Approach 2

A_{mid}

A | A_{left} | A_{right} | length = m

B_{mid}

B | B_{left} | B_{right} | length = n

if $A_{mid} < B_{mid}$

$$\left.\begin{array}{r} A_{left} < A_{right} \\ A_{left} < A_{mid} < B_{mid} < B_{right} \end{array}\right\}$$

A_{left} is not going to mix with ($A_{right} \cup B_{right}$)

same, B_{right} & ($A_{left} \cup B_{left}$)

if $A_{mid} > B_{mid}$

$$\left.\begin{array}{r} B_{left} < B_{right} \\ B_{left} < B_{mid} < A_{mid} < A_{right} \end{array}\right\}$$

B_{left} is not going to mix with ($B_{right} \cup A_{right}$)

Same, A_{right} & ($A_{left} \cup B_{left}$)

---

if $(m+n)$ is odd $K = \frac{m+n+1}{2}$ [median is K^{th} largest element]

if $(m+n)$ is even $K = \frac{m+n}{2}$ [median is $\frac{K^{th} \text{ largest element} + \text{next ele}}{2}$

if $K > (m+n)/2$

$A_{mid} > B_{mid}$

we can prune A_{left} or B_{left} because

$A_{mid} < B_{mid}$

their elements stay in the left half of the sorted list

$K = K -$ no. of elements in A_{left}

if $K \leq (m+n)/2$

$A_{mid} < B_{mid}$

we can prune A_{right} or B_{right} because

$A_{mid} > B_{mid}$

their elements stay in the right half of the sorted list

No need to upgrade K as we are pruning some elements from the right end of the sorted list

do this repeatedly until one array is empty then K^{th} element in other array is the answer

note: Aleft, Aright both include Amid, when they get pruned

Expt.No.

Approach 2 Example

Page No.

Date.

$\boxed{\cancel{1}\cancel{3}}$ 4 9 $m = 4$ $n = 3$ $(m+n)/2 = 3$

(1) $K = 4$ $\boxed{(m+n+1)/2}$

2 7 8 $4 > 3$ ✓

4 9 $m = 2$ $n = 3$ $(m+n)/2 = 2$

(2) 2 $\boxed{\cancel{7}\cancel{8}}$ $K = 2$ $[K - 2]$

$2 > 2$ ✗

4 9 $m = 2$ $n = 1$ $(m+n)/2 = 1$

(3) $\boxed{\cancel{2}}$ $K = 2$

$2 > 1$ ✓

$m = 2$ $n = 0$

(4) 4 9 $K = 1$ $[K - 1]$

K^{th} element i.e., first element

hence our median = 4

Expt.No.

Page No.

Date.

time complexity

$O(\log mn)$

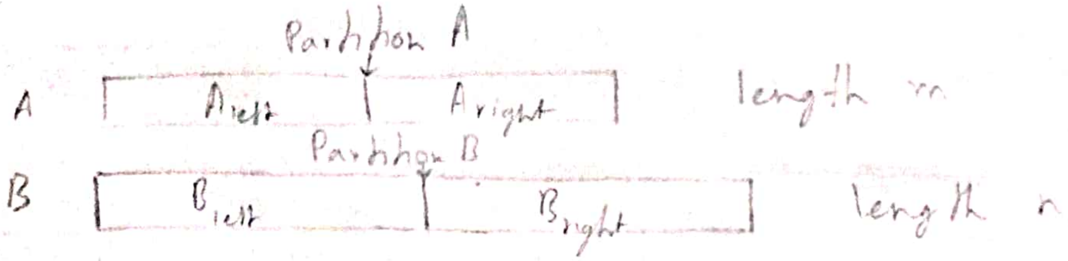
$\log m + \log n$

for the array of length m

for the array of length n

both of the array are splitted from the middle

Approach 3



We need to find those partitions such that,

$A_{left} + B_{left} \longrightarrow$ makes the smaller half of the sorted array

$A_{right} + B_{right} \longrightarrow$ makes the larger half of the sorted array.

- To get these partitions we find Partition A then

$$\text{Partition B} = \frac{m+n+1}{2} - \text{Partition A}$$

[$\because$ smaller half, made of $A_{left} + B_{left}$ will have $\frac{m+n+1}{2}$ elements]

$\boxed{\frac{m+n+1}{2} \text{ this 1 ensures inclusion of middle element in smaller half of sorted list is } m+n \text{ is odd}}$

- To get Partition A we use 2 variables left & right

Partition A is $\frac{\text{left} + \text{right}}{2}$

initially, left = 0 right = m [the search space, where we look for Partition]

$\boxed{\text{if it was } m-1 \text{ it would have failed if A.length} = 0}$

calculate Partition A & B if left $\leq$ right

- after Partition A & Partition B is calculated we can say

max_A_{left} = A [Partition A − 1] ($-\infty$ if Partition A = left or 0)

min_A_{right} = A [Partition A] ($+\infty$ if Partition A = m)

max_B_{left} = B [Partition B − 1] ($-\infty$ if Partition B = 0)

min_B_{right} = B [Partition B] ($+\infty$ if Partition B = n) right = $P_A - 1$, left = $P_A + 1$ thus 1 is to make sure that we never get same P_A for twice, thus '1' permanently throws P_A out of range (left → right)

- if $max\_A_{left} > min\_B_{right}$ we need to move Partition A towards left as max_A_{left} is too large to be in the smaller half of the sorted array so we can safely remove the part of A, right to Partition A & upgrade right = Partition A − 1 & return to calculate Partition A & B if left ≤ right (loop condition)
- if $min\_A_{right} < max\_B_{left}$ we need to move Partition A towards right as max_A_{right} is too small to be in the bigger half of the sorted array so we can safely remove the part of A, left to partition A & upgrade left = Partition A + 1 & return to calculate Partition A & B if left ≤ right (loop condition)
- if $max\_A_{left} \leq min\_B_{right}$ && $min\_A_{right} \geq max\_B_{left}$ we have got our partitions,

A_{left}	B_{left}	A_{right}	B_{right}

if m+n is odd max of smaller half will be the median i.e., $\max(max\_A_{left}, max\_B_{left})$

if m+n is even average of max of smaller half & min of larger half will be the median i.e.,

$$\frac{\max(max\_A_{left}, max\_B_{left}) + \min(min\_A_{right}, min\_B_{right})}{2}$$

Expt.No.

Page No.

Date.

$m = 4 \quad n = 5$

1 3 4 5 $\qquad left = 0 \qquad right = 4$

2 7 8 8 9 $\qquad P_A = 2 \qquad P_B = 3$

$min\_B_{left} > max\_A_{right}$

4 5 $\qquad left = P_A + 1 = 3 \qquad right = 4$

2 7 8 8 9 $\qquad P_A = 3 \qquad P_B = 2$

$min\_B_{left} > max\_A_{right}$

5 $+\infty$

2 7 8 8 9 $\qquad left = P_A + 1 = 4 \qquad right = 4$

$P_A = 4 \qquad P_B = 1$

$max\_A_{left} \leq min\_B_{right}$ && $min\_A_{right} \geq max\_B_{left}$

$m + n$ is odd

$max(5, 2) = 5$

median = 5

→ varified

1 2 3 4 5 7 8 8 9

OXFORD®

Teacher's Signature ....

Time complexity $O(\log(\min(m,n)))$ as the array of smaller size is splitted from the middle

Why A should be the smaller list? explained

2 7 8 8 9 $\quad m = 5 \quad n = 4$

1 3 4 5

left = 0 $\quad$ right = 5

$P_A = 2 \quad P_B = 3$

max_Aleft > min_Bright

---

left = 0 $\quad$ right = 1

2 7 8 8 9

1 3 4 5

$P = 0 \quad P_B = 5$

'A' must be the array with smaller size so that P_B cannot become an index way outside the array 'B' like what happened here