Lecture :- Heaps – 1

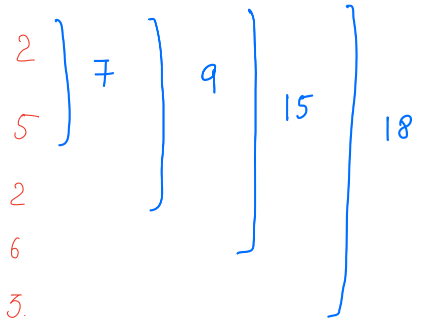
Agenda

- Connect n ropes
- Heaps with implementation
- Build min heap
- Merge n sorted arrays.

Qu Given n ropes with their length.

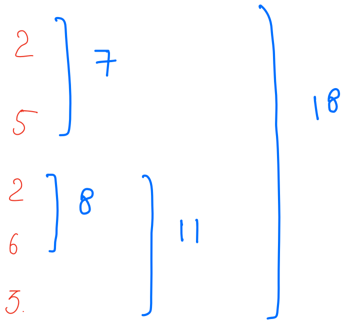
cost of connecting 2 ropes = sum of length of both.

find min cost of connecting all ropes



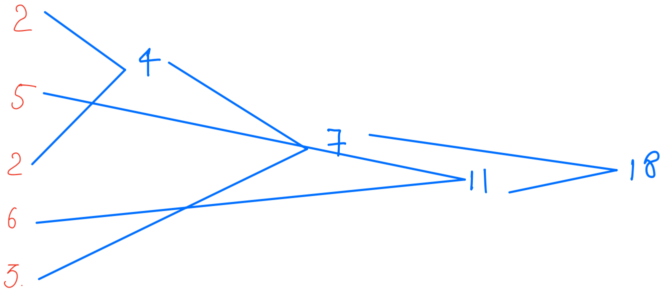
⟹ 7 + 9 + 15 + 18 = 43.

Overall cost = 43



7 + 8 + 11 + 18 = 44.

Overall cost = 44



4 + 7 + 11 + 18 = 40

overall cost = 40 ✓

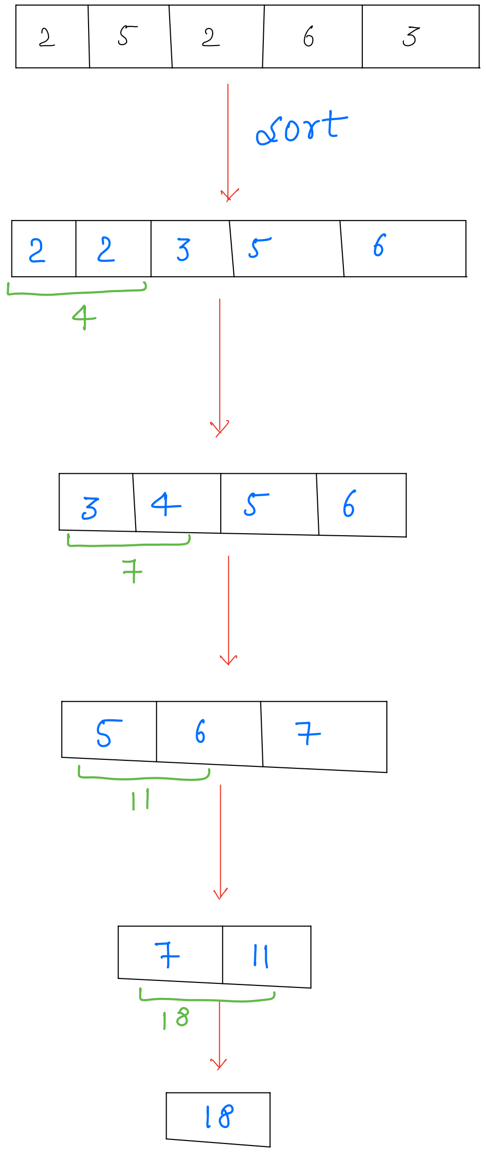
Observation

Lets say $x < y < z$

$x + y$ $+$ $x + y + z$	$y + z$ $+$ $y + z + x$.	$x + z$ $+$ $x + z + y$.

claim ⇒ Always connect min length rope first.

Brute force:



Total cost ⇒

$4 + 7 + 11 + 18$

⇒ 40

Time complexity: $n * n \log n = O(n^2 \log n)$

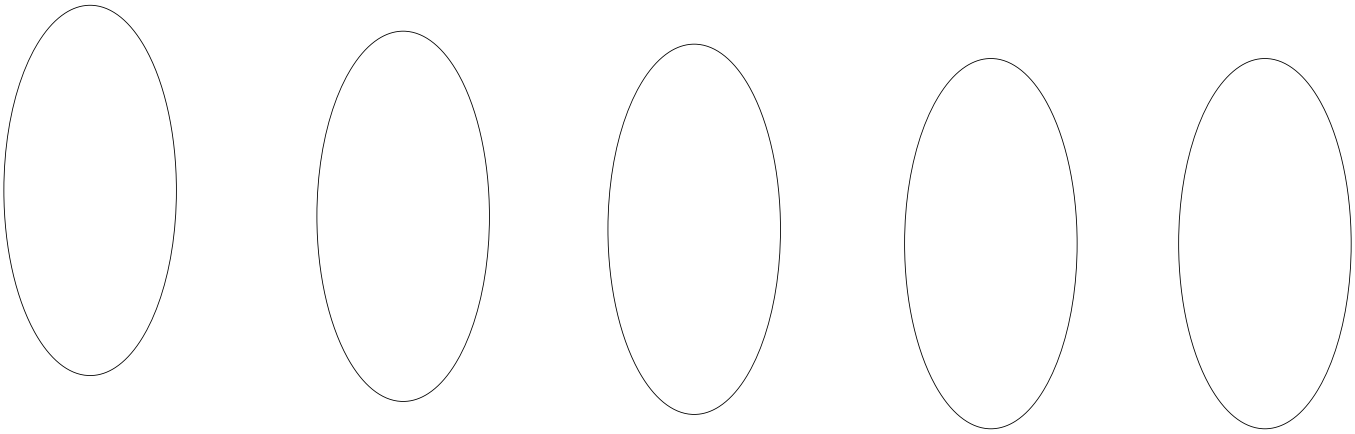
Improved idea

insert →

getMin() →

removeMin() →

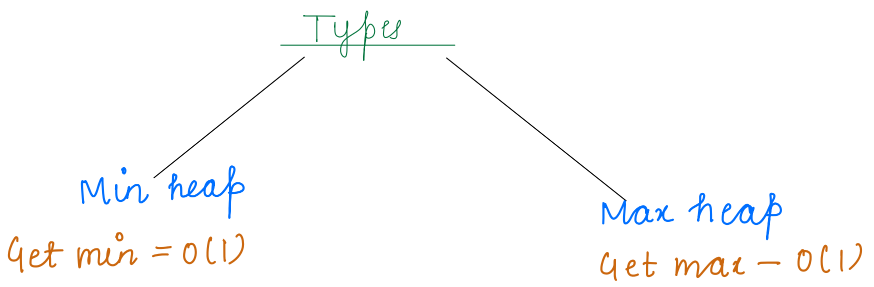
2	5	2	6	3



Heap Data structure

[Min or max value in optimal time]

Structure: complete binary tree.


Types

Min heap
Get min = O(1)

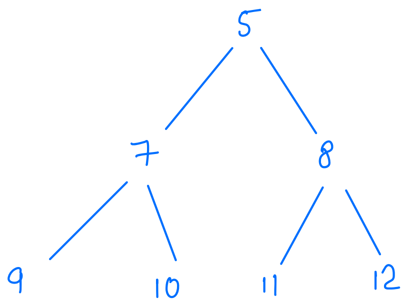
Max heap
Get max – O(1)

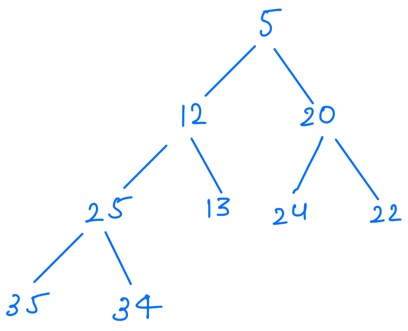
Heap order property Parent has higher priority than its children.

Min heap ⇒ Parent must be smaller than its children.

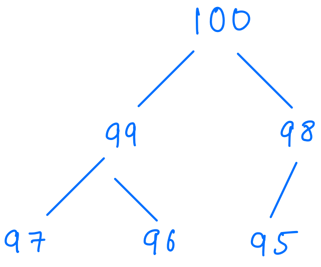
Max heap ⇒ " " " larger " " " .

Min heap





Max heap



Implementation and visualisation

0	1	2	3	4	5	6	7	8	9	10
3	5	10	6	8	12	13	10	12	15	11

0th idx ⇒ $lc = 2*0+1=1$

$rc = 2*0+2 = 2.$

1st idx ⇒ $lc = 2*1+1 = 3$

$rc = 2*1+2 = 4.$

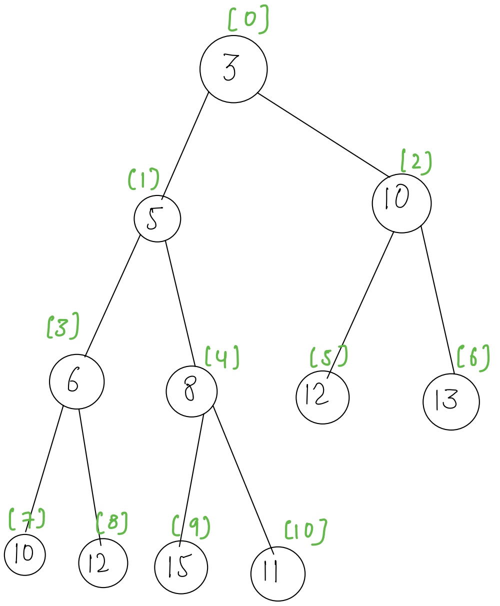
⋮

ith idx ⇒ $lc = 2i+1$

$rc = 2i+2.$

↓

$$parent = \frac{i-1}{2}$$

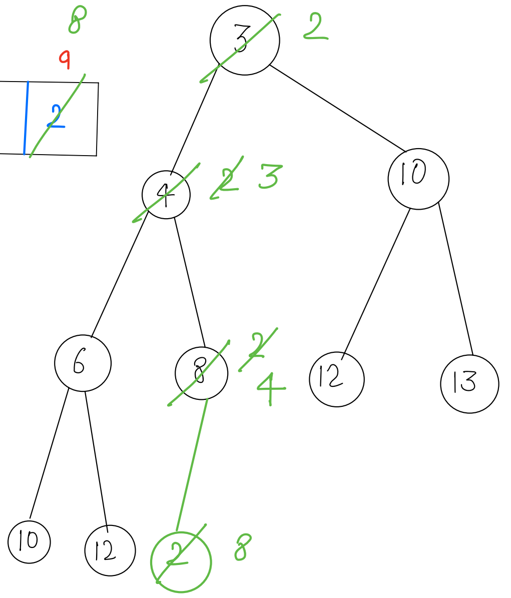


Insertion [Up-heapify]

0	1	2	3	4	5	6	7	8
3	4	10	6	8	12	13	10	12

insert(2)

2	~~2~~ 3			~~2~~ 4					8
0	1	2	3	4	5	6	7	8	9
~~3~~	~~4~~	10	6	~~8~~	12	13	10	12	~~2~~



A[9] = 2   parent = $\frac{9-1}{2} \Rightarrow 4$

A[4] = 8

swap(4th idx, 9th idx)

A[4] = 2   parent = $\frac{4-1}{2} = 1$

A[1] = 4

swap(4th idx, 1st idx)

A[1] = 2   parent = $\frac{1-1}{2} = 0$

A[0] = 3

swap(1st idx, 0th idx)

TC: O(log n)
↑
height of binary tree

Get min() ⟶ O(1)
return 0th idx of array.

Pseudo code

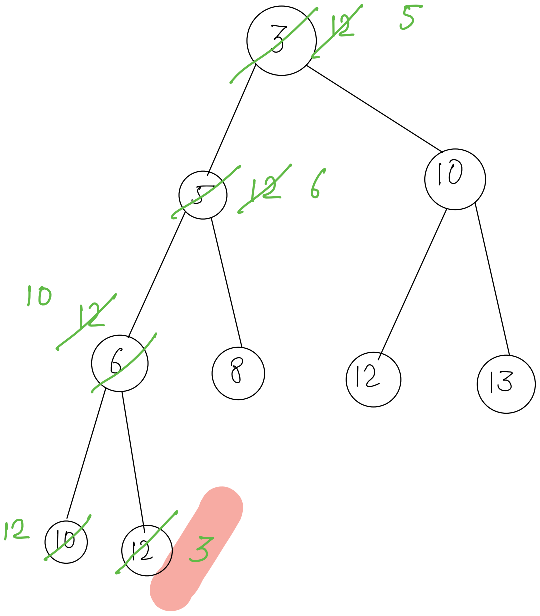
```
heap[ ];

heap.insert( val);      // Insert val at last.

i = heap.size() - 1;

while( i > 0) {

    pi = $\frac{i-1}{2}$ ;

    if ( heap [pi] > heap[i] ) {

        swap( heap, pi , i);

        i = pi ;

    } else {

        break;

    }

}
```

Delete minimum — Down heapify

0	1	2	3	4	5	6	7	8
~~3~~ ~~12~~ 5	~~5~~ ~~12~~ 6	10	~~6~~ ~~12~~ 10	8	12	13	~~10~~ 12	~~12~~ 3

swap 0th idx & last idx

Delete last idx



A[0]
↓
12

lc = 2*0+1 = 1 A[1] = 5

rc = 2*0+2 = 2 A[2] = 10

swap (A, 0, 1);

A[1]
↓
12

lc = 2*1+1 = 3 A(3) = 6

rc = 2*1+2 = 4 A[4] = 8

swap (A, 1, 3);

A[3]

lc = 2*3+1 = 7 A[7] = 10

rc = 2*3+2 = 8

swap(A, 3, 7)

TC: O(log n)

Pseudo code

h | w.

Qu Given arr[n] in any order. Create min heap [Medium-hard]

0	1	2	3	4	5	6	7	8	9	10
7	3	5	1	6	8	10	2	13	4	-2

Bruteforce: Sort the array.

0	1	2	3	4	5	6	7	8	9	10
-2	1	2	3	4	5	6	7	8	10	13

TC: $O(n \log n)$

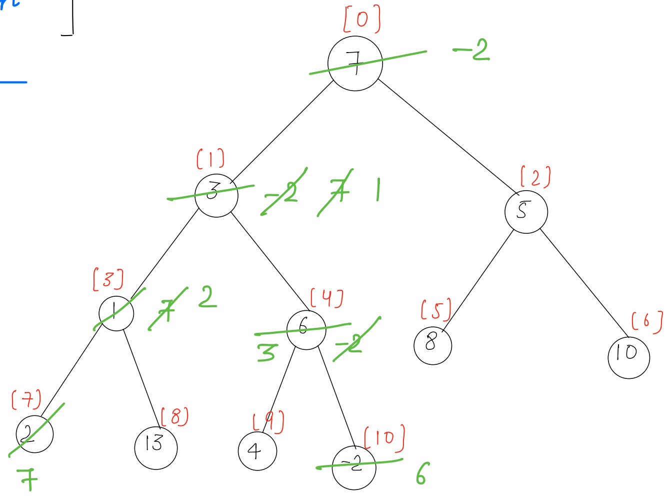
Inplace heap build

SC: O(1)

0	1	2	3	4	5	6	7	8	9	10
-2 ~~7~~	~~7~~ ~~-2~~ ~~3~~ 1	5	~~1~~ 7 2	3 ~~-2~~ ~~6~~	8	10	~~2~~ 4	13	4	6 ~~-2~~

Dry run: Non-leaf nodes ⇒ [4th − 0th]

↑ $\frac{\text{last node} - 1}{2}$



1.) 4th idx A[4] = 6

lc = 2*4+1 = 9 A[9] = 4

rc = 2*4+2 = 10 A[10] = -2

swap (A, 4th, 10th)

2.) 3rd idx A[3] = 1

lc = 2*3+1 = 7 A[7] = 2

rc = 2*3+2 = 8 A[8] = 13

do not swap

3.) 2nd idx A[2] = 5

lc = 2*2+1 = 5 A[5] = 8

rc = 2*2+2 = 6 A[6] = 10

do not swap

4.) 1st idx A[1] = 3

lc = 2*1+1 = 3 A(3) = 1

rc = 2*1+2 = 4 A[4] = -2

swap (A, 1, 4)

↓

4th idx → don't swap

5.) 0th idx A[0] = 7

lc = 2*0+1 = 1 A(1) = -2

rc = 2*0+2 = 2 A[2] = 5

swap (0, 1)

1st idx A[1] = 7

lc = 2*1+1 = 3 A(3) = 1

rc = 2*1+2 = 4 A[4] = 3

swap (1, 3)

3rd idx. A[3] = 7

lc = 2*3+1 = 7 A[7] = 2

rc = 2*3+2 = 8 A[8] = 13

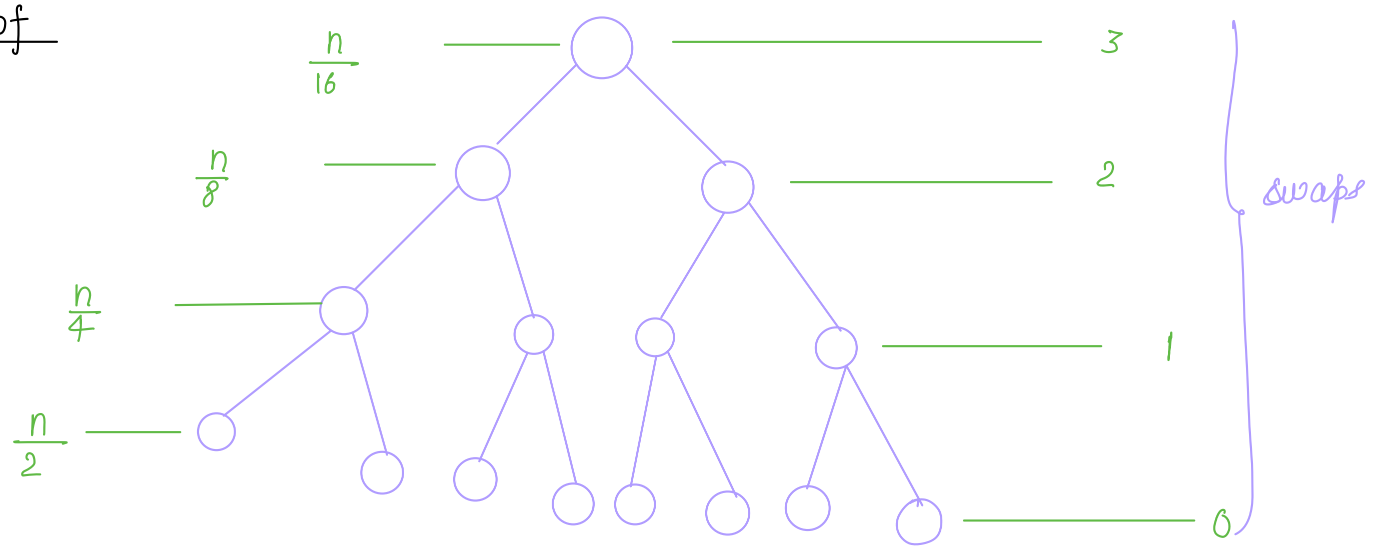
swap (3, 7)

Logic:

```
for(i = (n-1)/2; i>=0; i--) {
        downheapify(i);
}
```

TC: O(n) ⟶ Amazon.

Proof



Total swaps ⇒ $\frac{n}{2} * 0 + \frac{n}{4} * 1 + \frac{n}{8} * 2 + \frac{n}{16} * 3 \ldots\ldots$

$$\frac{n}{2}\left[\underbrace{0 + \frac{1}{2} + \frac{2}{4} + \frac{3}{8} \ldots\ldots}_{s\,=\,2}\right. \quad = \frac{n}{\cancel{2}} * \cancel{2} = n$$

$s = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} \ldots\ldots\ldots$ (I)

$\frac{1}{2} s = \frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \frac{4}{32} \ldots\ldots\ldots$ (II)

$-$ $-$

$\frac{1}{2} s = \underbrace{\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16}}_{GP} \ldots\ldots\ldots$ (I) $-$ (II)

$\frac{a}{1-r}$

$$\frac{s}{2} = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = \frac{\cancel{\frac{1}{2}}}{\cancel{\frac{1}{2}}}$$

$\frac{s}{2} = 1 \Rightarrow \boxed{s = 2}$

Break: 8:46 — 8:56

Qu 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	8

e =

-2	5	10	20

output:

-2	0	1	2	2	3	3	4	5 -	- -

Idea:

	# of pointers
Merge 2 sorted arrays	2
Merge 3 sorted arrays	3
Merge 4 sorted arrays	4
⋮	
Merge n sorted arrays	n

Approach

A =	2	3	11	15	20	
B =	1	5	7	9		
C =	0	2	4			
D =	3	4	5	6	7	8
e =	-2	5	10	20		

output: -2, 0, 1, 2, 2
⋮

min - heap

~~2, 1st, 0~~
~~1, 2nd, 0~~
~~0, 3rd, 0~~
3, 4th, 0
~~-2, 5th, 0~~
5, 5th, 1.
~~2, 3rd. 1~~
5, 2nd. 1
3, 1st, 1
4, 3rd, 2
⋮

TC: x * log(n)
(total el in all arrays)

Min heap java —— PriorityQueue< > = new PriorityQueue<>();

heap. add( ); —— O(logn)
heap. peek(); —— O(1)
heap. poll(); —— O(logn)

PriorityQueue< > = new PriorityQueue<>(Collections.reverseorder());
↑
Max heap

Thankyou 🙂

Doubts

Spotify

challaya	1 B
xyz	10M
abc	2M
cde	1.2B.

name
rollno

Student[] ⇒

Ayush,1	(8, 2)	(k,100)	U,(3).