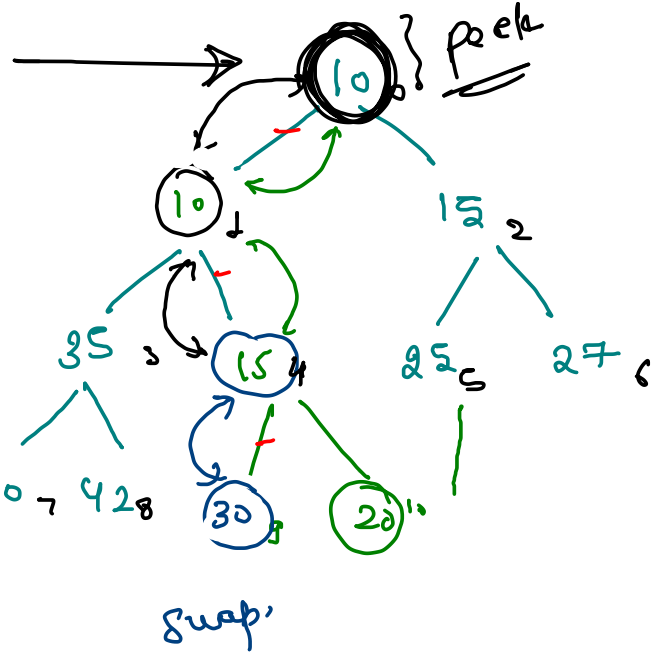
In Tree → we can move in downward direction only.

In ArrayList, we can move upward too →

child index = i

parent index = (i-1)/2 ;





$\log(n)$

$n = 9$

$= \log_2(9)$

$= \log_2(2^3)$

$= 3$

add complexity → $\log(n)$

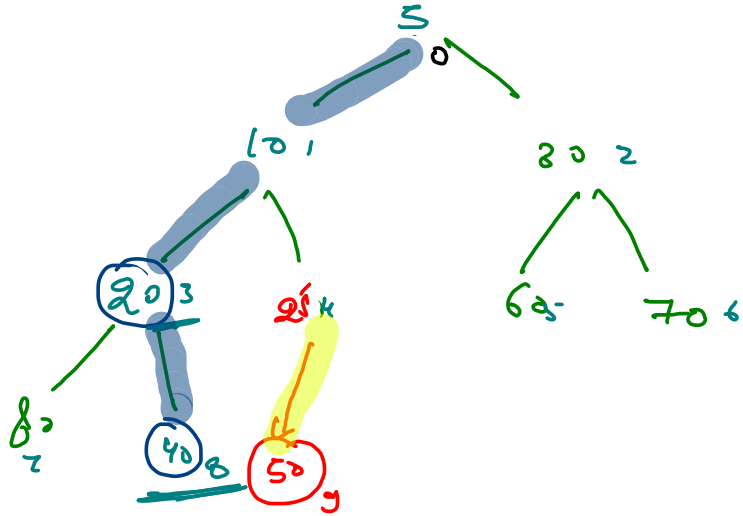
data.add(15) → O(1)

// maintain H.O.P. (Heap order property)

→ upheapify( ) complexity → $\log(n)$

pop ⟶ data.get(0);

Upheapify.



data.add(5)

data.add(25)

indx = ~~8~~ 4.

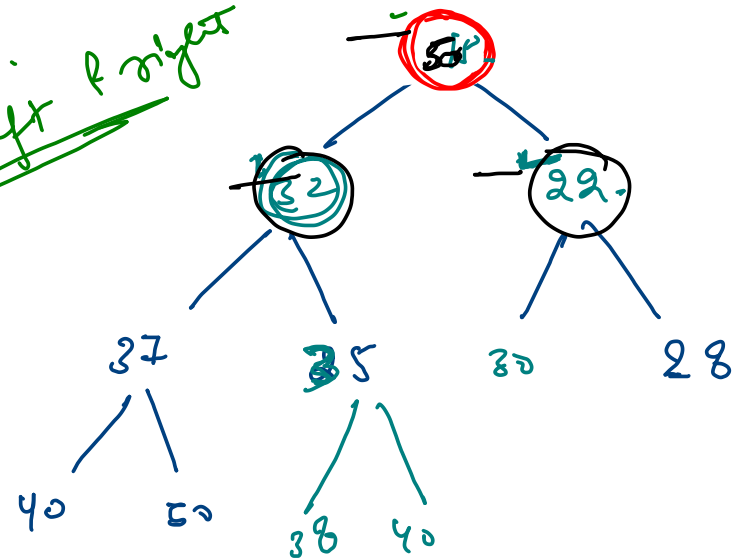
pi = ~~4~~ 1

Heap order property is maintained.

```
upheapify(indx){
    if(indx==0) return;
    pi = (indx-1)/2.
    if(data.get[indx] < data.get(pi)){
        swap(data, indx, pi);
        upheapify(pi);
    }
}
```

Remove =>

smallest b/w left & right



level order.

downheapify.

minIdx

peek() → data[0]

pq.remove() → log(n)

data.remove(0) → O(n)

remove

swap(0, data.size-1) O(1)

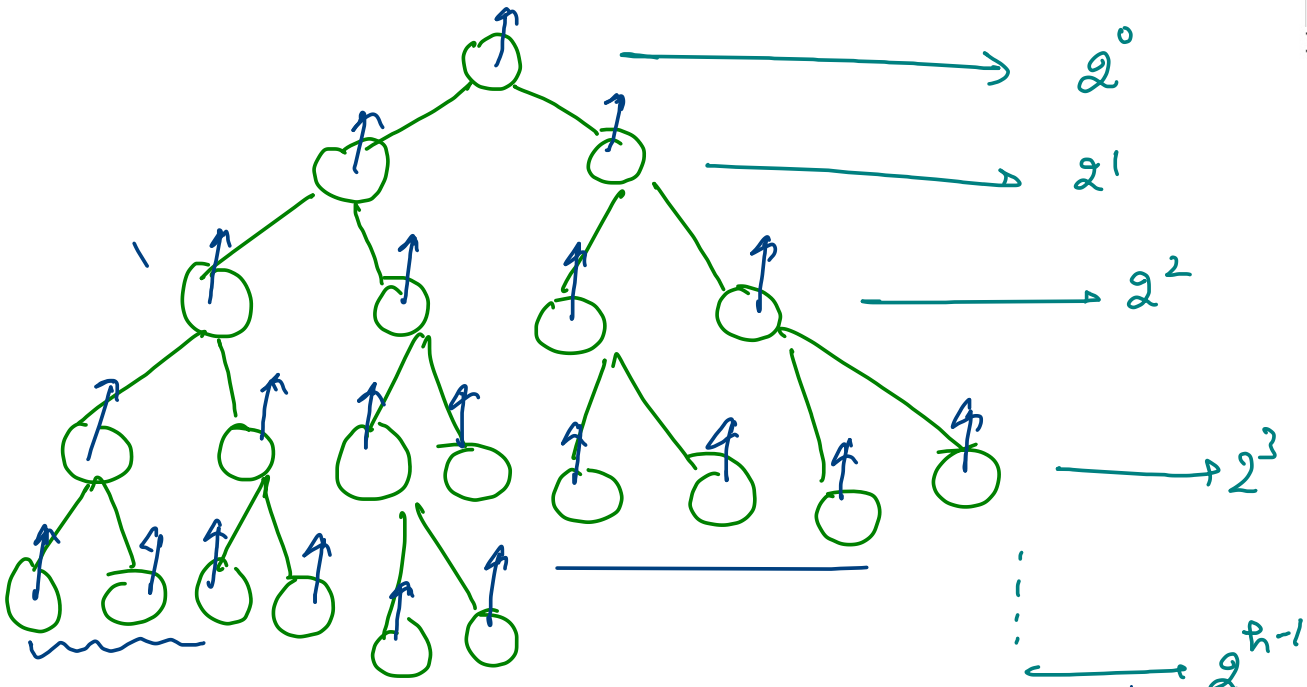
int val = data.remove(size-1);

// maintain heap ord. prop.

downHeapify(0);

return val;

array passing through constructor:




```
private void processArray(int[] arr) {
    for(int i = 0; i < arr.length; i++) {
        add(arr[i]);
    }
}
```

upheapify

} upheapify

Time complexity Required for one element = 'h'

h-levels

$h = \log n$

$T(n) = 2^0 \times 1 + 2^1 \times 2 + 2^2 \times 3 + \cdots + 2^{h-1} \times (h)$

$T(n) = 2^{h-1} \times h + \cdots + 2^0 \times 1$

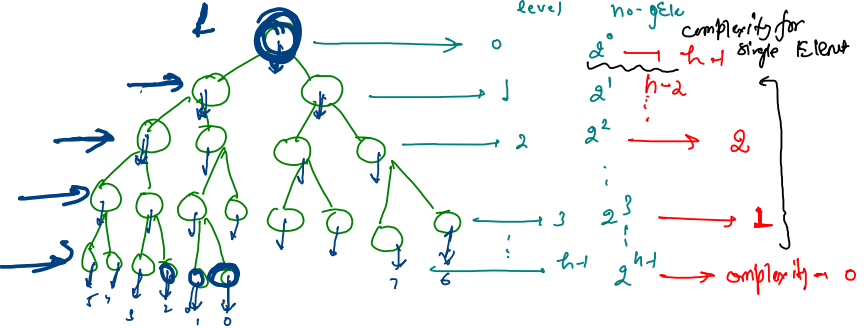
$= \frac{2^h}{2} \times h + \cdots + 2^0 \times 1$

$h = \log n$

$= \frac{2^{\log_2 n}}{2} \times \log n = \frac{n}{2} \times \log n$

⟶ if we consider only first term, it is $n \log n$

$a^{\log_a b} = b$



(1) fill array in data list

(2) down heapify from end of data list to 0 index

$$T(n) = 2^0 \times (h-1) + 2^1 \times (h-2) + 2^2 \times (h-3) + \dots + 2^{h-1} \times (0) \rightarrow (1)$$

$$2 \times T(n) = 2^1 \times (h-1) + 2^2 \times (h-2) + 2^3 \times (h-3) + \dots + 2^h \times 0 \rightarrow (2)$$

eq (2) − eq (1)

$$2T(n) - T(n) = -2^0 \times (h-1) + 2^1(h-1-h+2) + 2^2(h-2-h+3) + 2^3(h-3-h+4) + \dots + 2^{h-1}(1-0) + 2^h \times 0$$

$$T(n) = -h + 1 + 2^1 + 2^2 + 2^3 + \dots + 2^{h-1} + 0$$

sum of G.P $= \frac{a(r^n - 1)}{r-1}$

$$T(n) = -h + \frac{1(2^h - 1)}{2-1}$$

$$T(n) = -h + 2^h - 1$$

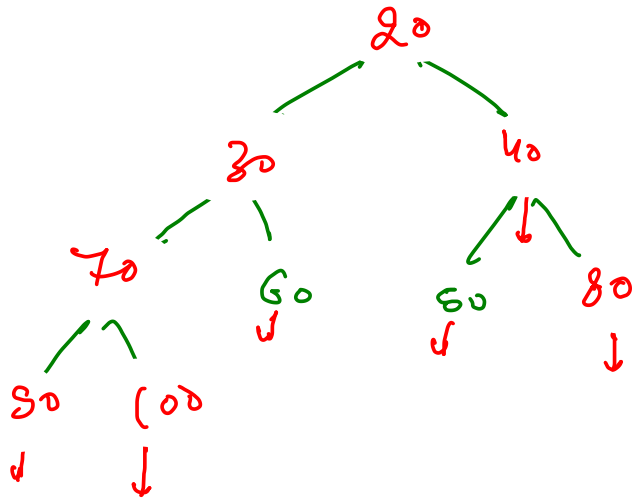
$h = \log n$

$$T(n) = 2^h - (h+1)$$

$$T(n) = 2^{\log n} - (\log n + 1)$$

$$T(n) = n - (\log n + 1)$$

$$T(n) = O(n)$$



① | 100 | 90 | 80 | 70 | 60 | 50 | 40 | 30 | 20 |

→ ② Down heapify from end to start in arraylist

Topic cover till now → all the question