Q) Find A^{th} magical number.

A no. is magical if it is divisible by B or C.

Eg: B = 2, C = 3, A = 8 — ans → 12

1 ✗, 2 ✓, 3 ✓, 4 ✓, 5 ✗, 6 ✓, 7 ✗, 8 ✓, 9 ✓, 10 ✓, 11 ✗, 12 ✓, 13, 14, 15

B.f.→ Consider all nos till count == A.

Maximum range for ans → $[1, A * \min(B,C)]$

2, 4, 6, 8, 10, 12, 14, 16

Range → $[\min(B,C), A * \min(B,C)]$

Q) Given B, C, x. find count of magical numbers from 1 to x.

B = 3, C = 5, x = 35. — 16

How many multiples of 3 are there in $[1,35] \Rightarrow \frac{35}{3} = 11$

" " " of 5 " " " $[1,35] \Rightarrow \frac{35}{5} = 7$

" " " of (3×5) " " " $[1,35] \Rightarrow \frac{35}{15} \rightarrow 2$

3, 5, 6, 9, 10, 12, 15, 18, 20, 21, 24, 25, 27, 30, 33, 35

Q) Count of magical no's from [1, 100], B = 9, C = 12.

$$\frac{100}{9} + \frac{100}{12} - \frac{100}{36} = 11 + 8 - 2 = 17$$

least common multiple of 9, 12.

Eg. A = 10, B = 2, C = 3

search space : [1, 20]

Q) Check if 16 is your 10th magical no.?

count of magical no's for 16 → $\frac{16}{2} + \frac{16}{3} - \frac{16}{6} = 8 + 5 - 2 = 11$

⇒ Move left

count of magical no's for 10 ⇒ $\frac{10}{2} + \frac{10}{3} - \frac{10}{6} = 7$

⇒ Move right

count of magical no's for 15 ⇒ $\frac{15}{2} + \frac{15}{3} - \frac{15}{6} = 10$

pseudo-code:

```
lft = min(B,C) , right = A * min(B,C)
while (lft <= right) {
    mid = (lft+right)/2;        // find count of magical no's till mid
    count = fun(B, C, mid);
    if (count == A) {
        return mid;             // ans = mid; right = mid-1;
    }
    else if (count < A) {
        lft = mid+1;
    }
    else {
        right = mid-1;
    }
}
```

trace for:

B = 5, C = 7, A = 3

[1, 15]

mid = $\frac{1+15}{2}$ = 8

count for 8 = $\frac{8}{5} + \frac{8}{7} - \frac{8}{35}$ = (2)

[9, 15]

mid = $\frac{9+15}{2}$ = 12

count for 12 = $\frac{12}{5} + \frac{12}{7} - \frac{12}{35}$ = (3)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
count:					1	1	2	2	2	3	3	3	3	4	5

ans = 12

→ move lft.

T.C → $\log_2(A * \min(B,C))$, S.C → O(1)

$$\frac{mid}{B} + \frac{mid}{C} - \frac{mid}{lcm(B,C)}$$

$$n1 \times n2 = gcd(n_1, n_2) * lcm(n_1, n_2)$$

#todo

→ How to minimize the search space ?

Google, Amazon, G.S

Q.1 Given N distinct elements in an array & array is unsorted. Find element at k^{th} index position in its sorted form.

Note → We can't modify the array. We can't use extra space.

Eg: arr → {2, 8, 3, 11, 14}

K = 2.

{2, 3, 8, 11, 14}
0 1 2 3 4

ans → 8.

arr → {11, 24, 18, 3, 5, 27, 34, 9, 40}

K = 4

less: 3 5 4

ans = 18

B.F → Consider every no. and for every no. find the count of no's which are less than the curr no.

if (count → K) ⇒ curr no. is our ans.

T.C → $O(n^2)$

$O(n \log n)$ → sorting → B.S → target, search space.

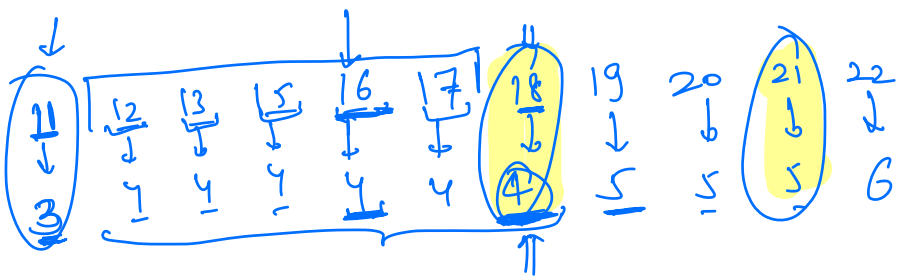
$O(n)$ ✗

→ [0, n-1] x

→ [min, max] → search space

arr → { 11, 24, 18, 3, 5, 27, 34, 9, 40 }, K=4

left	right	middle	(# count elements < middle element)	
3	40	21	5	⇒ (discard right portion)
3	20	11	3	⇒ (discard left portion)
12	20	16	4	⇒ 16 is our ans? X



pseudo-code.

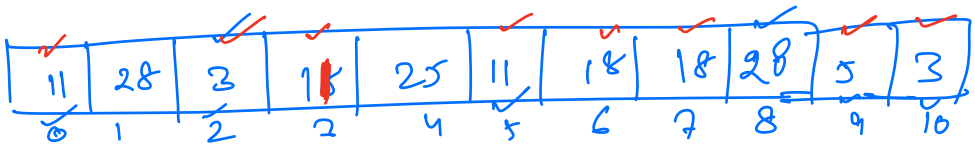
```
left = min , right = max
while (left <= right){
    mid = (left + right) / 2;
    x = countless(arr, mid);

    if (x <= k){
        ans = mid;
        left = mid+1;
    }

    else {
        right = mid-1;
    }
}
return ans;
```

No. of elements which are less than mid

T.C : O(N log (max - min))

S.C : O(1)

todo → what if duplicates are there?



k = 5

left	right	middle	count < middle
3	28	15	6
3	14	8	3
9	14	11	3
12	14	13	6
12	13	12	6
12	11	✗	

ans = 11

Q.) A[n] : [sorted]

B[m] :

find k^{th} position elements in merged sorted array.

```
left = Min[A[0], B[0]], right = Max(A[n-1], B[m-1])
while (left <= right) {
        mid = (left + right) / 2;                    B.S.
        x = countSortedless(a[], mid)
        x += "          "  (b[], mid)

        if (x <= k) {
            ans = mid;
            left = mid+1;
        }

        else {
            right = mid-1;
        }
}
```

T.C → $\log_2 n * \log(max - min)$

S.C → $O(1)$

Q: Find the median of two sorted arrays.

A : — — —

B : — — —

K = Total no. of elmnts / 2.

Q) Given N no. of sorted array ⇒ sorted matrix.

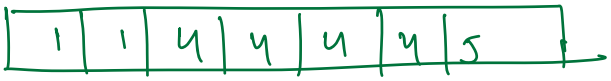
Find the median of that matrix.

min → ~~~ max → ~~~

```
while ( left <= right) {
    mid = (left + right) / 2;
    x = 0;
    for( i=0 ; i < N , i++) {
        x += countSortedLess (a[i], mid);
    }
    if (x <= K) {
        ans = mid;
        left = mid+1;
    }
    else {
        right = mid-1;
    }
}
```

T.C → <= O(n log n * log (max-min))

S.C → O (1)



k = 3

left	right	middle	count < mid
1	5	3	2
4	5	4	2
5	5		

x <= k.

update ans.

and update left = mid + 1

ans = 4