Heaps

Min Cost To connect all ropes
Intro to heaps
Insertion and deletion in heaps
Build heap in O(N)

Merge N sorted arrays

Priority queue is by default a min heap in STLs and Java Collection

Min Cost To connect all ropes

A B C D E
2 5 2 6 3

Ropes → Cost to Connect 2 ropes = Sum of length of 2 ropes

Find min Cost to connect all the ropes?

A B C D E
2 5 2 6 3
8 5 2 3
8 8 2
8 10
18

Cost: 8
8
10
18

Total Cost: 44.

~~5~~, 6, 3, 2+2 → 4
5, 6, 7 → 7
11, 7 → 11
18 → 18

40

Idea: Always pick the 2 smallest ropes.

Approach: Sort the array and keep picking two smallest ropes.

Sorted Arrays: a, b

a+b Insert and sort again (n times) $(N * N \log N) \approx N^2$

Operations to perform for above problem:-

1) Insertion
2) Deletion
3) getMin

} This is supported by heap.

Intro to heaps

Heap Data Structure

↓

Binary Tree

Structure

$\boxed{\text{Complete Binary Tree}}$

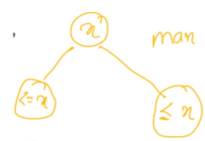
→ All levels should be completely filled.

→ exceptions → (last level may not be full but should be filled from left to right)

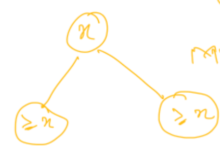
$\boxed{H = \log_2 N}$

Order of Elements

for ∀ nodes :

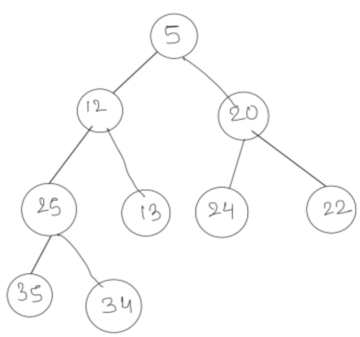


max heap

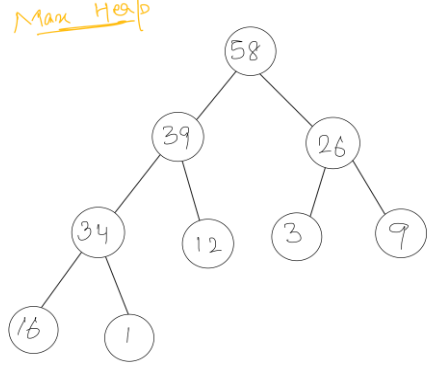


Min Heap.

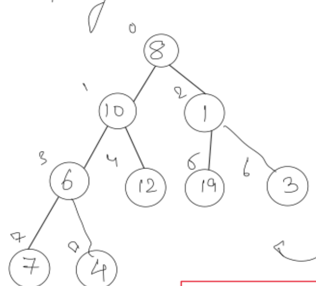
Min Heap



Max Heap



Array Implementation of Trees.



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

0 $\xrightarrow{\text{left}}$ 1, 0 $\xrightarrow{\text{right}}$ 2

2 → 5, 2 → 6

1 → 3, 1 → 4

3 → 7, 3 → 8

i $\xrightarrow{\text{left child}}$ $2i + 1$

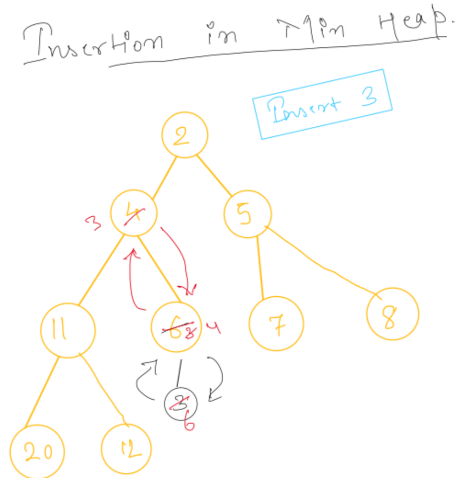
i $\xrightarrow{\text{right child}}$ $2i + 2$

i.e for any index i parent of i is

$= \boxed{(i-1)/2}$

Insertion and deletion in heaps

Insertion in Min Heap.



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

i	parent $= (i-1)/2$	ar[Par] > arr[i]
9	$\frac{9-1}{2} = 4$	Yes, So Swap
4	$\frac{4-1}{2} = 3$	Yes, So Swap.
1	$\frac{1-1}{2} = 0$	NO, Skip / return

```
Void insert ( int heap[], int x)
{
    heap.insert (x);    //insert element in last
    int i = heap.size() -1 ;
    while (i > 0)
    {
        int p = (i-1)/2 ;
        if ( heap [p] > heap[i] )
        {
            Swap (heap[p] , heap[i]);
            i = p;
        }
        else
            break;
    }
}
```

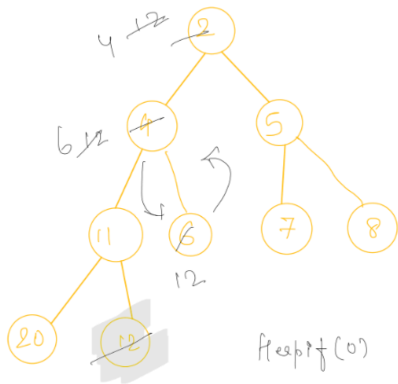
$$\boxed{T.C : O(\log_2 N)}$$

To build a new heap :-

Keep inserting all the elements one by one using above approach

So, T.C : O(N log N)

Extract Min :-



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

→ Replace last element with top (root)
→ remove last element
→ start heapify ( )

Heapify (0)

i	children	
0	1 → 4 2 → 5	Swap 12 & 4
1	3 → 11 4 — 6	Swap 12 & 6

```
void heapify (int [ ], int i)
{
    while (i < heap.size())
    {
        int x = heap[i];
        if (2i+1 < N)
        {   x = min (x, heap [2i +1])
        }
        if (2i+2 < N)
        {   x = min (x, heap [2i + 2])
        }
        if (heap[i] == x)
            return;
        if (x == heap [2i+1] )
            swap (heap[i], heap [2i + 1])
            i = 2i+1;
        } else
        {
            swap ( heap [i], heap[2i+2]
            i = 2i+2;
        }
    }
}
```

Pseudo Code for Connecting ropes :-

① Build the heap } N log N

② while (heap.size > 1)

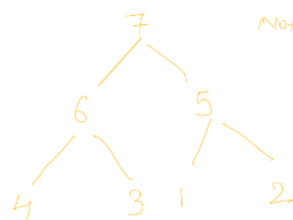
```
{
    int x = extractMin()
    int y = extractMin()
    ans = ans + (x+y)                    N log N
    heap.insert(x+y);
}
```

③ return ans;

O(N log N)

Build heap in O(N)

Build Min Heap



Not a min heap currently

0	1	2	3	4	5	6
7	6	5	4	3	1	2

(I) Sort the array to build the min heap
T.C: O(N log N)

(II) keep doing insertion one by one
T.C: O(N log N)

(III) → Start the heapify from the first non-leaf node because leaf node is already heapified
→ That is parent of last leaf node. parent(N-1) $= \frac{(N-1)-1}{2} = \frac{N-2}{2} = \frac{N}{2} - 1$

```
for (i = N/2 - 1 ; i >= 0 ; i--)
{  heapify ( heap, i);          // T.C : O(N)
}
```

T.C Analysis

No of Nodes	Total Iter
N/16	N/16 * 3
N/8	N/8 * 2
N/4	N/4 * 1
N/2	N/2 * 0

Total Iter $= \frac{N}{2} * 0 + \frac{N}{4} * 1 + \frac{N}{8} * 2 + \frac{N}{16} * 3 + \dots$

$$S = \frac{N}{2}\left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \dots\right]$$

$$\frac{S}{2} = \frac{N}{2}\left(\frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \dots\right] \quad \text{A GP}$$

$\boxed{1 - 2}$

$$\frac{S}{2} = \frac{N}{2}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots\right) \quad \text{GP}$$

$a = 1/2$
$r = 1/2$
$t = N$

$$\frac{S}{2} = \frac{N}{2}\left(\frac{1/2}{(1 - 1/2)}\right) = \frac{N}{2}(1)$$

$$\boxed{S = N}$$

$\therefore$ T.C : $O(N)$

Merge N sorted arrays

Merge N Sorted Arrays :

a = {2, 3, 11, 15, 20}
b = {1, 5, 7, 9}
c = {0, 2, 4}
d = {3, 4, 5, 6, 7}
e = {-2, 5, 10, 20}

Resultant Array

⇒ [-2, 0, . . .]

Approach :

→ Insert the first index of all the arrays in a heap.

→ Extract min() & keep pushing into resultant array.