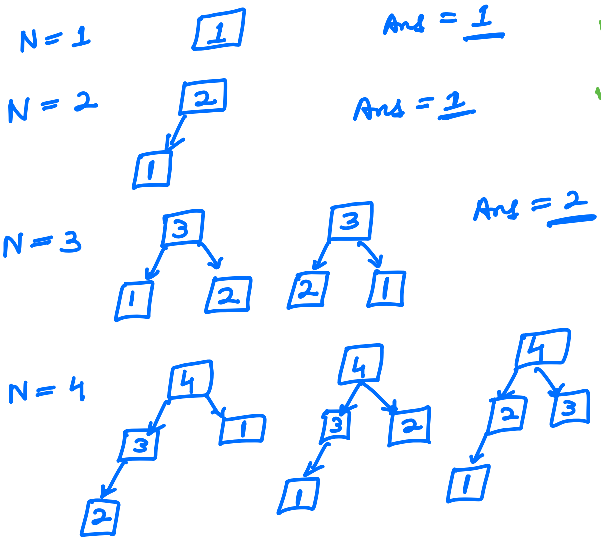
Q→ Ways to form max heap with N distinct elements ?

✓1) structure is complete binary tree.

✓2) root will have highest value & same applies to all subtrees.

{3) No relation b/w left & right subtree.

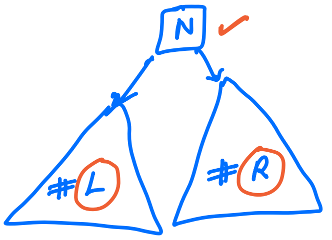


Ans = 1

Ans = 1

Ans = 2

Ans = 3 ✓



[1, 2, 3, 4 - - - N]

$$\boxed{L+R = N-1}$$

No of ways to select L elements out of (N-1) = ${}^{N-1}C_L$

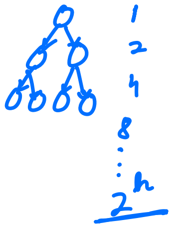
$$\boxed{R=(N-1)-L}$$

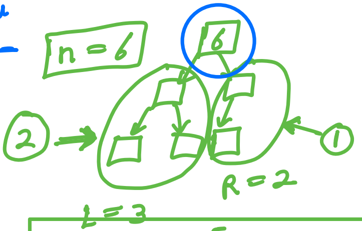
How to find L ?

$$\boxed{ways(n) = {}^{n-1}C_L * ways(L) * ways(R)} \leftarrow$$

1) height = log(N)

2) Max no. of nodes at i^{th} level = 2^i





$$\boxed{\begin{gathered} L+R = N-1 \\ \Rightarrow {}^{N-1}C_L = {}^{N-1}C_R \end{gathered}} \leftarrow$$

$$\boxed{ways(6) = {}^5C_3 * way(3) * ways(2)}$$

3) last level may or may not be completely filled. ✓

4) No of nodes at last level $= N - (1+2+4+ \text{—}\ 2^{h-1})$

$= N - (2^h - 1) \leftarrow$

5) # nodes till second last level in L $= \dfrac{(2^h-1) - \overset{\leftarrow \text{root}}{1}}{2}$ ✓ ←

6) # nodes at last level in L = $\min(2^{h-1}, N-(2^h-1))$ ←

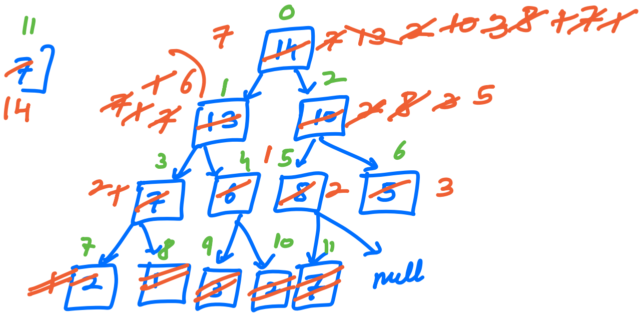
Max # nodes at last level in L $= \dfrac{2^h}{2} = 2^{h-1}$

Actual no. of nodes in last level $= N-(2^h-1)$

$$\boxed{\begin{aligned} ways(n) &= {}^{n-1}C_L * ways(L) * ways(R) \\ L &= \frac{(2^h-1)-1}{2} + \min\left(2^{h-1}, N-(2^h-1)\right) \end{aligned}} \leftarrow$$

Heap Sort

0	1	2	3	4	5	6	7	8	9	10	11
A = [~~14~~,	~~13~~,	~~10~~,	~~7~~,	~~6~~,	~~8~~,	~~5~~,	~~2~~,	~~1~~,	~~3~~,	~~2~~,	~~7~~]
~~7~~	~~7~~	~~2~~	~~1~~	1	2	3	~~1~~	8	10	3	14
	~~1~~	~~8~~	2				7				
7	~~7~~	~~3~~									
	~~1~~	5									
	6										



$i = 0 \rightarrow lc = A[1] = 13$ ✓
$\quad \hookrightarrow rc = A[2] = 10$

$i = 1 \rightarrow lc = A[3] = 7$ ✓ stop
$\quad \hookrightarrow rc = A[4] = 6$

$i = 0 \rightarrow lc = A[1] = 7$
$\quad \hookrightarrow rc = A[2] = 10$ ✓

$i = 2 \rightarrow lc = A[5] = 8$ ✓
$\quad \hookrightarrow rc = A[6] = 5$

$i = 5 \rightarrow lc = A[11]$
$\quad \hookrightarrow rc = A[12]$ > X

$$\underbrace{Array \rightarrow Heap}_{\substack{TC=O(N) \leftarrow \\ SC=O(1) ✓}} \underbrace{\rightarrow SortedArray}_{\substack{SC=O(1) ✓ \\ TC=\ ?}} ✓$$

$$\log(n) + \log(n-1) + \log(n-2) ... \log(1)$$
$$= \log(n!)$$

$$TC = O(\log(n!) + n)$$

$$\underbrace{\log(1) + \log(2) + ..\ \log(n)}_{n\ terms} \le \underbrace{\log(n) + \log(n) + ..\ \log(n)}_{n\ terms}$$

$$\boxed{\log(n!) \le n\log(n)}$$

$$\underbrace{\log(1) + \log(2) + ..\ \log(n)}_{n\ terms} \ge \underbrace{\log\left(\frac{n}{2}+1\right) + \log\left(\frac{n}{2}+2\right) + ...\ \log\left(\frac{n}{2}+\frac{n}{2}\right)}_{n/2\ terms}$$

$$\ge \underbrace{\log(n/2) + \log(n/2) + ---\ \log(n/2)}_{n/2\ terms}$$

$$\Rightarrow \boxed{\log(n!) \ge \frac{n}{2}\log\left(\frac{n}{2}\right)}$$

$$\boxed{\frac{n}{2}\log\left(\frac{n}{2}\right) \le \log(n!) \le n\log(n)} ✓$$

unstable sorting algo.

$--- x_1 --- x_2$

$x_1 == x_2$

x_1, x_2 ✓

$x_2\ x_1$ X

Q→ Given an integer array. ∀ $i \geq (k-1)$ find k^{th} largest element considering elements 0 – i

✓0 ✓1 ✓2 ✓3 ✓4 ✓5 ✓6

A = [10, 18, 7, 5, 16, 19, 3]

O/P → [7✓, 7✓, 10✓, 16✓, 16✓]

k = 3

Brute → $TC = O(K*(N-K)) \simeq O(K*N)$

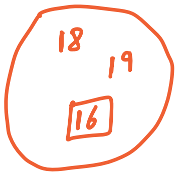
Among K elements → What is K^{th} largest element? → smallest element ✓

[1, 3, 8, 6, 10]

Use min heap with largest k elements from 0 – i

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

$SC = O(k)$



MinHeap ✓

Q→ For the input stream of integers. Find median at every step.

[Hackerearth]
[Directi]
[Deepen ai]

I/P → 9 8 7 3 6 4 1 ---

O/P → 9 8 8 7 7 6 6 ---

median → middle element in sorted order.

sorted → 1, 2, 4 → 2 (odd) ✓

1, 3, 6, 8 → 3 (even) ✓

solution

1) Brute → sort for every insertion → Ans = A[n/2] $TC \leq O(n * n\log(n))$

2) Insertion Sort → $TC = O(n^2)$

3) Using 2 heaps

Min Heap → large values

Max Heap → small values

5 4 3 2 1 6 7 8 9 10

O/P → 5 4 4 3 3 3 4 4 5 ←

1 2 3 4 5 — Max Heap

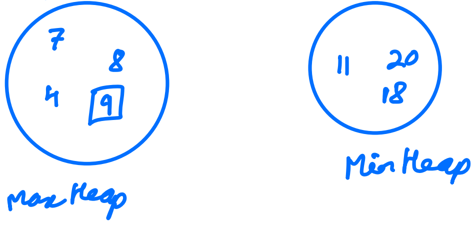
6 7 8 9 10 — MinHeap

Ans → Root of Max Heap ✓

[∴ size of max heap – size of min heap = {0, 1}] ✓ ← ✓

Dryrun → I/P → 9 8 7 4 11 20 18 - -

O/P → 9 8 8 7 8 8 9 - - -



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

$SC = O(N)$

```
1) if (x < root of max heap){
        insert x in max heap;
     → if (max heap size - min heap size > 1)
        { move root of max heap to
              min heap;
        }
   }
2) else { insert x in min heap;
     → if(min heap size > max heap size){
          move root to min heap to
              max heap;
   } }
```

Q→ Merge K sorted Linked List into one sorted LL?

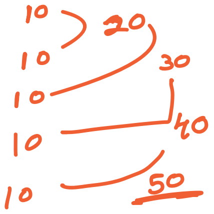
Eg→

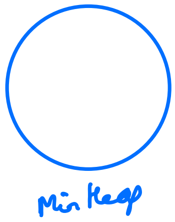
H1 $\overset{\cancel{x}}{1} \to \overset{\cancel{x}}{3} \to \overset{\cancel{x}}{7} \to \overset{\cancel{x}}{12} \to \overset{x}{\text{null}}$

H2 $\overset{\cancel{y}}{2} \to \overset{\cancel{y}}{6} \to \overset{\cancel{y}}{18} \to \overset{y}{\text{null}}$

H3 $\overset{\cancel{z}}{5} \to \overset{\cancel{z}}{10} \to \overset{\cancel{z}}{20} \to \overset{z}{\text{null}}$

K list of N length





$SC = O(K)$

$TC = (N * K * \log(K))$

50 elements

steps → 20 + 30 + 40 + 50 = 140 ✓

N
N
N
⋮
N
Klists

Total N*K elements ✓

$$\text{Steps} \approx 2N + 3N + 4N + \ldots KN$$

$$= N\left(\frac{K(K+1)}{2} - 1\right)$$

$$\approx O(K^2 * N)$$

O/P $\boxed{1} \to \boxed{2} \to \boxed{3} \to \boxed{5} \to \boxed{6} \to \boxed{7} \to \boxed{10} \to \boxed{12} \to \boxed{18} \to \boxed{20} \to \text{null}$

Head