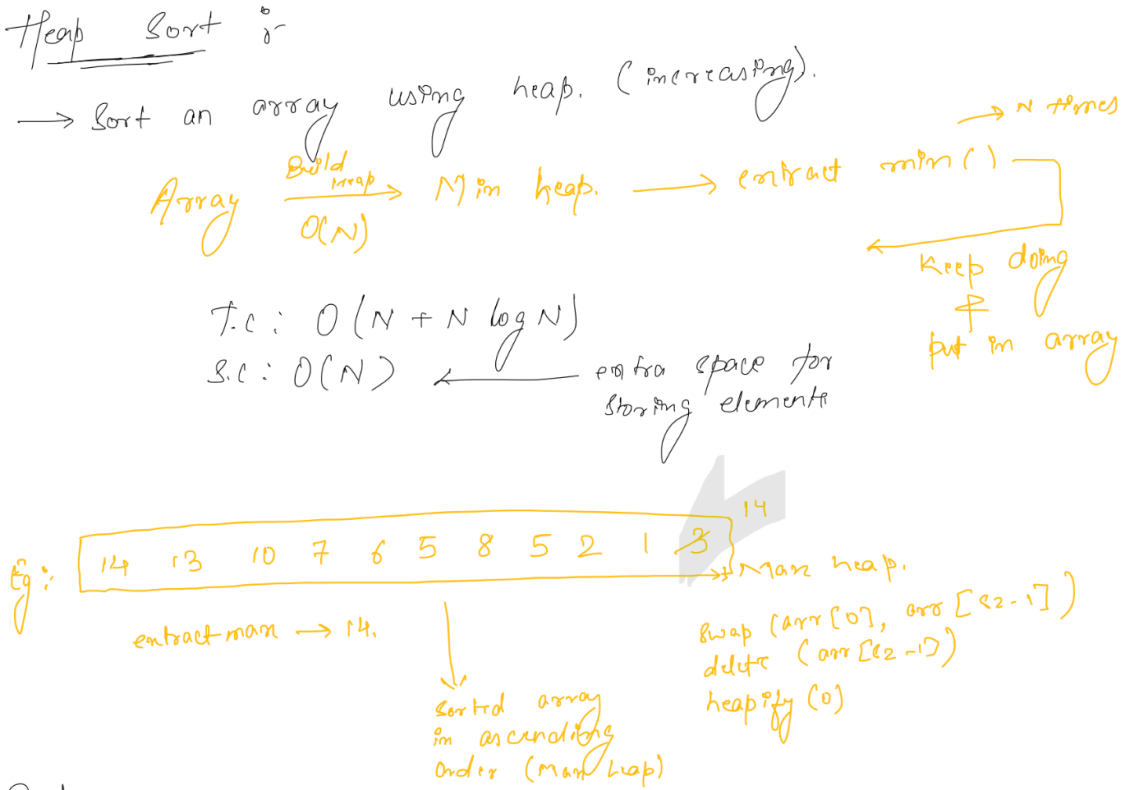
Heaps 2

Heap Sort
Kth Largest Element
Kth Largest Element in every prefix
K sorted Array
Running Median

Heap Sort



Pseudo code

```
1) Build Max heap  O(N)
2) s = N-1
   for (int i = N-1 ; i >= 0 ; i--
   {  swap (arr[0], arr[s2-1]);
      heapify (arr[0]);
   }
```

Kth Largest Element

Q:- Find

8	5	3	1	2	6	9	7

k = 3

ans = 7

Idea I

→ Sort the array in descending order & get k^{th} index.

↳ T.C : $O(N \log N)$

Idea II Use max heap :-

Build max heap of the array } $\log N$

↓

extractMax() k times } $k * \log N$

T.C : $O(N + K \log N)$

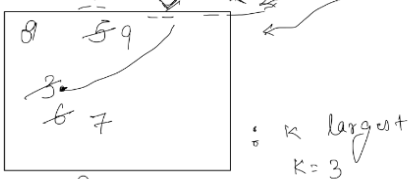
Idea III Min heap :-

8	5	3	1	2	6	9	7

k = 3

ans = 7

Min Heap



: k largest element

k = 3

Size of heap = k = 3.

8	9	7

↳ Min = 7 Ans

Pseudo Code :-

1) Insert first k elements simply into a min heap

```
for (int i=0 ; i<k ; i++)
{   insert (heap, arr[i])
}
```

} $O(k * \log k)$

2)

```
for (int i=k ; i<N ; i++)
{   if (arr[i] > getMin (heap))
    {   extractMin (heap);      log k
        insert (heap, arr[i]);  log k
    }
}
```

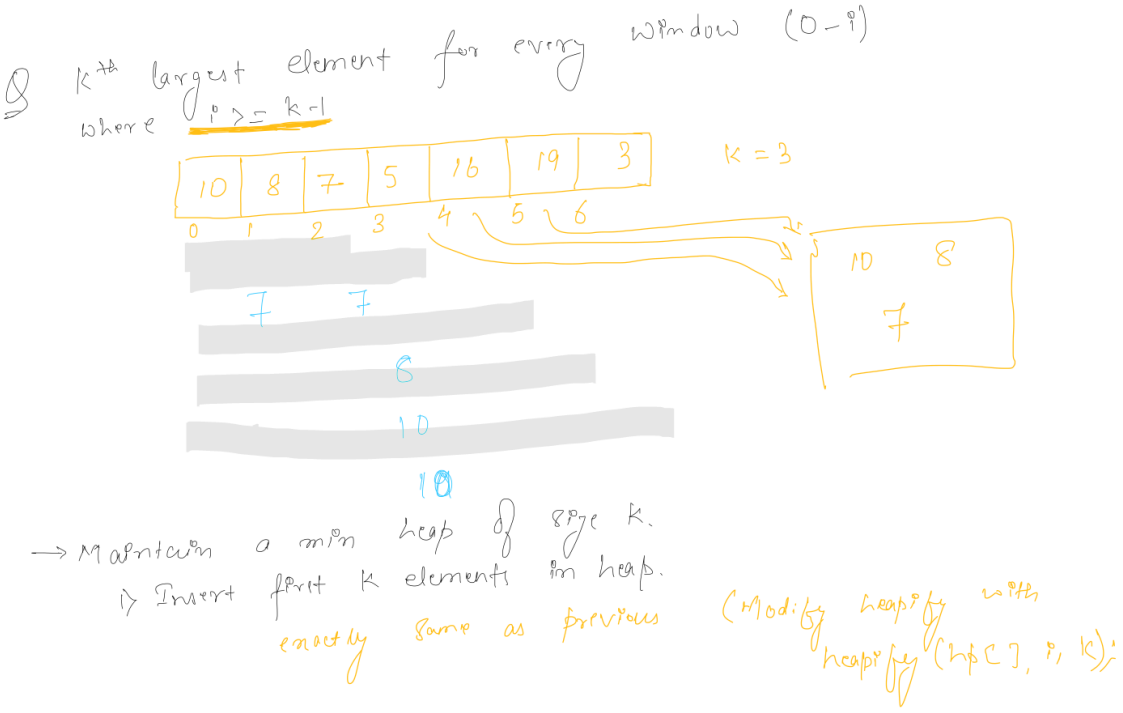
$O((2 \log k) * (N-k))$

3) ans = getMin() ; $O(1)$

T.C : $O(k \log k + (N-k) \log k)$

T.C : $O(N \log k)$

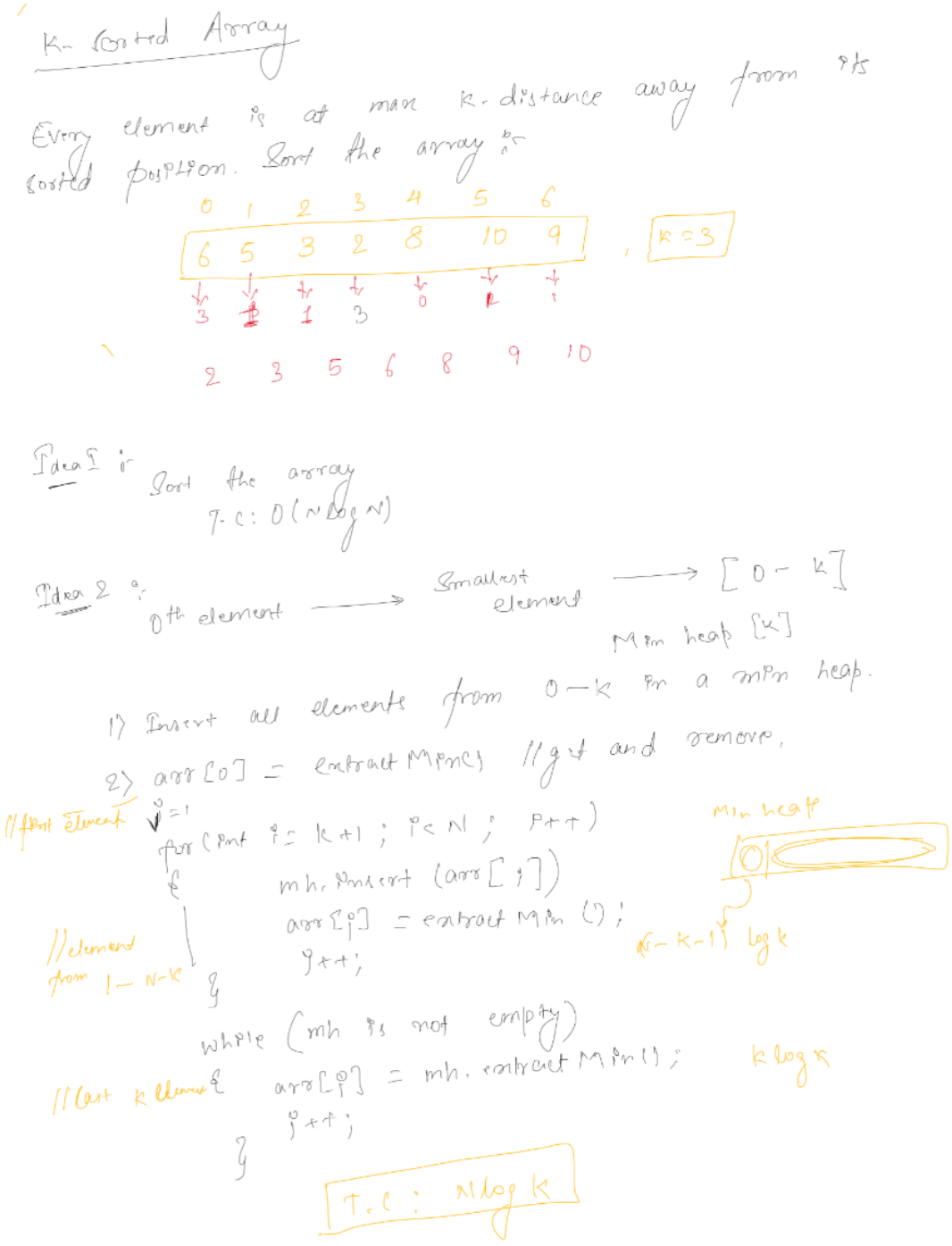
Kth Largest Element in every prefix



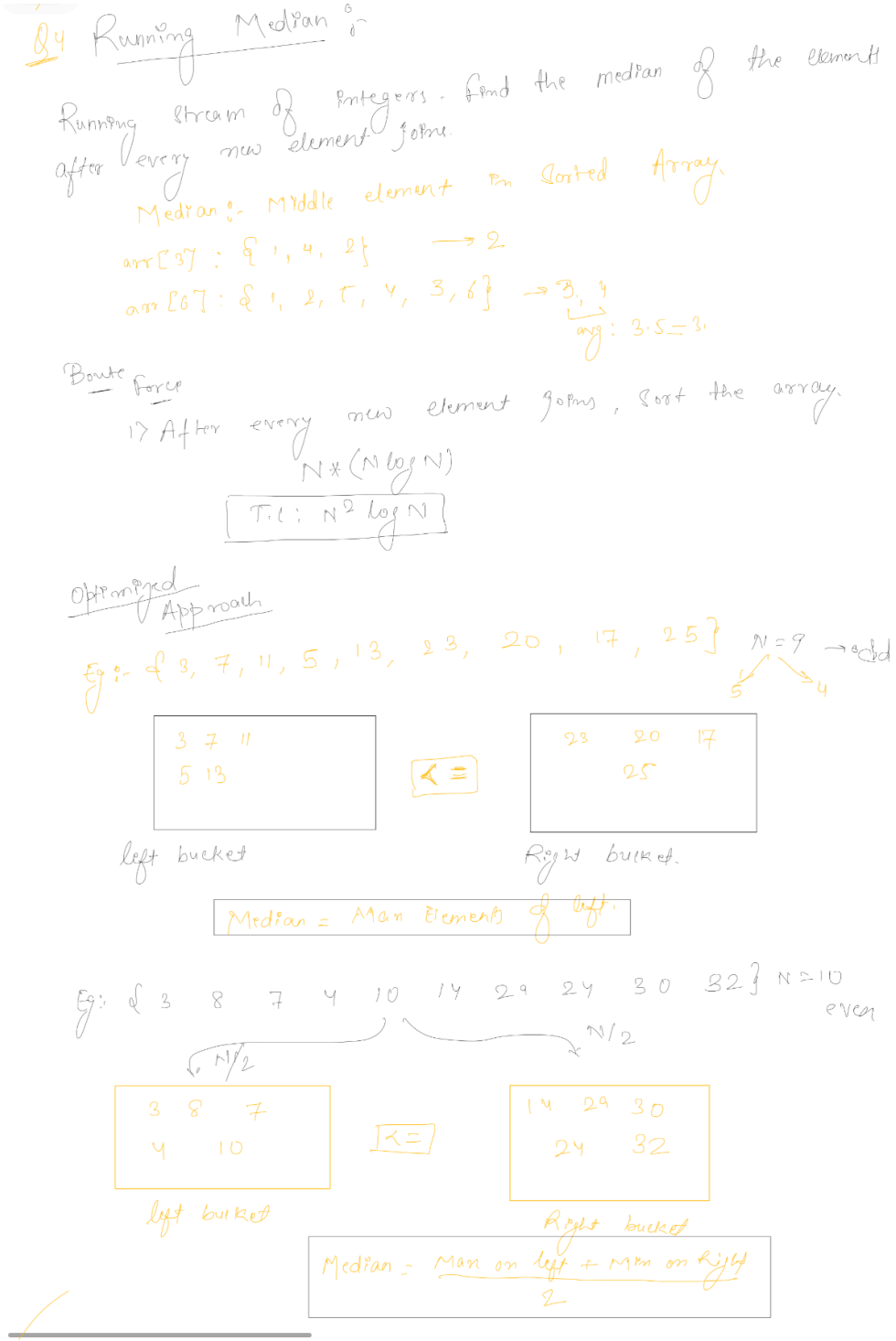
→ Maintain a min heap of size k.
1) Insert first k elements in heap.

exactly same as previous (Modify heapify with heapify(hp[], i, k);

K sorted Array



Running Median



Operation on:-

arr[] : {4, 9, 6, 2, 1, 10, 9, 7, 3, 5}

6 > max of left bucket

Median : 4 6 6 5 10

```
if (left.size() < right.size())
{  left.insert(right.extractMin())
}
if (left.size() >= right.size() + 1)
{
   right.insert(left.extractMax());
}
// Check for even & odd
if (left.size() == right.size())
{  //(left.getMax() + right.getMin()) / 2
}
else {
    // left.getMax();
}
```

```
left.insert(arr[0])
print(arr[0]);
for(i=1; i<N; i++)
  if (arr[i] <= left.getMax())
     left.insert(arr[i]);
  else
     right.insert(arr[i]);
```

for multiple streams that keeps coming :)

T.C: N log N

Use multiple Priority Queue