# Heaps 2

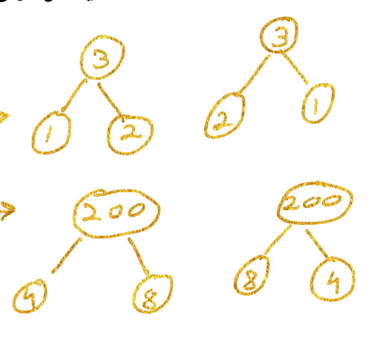
Wednesday, 6 October 2021 9:02 PM

Find the no. of ways to form Max Heap with N distinct elements.

$\{1, 2, 3\} \rightarrow$

$\{4, 8, 200\} \rightarrow$

$\{3, 18, 60\}$



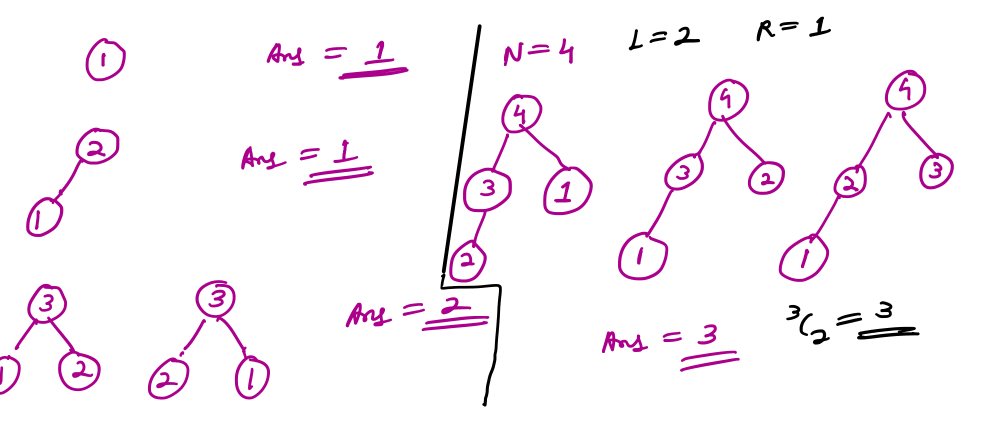
$N=1$ Ans $= 1$

$N=2$ Ans $= 1$

$N=3$ Ans $= 2$

$N=4$ $L=2$ $R=1$

Ans $= 3$ ${}^3C_2 = 3$



Ideas 1) The structure is same in all cases. ✓

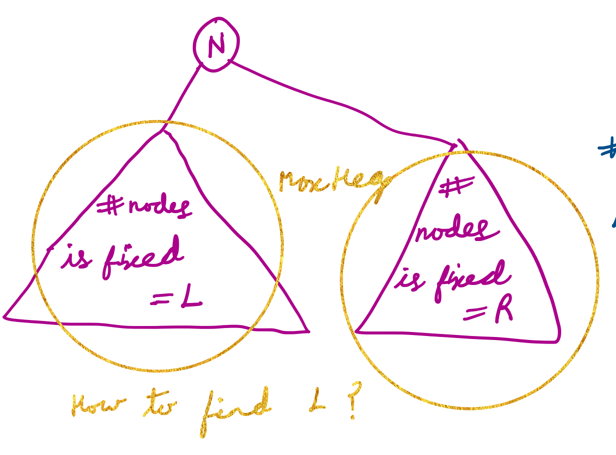
2) Root is fixed (even for subtrees root is fixed if we know the values). ✓

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

$\{1, 2, 3 \ldots N\}$

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

L & R are independent.



MaxHeap

#nodes is fixed $= L$

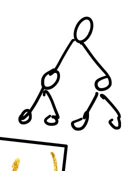
# nodes is fixed $= R$

# ways to select L elements from $(N-1)$ elements $= {}^{N-1}C_L$

How to find L ?

1) Height of complete binary tree $= \lfloor \log_2(N) \rfloor$ ✓

2) Max no. of elements present at any level $= 2^i$ (Root = level 0) ✓



1
2
4
8
16

last level may not be completely filled ?

$\boxed{3)}$ No. of elements present at last level $= N - \boxed{(2^H - 1)}$

#elements present till second last level. ✓

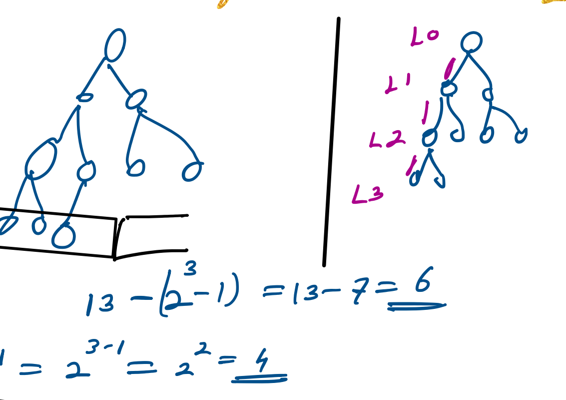
$\rightarrow (1 + 2 + 4 + \ldots 2^{H-1} = 2^H - 1)$ ✓

4) Max no. of elements in last level of left subtree $= \frac{2^H}{2} = 2^{H-1}$ ✓

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

(1) → Root

$$\frac{2^H - 1 - 1}{2} = \frac{2^H - 2}{2} = 2^{H-1} - 1$$



L0
L1
L2
L3

$H = 3$

$N = 13$ $13 - (2^3 - 1) = 13 - 7 = 6$

$2^{H-1} = 2^{3-1} = 2^2 = 4$

```
ways(N){
    if (N <= 2) return 1;
    if (ans[N] > 0) return ans[N];
    H = log2(N);
    L = min(N - (2^H - 1), 2^(H-1)) + (2^(H-1) - 1)
    R = (N-1) - L
    ans[N] = (N-1)C(L) * ways(L) * ways(R);
    return ans[N];
}
```

$TC = O(N)$
$SC = O(N)$

$L < N$
$R < N$

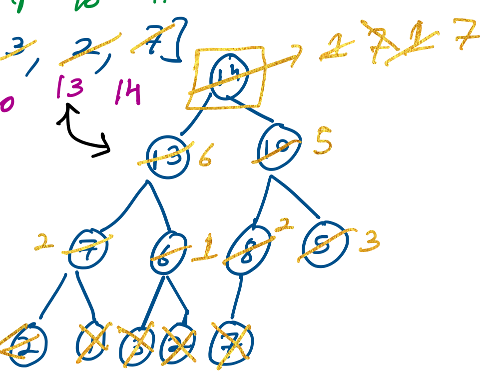
---

## Heap Sort

Eg → $A = [7, 3, 5, 1, 6, 8, 10, 2, 13, 14, 2, 7]$

↓ Inplace heap build $TC = O(N)$ $\boxed{SC = O(1)}$ ✓

$A = [14, 13, 10, 7, 6, 8, 5, 2, 1, 3, 2, 7]$ (indices 0–11)



$$TC = \log(N) + \log(N-1) + \log(N-2) \ldots \log(1)$$
$$= \log(N * (N-1) * (N-2) \ldots 1) = \boxed{\log(N!)}$$

$\boxed{\log(a) + \log(b) = \log(ab)}$

$$\log(1) + \log(2) + \ldots \log(N) \le \log(N) + \log(N) + \ldots \log(N)$$

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

last N/2 terms

$$\boxed{\log(1) + \log(2) + \ldots \log(N)} \ge \log\left(\frac{N}{2}+1\right) + \log\left(\frac{N}{2}+2\right) + \ldots \log(N)$$
$$\ge \log\left(\frac{N}{2}\right) + \log\left(\frac{N}{2}\right) + \ldots \log\left(\frac{N}{2}\right)$$

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

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

Heap Sort is unstable sorting technique.

2 3 7 8 3 5 → $\boxed{2\ 3\ 3\ 5\ 7\ 8}$

$x_1$ $x_2$ → $x_1$ $x_2$

Name
Marks
Roll. No.
⋮

---

## Q → Running Median (Hackerrank)

From the given input stream, find median at every step.

median → middle element in sorted order.

$\{1, 4, 2\} \rightarrow \{1, \boxed{2}, 4\}$

$\{1, 3, 4, 2\} \rightarrow \{1, \boxed{2}, 3, 4\}$

Eg → 9 8 7 3 6 4 . . .

Ans → 9 8 8 7 7 6 . . .

$SC = O(N)$

Brute → Sort at every step & return middle element as answer. $TC = O(N * N\log(N))$

Insertion Sort → Place new element in its correct position. $TC = O(N^2)$

Heaps → Store smaller values in max heap.
→ Store large values in min heap.

$\boxed{\text{Ans} = \text{Root of Max Heap}}$ → $\boxed{\text{size of max heap} - \text{size of min heap} = \{0, 1\}}$ ✓

min: 7, 10

max: 2, 3, 5

2 3 5 7 10

if ( x ≤ root (maxHeap)) → insert in max heap else insert in min heap & check size.

I/P → 9 8 7 3 6 10 14 . . .

O/P → 9 8 8 7 7 7 8 . . .

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

$SC = O(N)$

Max Heap: 8, 3, 6, 7

Min Heap: 14, 9, 10