Todays Content : 40

1. k^{th} index ele in unsorted distinct array
2. k^{th} index ele in unsorted array
3. k^{th} index ele in 2 sorted arrays A[] & B[]
4. k^{th} index ele in N sorted arrays

1Q. Given unsorted arr[] of N distinct elements

Find k^{th} index pos in it's sorted form of arr[].

Note: We cannot modify arr[] & we cannot use extra space

	0	1	2	3	4
Ex1: ar[5] =	2	8	3	11	14

k=2 : ans = 8

	0	1	2	3	4	5	6	7	8
Ex2: ar[9] =	11	24	18	3	5	27	34	9	40

k=6 : ans = 27

Hint:

{0 1 2 .. k-1 k}

$\boxed{\text{No: of element} < ar[i] = k, \text{ for element at } k^{th} \text{ Index}}$

Dry Run:

	0	1	2	3	4	5	6	7	8
ar[9] =	11	24	18	3	5	27	34	9	40
k=6	✗	✗	✗	✗	✗	✓			
#count less =	3	5	4	0	1	6			

0 1 2 .. 5   6

{6 elements} $\boxed{27}$

#Idea1: For every ar[i] {
    Iterate on arr[] & calculate count of ele < ar[i] = c.
    if(c==k){
        return ar[i];
    }
}

TC: O(N*N) = O(N²)   SC: O(1)

Idea 2: Using Binary Search

Target: k^{th} index element in sorted form of arr[]

Search space: In arr[]

Discard:

$k = 6$

	l								h
	0	1	2	3	4	5	6	7	8
arr[9] =	{ 11	24	18	3	$\boxed{5}$	27	34	9	40 }

Dry Run:

l	h	m
0	8	4

4: # For arr[m] = arr[4] Iterate & calculate ele < arr[4]

count = 1; # no: of ele < arr[4].

count < k; # Means, the count value; By the mid value, It will also the, no: of ele less than that.

Issue: We cannot decide which side to go, because going to left or right can the value, means we cannot discard search space, hence we cannot apply BS on above search space.

Hint2: Change search span

Target: k^{th} index element in sorted form of arr[]

Search span: lo = min(arr[])   hi = max(arr[])

#It's a gurantee for every input, ans lies in Search span

Discard:

	0	1	2	3	4	5
ar[6] =	{ 4	1	5	15	6	2 }

k = 3

l	h	m	#Count of ele < m
1	15	8	#eles < 8 = 5 > 3, h = m-1;

$\{ a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad \boxed{8} \quad \ldots\ldots \}$

l   h m

l	h	m	#Count of ele < m
1	7	4	#eles < 4 = 2 < 3, l = m+1

$\{ a_0 \quad a_1 \quad \boxed{4} \quad a_2 \quad a_3 \quad a_4 \}$

l	h	m	#Count of ele < m
5	7	6	#eles < 6 = 4 > 3 h = m-1

$\{ a_0 \quad a_1 \quad a_2 \quad a_3 \quad \boxed{6} \quad \ldots\ldots \}$

l	h	m	#Count of ele < m
5	5	5	#eles < 5 = 3 = 3

return m; 5

Edge Case:

ar[10] = { 11 24 30 3 5 27 34 9 40 }

k = 4

l	h	m	
3	40	21	# count of ele ≤ m # eles ≤ 21 = 4 = 4 ans = 21, l = m+1

1st 4th 5th 6th 3rd 2nd

{ a_0 a_1 a_2 a_3 | $\boxed{20}$ $\boxed{21}$ $\boxed{22}$ $\boxed{a_4}$ | 25 a_5 a_6 a_7 a_8 a_9 }

eles < 4 | 4 4 4 4 | # eles > 4

# goto right
l = m+1

update ans
& look for
better in right
ans = m
l = m+1

# goto left
h = m-1

21	40	30	# eles ≤ 30 = 5 > 4, h = m-1
21	29	25	# eles ≤ 25 = 5 > 4, h = m-1
21	24	22	# eles ≤ 22 = 4 ans = 22, l = m+1;
23	24	23	# eles ≤ 23 = 4 ans = 23, l = m+1;
24	24	24	# eles ≤ 24 = 4 ans = 24, l = m+1

25 > 24 # stop & return ans = 24.

TC: $O(N * \log_2(h-l+1))$ → # h-l+1, Binary Search Search space size.

→ # for each iteration, calculate no. of ele less than itself.

```
int k^th Index(vector<int> &arr, int k){

}
```

ar[10] = { 11 24 30 3 5 27 34 9 40 }

k = 4

l h m #Count of ele < m

Q2. Given unsorted array Find k^{th} order element

	0	1	2	3	4	5	6	7
Ex: ar[8] =	{15	4	15	10	16	19	10	15}

k = 4

Search Span {4 19} : k = 4

4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

#cle k

	0	1	2	3	4	5	6	7
Ex: ar[8] =	{15	4	15	10	16	19	10	15}

k = 4 :

Dry Run:

l	h	m

#Count of ele ≤ m

28. Given 2 sorted arrays, find the k^{th} index in overall sorted data

$A[N]$ $B[M]$

Ex: $k=8$

$$\begin{array}{lcccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A[8] = \{ & \cancel{3} & \cancel{3} & \cancel{6} & \cancel{7} & \cancel{7} & 11 & 14 & 17 \} \end{array}$$

ans = 10

P_1

$c = \cancel{1}\ \cancel{2}\ \cancel{3}\ \cancel{4}\ \cancel{5}\ \cancel{6}\ \cancel{7}\ 8^{th}$ ele: We are going to remove = 10.

$B[7] = \{ \cancel{2}\ \cancel{2}\ 10\ 10\ 13\ 20\ 20 \}$

P_2

Idea1: $P_1 = 0$, $P_2 = 0$

Which ever ele is smaller: inc that pointer & count.

if $C == K$: { Min of $A[P_1]$ & $B[P_2]$ is ans }

TC: $O(N+M)$ SC: $O(1)$

Idea2: Target: k^{th} index element

Search Space: { l: $\min(A[0], B[0])$ $h = \max(A[N-1], B[M-1])$ }

$k=8$

$$\begin{array}{lcccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A[8] = \{ & 3 & 3 & 6 & 7 & 7 & 11 & 14 & 17 \} \end{array}$$

$B[7] = \{ 2\ 2\ 10\ 10\ 13\ 20\ 20 \}$

l	h	m	# count of ele < m in A+B, k = 8
2	20	11	# eles < 11 : 9 > 8 : h = m-1;
2	10	6	# eles < 6 : 4 <= 8 : ans = 6; l = m+1;
7	10	8	# eles < 8 : 7 <= 8 : ans = 8; l = m+1;
9	10	9	# eles < 9 : 7 <= 8 : ans = 9; l = m+1;
10	10	10	# eles < 10 : 7 <= 8 : ans = 10; l = m+1;

11 > 10 : Stop & return ans = 10.

Given sorted array return count of elements < k.

```
int countless(int[] ar, int k){
    int N = ar.length, ans = -1;
    int l=0, h = N-1;
    while( l<=h){
        int m= (l+h)/2;
        if( ar[m] < k){ ans = m; l = m+1; }
        else { h = m-1;
    }
    return ans+1;
}
int kthSmallest(int[] A, int[] B, int k){
    int N= A.length, M= B.length;
    int l= Math.min(A[0], B[0]);
    int h= Math.max( A[N-1], B[M-1]);
    while( l<=h){
        int m= (l+h)/2;
        // Count no: of elements < m in A[] & B[];
        int c=0;
        c= c+ countless(A, m);  } -> logN
        c= c+ countless(B, m);  } -> logM
        if( c <= k){ //
            ans = m; l= m+1;
        }
        else{ // c > k
            h=m-1;
        }
    }
    return ans;
}
```

Search Space = $\{h - l + 1\}$

BS iterat = $\log_2^{h-l+1}$

$$\boxed{\begin{array}{l}\text{Total TC} = \log_2^{h-l+1} * \{\log N + \log M\} \\ SC = O(1)\end{array}}$$

Q: Calculate median / k^{th} index for 2 sorted arrays

TC: $\log_2^{N+M}$: TODO

Q: Given mat[N][M] every row sorted, find k^{th} index element in Overall Sorted data.

```
int countless(int[] ar, int k){
    int N = ar.length, ans = -1,
    int l=0, h = N-1;
    while(l<=h){
        int m= (l+h)/2;
        if(ar[m] < k){ ans = m; l = m+1; }
    } else { h = m-1;
    return ans+1;
}
```

Ex: mat[3][4] = $\begin{bmatrix} 0 & 3 & 6 & 8 \\ -2 & 1 & 4 & 11 \\ 2 & 3 & 4 & 6 \end{bmatrix}$

k=6

```
int kthIndex(int[][] mat, int k){        N*M
    int N= mat.length, M = mat[0].length;
    int l = min 0th col , h= Max last col;
    while(l<=h){
        int m= (l+h)/2;
        // Count no: of elements < m in each row in mat[]
        int c=0;
        for(int i=0; i<N; i++){        → NlogM
            c= c+ countless(mat[i], m); }   → logM
        }
        if(c<=k){ //
            ans= m; l= m+1;
        }
        else{ // c>k
            h= m-1;
        }
    }
    return ans;
}
```

Search Spa = {h - l + 1}

BS iterat = $\log_2^{h-l+1}$

$TC = O(\log_2^{h-l+1}) * N\log M + 2N$

min & max

$SC = O(1)$