Median of 2 sorted Arrays :-

(self revision)

eg:- arr1 = [1, 3, 4, 7, 10, 12]

arr2 = [2, 3, 6, 15]

what we cannot do?

→ we cannot directly apply "BS on real domain"

~~by keeping the check for~~

using the logic :- median is a value such that (#elem ≤ median) == (#elem ≥ median)

bcz

above median is $\frac{4+6}{2} = 5$

& based on above logic we could be getting any real no betn (4, 6)

what to do?

sorted array :- [1 2 3 4 | 6 7 10 12 15]

if arr = [1 2 3 | 4 5]

So to find median ⇒ we need the left & right elem from pt of symmetry (for even len)

& for odd :- left from symmetry.

bcz that is the only way to find median

⇒ Even len

↳ we know both left & right half will have equal no of elem

& that no = $\left(\frac{n1+n2+1}{2}\right)$ or $\left(\frac{n_1+n_2}{2}\right)$

(applies for both even & odd) (L&R) (only in L)

---

* So. we need to put the shortest cut = $\left(\frac{n_1+n_2+1}{2}\right)$ elem on the left side of symmetry.

options are:- (take from beginning)

take 0 from arr1, cut from arr2

1 " , cut-1 "

2 " , cut-2 "

⋮

cut " , 0 "

one of them is a valid configuration

bcz median is a specific value only

& we need greatest from left & ~~largest~~ smallest from right

Options :-

take(# elem from arr1)

= 0 ⟶ min(cnt, arr1.size())

i.e max^m elem we should / can pick from arr1.

Eg:- arr1 = [1, 2, 3]
arr2 = [- - - -]
median = 5th elem

Now, if we do linearly pick up.

0 ⟶ min(cnt, arr1.size())

then T.C = O(N)

So can we do it using B.S ?

general format to B.S on a search space is check[mid] :- _ 000111 _

But here :- (only 1 configurn is valid)

Eg:- arr1 = [1, 3, 4, 7, 10, 12]

arr2 = [2, 3, 6, 15]

$\frac{n_1+n_2+1}{2} = \frac{11}{2} = 5$. ⟶ 5 elem on left

Option 2

~~can't pick~~

5 elems on left.

so for picking from the smaller array

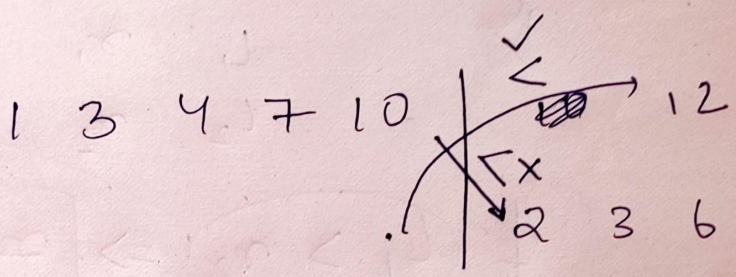
i.e ((0 elem, 1 elem, ... - min(cut, smaller arr. size)) + rest from other array)

to make 'cut' elems on left

options:-

0	1	2	3	4
×	×	✓	×	×

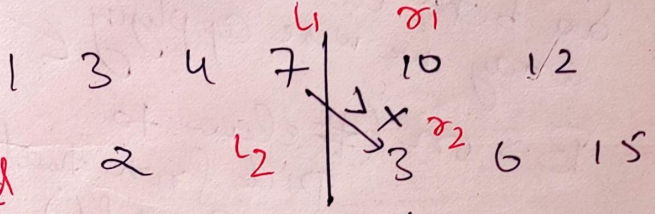
1) (0) from arr 2.



so not valid

---

2) (1)



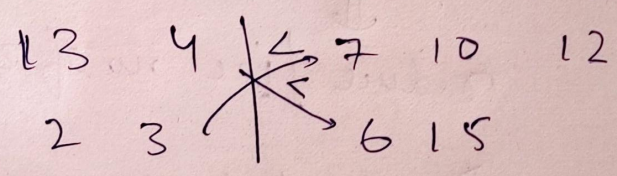
i.e comparing

$L_1 < r_2$ for valid

$L_2 < r_1$

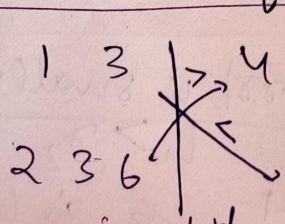
3) (2)

not valid



valid configuration.

4) (3)



invalid

options : 0 1 2 3 4

ans : x x ✓ x x

Can we perform a Binary search on this?

↓

~~Only~~ BS

create a check fn :

Some Observⁿ from how we decide what is ans.

1) l_1 r_1

l_2 r_2

if ($l_1 > r_2$) ⇒ [remove bigger elemⁿ from arr1. ⇓ inc elem on arr2]

So betn whichever array we were applying BS on (# elem to pick from that array for left)

⇓

reduce / inc no of elem

2) so this way we can B to go left / right

→ eg: if arr1 is smaller ~~array~~ arr & $l_1 > r_2$ then use elem from arr1

hi = mid-1 ⇐ so left

Note :- There'll never be a case when $l_1 > r_2$ && $l_2 > r_1$ ✗

---

* → Similarly if arr2 was the smaller array then for $(l_1 > r_2)$ → now decrease / remove biggest elem from arr1. ↵

so right (# elem from arr2 for left) ≤ i.e (else) elem to arr2

lo = mid+1

&& once $(l_1 < r_2$ && $l_2 < r_1)$

(mid) rep (no of elem to take from smaller arr for left)

even sized

$$\text{median} = \frac{\max(l_1, l_2) + \min(r_1, r_2)}{2}$$

Similarly for odd sized array

$$\text{median} = \frac{\max(l_1, l_2)}{2}$$