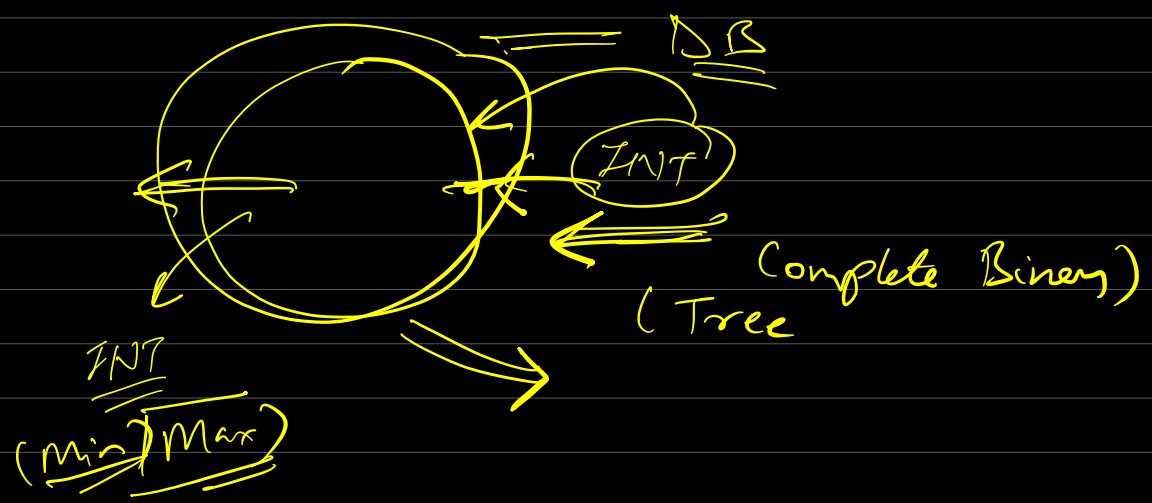
N integer ⇒ min / max

(Heaps)



Q Given an array. Perform B negations (Input)

Find the max sum after B negations

$A = [2, -5, 10, -3, -6] =$

$B = 3$

$A[i] = A[i] * (-1)$

$A = [\cancel{-5}, \cancel{-4}, \cancel{-3}, \cancel{-1}, 2, 3, 8, 7, 11]$

$\Rightarrow [5, 4, 3, 1]$

$\Rightarrow$ $x < B$ min $\quad B = \cancel{4}\ 5$

$(B - x)$ $\quad 4$

$\cancel{1}\ \cancel{1}\ \cancel{1}\ \cancel{1}\ 1$

(1) 1) Sort the array

2) Negate all negative numbers (# x)

if $(x \geq B)$ (problem is already solved)

if $(x < B)$

min of negation of the largest -ve no. or smallest +ve no. in the original array

$\Downarrow$

if $(B - x)$ is even, smallest element is as it is $O(1)$

else negate the smallest element

$$T.C. = O(N \log N + N)$$

$$= O(N \log N)$$

(2) $\Rightarrow$ At every step, we are negating the smallest element.

Min Heap

1) Delete the min
2) Negate the min
3) Insert back the negated element.

T.C. of 1 operation = $\log N \times$ (B operation)

$B \log N$

Sort : Heap faster

1) $B < N$ $\quad N \log N > B \log N$

2) $B = N$ — $\quad N \log N = N \log N$

3) $B > N$ — $\quad N \log N = N \log N$

3

4

2, 5, 10, 3, 6

7

IS -ve

+ve / -ve

2 -2 2

S.C. =

Q-2 Given an array

3, 0, 2

	0	1	2	3	4	5	6
A ⇒	3,	0,	-3,	2,	-10,	1,	6
	ACD	ND	0,	0	0,	6,	36

for $\forall i \geq 2$

find the product of largest ~~3~~ K elements in the array till i (100)

Sol^n

(1) 3 variables

min

K

New Element

max₁, max₂, max₃

~~3~~ 6 _ ~~0~~ 2 3, -3 ~~0~~ 1 2

⇒ If the new element < max_3

the 3 max would remain as it is

⇒ If the new element > max_3,

⇒ max_3 will be removed.

Observations

	0	1	2	3	4	5	6
A ⇒	3,	0,	−3,	2,	−10,	1,	6

Heap

3, 0, 2 ⇒ Min

−10

−3

2

Max Heap

⇒ Remove k elements $K \log K$

⇒ Reinsert k max $K \log K$

Min Heap

$\log K$

$(K \log K)$

Steps

1) Insert the first K elements into the min heap $K \log K$ (K)

2) for (i = K+1 to N) { $\Rightarrow$ $N-K$

Check if (A[i] > heap.min) { $O(1)$

remove heap.min $\log K$

insert A[i]; $\log K$

}

$(N-K) \log K$

$$T.C. = O(K \log K + (N-K) \log K)$$

$$= (N \log K)$$

$$S.C. = O(K)$$

Only for explanation

Max Heap

1) Insert first K elements ($K \log K / K$)

2) for (i = K+1 to N) { ($N-K$) ($K \log K$)

⇒ 1) Extract all element ($K \log K$

2) Insert K max ($K \log K$)

}

$N K \log K$ ✗✗

NIKHIL

H.W. How will you calculate the product for every i optimally

---

Q Given N distinct integers.

find the no. of ways to form max Heap.

$N \Rightarrow [1, 2, 3, \ldots N]$

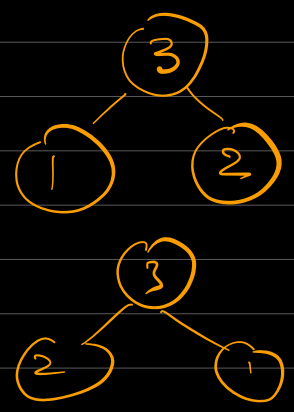
$N = 1$

1

1

$N = 2$ $[1, 2]$

$N = 3$ $[1, 2, 3]$



$N = 4$ $[1, 2, 3, 4]$ $[1, 2, 3, 4]$

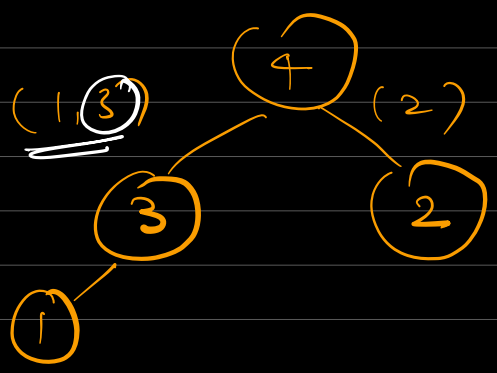
$\Rightarrow 4$

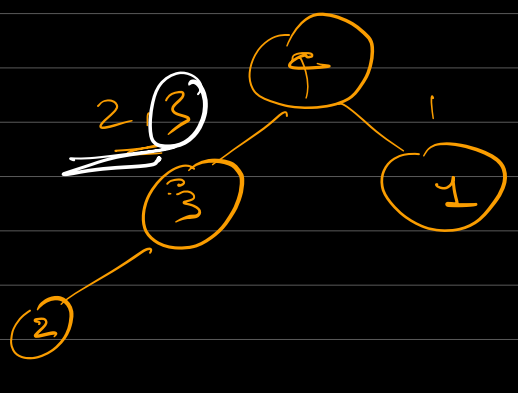
$^3C_2 = 3$

$^3C_1$

(1, 2)

(3)





NO / NO

Observations

1) Root is always fixed (including subtrees)
   ↳ largest

2) Structure for a given tree for a value of N is fixed.

3) No relation b/w the LST & RST

Hence we need to consider all possibilities.

Generalising

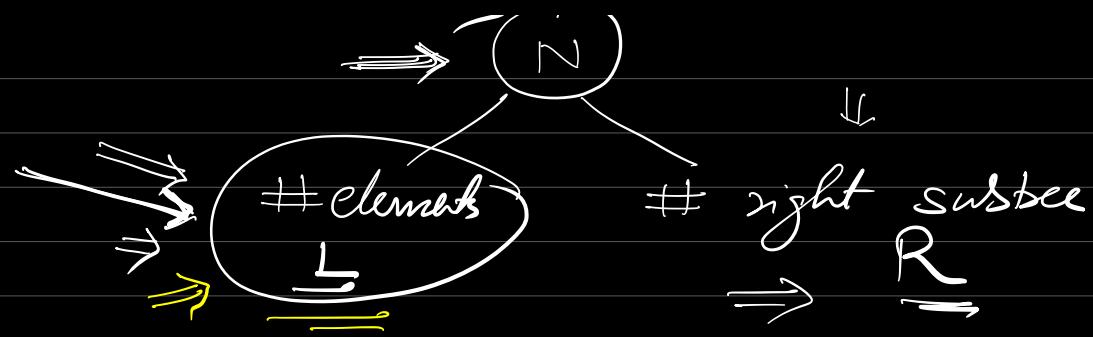
⇒ Lets say we have N element

1, 2, 3, 4, 5. . . . . N

⇒ N will be the root of the max heap.

⇒ Structure of the tree will be fixed

N



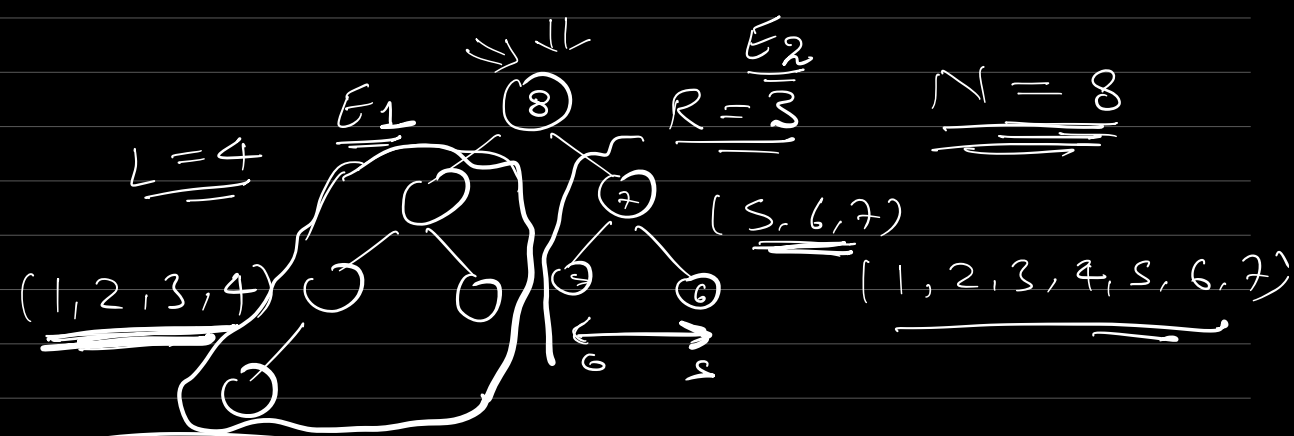
$^{N-1}C_L$ (Total ways to select L elements for LST)

$$^{N-1}C_L = {}^{N-1}C_R$$

$$L + R = N - 1$$

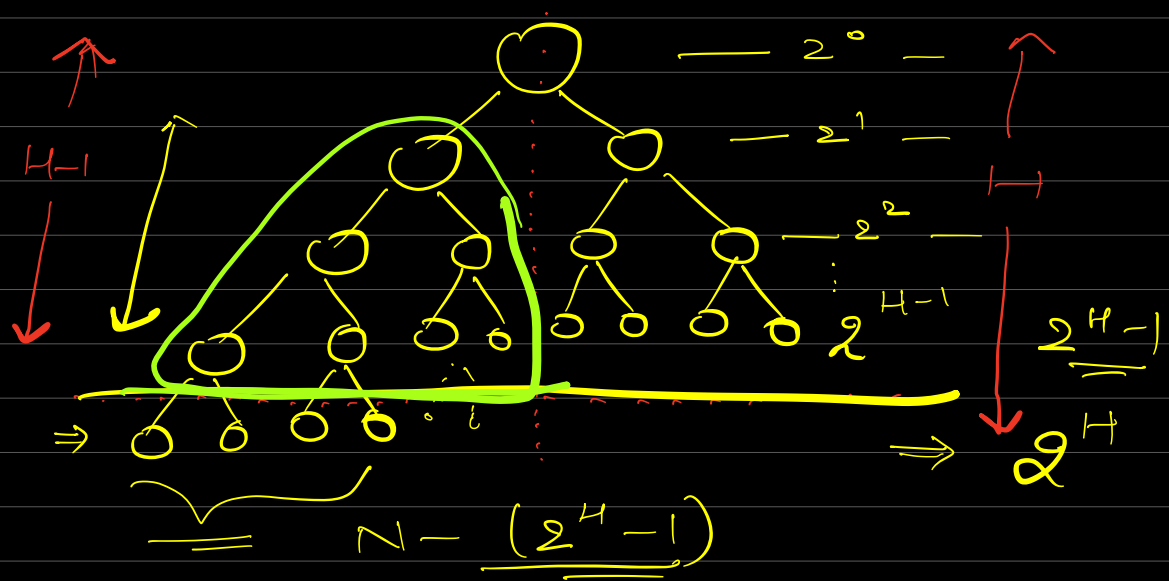
$^{N}C_R = {}^{N}C_{N-R}$

$$\boxed{\text{ways}(N) = {}^{N-1}C_L \times \text{ways}(L) \times \text{ways}(R)}$$

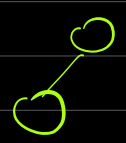
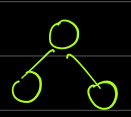
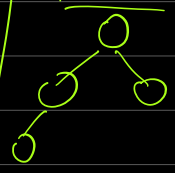
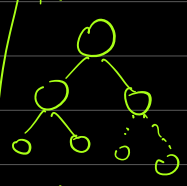


$$\left({}^{7}C_4\right) \times 3 \times 2$$

Q How to calculate L (N)



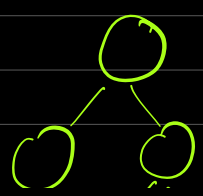
⇒ Height of a complete BT.

N=1	N=2	N=3	N=4	N=5	N=6	N=7
					H=2	H=2
H=0	H=1	H=1	H=2	H=2		

N=8
H=3

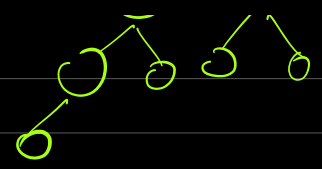
$$\boxed{\text{Height} = \lfloor \log_2 N \rfloor}$$

(H)



$0 = 2^0$

$1 = 2^1$



$2 = 2^2$

$8 = 2^3$

$\underline{\underline{H}}$

elements at the ith level $= 2^i$

elements in the tree except the last level

$\Rightarrow$

$$= 2^0 + 2^1 + 2^2 \ldots\ldots 2^{H-1} \Rightarrow GP$$

$$1\,\frac{(2^H - 1)}{2-1} = 2^H - 1 - \frac{1}{\downarrow}$$

root

2

elements in the last level

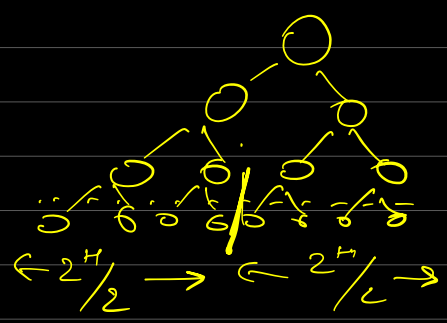
$$= \underbrace{N}_{\text{Total}} - \underbrace{(2^H - 1)}_{\text{all elements except last level}}$$

$\Rightarrow$

* Max no. of elements in the last

level 0

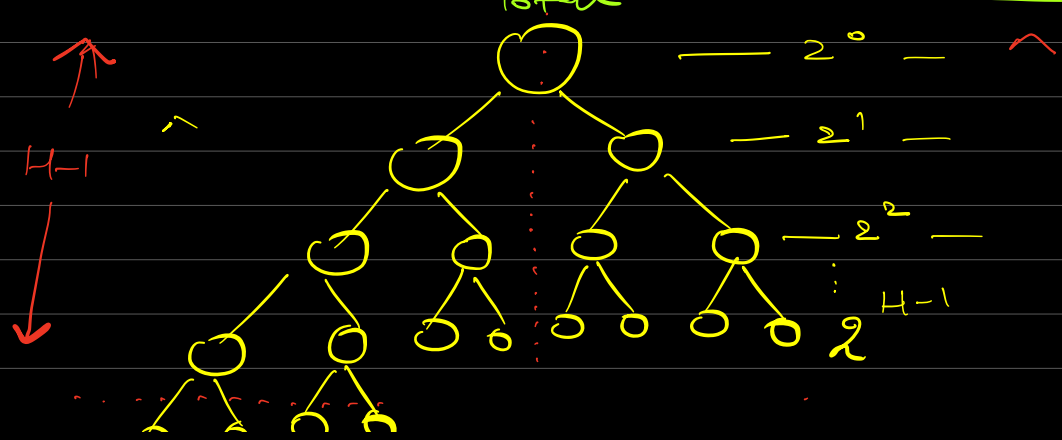
$= 2^H$



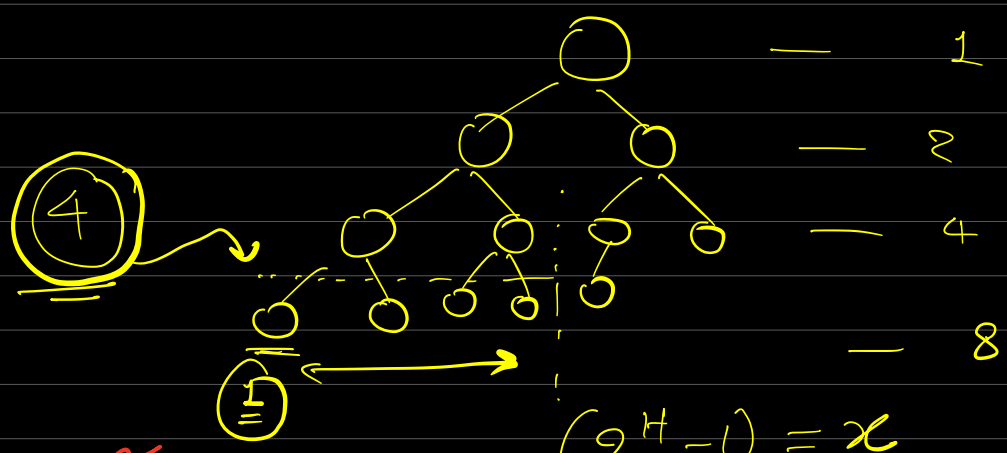
Max no. of element at the last level of LST $= \dfrac{2^H}{2} = 2^{H-1}$

Elements in the last level of L

$$\boxed{= \min\left(\underbrace{N-(2^H-1)}_{\text{Total}},\ \underbrace{2^{H-1}}_{\text{Max in last level of LST}}\right)}$$



$$\underbrace{\left[N-(2^H-1)\right]}_{} \leq 2^{H-1} \Rightarrow$$



$$(2^H-1) = x \qquad \frac{2^H}{2} = \frac{x+1}{2}$$

$$\boxed{L = \frac{(2^H-1-1)}{2} + \min\left(2^{H-1}, (N-(2^H-1))\right)}$$

$$\boxed{way(N) = {}^{N-1}C_L \times ways(L) \times ways(R)}$$

$$\boxed{R = (N-1) - L}$$

$$L = \left(\frac{x-1}{2}\right) + \min\left(\frac{x+1}{2}, N-x\right)$$

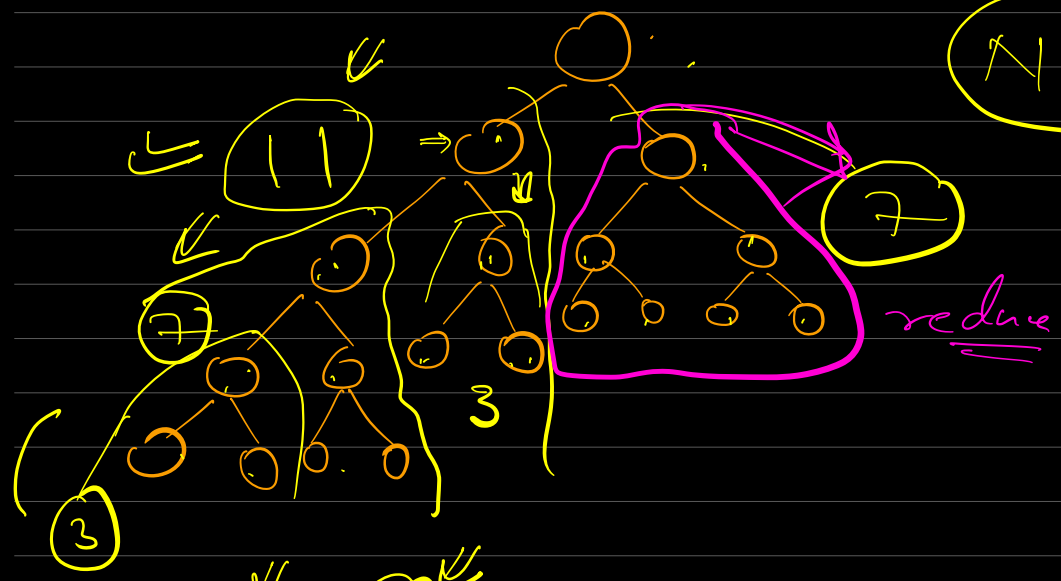
Pseudo Code

```
no. of Heap (N) {
    if (N ≤ 2) return 1;   // Base Case
    H = log2 N;
    x = 2^H - 1;
    L = (x-1)/2 + min( N-x, (x+1)/2 );
    R = (N-1) - L
    y = N-1C_L × no. of Heap(L) × no. of Heap (R);
    map.push(N, y);
    return y;
}
```

```
if (map.contains (N) {
    return map[N];
}
```

Q Can we optimize this ??

N = 16



3, 7, 11

⇒ for some values, we might be calculating again & again

⇒ Store the results once we calculate

Hashmap < No. of nodes , no. of heaps >

$N < [1 \; \oplus \; (N-1)]$

T.C. $\leq O(N)$

S.C $\leq O(N)$

1) Heap Sort
2) Ques. of Heap
3) Merge K sorted list

Priority Queues (Heap)

2) Problems from Tries