16/8/2023

Heaps-1 – Introduction.

Q Given N ropes with their lengths.
Cost of connecting 2 ropes = sum of length of both.

Find min cost to connect all ropes.

2, 5, 2, 6, 3

$\{2, 5\} \to 7$, $\{2, 6\} \to 8$, $\{8, 3\} \to 11$, $\{7, 11\} \to 18$

cost = (2+5) + (2+6) + (8+3)
= (11+7)
⇒ 7 + 8 + 11 + 18
⇒ 44

2, 5, 2, 6, 3

$\{2, 5\} \to 7$, $\{7, 2\} \to 9$, $\{6, 3\} \to 9$, $\{9, 9\} \to 18$

cost = (2+5) + (7+2) + (6+3)
= (9+9)
⇒ 7+9+9+18
= 43

2, 2, 3, 5, 6

$\{2, 2\} \to 4$, $\{4, 3\} \to 7$, $\{5, 6\} \to 11$, $\{7, 11\} \to 18$

cost = (2+2) + (4+3) + (5+6)
+ (7+11)

= 4+7+11+18
⇒ 40

Observation: Connect smaller length ropes.

$\underset{1}{x} < \underset{2}{y} < \underset{3}{z}$ (lengths)

$\boxed{x+y \quad (x+y)+z \quad 3+6} < \boxed{x+z \quad (x+z)+y \quad 4+6} < \boxed{y+z \quad (y+z)+x \quad 5+6}$

Sorting

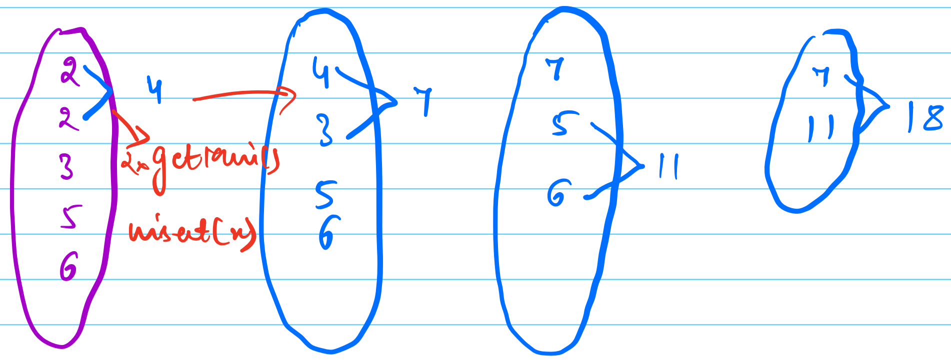
2 2 3 5 6 10

4 7

sort at every step → insertion sort

TC: $O(N^2)$

SC: $O(1)$



DS → insert(x) , getmin() > $O(\log_2 N)$

↳ also removes from DS

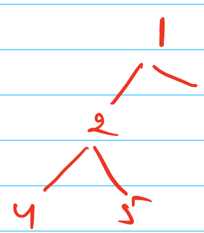
$3 \times (\log_2 N) \times (N-1) \Rightarrow$ TC: $O(N \log_2 N)$

SC: $O(N)$

Heaps (Priority Queue)

1> **Structure** → complete binary tree.

CBT → every level is complete except (maybe) the last level. All nodes of the last level are as left as possible.



2) **Types** → Min heap ∀ nodes

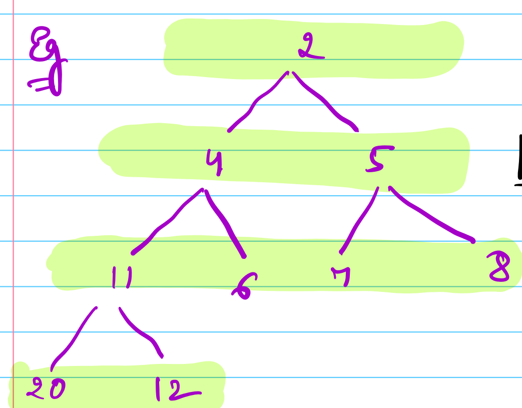
node.data <= children.data

→ Max heap ∀ nodes

node.data >= children.data

3> No relationship b/w left & right subtrees.

Eg



Storing in array -

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

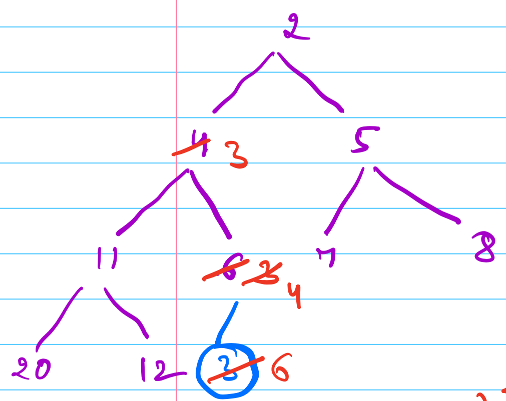
∀ nodes i

left child $= 2*i+1$

right child $= 2*i+2$

parent $= (i-1)/2$

Insertion in heap



insert(3)

TC: $O(\log_2 N)$

insert(x)

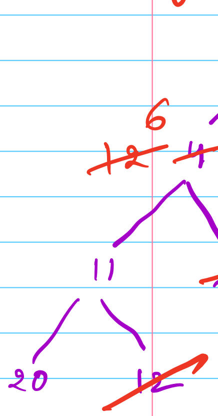
A =

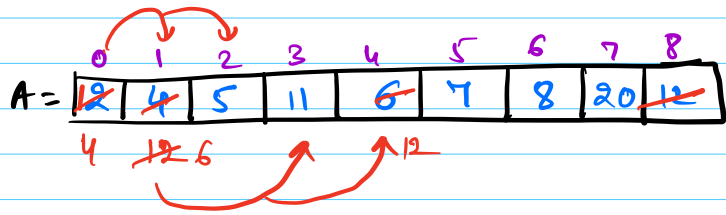
0	1	2	3	4	5	6	7	8	9
2	4	5	11	~~6~~ 3	7	8	20	12	~~3~~ 6

Heapify → maintain properties of heap after any operation.

```
A[N] = x , i = N , N += 1

while ( i > 0 ) {
    p = (i-1)/2

    if ( A[p] > A[i] ) {
        swap ( A[i] , A[p] );

        i = p;
    }
    else {
        break;
    }
}
```

getMin [Min heap]






```
ans = A[0]    A[0] = A[N-1] , N = N-1
                              remove last index.
 i = 0
 while ( i<N ) {
   lc = 2*i+1
   rc = 2*i+2.   // lc+1

   if (rc<N){
     if (A[lc] <= A[i] && A[lc] <= A[rc]){
        swap(A[lc], A[i]);
        i = lc;
     }
     else if(A[rc] <= A[i] && A[rc] <= A[lc]){      2 child nodes
        swap (A[rc], A[i]);
        i = rc;
     }
     else{
        break;
     }
   }
   else if (lc<N) {
     if (A[lc] <= A[i]){
        swap (A[lc], A[i]);
        i = lc;                                     Single child
     }
     [else{ break]
   }
no child ⟶ else break;
 } return ans;
```

Searching → TC: O(N)

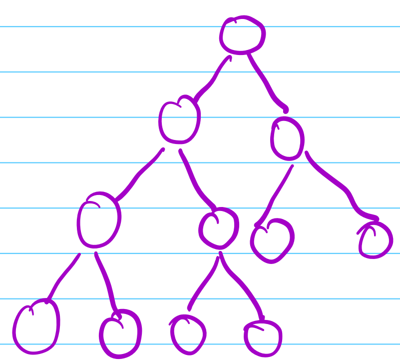
Build

1> Insert all elements 1-by-1 → TC: O(N log N)

2> If all elements are known.

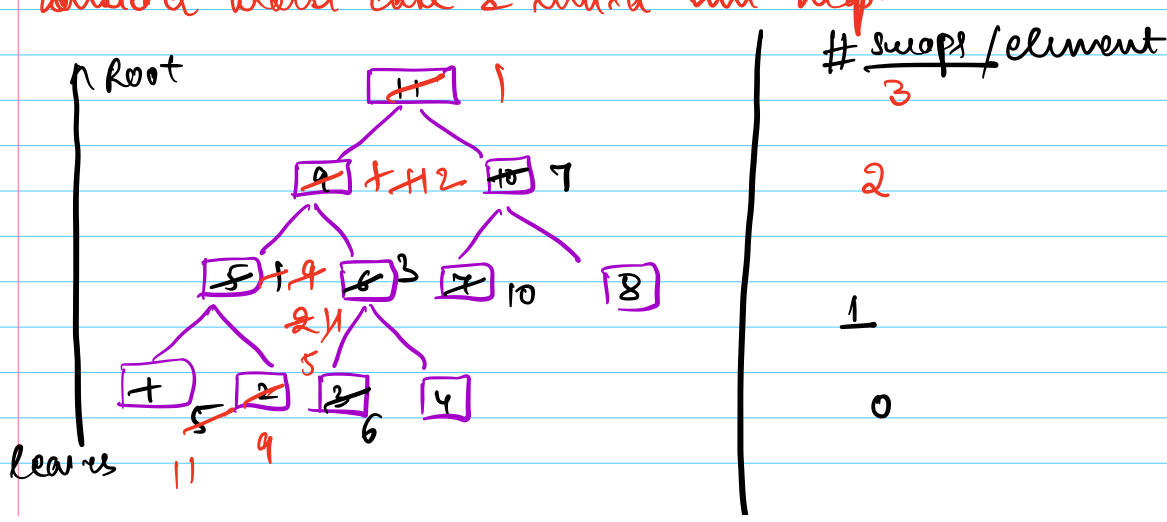
N →	1	2	3	4	5	6	7	8	9	10	11
	1	1	2	2	3	3	4	4	5	5	6

leaves in complete BT = $(N+1)/2$



	0	1	2	3	4	5	6	7	8	9	10
A =	5	8	6	1	7	2	3	4	9	11	10

consider worst case & build min heap.



Total swaps

$\Rightarrow \frac{N}{2} * 0 + \frac{N}{4} * 1 + \frac{N}{8} \times 2 \cdots\cdots$

$= \sum \frac{N}{2^{i+1}} * i = \frac{N}{2} \left(\sum \frac{i}{2^i} \right) \Rightarrow \frac{N}{2} * 2 \Rightarrow N$

$\sum \frac{i}{2^i} \rightarrow 2$

TC: O(N)

$S = 1/2^1 + 2/2^2 + 3/2^3 \cdots$ } AGP

$S/2 = 1/2^2 + 2/2^3 + 3/2^4 \cdots\cdots$

$\left(S - S/2\right) = \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} \cdots\cdots$ GP

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

$S = 2$

Inplace Heap build (Max heap)

	0	1	2	3	4	5	6	7	8
A =	~~7~~ 8	~~3~~ 5	~~5~~ ~~8~~ 7	~~1~~ 4	~~5~~ 3	6	~~8~~ 5	2	~~4~~ 1

→ Leaves

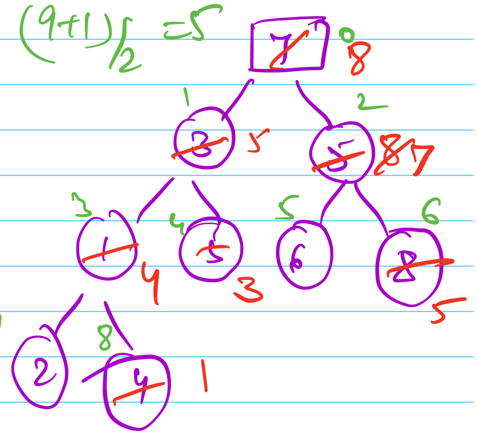
leave = (9+1)/2 = 5

lc = A[1]
rc = A[2]

lc = A[3]
rc = A[4]

lc = A[5]
rc = A[6]

lc = A[7]
rc = A[8]



TC: O(N)
SC: O(1)

$A = [8 \quad 5 \quad 7 \quad 4 \quad 3 \quad 6 \quad 5 \quad 2 \quad 1]$