Today's Content

Connecting the Ropes
Heap Introduction
Insertion
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

Ans = 40

		Cost
2 + 5 = 7	(7, 2, 6, 3)	7
		+
2 + 6 = 8	(7, 8, 3)	8
		+
8 + 3 = 11	(7, 11)	11
		+
7 + 11 = 18	18	18

Total cost = 44

2, 5, 2, 6, 3

Cost

① 2 + 2 = 4 — 4

+

4, 5, 6, 3

② 3 + 4 = 7 — 7

7, 5, 6

+

③ 5 + 6 = 11 — 11

+

7, 11

④ 7 + 11 = 18 — 18

18

Total cost = 40

Idea: Always pick 2 smallest ropes and merge them

$x < y < z$

$\boxed{\bar{x}\ \bar{y}}\ \bar{z}$ $\boxed{\bar{y}\ \bar{z}}\ \bar{x}$ $\boxed{\bar{x}\ \bar{z}}\ \bar{y}$

Case	1	2	3
step 1 :	$x+y$	$y+z$	$x+z$
step 2 :	$x+y+z$	$x+y+z$	$x+z+y$

Case 1 $\le$ case 3 $<$ case 2

$$[2,5,2,6,3] \xrightarrow[N\log N]{\text{sort}} \text{①}\ [2,2,3,5,6] \xrightarrow{4} [3,5,6,4] \xrightarrow{\text{sort}} \text{②}\ [3,4,5,6]$$

$$[3,4,5,6] \xrightarrow{7} [7,5,6] \xrightarrow{\text{sort}} \text{③}\ [5,6,7] \xrightarrow{11} [11,7] \xrightarrow{\text{sort}} \text{④}\ [7,11] \xrightarrow{18} [18]$$

TC : $O((N-1)N\log N) = O(N^2 \log N)$

Use insertion sort : $N\lg N + (N-1)N$

$\Rightarrow$ TC : $O(N^2)$

1 connection $\rightarrow 3N$

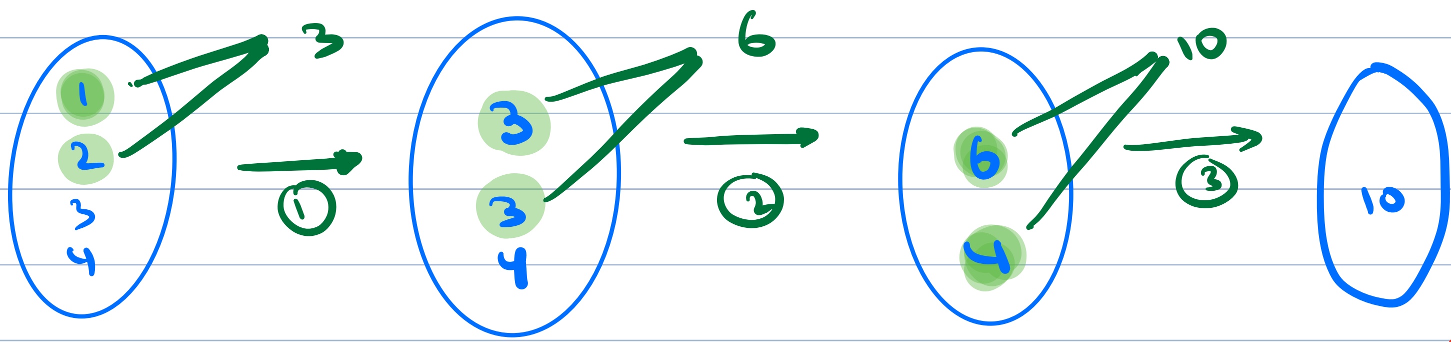
$N-1$ connections $\rightarrow (N-1)N$

Quiz [1, 2, 3, 4]

		Cost
1+2 = 3	[3, 3, 4]	3
		+
3+3 = 6	[6, 4]	6
		+
6+4 = 10	[10]	10
		19

We need a DS which is optimized in foll. operations

Heap
- Insert (log N)
- Extract Min (log N)

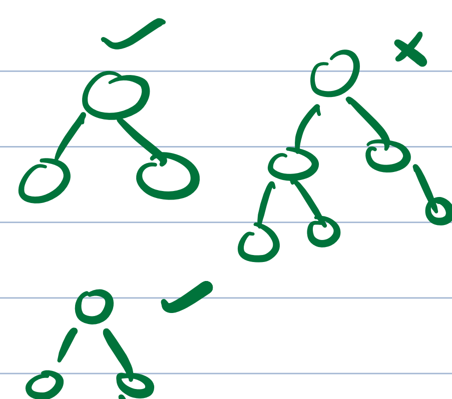


1 connection → 3 log N

TC: O((N-1) log N)

① Complete BT (CBT)

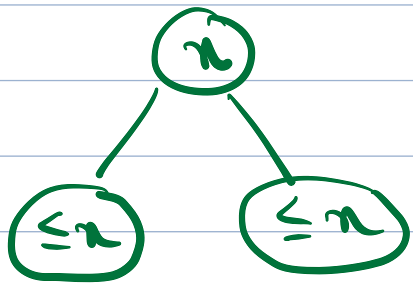
All levels are filled completely except last level, data can be filled from left to right



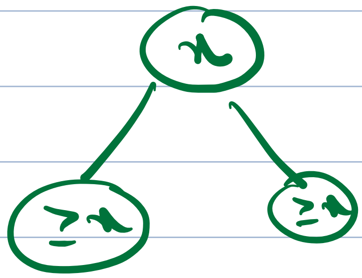
② Order of elements

↓

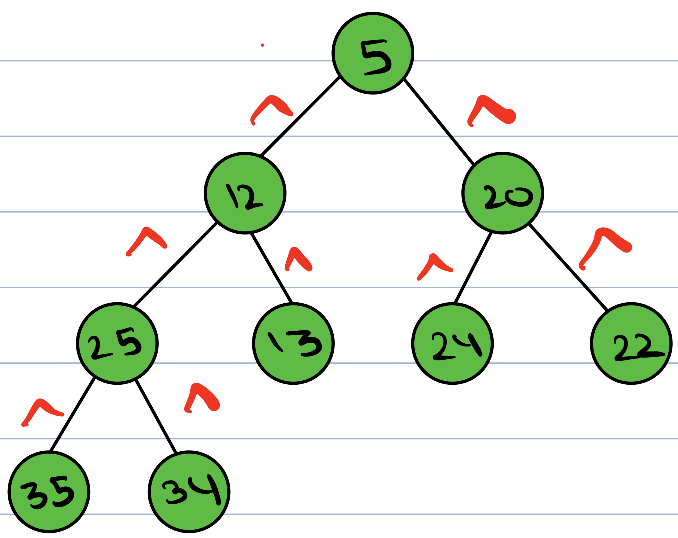
Heap Order Property [HOP]



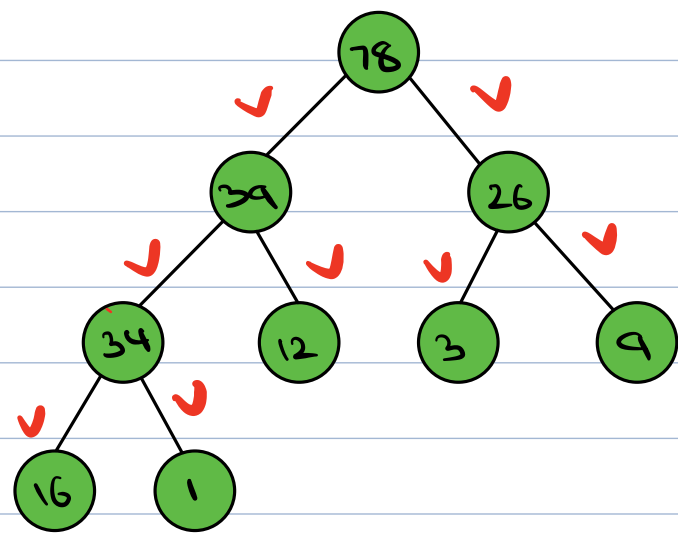
Max Heap   Node ≥ Children



Min Heap   Node ≤ Children



1. Complete BT
2. Min Heap



1. Complete BT
2. Max Heap

Min Heap → min ele is at root O(1)

Max Heap → Max ele is at root O(1)

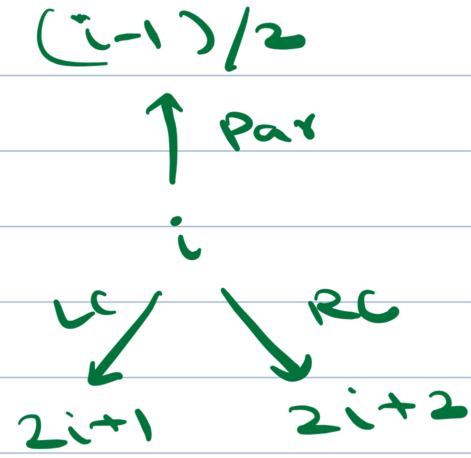
Heap → CBT (can be implemented using array)

Level order Traversal

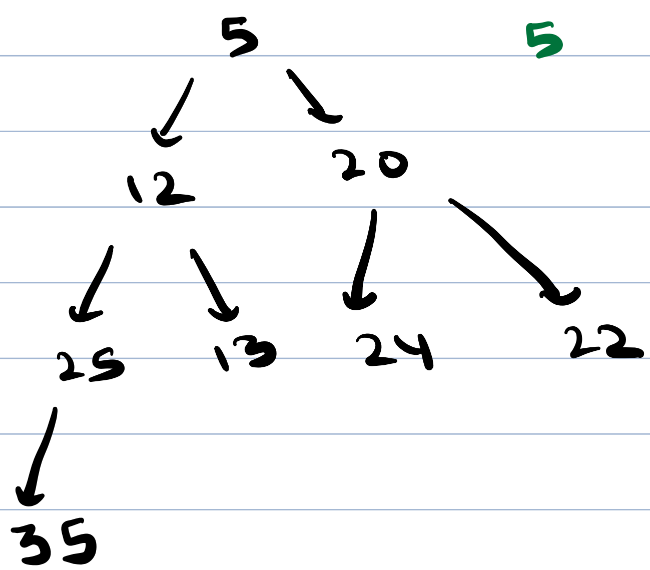
8	10	1	6	12	19	15	3	7
0	1	2	3	4	5	6	7	8

Par idx	LC idx	RC idx
0	1	2
1	3	4
3	7	8
i	$2i+1$	$2i+2$

Node idx	Par idx
3	1
6	2
5	2
i	$(i-1)/2$



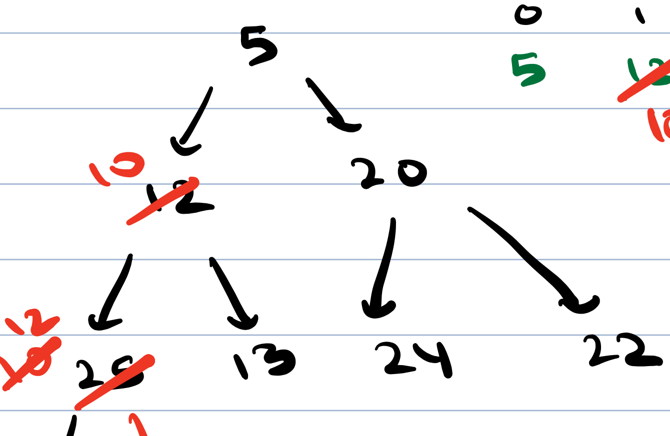
Insertion in Min Heap

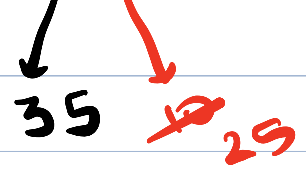


0	1	2	3	4	5	6	7
5	12	20	25	13	24	22	35

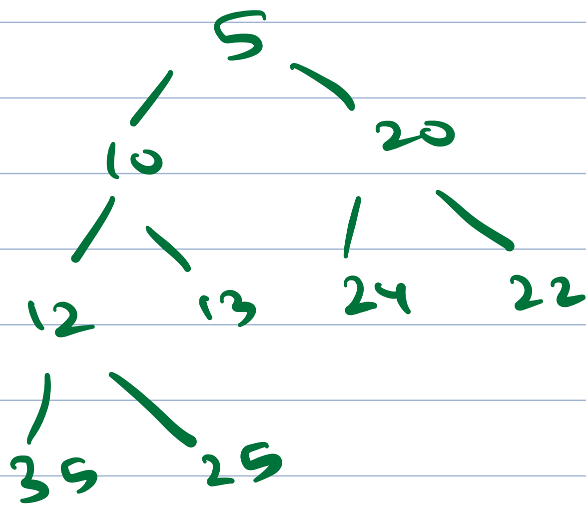
① Insert 10

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





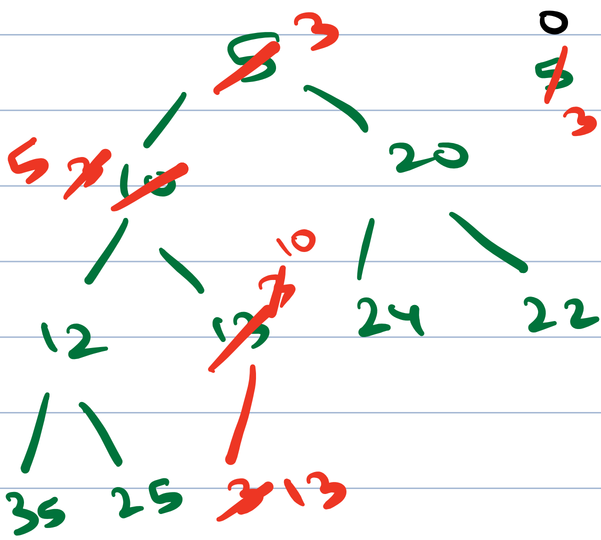
$(i-1)/2$



i	par idx	a[par] < a[i]
8	3	25 < 10 ? No swap
3	1	12 < 10 ? No swap
1	0	5 < 10 YES STOP

---

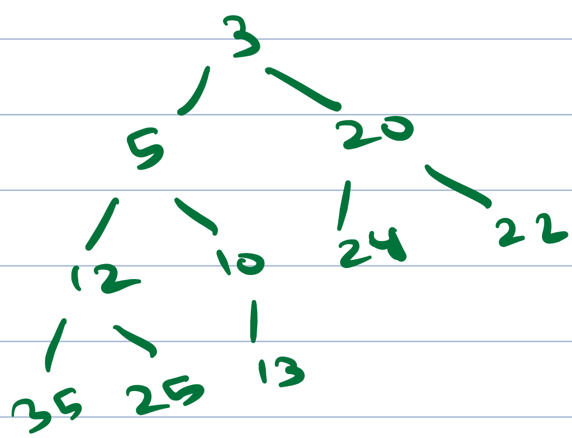
② Insert 3



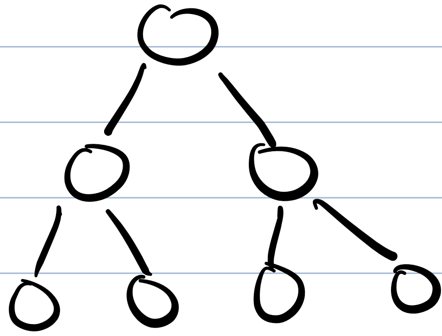
0	1	2	3	4	5	6	7	8	9
~~5~~ 3	~~10~~ ~~3~~ 5	20	12	~~13~~ ~~3~~ 10	24	22	35	25	~~3~~ 13

$(i-1)/2$

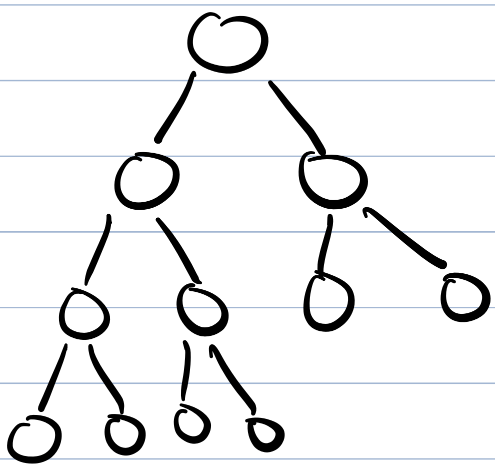
i	par idx	a[par] < a[i]
9	4	13 < 3 ? No swap
4	1	10 < 3 ? No swap
1	0	5 < 3 ? No swap
0		i == 0 STOP



Height of tree



N	$\log_2 N$
7	$\log_2 7 = 2.$ —
11	$\log_2 11 = 3.$ —



TC: $O(H) \xrightarrow{CBT} O(\log_2 N)$

```
void insert(list<int> heap, int x){
    heap.add(x)            // val -> last
    i = heap.size() - 1
    while( i > 0 ){
        pi = (i-1)/2
        if(heap[pi] > heap[i]){
```

swap (heap[pi], heap[i])
i = pi
}

else {
break
}
}
}

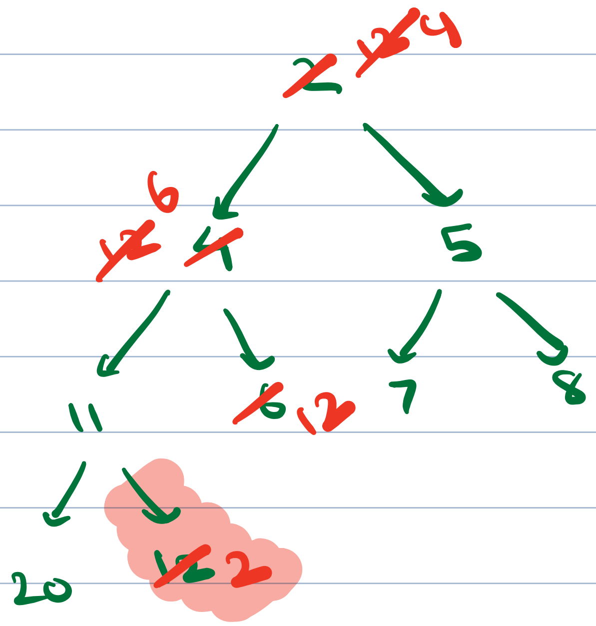
10:35

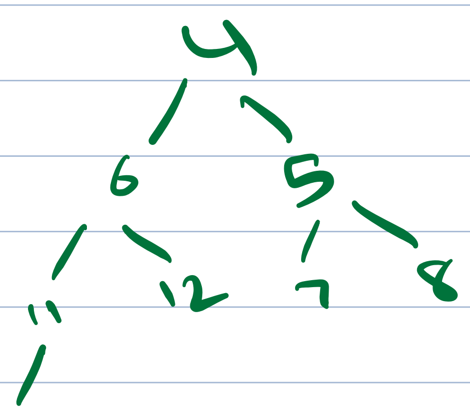
https://notability.com/n/Y0MX0ggK8pdO9DWAfBA~h

Extract Min

getMin () → heap[0]

0	1	2	3	4	5	6	7	8
~~2~~ ~~12~~ 4	~~4~~ ~~12~~ 6	5	11	~~6~~ 12	7	8	20	~~12~~ 2





i	LC $2i+1$	RC $2i+2$	min (heap[i], LC, RC)
0	1	2	min (12, 4, 5) = 4 swap (12, 4) Swap(0 idx, 1 idx)
1	3	4	min (12, 11, 6) = 6 swap (12, 6)

20

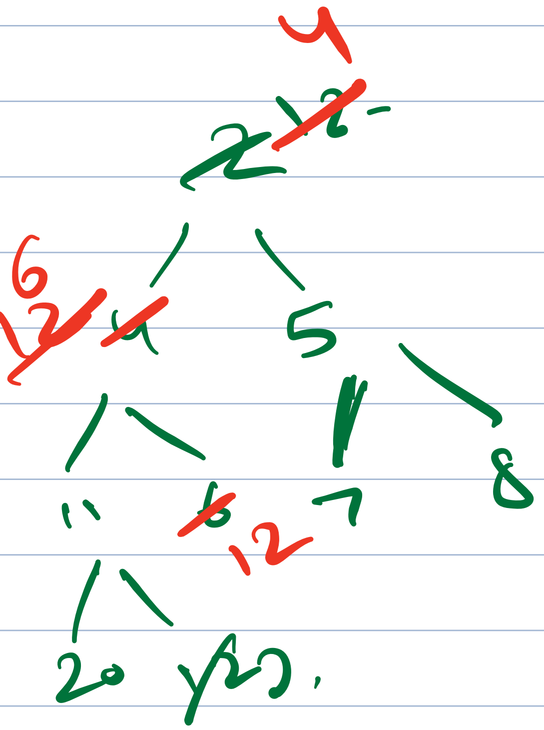
		swap(1 idx, 4 idx)
4	9	10 LC idx is invalid ↓ leaf node ↓ $\boxed{\text{STOP}}$

```
// list <int> heap
swap (heap [0], heap [heap.size() - 1])
heap.remove()      // remove last
heapify (0, heap)

void heapify (int i, list <int> heap) {
                                   non leaf node
  while (2i+1 < heap.size()) {
      l = 2i+1,   r = 2i+2
      x = min (heap[i], heap[l])
      if (r < heap.size())
          x = min (x, heap[r])

      if (heap[i] == x) {
          break
      }
      else if (heap[l] == x) {
          swap (heap[i], heap[l])
          i = l
      }
```

```
else {        // heap[r] = x
    swap(heap[i], heap[r])
    i = r
}
}
}
}
```



$i \quad N = 8$

0	1	2	3	4	5	6	7	8
~~2~~ ~~12~~ 4	~~4~~ ~~12~~ 6	5	11	~~6~~ 12	7	8	20	~~12~~ 2

$TC : O(H) = O(\log_2 N)$

10 : 59

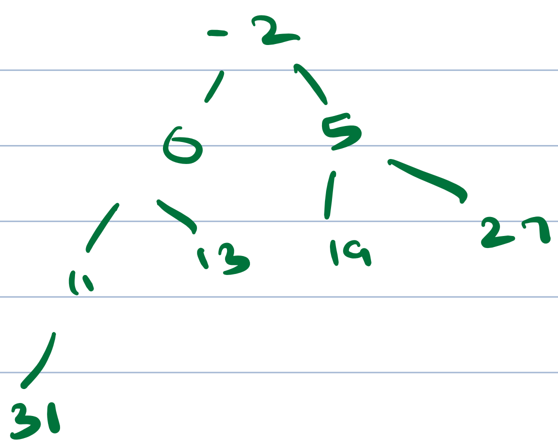
Build Heap

Given an array, make a min heap out of it.

Ex : [5 13 -2 11 27 31 0 19]

Approach 1 : Sort the array

[-2 0 5 11 13 19 27 31]



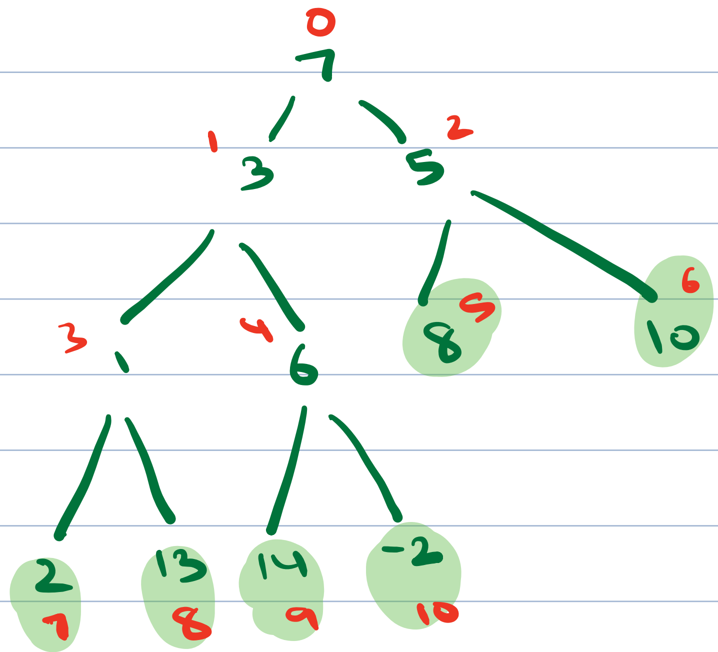
TC : O(N log N)

Approach 2: Start with an empty heap and call insert N times.

TC : O(N log N)

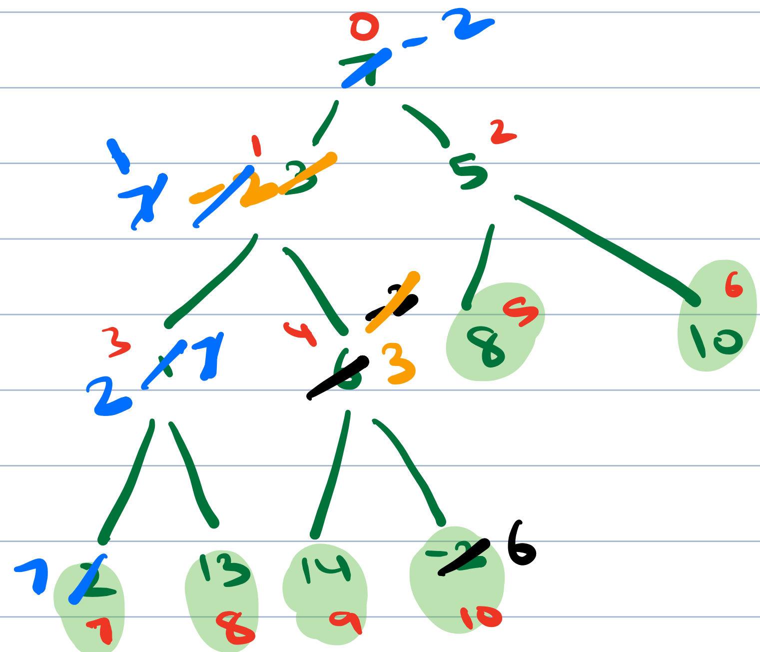
Approach 3 : Build heap → linear time

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



① Non Leaf Node

② Heapify from non-leaf node till 0th node



heapify (4)

↓

heapify (3)
no action

↓

heapify (2)
no action

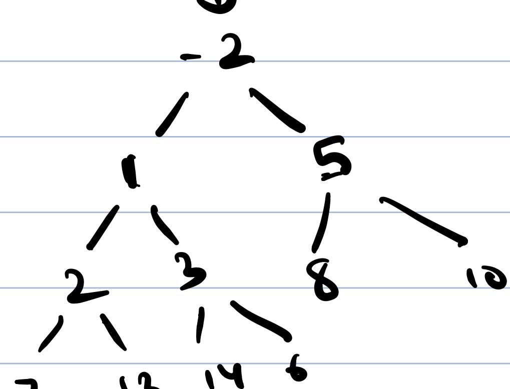
↓

heapify (1)

↓

heapify (0)

↓



$$\text{for } (i = (N-2)/2;\ i \geq 0\ ;\ i--)$$

$$\text{heapify}(i, arr)$$

Idx of last non-leaf node

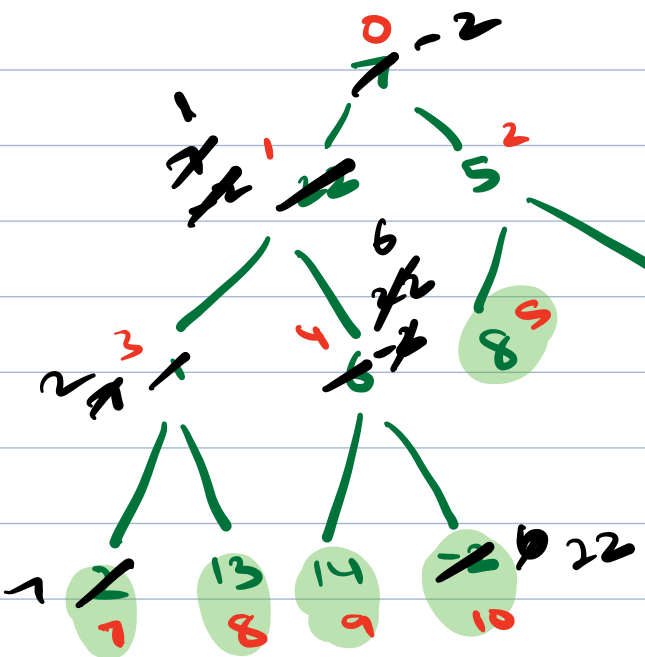
↓

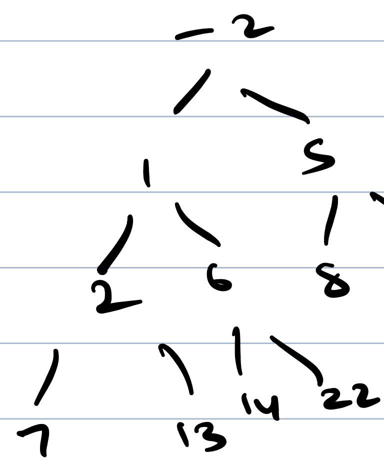
parent of last leaf

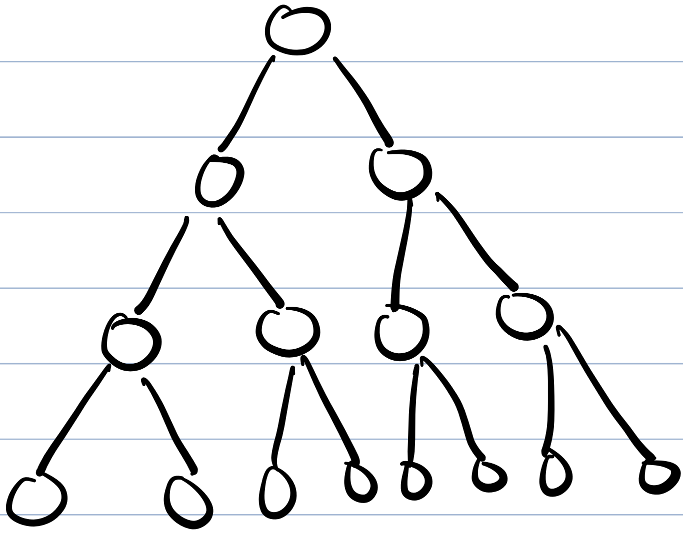
Last leaf → par $(i-1)/2$

$N-1$

$$\frac{(N-1-1)}{2} = \frac{(N-2)}{2}$$







Nodes	Swaps
1	
⋮	
N/8	2
N/4	1
N/2	0

TC: $\overset{0}{\cancel{N/2 \times 0}} + N/4 \times 1 + N/8 \times 2 + N/16 \times 3$ ......

$$= N/2 \left(\underbrace{\frac{1}{2} + \frac{2}{4} + \frac{3}{8}}_{\text{AGP}} + \ldots\ldots \right)$$

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

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

$\frac{1}{2} S = \underline{1/2 + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots\ldots}$

GP ∞ series sum $= \frac{a}{1-r}$

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

$\Rightarrow S = 2$

TC : $\frac{N}{2}(AWP) = \frac{N}{2}(2) = O(N)$