sort a given array

min heap → extractMin()

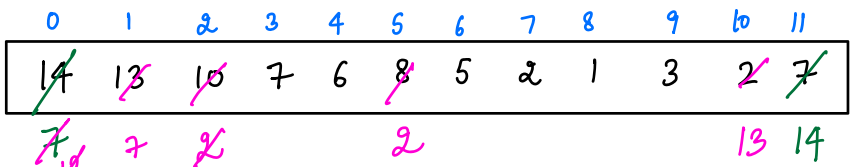
O(N)

NlogN

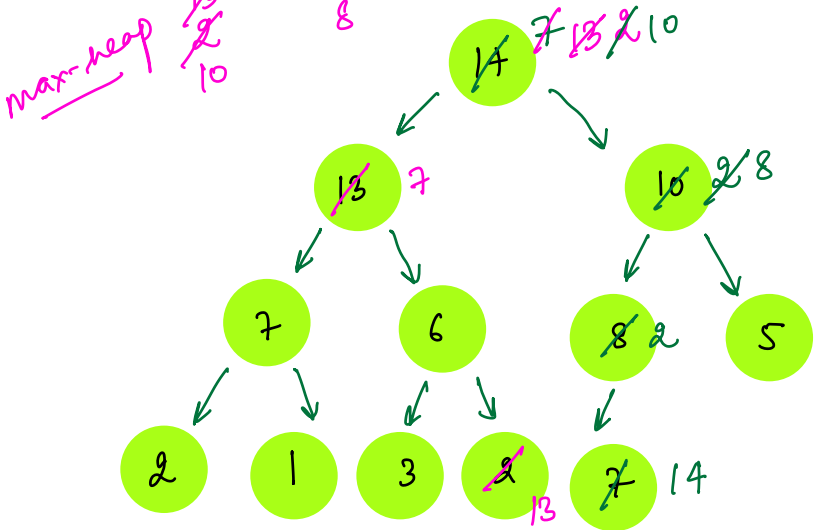
put the min in the new array

S.C: O(N)

max value should come at the last



max-heap



extractMax()

size--;

extract

T.C: ≃ NlogN

S.C: O(1)

Not stable

HeapSort

array → build → maxheap → sorted array

Find K^{th} largest element in an array

8 5 1 2 4 9 7 $K = 3^{rd}$

Heaps ⇒ max heap ⇒ extract max

3 times (K times)

Q K^{th} largest element ∀ (0 - i) windows

$i >= K-1$

[10 18 7 5 16 19 3] $K = 3$

7 7 10 16 16

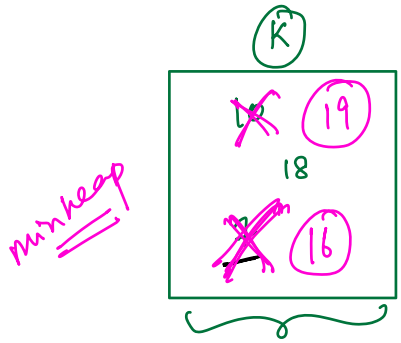
B.F ⇒ 19 18 16 10 7 5 3 T.C: $O(n^2)$ $k \log k$ k^2

5^{th} largest

10 1 9 5 8

↓

min



min = 7

⇒ ~~7~~ 16

always maintain largest K elements

ans = min of min-heap

1) Build minheap for first K elements

2) $\log_2 K$ — $O(K)$ extract next

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

K^{th} smallest → max heap

Nearly sorted array, sort the array

⇓

every element is atmost k position away from it's sorted position

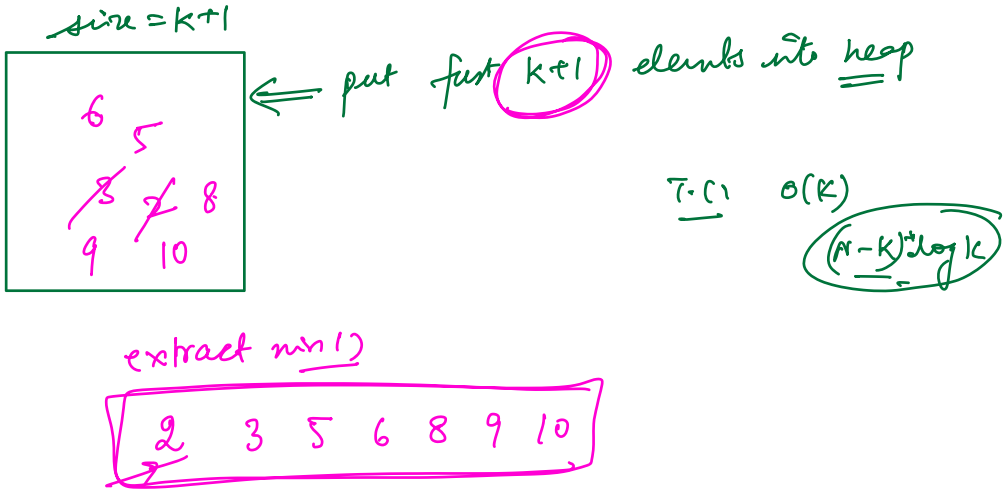
0	1	2	3	4	5	6
6	5	3	2	8	10	9

qi ← K=3

sorted	2	3	5	6	8	9	10

B-f:- 1) sort the data — nlogn

minimum element will dif in first K+1 elements


size = k+1
6
5
8
8
9
10
put first k+1 elemts into heap
T.C: O(K)
(N-K)*log K
extract min ()
2 3 5 6 8 9 10

infinite stream of integers, Find median of the recent set of elements

9 8 7 3 6 4 1 ....

ans: 9 8 8 7 7 6 6 ....

median

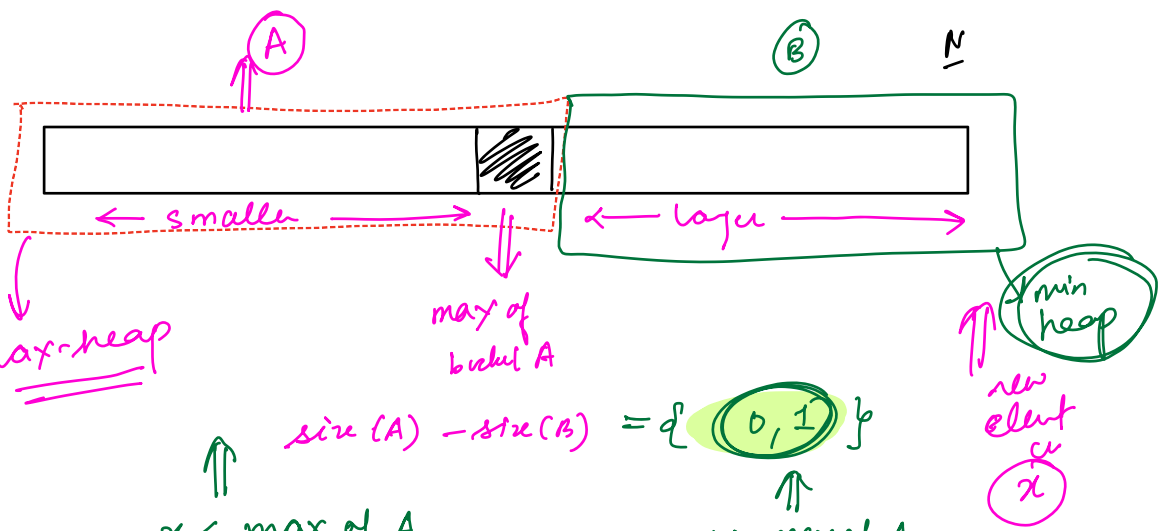
1 (2) 3 odd

1 (3) 5 6 even

$k = \left(\frac{N-1}{2}\right)^{th}$

B.F:-

- sort again & again
- insertion sort $- O(n^2)$



size(A) − size(B) = { 0, 1 }

$x <$ max of A

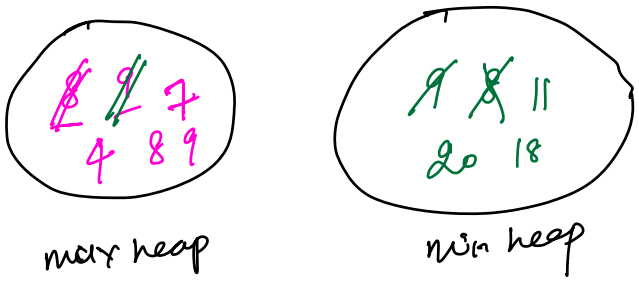
max of A will move to B

if size > 1 do =

$x >$ max of A

shift min of B

9 8 7 4 11 20



9 8 8 7 8

7 8 9

(x)

if(x < max of maxheap)
{
    insert x in maxheap
    if(size(max) - size(min) > 1)
    {   extract max() & insert it
        in min heap()
    }
}
else
{
    insert x in minheap
    if(size(min) > size(max))
    {
        extract min() & insert
        into maxheap()
    }
}
ans. insert(getmax() from maxheap);

priority que ≈ heaps

Doubts

Merge k sorted arrays