Searching 3
Friday, 12 March 2021 7:13 PM

1Q) Given 2 sorted arrays of size of N & M
find the median, of data if merged.
⇒ N+M is odd, [All Elements are distinct]
↳ VIMP condition

A[N] : 2 6 11 19 25 30
B[M] : 8 9 15 29 40

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C[N+M] : | 2 | 6 | 8 | 9 | 11 | 15 | 19 | 25 | 29 | 30 | 40 |

⇒ Median element index = (N+M)/2

Sol (Merge 2 Arrays & get Middle)
TC: O(N+M) SC: O(N+M)

Hint for SC:
While merging every time we move a pointer either P1 & P2 increase count by 1

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

| | P1 | P2 |
|---|---|---|
| c=0 | | |
| c=1, | 1 | 0 |
| c=2, | 2, | 0 |
| c=3, | 2, | 1 |
| c=4, | 2, | 2 |
| c=5, | 3, | 2 |

c = (N+M)/2

TPII now we neglected 5 value, Can I say next element I want to insert is my ans

⇒ median = min(A[P1], B[P2])

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C[N+M] = | 2 | 6 | 8 | 9 | 11 | 12 13 14 15 | 19 | 25 | 29 | 30 | 40 |

ans=13 [Actually ans=15]
#ele = 5

Hint1:
an element is said to be median if no: of ele less than that = 5 ⇒ [(N+M)/2]
In above Case

TC: Given lo & hi
Iterations of BS.
$\log_2(hi - lo)$

Overall TC of above code
(BS Iterations) * [ ]
$\log_2(hi - lo) * [\log(M) + \log(N)]$
TC:

| min(A[0], B[0]) lo | max(A[N-1], B[N-1]) hi | mid |
|---|---|---|
| 2 | 40 | 21<br>#Elements < 21<br>7 > 5 [Not mid]<br>// we need to goto left<br>high = mid-1 |
| 2 | 20 | 11<br>#Element < 11<br>4 < 5 [Not mid]<br>// we need to goto right<br>low = mid+1 |
| 12 | 20 | 16<br># Elements < 16<br>6 > 5 [Not mid]<br>high = mid-1 |
| 12 | 15 | 13<br># Elements < 13<br>5 == 5<br>ans = 13;<br>// look for a better ans on right side |
| 14 | 15 | 14<br># Elements < 14<br>5 == 5<br>ans = 14;<br>goto right |
| 15 | 15 | 15<br># Element < 15<br>5 == 5<br>ans = 15;<br>goto right; |
| 16 ⟷ | 15 | ⇒ ans = 15 |

[We stop]

⇒ Apply BS in both A[N] & B[M] & get no: of elements < 21, & add them.
TC: logN + logM

int Kthpos (int A[], int B[], int M, int N, int k){
int lo = min(A[0], B[0]), hi = max(A[N-1], B[M-1])
int ans = lo;
while ( lo ≤ hi ) {
int mid = (lo+hi)/2
// How many elements are than mid in both arrays combined
[Can we implement Using BS]
int c = countLen(A[], N, mid) → log(N)
+ countLen(B[], M, mid) → log(M)
In a given array it will simply calculate no: of elements < than given element, Implement on own
if(c < k__ ){
// go to right
lo = mid+1
}
else if(c > k__ ){
// go to left
high = mid-1
}
else{
ans = mid;
low = mid+1
} c == (k)
update ans, look for a Better ans
It is on right side
}
return ans;
}

// We can get $k^{th}$ pos element in merged data using above function

Note: In beside logic will work even if (N+M) is even you can get any position you want.

If N+M is even // How many medians are there?

Median pos1 = (N+M)/2 - 1 : $\left(Kth\,pos(A, B, M, N, (N+M)/2 - 1) + Kth\,pos(A, B, M, N, (N+M)/2)\right)/2$ Median

Median pos2 = (N+M)/2 ;

2Q) Given 2 sorted arrays A[] & B[]
find $k^{th}$ pos of arrays when merged.
→ Elements can repeat as well

A[] : 2 4 4 6 8 8
B[] : 3 4 5 5 6 7

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C[] : | 2 | 3 | 4 | 4 | 4 | 5 | 5 | 6 | 6 | 7 | 8 | 8 |

Calculating l & r

How many element < 5 = 5
How many element = 5 = 2
stfind = 5 endfind = 6

How many element < 4 = 2
How many elmt = 4 = 3
stfind = 2 endfind = 4

How many element < mid = $C_1$
How many element = mid = $C_2$
$l = C_1$ $r = C_1 + C_2 - 1$
stfind endfind

| val | stfind | endfind | |
|---|---|---|---|
| 4 | 2 | 4 | // Is this element present at 3<br>YES 3 inclusive [2 4] |
| 5 | 5 | 6 | // Is this element present at 8<br>NO 6 < 8 ⇒ [end] |
| 6 | 7 | 8 | // Is this element present at 5<br>NO 5 < 7 [start] |

| val | l | r | |
|---|---|---|---|
| | | | // Is this element present at k<br>If k in between [l r] val is my ans |

// When can we say val is coming before $k^{th}$ index
[l r] < $k^{th}$ (val)
if r < k ⇒ we should inc
⇒ goto right
⇒ [lo = mid+1]

// When can we say val is coming after $k^{th}$ index
$k^{th}$ < [l r] (val)
if l > k ⇒ we should dec
⇒ goto left
⇒ [hi = mid-1]

// Assume k = 1
val = 5

$k^{th}$ / 1
st ↔ ed
5 6
val = 5

int Kthpos (int A[], int B[], int N, int M, int k){
int lo = min(A[0], B[0]), high = max(A[N-1], B[M-1]), ans = A[0]
while ( lo <= hi ){
int mid = (lo+hi)/2
// for given mid we have to find l & r
int $C_1$ = countLess(A, N, mid) + countLess(B, M, mid) + __
int $C_2$ = countEqual(A, N, mid) + countEqual(B, M, mid) + __
int l = $C_1$, r = $C_1 + C_2 - 1$
// Say we have l & r
if( l > k){
// k [l r] mid
// decrease mid;
high = mid-1;
}
else if(r < k){
[l r] k
// increase mid
low = mid+1
}
else{
return mid;
}
}
// overall TC for X arrays
$\log_2(hi - lo) * [X \log N]$

// Extend same logic N rows 2D Matrix
If we have x-arrays, mat[X][N]
first calculate $C_1, C_2$ for mid & get l & r
int $C_1 = 0$, $C_2 = 0$
for(int i = 0; i < X; i++){
$C_1 = C_1$ + countLess(mat[i], N, mid)
$C_2 = C_2$ + countEqual(mat[i], N, mid)
}
l = $C_1$ & r = $C_1 + C_2 - 1$
TC: X logN
X arrays

⇒ Given a 2D matrix, where every row is sorted, you are given random [k] positions. [find sum of data points in these k positions in merged N arrays data]

mat[ 3 3 4 5
6 8 8 10
1 1 2 3
9 9 10 10 ]

k = 3 = { 1, 4, 5 }
{ Calculate sum of all These k positions After we merge data }