Min Heap :

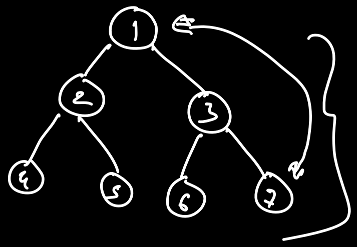
getMin() $\longrightarrow$ O(1)

extractMin() $\longrightarrow$ O(log N)

insertInHeap(x) $\longrightarrow$ O(log N)

buildHeap(A[]) $\longrightarrow$ O(N)

delete(x) $\longrightarrow$ O(N)



Heap

Q Given an array. Find the K smaller elements.

A : 8, 3, 10, 4, 11, 2, 7, 6, 5, 1 K = 4

[1, 2, 3, 4]

* Given array can not be modified

App 1

Sort the array

ret first K elements

TC : O(N log N)

SC : O(N) / O(log N)*

App 2

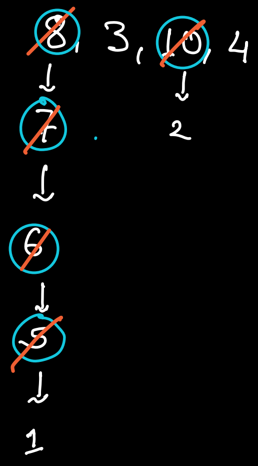
Convert array to min-heap. ⇒ O(N)

Do K extractMin() ⇒ O(K log N)

TC : O(N log N)

SC : O(N)

A : 8, 3, 10, 4, 11, 2, 7, 6, 5, 1



getMax()
extractMax()
Max Heap

* Create a max heap of size K. ⇒ O(K)
* Iterate over remaining N-K elements ⇒ N-K

  if element < heap.top

  { extract Max() O(log K)
  insert (curr element) O(log K) }

$O(K) + O((N-K)\log K)$

K→N/2

$O(N) + O(N\log N)$

Amazon

Given a nearly sorted array. Sort the array.

↳ K-Sorted

every element is atmost K positions away from its sorted position

Given ↓ $K = 3$

$A : \overset{0}{6} \quad \overset{1}{5} \quad \overset{2}{3} \quad \overset{3}{2} \quad \overset{4}{8} \quad \overset{5}{10} \quad \overset{6}{9}$

$\overset{0}{2} \quad \overset{1}{3} \quad \overset{2}{5} \quad \overset{3}{6} \quad \overset{4}{8} \quad \overset{5}{9} \quad 10$

Smallest element ⇒ min of first K+1 elements.

Create a min heap of size K+1 ⇒ $O(K)$

iterate over $(N-K-1)$ elements $O((N-K) \log K)$

& extract min()

add element()

$O(K) + O((N-K) \log K)$

Worst case $K = N/2$

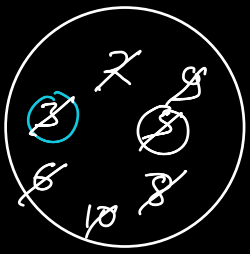
$O(N) + O(N \log N)$

if $K << N$

TC $\approx O(N)$

$\overset{0}{6} \quad \overset{1}{5} \quad \overset{2}{3} \quad \overset{3}{2} \quad \overset{4}{8} \quad \overset{5}{10} \quad \overset{6}{9}$

$K = 3$



2, 3, 5, 6, 8, 9, 10

Q Given an array. Sort it in-place in ASC order
↳ No extra space
SC : O(1)

Insertion Sort
Bubble sort
Selection Sort

SC : O(1)
TC : $O(N^2)$

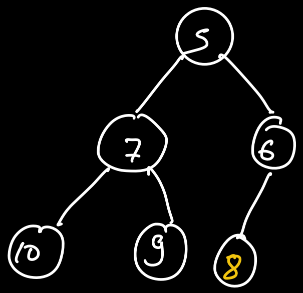
0	1	2	3	4	5	6	7	8
8,	3,	7,	6,	1,	5,	10,	4,	9

Selection Sort

* get min → O(N) $\xrightarrow{\text{Min Heap}}$ O(log N)

* Put in correct position

							Size ↓	
0	1	2	3	4	5	6	7	
3	4	6	5	9	8	7	10	1



↓ extract Min ( )

						Size ↓		
0	1	2	3	4	5	6	7	
4	5	6	10	9	8	7	3	1

↓ extract Min ( )

					Size ↓			
0	1	2	3	4	5	6	7	
5	7	6	10	9	8	4	3	1

↓ extract Min ( )

⋮

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

To Sort in ASC order

$\longrightarrow$ Use Max heap.

Heap Sort

TC : $\underline{O(N)} + O(N \log N)$

SC : $O(1)$

if N is large

then O(N) affect the execution time.

+

No of Swaps.

```
// buildHeap(A);          ---> Build a max heap
for (i=0; i<N; i++){
      {extract Max();
}                                 //
```

Q Given a stream of integers. Find the median until every new insertion.

Amazon
Google
Facebook
MS
Adobe
Uber
Ola
Flipkart
Snapdeal
⋮

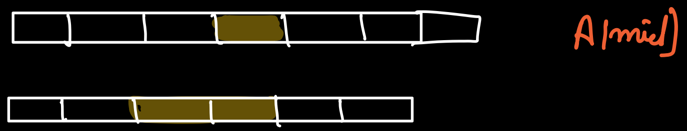
9 ⟶ 9

9, 6 ⇒ 6, 9 ⟶ 7

9, 6, 3 ⇒ 3, 6, 9 ⟶ 6

9, 6, 3, 10 ⇒ 3, $\boxed{6, 9}$, 10 ⟶ 7 (7.5)

9, 6, 3, 10, 4 ⇒ 3, 4, $\boxed{6}$, 9, 10 ⟶ 6



App1

Sort every time you get a new element.

TC : $O(N^2 \log N)$

App2

Insertion sort : Every time, we get a new val insert it to correct position in sorted order.

TC : $O(N^2)$

n/2 smaller no. | n/2 larger no.

0	1	2	3		4	5	6	7
3	7	$\boxed{11}$	5	|	$\boxed{13}$	23	20	17

$3 \quad 5 \quad 7 \quad \boxed{11 \quad 13} \quad 17 \quad 20 \quad 23$

↑ Max Heap

↑ Min Heap

$$|left| == |right|$$

$$med \rightarrow \frac{MaxHeap.top + MinHeap.top}{2}$$

0	1	2	3	4		5	6	7	8
3	7	11	5	$\boxed{13}$	|	23	20	17	21

0	1	2	3		4	5	6	7	8
3	7	11	5	|	$\boxed{13}$	23	20	17	21

$|left| > |right|$

$med \longrightarrow MaxHeap.top$

$|right| > |left|$

$med \rightarrow MinHeap.top$

Stream	Max Heap	Min Heap	Partition		Med
9	9		9		9
9, 6	⑥	⑨	6	9	7
9, 6, 4	⑥ 4	⑨	4, 6	9	6
9, 6, 4, 2	2 ④	9 ⑥	2, 4	9, 6	5
9, 6, 4, 3, 10	2 ④	9 ⑥ 10	2, 4	9, 6, 10	6
9, 6, 4, 2, 10, 12	⑥ 2 4	⑨, 12 10	2, 6, 4	9, 12, 10	7

TC : $O(\log N)$ per insertion

$\Rightarrow O(N \log N)$