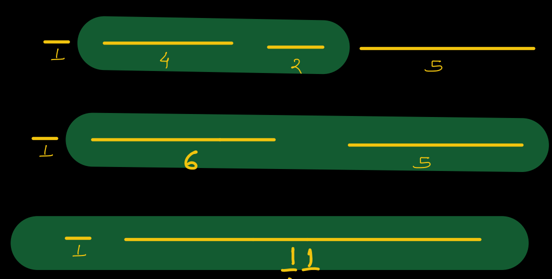
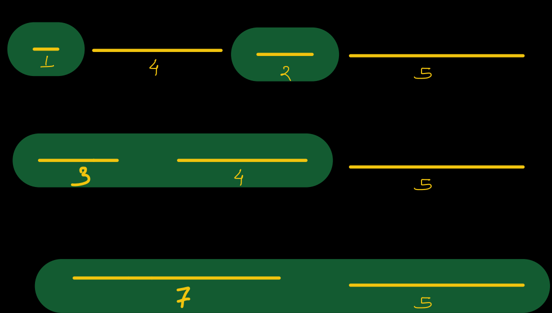
Amazon
MS
Paypal

Q. Given lengths of N ropes.
Merge all of them to form a single rope.
In one merge → pick 2 ropes & merge them.
Cost (merge (r_1 , r_2)) ⇒ len(r_1) + len(r_2)
Minimize the overall cost of merging.



⇒ 4+2 = 6
⇒ 6 + 5 = 11
⇒ 1 + 11 = 12

total cost = 29



⇒ 1 + 2 ⇒ 3
⇒ 3 + 4 = 7
⇒ 7 + 5 = 12

total cost = 22

$r_1 < r_2 < r_3$

$merge(r_1, r_2) \rightarrow r_1 + r_2$

$merge(r_1\text{-}r_2, r_3) \rightarrow r_1 + r_2 + r_3$

$\Sigma = (r_1 + r_2) + (r_1 + r_2 + r_3)$

$merge(r_1, r_3) \rightarrow r_1 + r_3$

$merge(r_1\text{-}r_3, r_2) \rightarrow r_1 + r_2 + r_3$

$\Sigma = (r_1 + r_3) + (r_1 + r_2 + r_3)$

$merge(r_2, r_3) \rightarrow r_2 + r_3$

$merge(r_{23}, r_1) \rightarrow r_1 + r_2 + r_3$

$\Sigma = (r_2 + r_3) + (r_1 + r_2 + r_3)$

Optimal Strategy : Always pick two ropes of min possible size.

4, 3, 5, 8, 6, 9 $\xrightarrow{\text{sort}}$ 3, 4, 5, 6, 8, 9

7

$\rightarrow O(N)$

5, 6, 7, 8, 9

11

$\rightarrow O(N)$

7, 8, 9, 11

$\rightarrow O(N)$

TC : $O(N \log N) + O(N^2)$

getMin() / extract Min() $\rightarrow O(\log N)$

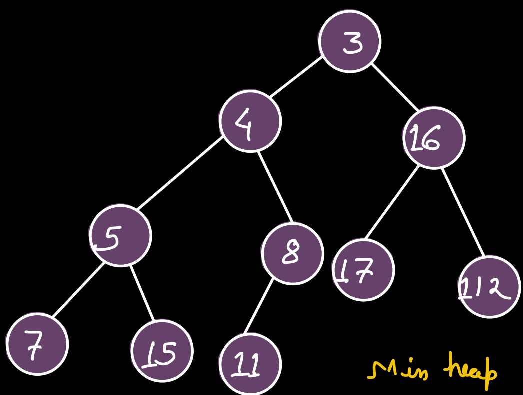
insert(x) $\rightarrow O(\log N)$

Heap

Heap

- Complete binary tree → All levels are completely filled except possibly the last level.
  last level nodes → left aligned.
  - Ht → $\log N$
  - Nodes in last level $\approx N/2$

- Min OR max property → Value of a node must be greater / smaller than both LST & RST
  (Max Heap) (Min Heap)
  - Has to be followed by all the nodes.



Min heap

Store heap in array

* No need of left & right pointer

* Move from child to parent in $O(1)$

$i \xrightarrow{\text{left}} 2i+1$

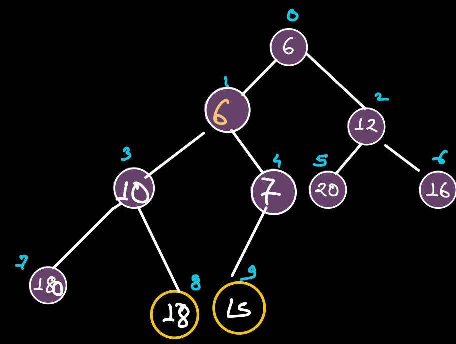
$i \xrightarrow{\text{right}} 2i+2$

$j \xrightarrow{\text{Parent}} \frac{j-1}{2}$

Insert a val in a heap

0	1	2	3	4	5	6	7	8	9
6	6	12	10	7	20	16	180	18	15

↑ size



$i = 8$

$\text{Parent}(i) = \frac{8-1}{2} = 3$

$i = 3$

$\text{Parent}(i) = \frac{3-1}{2} = 1$

$i = 1$

$\text{Parent}(i) = \frac{1-1}{2} = 0$

TC: $O(\log N)$

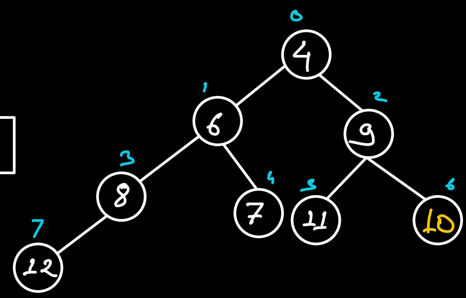
```
void insert ( int K) {
        size++;
        A[size] = K;
        i = size;
        while ( i > 0 ) {
            P = (i-1)/2 ;
            if ( A[P] > A[i]) {
                swap( A[P], A[i]);
                i = P;
            }
            else {
                break;
            }
        }
}
```

Percolate up
or
Shift up

Delete min from min-Heap

0	1	2	3	4	5	6	7	8	9	10
4	6	9	8	7	11	10	12	3		

↑ size

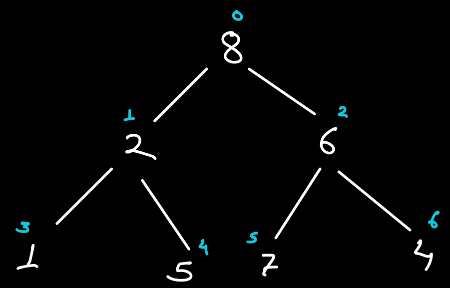


Percolate down
Or
Shift down

```
int extract Min ( ) {
    ans = A[0];
    swap( A[0], A[size]);
    Size--;
    i = 0;
    While ( i < size ) {
        int minIdx = i;
        int l = 2i+1 , r = 2i+2;
        if ( l <= size && A[l] < A(minIdx) {
            minIdx = l;
        }
        if ( r <= size && A[r] < A(minIdx) {
            minIdx = r;
        }
        if( minIdx == i) {
            break;
        }
        swap( A[i] , A[ minIdx ]);
        i = minIdx;
    }
    ret ans;
}
```

Q Given an array. Convert it to a min heap.

0	1	2	3	4	5	6
8	2	6	1	5	7	4

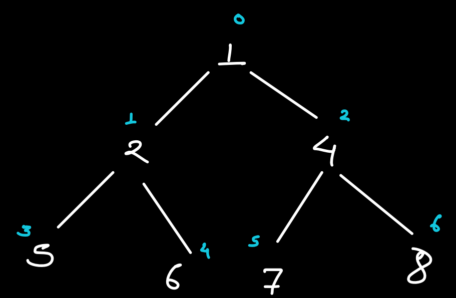


Approach 1

Sort the array

0	1	2	3	4	5	6
1	2	4	5	6	7	8

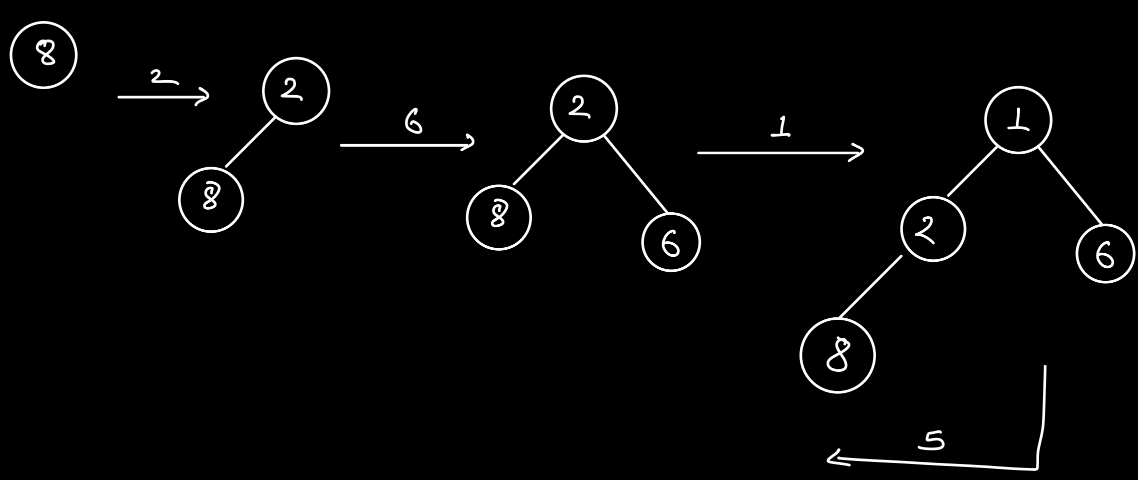
TC : O(N log N)



Approach 2

0	1	2	3	4	5	6
2	8	6	1	5	7	4

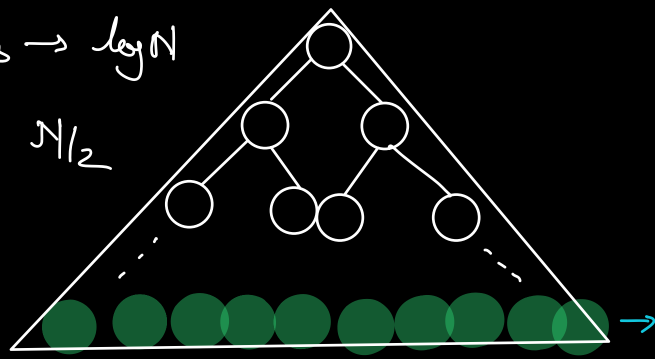
size



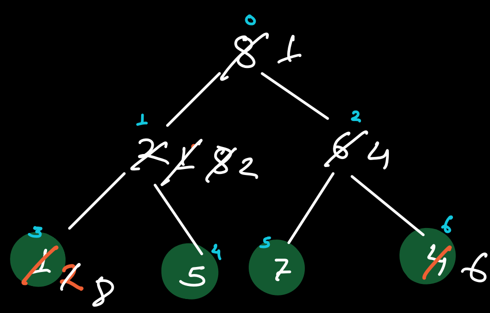
height for last level nodes → $\log N$

Count of last level nodes → $N/2$

TC : $O(N \log N)$



0	1	2	3	4	5	6
8	2	6	1	5	7	4



$N/2 = 0$

$N/4 = 1$

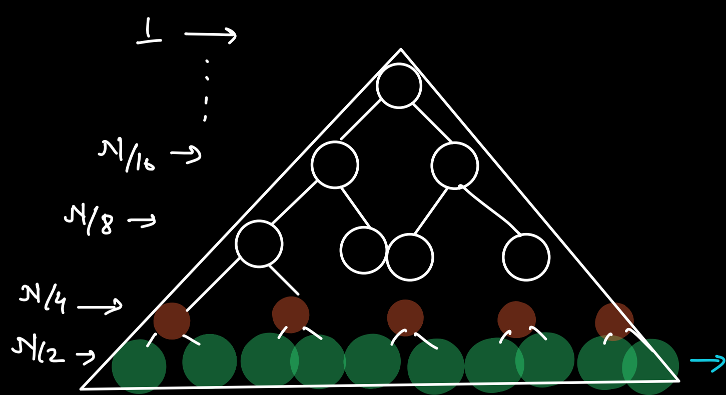
$N/8 = 2$

$N/16 = 3$

$N/32 = 4$

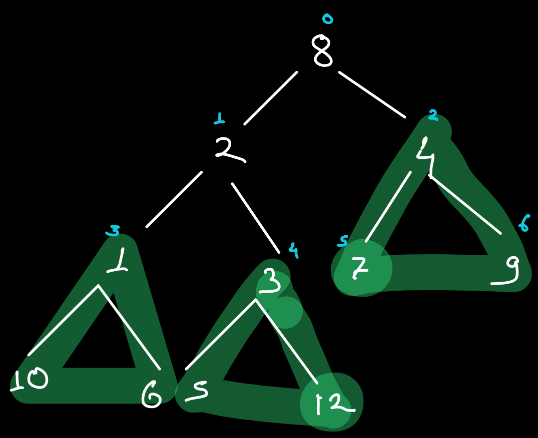
$\vdots$

$1 = \log N$



TC : $O(N)$

Maths



Total no. of iterations (swaps)

$$= \frac{N}{2} \times 0 + \frac{N}{4} \times 1 + \frac{N}{8} \times 2 + \frac{N}{16} \times 3 + \cdots + 1 \times \log N$$

$$= \frac{N}{2^1} \times 0 + \frac{N}{2^2} \times 1 + \frac{N}{2^3} \times 2 + \frac{N}{2^4} \times 3 + \cdots$$

$$= \sum_{d=0}^{\log N} \frac{N}{2^{d+1}} \times d$$

$$= \frac{N}{2} \left[\underbrace{\sum_{d=0}^{\log N} \frac{d}{2^d}}_{2} \right]$$

$\log N \longrightarrow \infty$

$$\sum_{d=0}^{\infty} \frac{d}{2^{d+1}} = \left[\frac{0}{2^0} + \frac{1}{2^1} + \frac{2}{2^2} + \underbrace{\cdots}_{\xrightarrow{\infty \text{ ter}}} \right] \longrightarrow 2$$

$$= \frac{N}{2} \times 2 \longrightarrow N$$

TC: O(N)

Java $\longrightarrow$ PriorityQueue

C++ $\longrightarrow$ priority_queue

Python $\longrightarrow$ heapq

JS $\longrightarrow$?