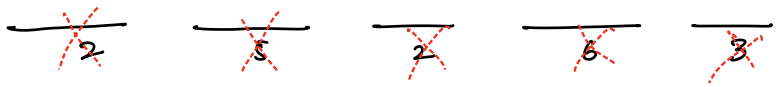
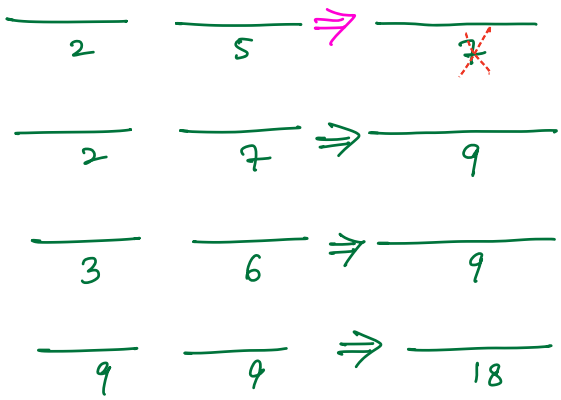


cost of connecting 2 ropes = sum of length of 2 ropes

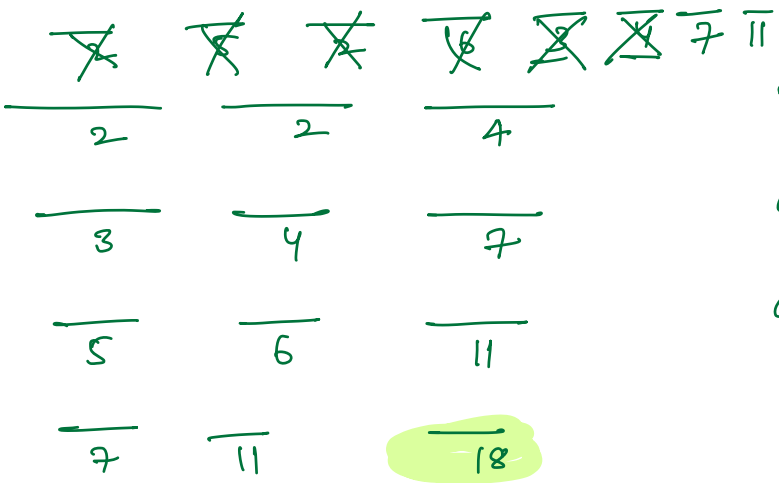
Find minimum cost to connect all the ropes, connecting two at a time.

1)



cost = 0
cost += 7
cost += 9 (16)
cost += 9 (25)
cost += 18 (43)

2)



cost = 4
cost += 7 = 11
cost = 22
cost += 18 = 40

x < y < z

x y z

x+y+
x+y+2

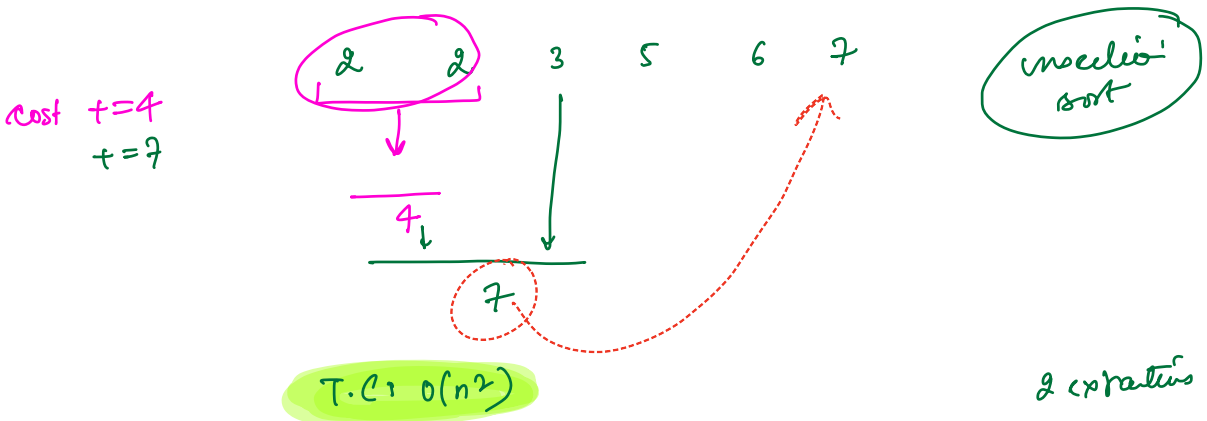
x z y

x+z+
x+z+y

z y

y+z
y+z+x

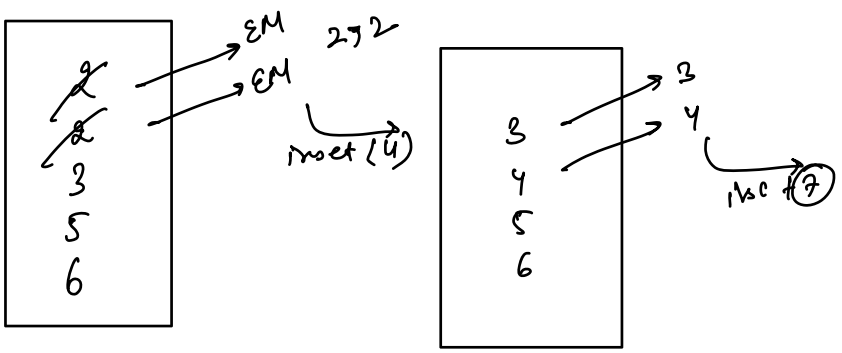
Choose two smallest rope to connect.



T.C: O(n²)

2 extractions + 1 insertion

DS → extractMin() — $\log_2 N$
DS → insert() — $\log_2 N$
} N log N




```
any i/p given
1) Build min heap
for(i=0; i<n-1; i++)
{
  x = extractMin()
  y = extractMin()
  cost += (x+y);
  insert(x+y);
}
```

Heaps (Binary Heap) → Binary Tree

1) structural

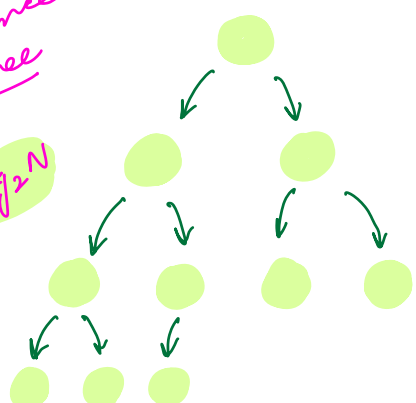
complete Binary Tree

Tree which has all levels completely filled, last level ≡ left to right.

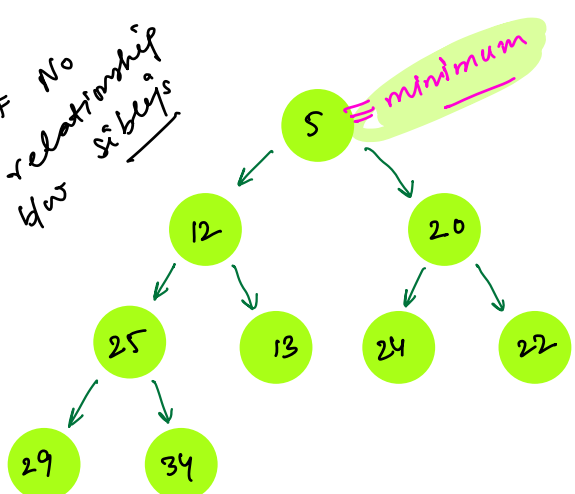
CBT ≡ Balanced Tree

||

$H = \log_2 N$



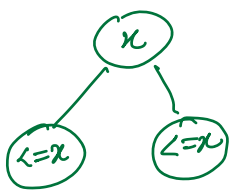
No relationship b/w siblings



min-heap

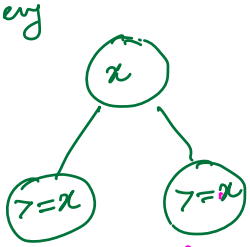
2) elements order

a) for every node

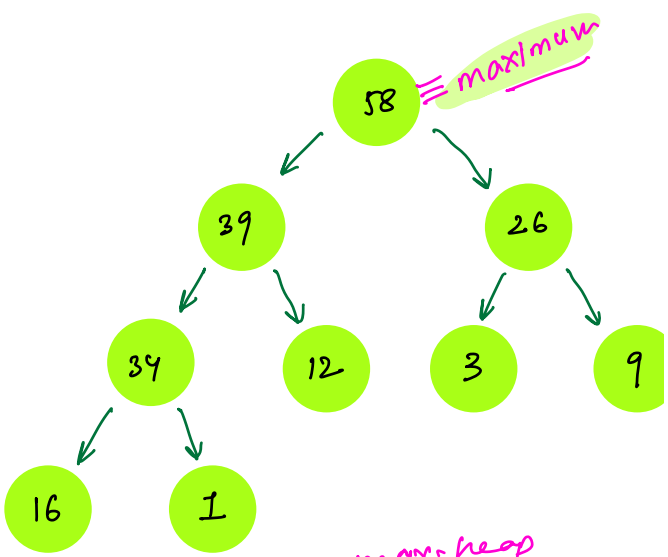


max-heap

for eg.



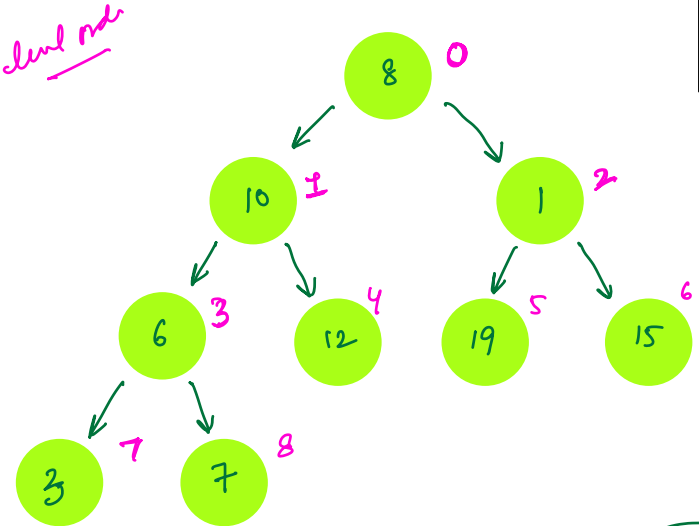
min-heap



max-heap

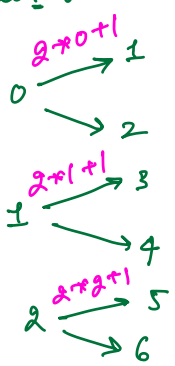
Array implementation of trees

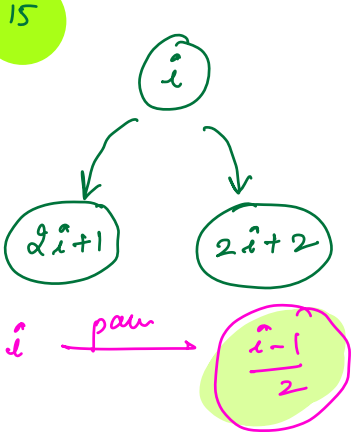
level order



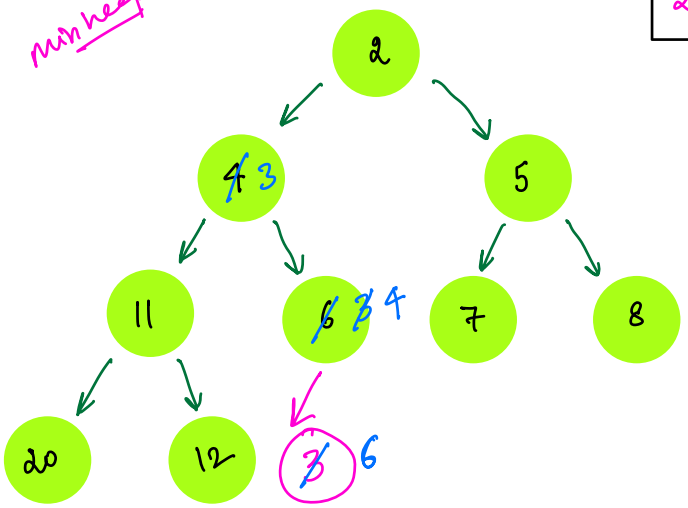
0	1	2	3	4	5	6	7	8
8	10	1	6	12	19	15	3	7

root ≡ arr[0]





insert in Heap



T.C: $\log N \approx O(H)$

heap →

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

3 3 4 6

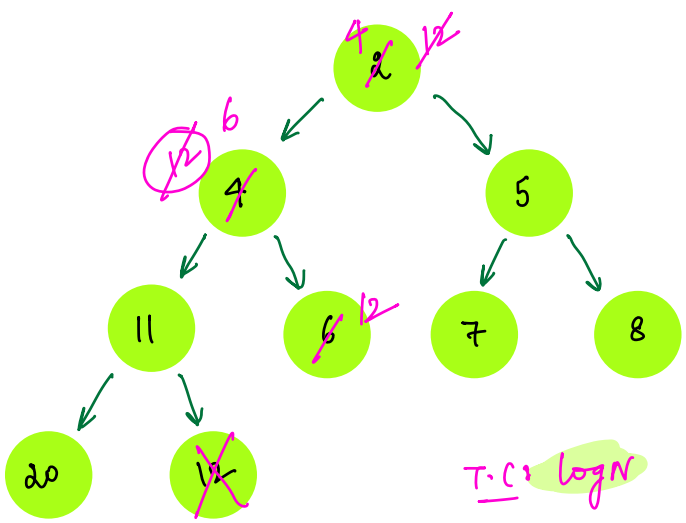
insert 3

heap.insert(3); (at end)

i	parent	
9	$\frac{(9-1)}{2} = 4$	arr[4] > arr[9] swap
4	$\frac{(4-1)}{2} = 1$	arr[1] > arr[4]
1	0	arr[0] < arr[1]

stopping cond^n → reach to root (i=0)

extract min — return the min element



T.C: logN

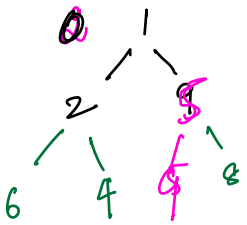
0	1	2	3	4	5	6	7	8
2	4	5	11	6	7	8	20	12

12 4 12 6 12 2

swap(arr[0], arr[size-1])
size--;

i	children	
0	1 → 4 2 → 5	min(4,5) swap
1	3 → 11 4 → 6	min(11,6) swap
4		

Build



0	1	2	3	4	5
9	1	9	6	4	8

1 2 5 9

inplace

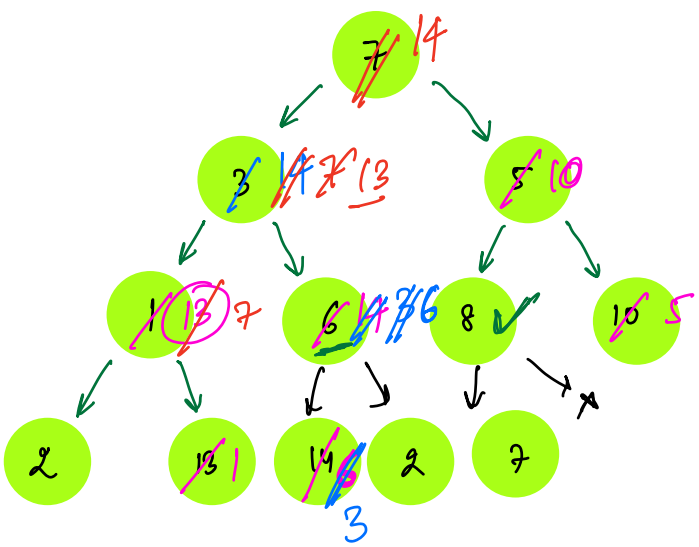
again & again insert fun^n

Heapify

0	1	2	3	4	5	$N/2-1$	$N-1$
7	3	5	1	6	8	10	2	13	14	2	7

↳ first non leaf

max-heap



$i = 5$
$i = 4$
$i = 3$
$i = 2$

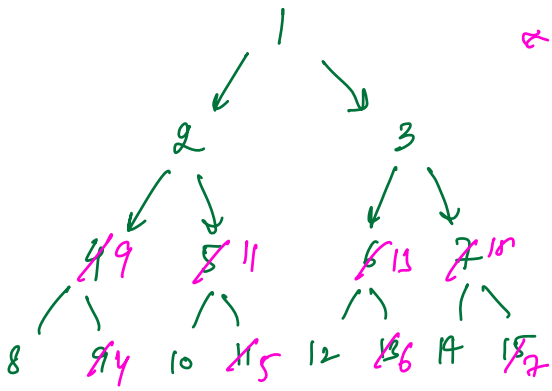
✓ swap max(4,2)

$$\frac{N-1}{2} - 1$$

$$= \frac{N-2}{2}$$

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

elements in max heap



T.C: $N/2 * 0 + N/4 * 1 + N/8 * 2 + N/16 * 3 \ldots$

$= N/2 \left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} \ldots \right)$

T.C = $N/2 * 2 = O(N)$ → AGP

$S = 1/2 + 2/4 + 3/8 + 4/16 + 5/32 \ldots$

$S/2 = \quad 1/4 + 2/8 + 3/16 + 4/32 \ldots$

$S/2 = 1/2 + 1/4 + 1/8 + 1/16 + 1/32$

$S/2 = 1$

$S = 2$

$S_\infty = \frac{a}{1-r}$

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

```
heapify( heap[ ], int i)
{
    while( 2i+1 < size )
    {   x = max( heap[i], heap[2i+1], heap[2i+2]);

        if( x == heap[i])
          {    return ;}
        else if(   x == heap[2i+1])
             {        swap( heap[i], heap[2i+1]);
                      i = 2i+1;
             }
        else
          {
                    swap( heap[i], heap[2i+2]);
                      i = 2i+2;
          }
    }
}
```

Q N chocolate bags, each having A[i] chocolates.

Kid ⇒ select the bag with max no of chocolates & eats it.

Magician ⇒ Fill the bag again by A[i]/2 chocolates.

Find no of chocolates kid can eat in k steps.

K = 4

A = [~~10~~ 5, 3, ~~18~~ 9, ~~8~~ 4, 4] K = 5

ans t = 15

t = 10

t = 8

t = 7

max-heap

1) Build
2) extract
   insert
   (k > 0)