Lecture : Heaps 1

Agenda

- Connect ropes
- Implementation of heaps
- Insertion and deletion of heaps
- Inplace heap build
- Merge K sorted lists.

Qu. Given n ropes with their length.

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

find min cost of connecting all ropes.

Sol

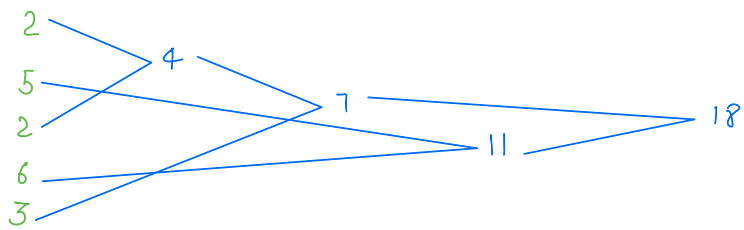
2
5 — 7
2 — 9
6
3 — 9
— 18

= 7 + 9 + 9 + 18 = 43

Overall cost = 43

2
5 — 2 + 5 = 7
2
6 — 2 + 6 = 8
3 — 8 + 3 = 11
— 7 + 11 = 18

Overall cost = 7 + 8 + 11 + 18 = 44



Overall cost = 40

Analogy Lets say $x < y < z$

1.
$x + y$
$+$
$\underbrace{x + y + z}$

2
$y + z$
$+$
$\underbrace{y + z + x}$

3
$x + z$
$+$
$\underbrace{x + z + y}$

Always connect min length ropes first

Brute force idea Sorting.

Ex: 2 5 2 6 3

$\downarrow$ sort

$\underbrace{2 \quad 2}_{4}$ 3 5 6

cost = 4 + 7 + 11 + 18

Step1:

4 3 5 6

$\downarrow$ sort

$\underbrace{3 \quad 4}_{7}$ 5 6

Step 2

7 5 6

$\downarrow$ sort

$\underbrace{5 \quad 6}_{11}$ 7

Step3

11 7

$\downarrow$ sort

$\underbrace{7 \quad 11}_{18}$ stop

Insertion sort
$\uparrow$
TC: $O(n\log n * n) \simeq O(n^2)$

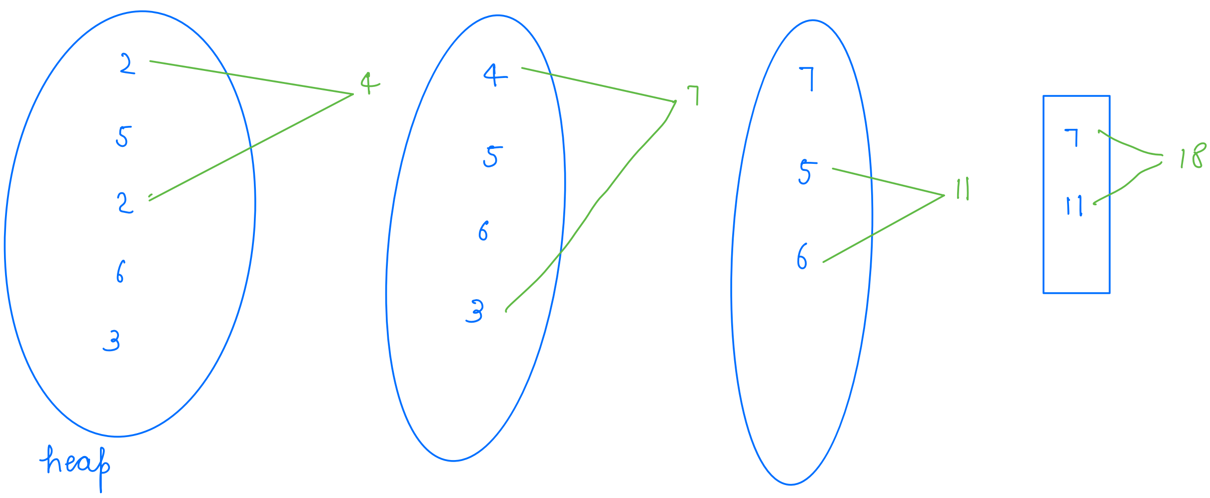
Improved Idea

Data structure $\longrightarrow$ insert() — log(n)
[Heap]

getMin() — O(1)

removeMin() — log(n)

arr[] = [2 5 2 6 3]



TC: n log n.

Heap Data Structure [Binary Heap]

1) **Structure:** Complete binary tree

All levels are completely filled except for last level and nodes should be in L-R order

2) **Types of heaps**

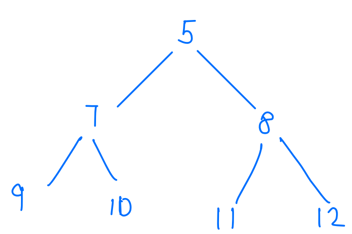
- Min heap — GetMin() – O(1)
- Max heap — GetMax() – O(1)

3) **Heap order property** Parent has higher priority than its children

Min heap:- Parent must always be smaller than its children.

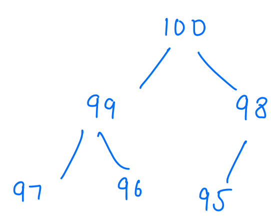
Example:

Min heap



No relation b/w left and right child

Max heap



Implementation and visualization

array

Actual heap

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

0th idx — $lc = 2*0+1 = 1$

$rc = 2*0+2 = 2$

4th idx — $lc = 2*4+1 = 9$

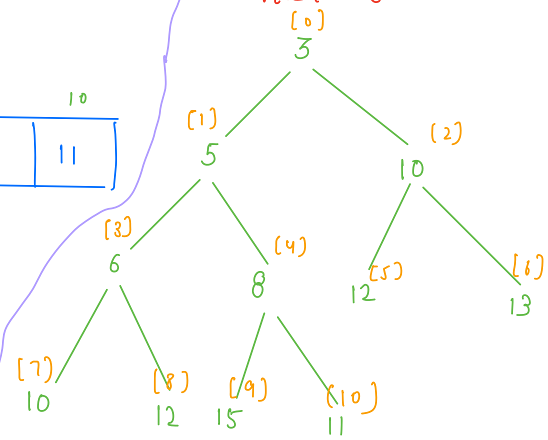
$rc = 2*4+2 = 10$

ith idx — $lc = 2i+1$

$rc = 2i+2$

$$\text{parent} = \left(\frac{i-1}{2}\right) \text{idx}$$

visualization



0th idx — $lc = 1$

$rc = 2$

4th idx — $lc = 9$

$rc = 10$

Insertion = up heapify

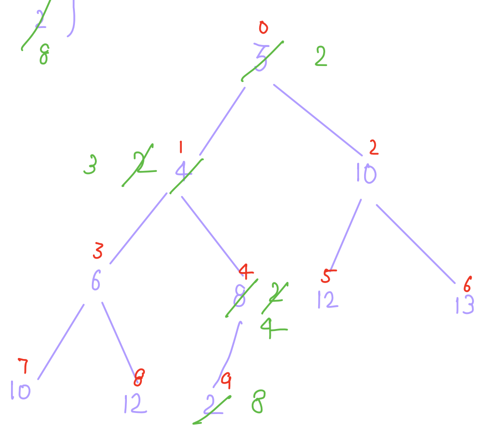
A[] = {	0	1	2	3	4	5	6	7	8	}
	3	4	10	6	8	12	13	10	12	

insert(2)

Step1:

A[] = {	0	1	2	3	4	5	6	7	8	9 }
	$\cancel{3}$ 2	$\cancel{4}$ $\cancel{2}$ 3	10	6	$\cancel{8}$ $\cancel{2}$ 4	12	13	10	12	$\cancel{2}$ 8

Visualisation



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

A[4] = 8 swap

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

A[1] = 4 swap

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

A[0] = 3 swap

TC: O(log n)

↑ height of complete binary tree.

* getMin() — TC: O(1)

return 0 th idx of the array

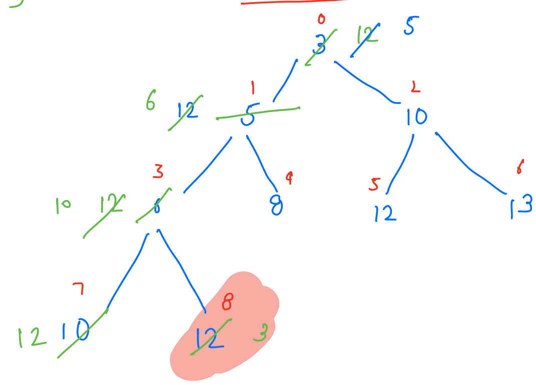
Delete min() = Down heapify

arr[] =

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

Step1: swap 0th idx with last idx.
Delete the last idx
O(1)

visualisation



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]

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 → A[8] = Invalid idx

swap(A, 3, 7)

Stop right here

TC: O(log n)
↓
height of complete B.T.

Break: 8:13 – 8:25

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

	0	1	2	3	4	5	6	7	8	9	10
arr[] =	7	3	5	1	6	8	10	2	13	4	-2

Brute force: heap[]

↓

call insert() function

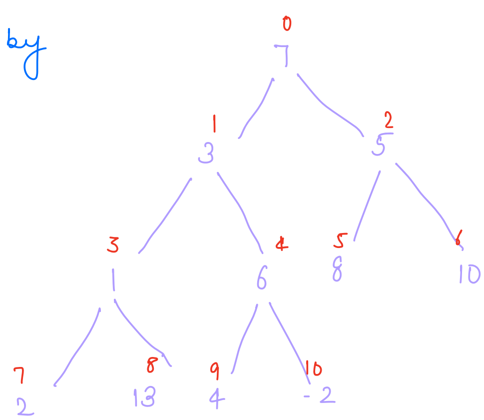
TC: O(n logn)

SC: O(n)

Inplace heap build SC: O(1)

	0	1	2	3	4	5	6	7	8	9	10
arr[] =	7	3	5	1	6	8	10	2	13	4	-2

1) Leaf nodes: Heap order property is already satisfied by leaf nodes



2) Heap order property is missing from non-leaf nodes

first non-leaf node:- Parent of last idx

↓

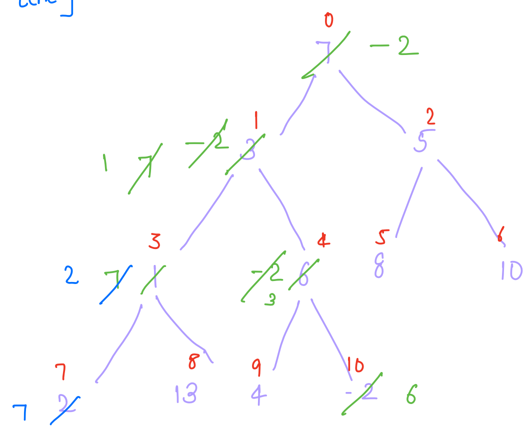
$$\frac{\text{last idx} - 1}{2} = \frac{n-1-1}{2} = \frac{n-2}{2}$$

Last non-leaf node:- 0th idx.

arr[] =

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

Dry run: Non leaf nodes [4th — 0th idx]



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

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

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

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)

5) 0 th idx

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

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

swap(A, 0, 1)

1st idx ÷   A[1] = 7

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

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

swap(A, 1, 3)

3rd idx   A[3] = 7

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

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

swap(A, 3, 7)

Logic:

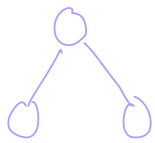
```
for (i = (n-2)/2 ; i >= 0; i--) {

    downHeapify( arr, i);

}
```

TC: O(n) [swaps] SC: O(1)

Amazon.

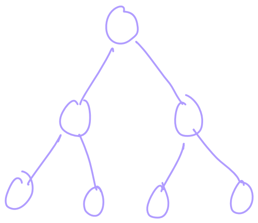
Proof: Complete binary tree = n nodes

Last level $= \frac{n+1}{2}$ nodes $\simeq \frac{n}{2}$.



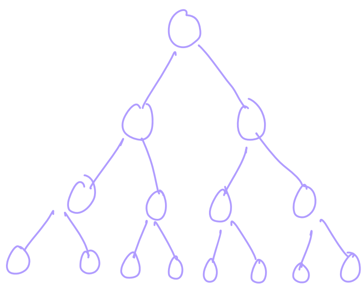
Nodes = 3

Last level = 2



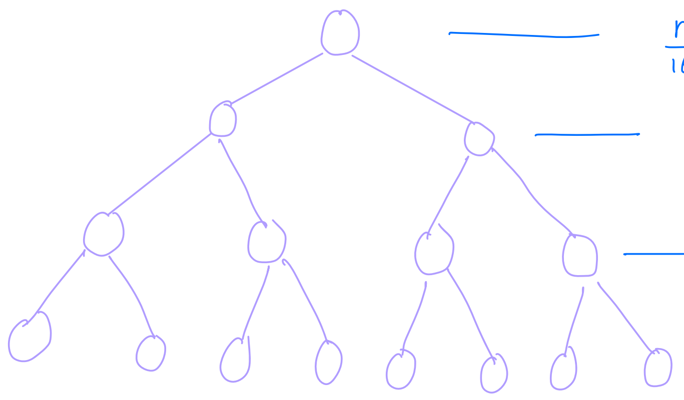
Nodes = 7

LL = 4



Nodes = 15

LL = 8



$\frac{n}{16}$ 3

$\frac{n}{8}$ 2

$\frac{n}{4}$ 1

$\frac{n}{2}$ 0

TC: Total swaps

Total swaps: $\left[\frac{n}{2} * 0 + \frac{n}{4} * 1 + \frac{n}{8} * 2 + \frac{n}{16} * 3 \text{ - - - - - -} \right]$

$\frac{n}{2} \left[0 + \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \text{ - - - - - -} \right]$

$\frac{n}{2} \left[\underbrace{\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \text{ - - - - - -}}_{s = 2} \right]$

$s = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} \text{ - - - - - -}$

$\frac{1}{2} s = \quad \frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \frac{4}{32} \text{ - - - -}$

$-$ $\qquad$ $-$

$\frac{s}{2} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} \text{ - - - -}$

[GP] $= \frac{a}{1-r}$

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

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

Total swaps $= \frac{n}{\cancel{2}} * \cancel{2} = n.$

Qu Merge K sorted LL into one sorted LL

ip

1 ⟶ 3 ⟶ 7 ⟶ 12 ⟶ null

2 ⟶ 6 ⟶ 18 ⟶ null

5 ⟶ 10 ⟶ 20 ⟶ null

7 ⟶ 19 ⟶ null

output: 1 ⟶ 2 ⟶ 3 ⟶ 5 ⟶ 6 ⟶ 7 ⟶ 10 ⟶ 12 ⟶ 18 ⟶ 19 ⟶ 20 ⟶ null

Brute force: Lets say k ll of size = n

$$\underbrace{\underbrace{\underbrace{n \quad n}_{2n} \quad n}_{3n} \quad n}_{4n} \quad \cdots \cdots \quad n$$

$$2n + 3n + 4n + 5n \cdots\cdots kn$$

$$n(\underbrace{2+3+4+5 \cdots\cdots k}_{\simeq k^2})$$

TC: $O(\boxed{n \cdot k^2})$

Idea 2: Min heap

1 → 3 → 7 → 12 → null
h1 h1 h1 h1

2 → 6 → 18 → null
h2 h2 h2

5 → 10 → 20 → null
h3 h3 h3

7 → 19 → null
h4 h4

```
Pair {
    int num;
    Node head;
}
```

output:

min heap

~~1, h1~~ ~~3, h1~~ ~~7, h1~~ ~~12, h1~~

~~2, h2~~ ~~6, h2~~ ~~18, h2~~

~~5, h3~~ ~~10, h3~~ ~~20, h3~~

~~7, h4~~ ~~19, h4~~

output: 1 2 3 5 6 7 7 10 12 18 19 20

Algo:

1) Initially insert head of every list in heap
2) Pick min from heap

   Print the min el

   Whichever list el you considered, add its next el in heap

Thankyou :)