Today's Content:

- → A^{th} magical number
- → k^{th} pos in unsorted distinct array → (Amazon / Google)
  - → k^{th} pos in unsorted array
  - → k^{th} pos in 2 sorted arrays, when merged
  - → k^{th} pos in N sorted arrays, when merged

18) Given 2 numbers A & B, find LCM (A, B)

LCM (A, B) = least comm multiple between A & B

LCM (6, 5) = 30
LCM (12, 8) = 24

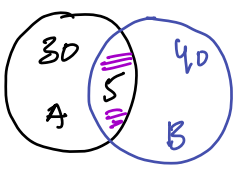
LCM (A, B) = A * B / gcd (A, B)

LCM (6, 5) = 6 * 5 / gcd (6, 5) = 30/1 = 30

LCM (12, 8) = 12 * 8 / gcd (12, 8) = 96/4 = 24

→

Venn diagrams:



A : Indicates Students who play cricket : 30

B : Indicates Student who play football : 40

Calculate students who play Cricket & football : 5

Calculate Students who play cricket or football : 30 + 40 - 5 = 65

$A \cup B = A + B - (A \cap B)$

↳ { Counted 2 times in both A & B
Hence we are subtracting 1 time }

3(8) Given A & B, find no of multiples of A from [1 - B]

ans: B/A

	A	B								
Eg1:	3	20	:	3	6	9	12	15	18	: $\frac{20}{3} \Rightarrow 6$
Eg2:	4	30	:	4	8	12	16	20	24	28 : $\frac{30}{4} \Rightarrow 7$
Eg3:	7	35	:	7	14	21	28	35 :		$\frac{35}{7} \Rightarrow 5$

4(8) Given A, B, C find no of multiples of B or C from [1 - A]

Eg1: // generalized: $A/B + A/C - A/LCM(B,C)$

common (repeating)

B ⊆ A

3 5 35

mul of 3 ⇒ {3 6 9 12 15 18 21 24 27 30 33}

mul of 5 ⇒ {5 10 15 20 25 30 35}

⇒ 11 + 7 − 2 = 16

9 12 100

LCM(9, 12) = 36

mul of 9 {9 18 27 36 45 54 63 72 81 90 99} ⇒ $\frac{100}{9}$

mul of 12 {12 24 36 48 60 72 84 96} ⇒ $\frac{100}{12} \Rightarrow 8$

mul of 36 {36, 72} → $\frac{100}{36} \Rightarrow 2$ $\left\{\frac{100}{9} + \frac{100}{12} - \frac{100}{36}\right\}$

12 8 100

LCM(12, 8) = 24

mul of 12 : {12 24 36 48 60 72 84 96} ⇒ $\frac{100}{12} \Rightarrow 8$

mul of 8 : {8 16 24 32 40 48 56 64 72 80 88 96} ⇒ $\frac{100}{8}$

mul of 24 : {24 48 72 96} ⇒ $\frac{100}{24} \Rightarrow 4$ $\left\{\frac{100}{12} + \frac{100}{8} - \frac{100}{24}\right\}$

Q) A^{th} Magical Number :

Given A, B, C, find A^{th} magical number.

Note: If a number is said to be Magical Number, If it is a divisible by B or C

	B	C	A
Ex1:	2	3	8

{ 1 ✓ 2 ✓ 3 ✓ 4 ✗ 5 ✓ 6 ✗ 7 ✓ 8 ✓ 9 ✓ 10 ✗ 11 ✓ 12 } — 8th magical number

Ex2: 4 6 5 — 5th magical number

{ 1 2 3 ✓ 4 5 ✓ 6 7 ✓ 8 9 10 11 ✓ 12 14 15 ✓ 16 }

Ideas:

loop untill you get A magical numbers

co = 0;

for(i = 1; ; i++) { — at max till what value we need to go to get A^{th} magical number

  if(i % B == 0 || i % C == 0) {

    co++

    if(co == A) { return i; }

  }

}

obs: [1, min(B,C) * A] => we will get A^{th} magical number

Ex:

B	C	A	min(B,C) * A
3	10	3	=> [9]
4	20	4	=> [16]
2	3	10	=> [20]
2	4	10	=> [20]

min(B,C) * A

TC:

SC: O(1)

// Idea 2: Searching

Target : A^{th} magical Nuber

SearchSpace : [1, min(B,C)*A]

If we can discard Space ?

	B	C	A
Ex:	4	6	10

Is 36 your 10^{th} magical number:

[1 36] : multiple of 4 → 36/4 → 9

[1 36] : multi of 6 → 36/6 → 6

[1 36] : multi of 12 → 36/12 → 3

[1 36]:
9 + 6 - 3 → 12

{ number of mult [1-36] > 10 }

36 37 38 39 -- } goto left

Is 24 your 10^{th} magical number

[1 24] : multiple of 4 → 6

[1 24] : multi of 6 → 4

[1 24] : multi of 12 → 2

{ number of mul [1-24] → 8 < 10

21 22 23 24 goto right

Tracing:

Learn Space:

B/5 C/7 A/3 } [1, min(5,7)*3]

l/1 h/15 m/8 : [1 8] → How many factors of 5 or 7 = 8/5 + 8/7 - 8/35 = 2

2 < A : goto right : l = m+1

9 15 12 : [1 12] → How many factors of 5 or 7 = 12/5 + 12/7 - 12/35 = 3

3 == A : { ans = 12, goto left; h = m-1 }

9 11 10 : [1 10] → How many factors of 5 or 7 : 10/5 + 10/7 - 10/35 = 3

3 == A : { ans = 10, goto left, h = m-1 }

9 9 9 : [1 9] → How many factors of 5 or 7 : 9/5 + 9/7 - 9/35 = 2

2 < A : goto right, l = m+1

10 9 : (Forcefully) return ans = 10

B=5, C=7

1st occurrence with magical number = 3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	1	1	2	2	2	3	3	3	3	4	5

// Pseudo code :

```
magical ( int A, int B, int C) {
    lo = 1, hi = min(B,C) * A, ans = ___
    lcm = LCM(B,C)
    while( lo <= hi) {
        int m = (l+h)/2   [number of magical [1 - m]]
        int co = m/B + m/C - m/lcm
        if ( co < A) {
            [1  m] < A : goto right
              ↳ no: of magical
            l = m+1
        }
        else if ( co > A) {
            [1  m] ≥ A : goto left
              ↳ no: of magical
            h = m-1
        }
        else // number of mag [1 m] = A
            ans = mj : goto left
            h = m-1
    }
    return ans;
}
```

TC: $\log_2(hi - lo+1) + 1$

TC: $\log(\min(B,C)*A)$

SC: O(1)

10:40pm

lomgins mrcat

208) Given unsorted array of N distinct elements, find k^{th} index pos in it's sorted form

Note: We cannot modify array / We cannot Use Extra Space

Eg1: ar[5] = { 2 8 3 11 14 }

k = 2 Sorted form { 2 3 8 11 14 }

0 1 2 3 4

ans = 8

Eg2: ar[9] = { 11 24 18 3 5 27 34 9 40 }

0 1 2 3 4 5 6 7 8

k = 4

// Elements < 3 5 4

↳ 18 is ans:

Ideas:

↓

ar[N] k^{th} index in sorted form

0 1 2 --- k-1 < k^{th} index

k Elements

obs: no: of elems < k^{th} index == k

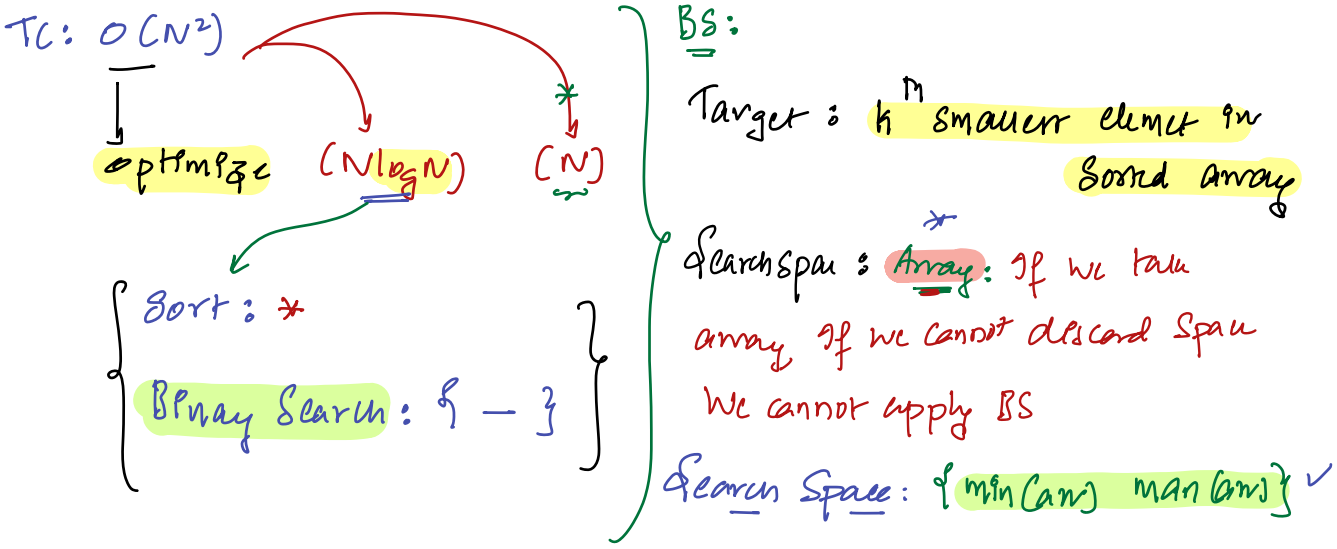
```
int countLess(int ar[], int ele){
    c = 0;
    for(i=0; i<N; i++){
        if(ar[i] < ele){ c++ }
    }
    return c;
}
```

Idea 1:

For every element count no: of Elements less than that == k

```
for(i=0; i<N; i++){
    c = countLess(ar[], ar[i])
    if(c==k){
        return ar[i]
    }
}
```

TC: N^2 SC: O(1)



Eg: { 4 1 5 15 6 2 } k=3

l	h	m	# how many elements ≤ m { iterate in array & get count }
1	15	8	cnt ≤ 8 : 5 > 3 : goto left; h = m-1

8 9 10 --
* * *

1	7	4	# how many elem ≤ m

cnt ≤ 4 : 2 < 3 : goto right : l = m+1

Pseudo code:

```
int posElement(int ar[], int N, int k){
    l = min(ar[]), h = max(ar[])
    while(l <= h){
        m = (l+h)/2          --> you need to get no: of elements on & m in array.
        int c = CountLess(ar, m)
        if(c > k){
            // goto left
            h = m-1
        }
        else if(c < k){
            // goto right
            ans = m
            l = m+1
        }
        else{ --> {Issues}
            ans = m;
            l = m+1
        }
    }
    return ans;
}
```

TC: $\log_2(h-l+1) * N$

SC: $O(1)$

```
if(c > k){ h = m-1 }
else{
    ans = m;
    l = m+1
}
```

Eg: { 11 24 20 3 5 27 34 9 40 } k=4

l	h	m	Element at m	
3	40	21	:	# <21 : 5 > 4 : goto left, h = m-1
3	20	11	:	# <11 : 3 < 4 : goto right, l = m+1
12	20	16	:	# <16 : 4 == 4 { ans = 16, l = m+1 }
17	20	18	:	# <18 : 4 == 4 { ans = 18, l = m+1 }
19	20	19	:	# <19 : 4 == 4 { ans = 19, l = m+1 }
20	20	20	:	# <20 : 4 == 4 { ans = 20, l = m+1 }

4 -> 20 : { Break }

{ We need last 4 occurance here }

//Search Space :

3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22 ...	40
✓		✓				✓		✓												
0	1	1	2	2	2	2	3	3	4	4	4	4	4	4	4	4	4	5	5	

cut < k : ans = m { no effect } : l = m+1

cut = k : ans = m : l = m+1

cut > k : h = m-1

// Elements are repeating ? - {code works}

```
int posElement(int ar[], int N, int k){
    l = min(ar[]), h = max(ar[])
    while(l <= h){
        m = (l+h)/2        // you need to get no. of elements <= m in array.
        int c = Count(ar, m)
        if(c > k){ h = m-1}
        else
        {
            ans = m;
            l = m+1
        }
    }
    return ans;
}
```

Duplicates:

ar[8] = { 15 4 15 10 16 19 10 15 } → k = 4

min	max	m	Elements < m
4	19	11	# < 11 : 3 < k = 4 : ans = 11, l = m+1
12	19	15	# < 15 : 3 < k = 4 : ans = 15, l = m+1
16	19	17	# < 17 : 7 > k : h = m-1
16	16	16	# < 16 : 6 > k : h = m-1
16 > 15			break : { return ans } return 15

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

0 1 1 1 1 1 1 3 3 3 3 3 6 7 7 7

count > k :

h = m-1

count < = k

: ans = m

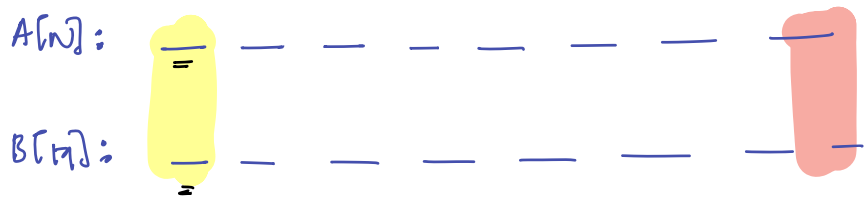
: l = m+1 }

Eg: ar[] : { 4 4 4 4 4 16 17 18 }

ar[] : { 4 5 5 5 5 16 17 18 }

ar[] : { 4 10 10 11 11 16 17 18 }

Q8): 2 sorted array, find k^{th} pair in merged array:



↳ possible (A[], B[], k)

lo = min(A[0], B[0]), h = max(A[N-1], B[M-1])

while (lo <= hi) {

m = (l + h)/2 → no: of elements <= m in A + B

c = countLess sorted (A, m) : Calculate no: of elements less than m in given sorted array : (log N)

c = c + countLess sorted (B, m)

if (c > k) { h = m - 1 }

else { ans = m
l = m + 1 }

}

return ans;

TC: $\log_2(hi - lo) \times \{\log_2 N + \log_2 M\}$

SC: O(1)

3

A[10] → Total Elements: 17 → median $\frac{17}{2} \rightarrow 8^{th}$

B[7]

// N Sorted Arrays in, find k^{th} index pos in merged arrays:

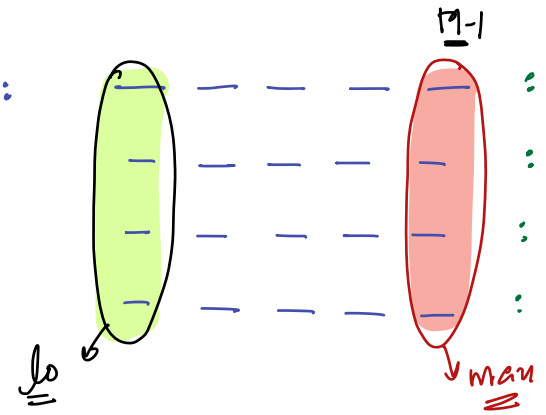
Mat[N][M]

i → no:of rows

j → no:of cols

In every row m Elements

mat[4][5]:



posmatrix(int mat[][], int N, int M, int k){

lo = min (All 0^{th} col Elements) hi = max (All $M-1^{th}$ col elements)

while(lo < hi){

mid = (lo + hi)/2 → no:of Elements < mid in all arrays

c = 0;

for(i=0; i<N; i++){

c = c + countlessorEq(mat[i], mid)

}

if(c > k) { h = mid-1 }

else{

ans = mid;

l = mid+1

}

}

return ans;

}

TC: $= \log_2(hi-lo+1) * N * \log_2^M$

SC: O(1)

// mat[5][5] → elements → 25 → mid: 25/2 → 12th Index position