Today's Content

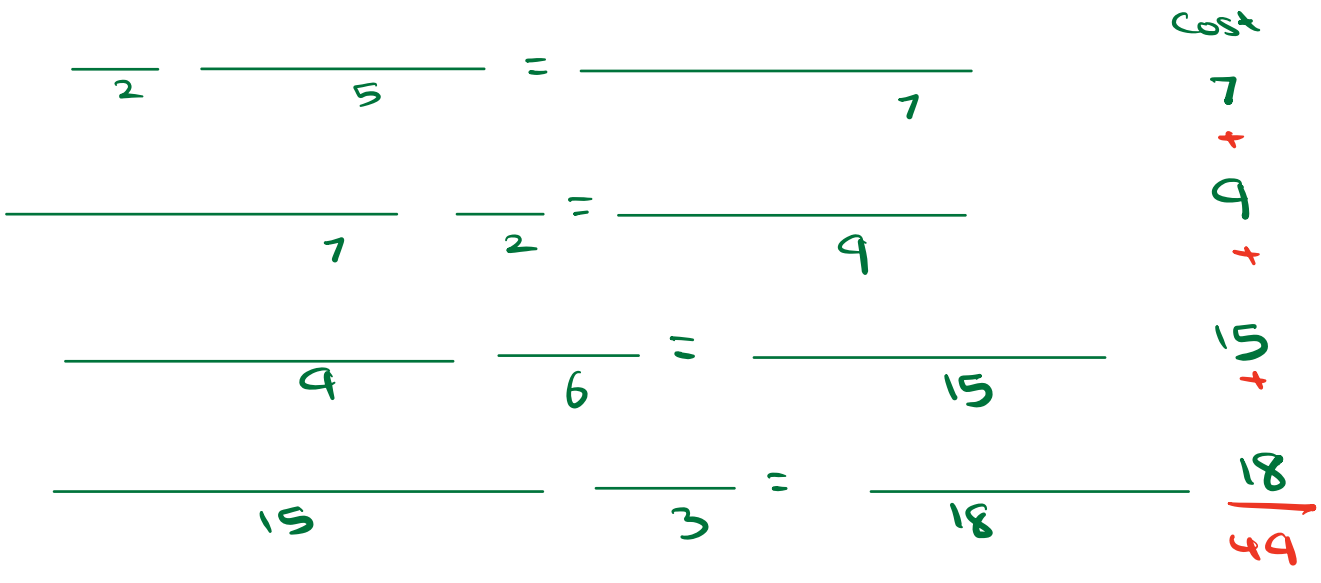
- Connecting the Ropes
- Heap Introduction
- Insertion
- Heapify
- Extract Min
- Build Heap

Q. Connecting the Ropes

You can connect any two ropes together, there's a cost associated to connect them = sum of length of ropes that you're connecting.

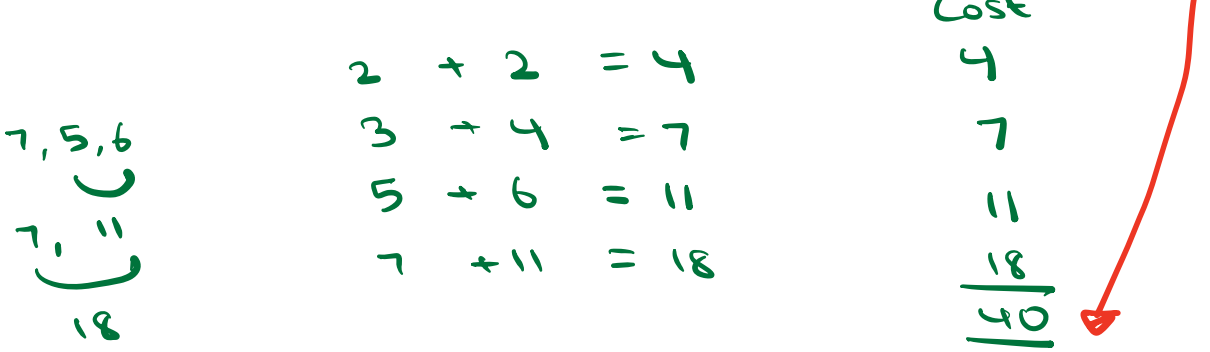
Find min. cost required to connect all ropes.

[2, 5, 2, 6, 3]

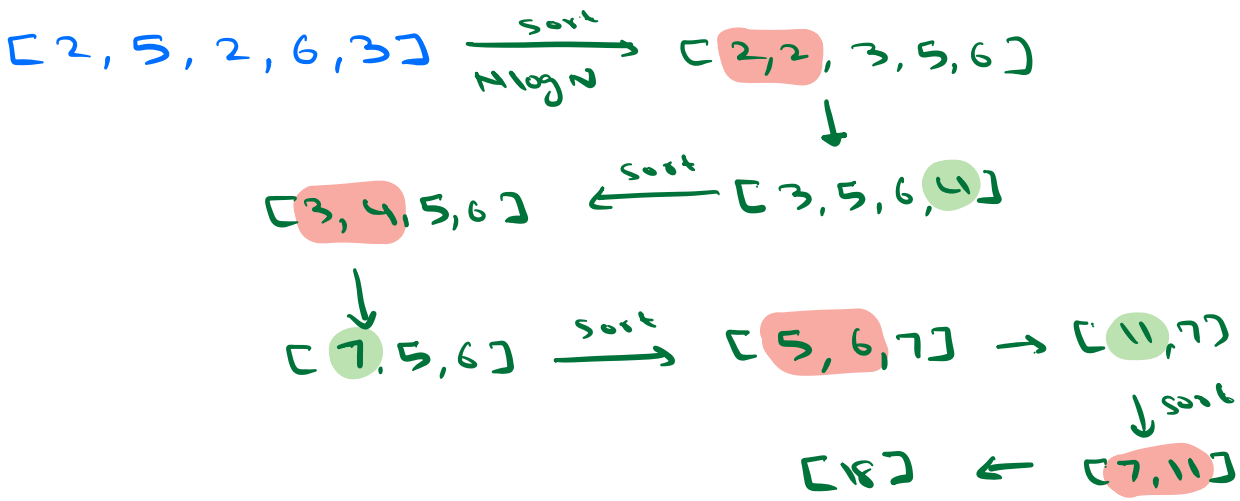


Sort [2, 5, 2, 6, 3] → [2, 2, 3, 5, 6]

Reduce



Idea: Always pick 2 smallest ropes and combine them.



TC: $N \times N \log N$

$O(N^2 \log N)$

We need a DS which is optimized for finding min element

Heap DS

↓

Binary Tree

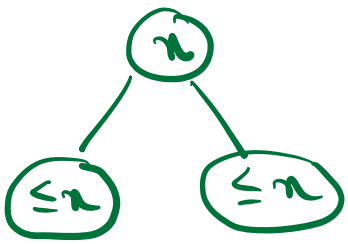
① Complete BT

All levels are filled completely except last level, data can be filled from left to right

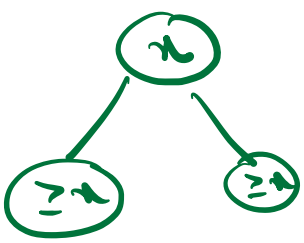
② Order of elements

↓

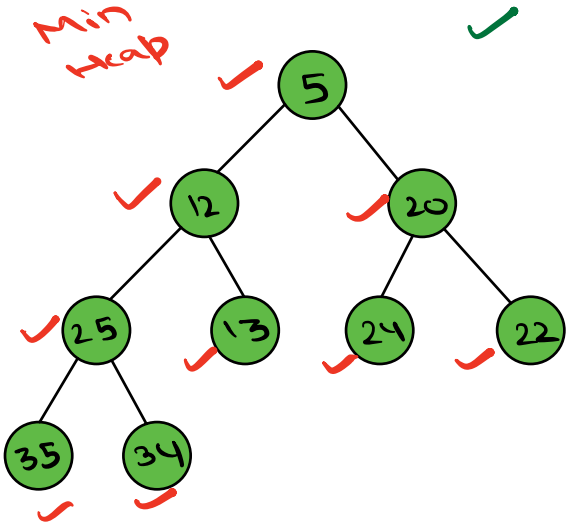
Heap Order Property [HOP]



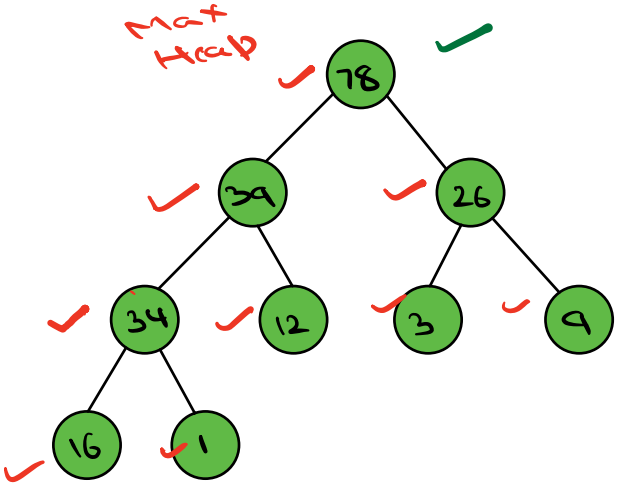
Max Heap Node ≥ children



Min Heap Node ≤ children



1. Complete BT
2. HOP



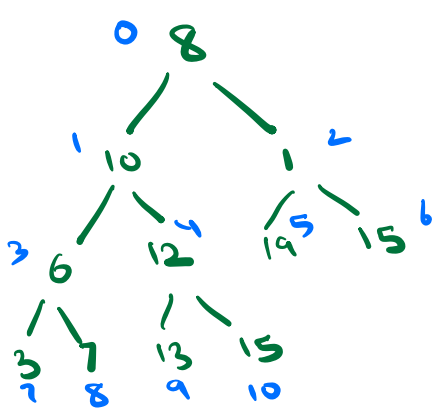
1. Complete BT
2. HOP

Min Heap → Min ele is at root O(1)

Max Heap → Max ele is at root O(1)

Visualize array as Tree (BT)

8	10	1	6	12	19	15	3	7	13	15
0	1	2	3	4	5	6	7	8	9	10



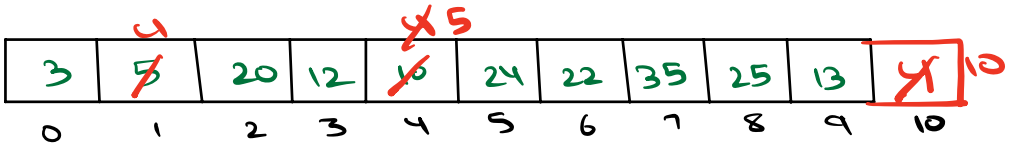
Parent	Children
0	1,2 → [2*0+1, 2*0+2]
3	7,8 → [2*3+1, 2*3+2]
4	9,10 → [2*4+1, 2*4+2]

i → LC: 2i+1, RC: 2i+2

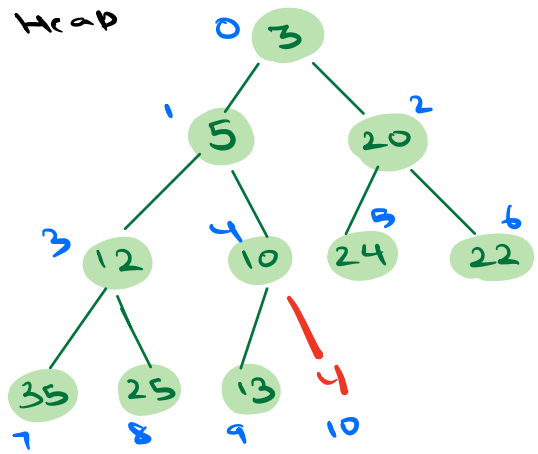
$c_i \xrightarrow{\text{Par}} \frac{c_i - 1}{2}$

Insertion in minHeap

Insert 4



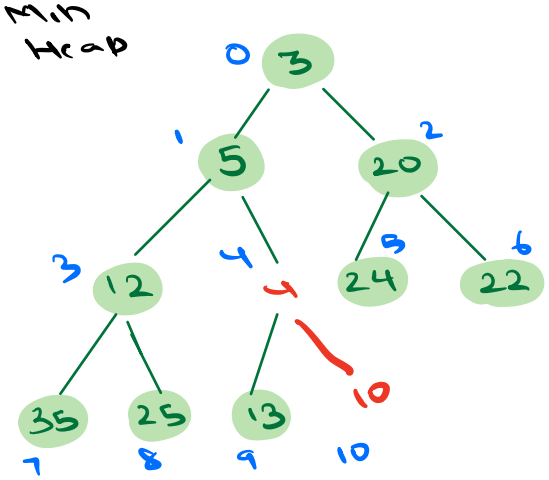
Min Heap



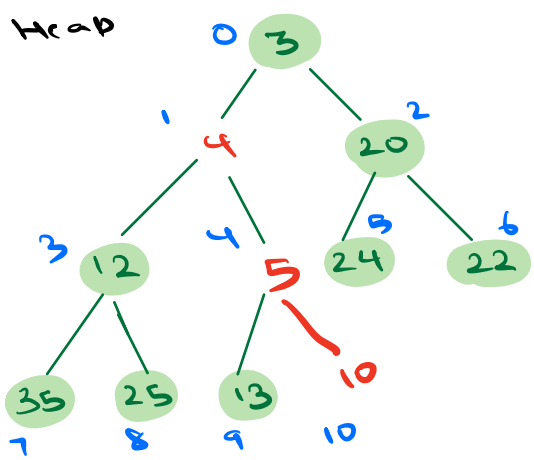
$\frac{(i-1)}{2}$

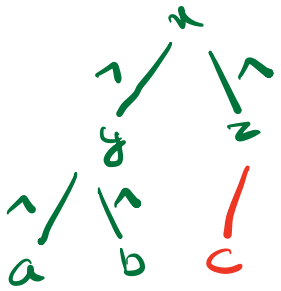
i	P_i	arr[i] > arr[P_i]
10	4	4 > 10 swap No
4	1	4 > 5 swap No
1	0	4 > 3 Yes break

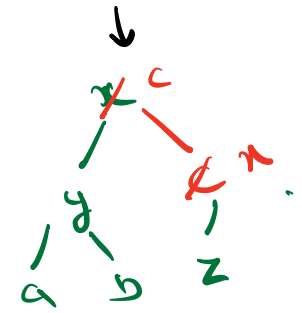
Min Heap



Heap







x < y < a
x < y < b
x < z

On inserting new element c, it should ideally be bigger than par (z)

But if c < z, swapped
Similary if c < x, swapped

if x < y < a and c < x
x < y < b

Then c is definitely less than y and its children

Dynamic array

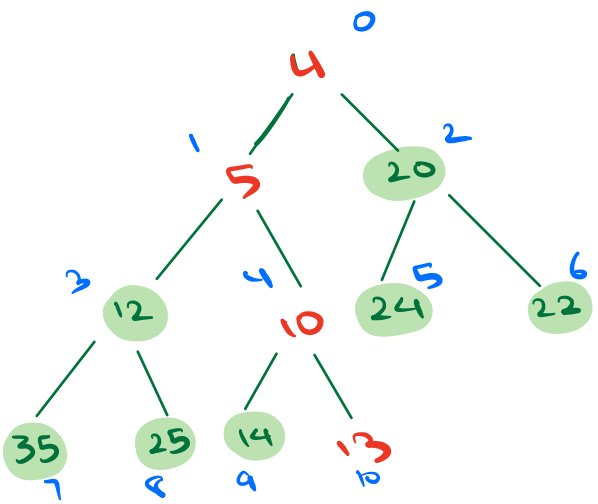
```
void insert(int[] heap, int x) {
    heap.addlast(x)  → dynamic_arr.addlastlast
    i = heap.size() - 1
    while (i > 0) {
        P_i = (i-1)/2
        if (heap[i] < heap[P_i]) {
            swap(heap, i, P_i)
            i = P_i
        }
        else {
            break
        }
    }
}
```

TC: O(Height of Tree)
↓
TC: $O(\log_2 N)$
SC: O(1)

Heapify

Given a min heap, all elements are following heap property except for first ele, fix the heap

0	1	2	3	4	5	6	7	8	9	10
~~13~~ 5	~~4~~ 12 ~~5~~	20	12	~~5~~ 10	24	22	35	25	14	~~10~~ 13



i	idx Child	min (a[i], a[2i+1], a[2i+2])
0	1, 2	min(13, 4, 20) = 4 idx → 1 swap(0, 1)
1	3, 4	min(13, 12, 5) = 5 idx → 4 swap(1, 4)
4	9, 10	min(13, 14, 10) = 10 idx = 10 swap(4, 10)
10		break

N = 11 (heap size)

↓

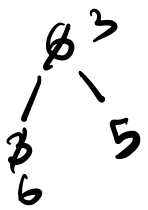
valid idx 0 - 10

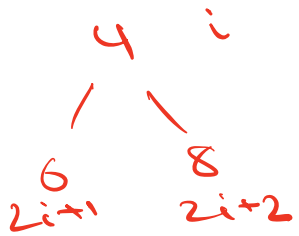
i	2i+1	2i+2
5	11 X	12 X
6	13 X	14 X
7	15 X	16 X
⋮		

↓

children idx ≥ 11 → invalid

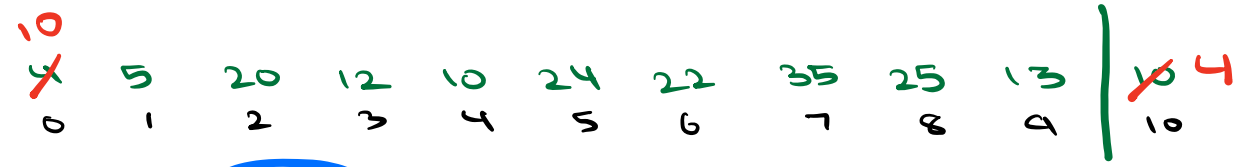
```
void heapify (heap[], N, i){
    while (2i+1 < N) {                          NOTE *
        x = min(heap[i], heap[2i+1], heap[2i+2])
        if (x == heap[i])
            break
        else if (x == heap[2i+1]) {
            swap(heap, i, 2i+1)
            i = 2i+1
        }
        else if (x == heap[2i+2]) {
            swap(heap, i, 2i+2)
            i = 2i+2
        }
    }
}
```

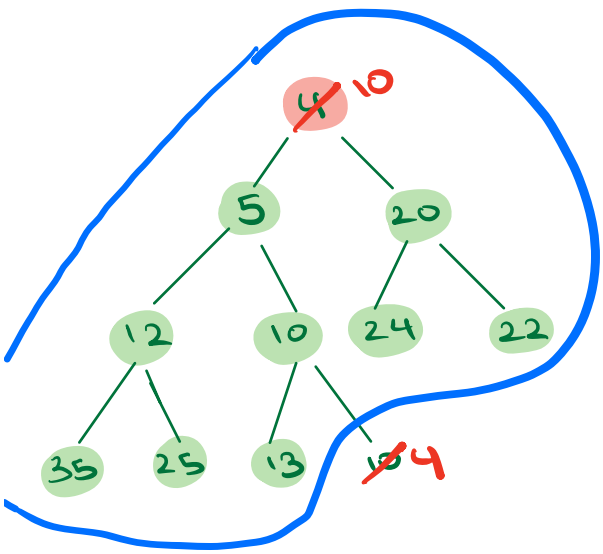




* Note: Some node might only have left child. So when trying to find min, check if right child exists i.e. $2i+2 < N$.

Extract / Remove min





Min → heap[0]

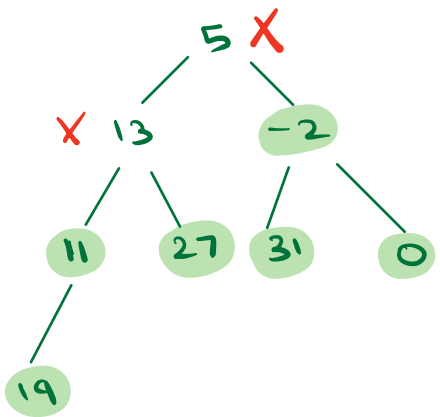
1. swap heap[0] with heap[last]
2. remove last elt
3. heapify with idx 0

TC : O(log N)

SC : O(1)

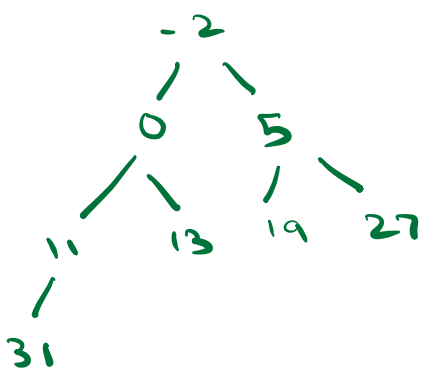
Build Heap [Min Heap]

[5 13 -2 11 27 31 0 19]



Idea 1 : Sort the arr

[-2 0 5 11 13 19 27 31]

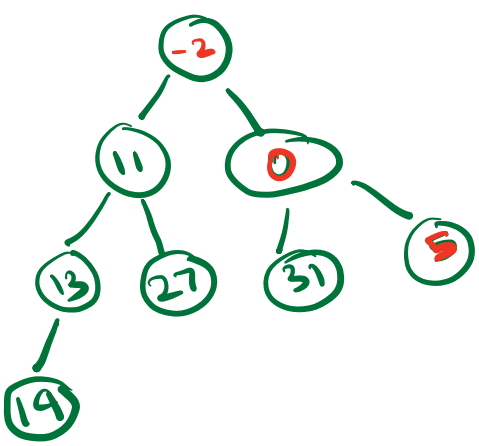


TC : O(N log N)

↓

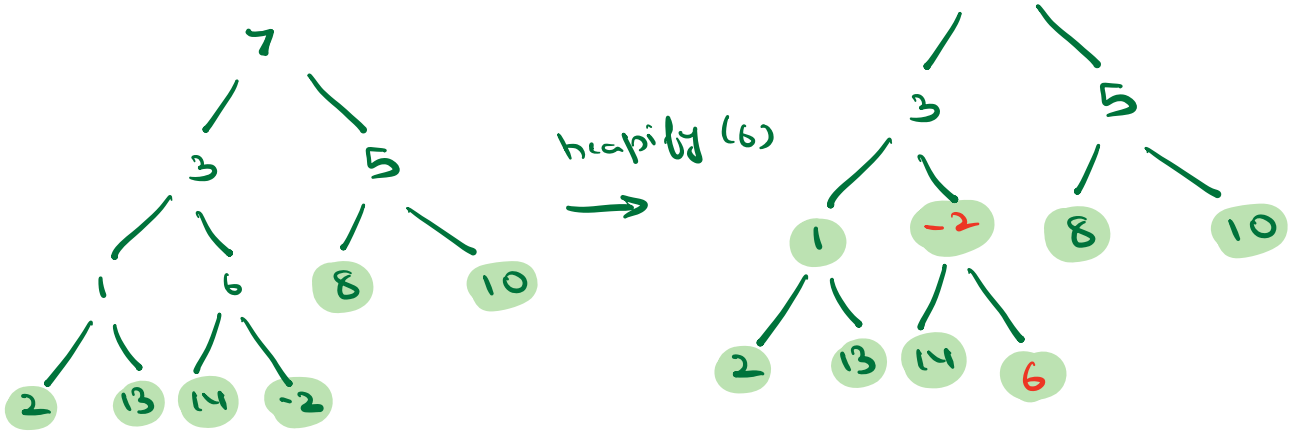
[5 13 -2 11 27 31 0 19]

Idea 2: Insert one-one element into heap

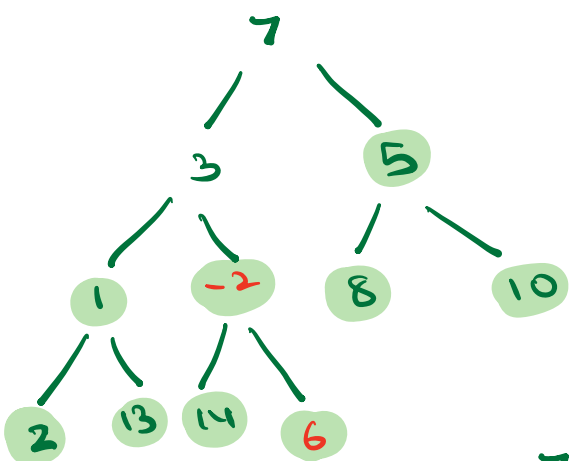


TC: O(N log₂ N)

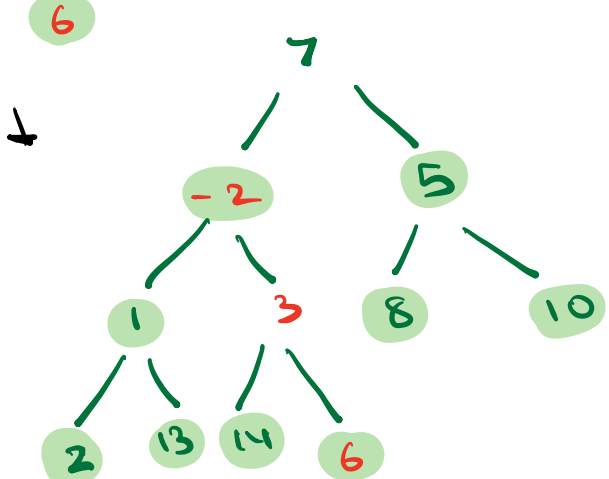
Idea 3 : [7, 3, 5, 1, 6, 8, 10, 2, 13, 14, -2]



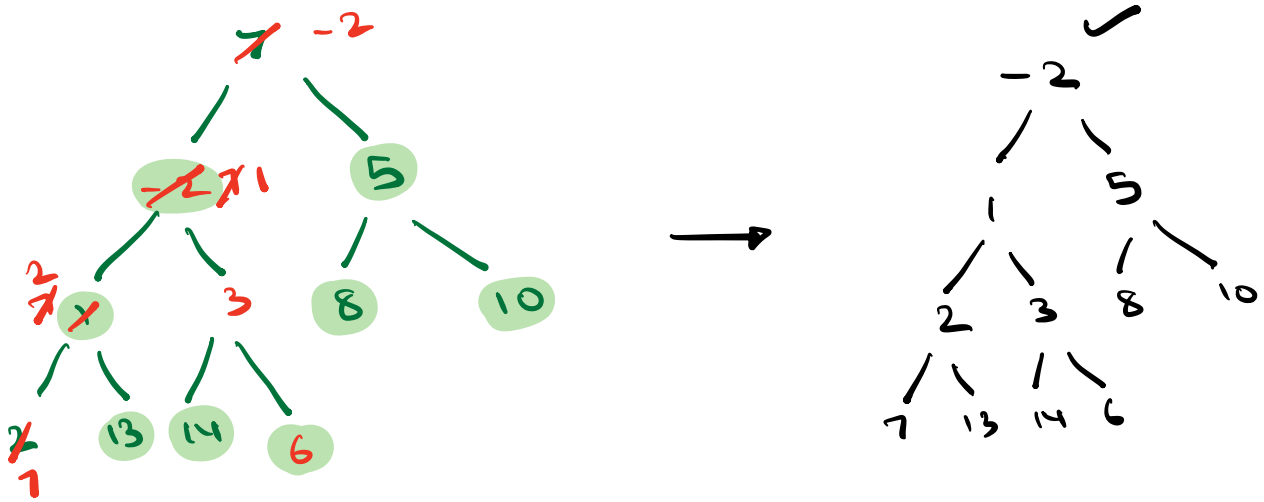
heapify (1)



heapify (5)
heapify (3)



heapify (7)



First non-leaf node

↓

Parent of last leaf

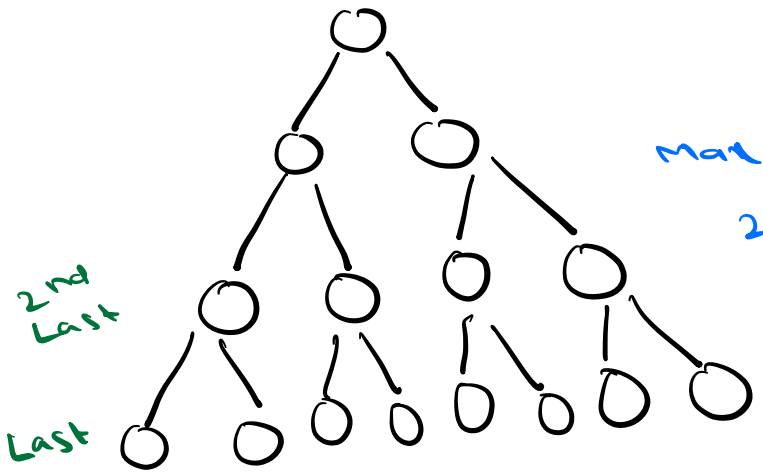
last leaf (idx) → N-1

Parent → $\frac{i-1}{2}$ → $\frac{N-1-1}{2}$ → $\frac{N-2}{2} = \frac{N}{2} - 1$

```
for (i = N/2 - 1 ; i >= 0 ; i--) {
    heapify (heap, i)
}
```

$\times$ $N \log_2 N$

N nodes



Max no. of nodes in last level = $N/2$

Max No. of nodes in 2nd last level = $N/4$

3rd last = $N/8$

3 → 2
7 → 4
15 → 8

Total swaps

$$= \underbrace{\frac{N}{2} * 0}_{0} + \frac{N}{4} * 1 + \frac{N}{8} * 2 + \frac{N}{16} * 3 + \dots$$

$$= \frac{N}{2}\left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \dots\right)$$

AGP

$$S = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \dots$$

$$-\frac{S}{2} = \frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \dots$$

$$S/2 = \frac{1/2 + 1/4 + 1/8 + 1/16 + \dots}{r = 1/2}$$

$$\text{Sum} = \frac{a}{1-r} \qquad (r < 1)$$

$$\frac{S}{2} = \frac{1/2}{1-1/2}$$

$$\frac{S}{2} = 1 \Rightarrow S = 2$$

Total swaps

$$= \frac{N}{2}\left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \cdots\right)$$

$$= N/2\ (2) = N$$

TC: O(N)

Doubt