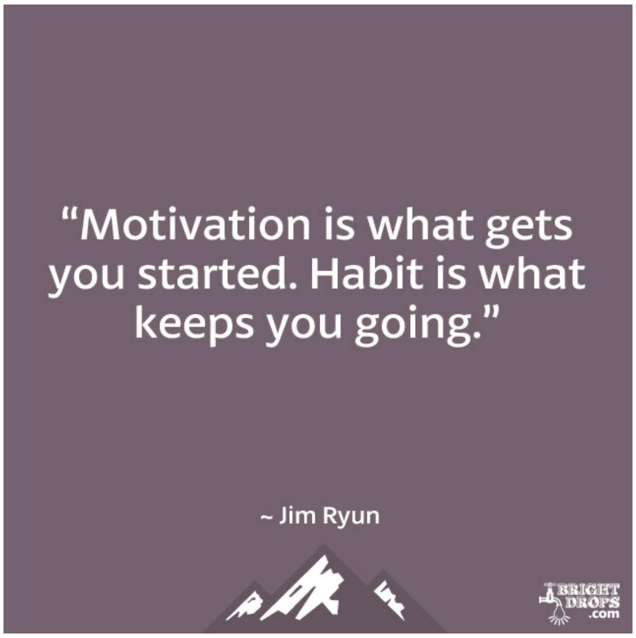
BINARY SEARCH 2



Good

Evening

Today's content

01. Sqrt (N)
02. Search in sorted Rotated Array
03. A* magical no.
04. Median in a matrix (Amazing ques)

Q1 Given a positive N, find sqrt(N) /

Given N, find greater i such that $i * i \le N$

floor(sqrt(n))

sqrt(25) = 5

sqrt(16) = 4

sqrt(47) = 6

sqrt(10) = 3

Idea 1 → Start from $i=1$ & keep on updating the ans until $i*i \le n$

$n = 40$

i	$i*i \le n$	ans
1	$1*1 \le 40$	1
2	$2*2 \le 40$	2
3	$3*3 \le 40$	3
4	$4*4 \le 40$	4
5	$5*5 \le 40$	5
6	$6*6 \le 40$	6
7	$7*7 \le 40$ → Break	

Ans = 6

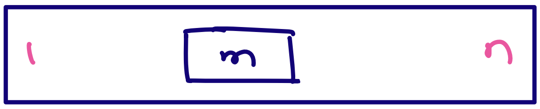
Target = sqrt(n)

Search space = 1 n

lo hi

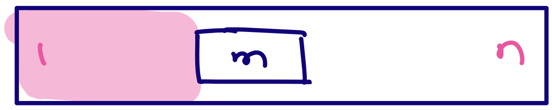
To reduce the search space

01. Case 1



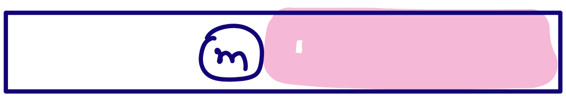

```
if (m*m == n) return m;
```

02 Case II




```
if (m*m < n){
    ans = m;
    //go to right
}
```

03. Case III




```
if (mid * mid > n)
{
    go to LHS
}
```

$n = 50$

lo	hi	m	compare m*m & n	
1	50	25	$25 * 25 > 50$	: hi = m − 1
1	24	12	$12 * 12 > 50$	: hi = m − 1
1	11	6	$6 * 6 < 50$	ans = 6 l : m + 1
7	11	9	$9 * 9 > 50$	: hi = m − 1
7	8	7	$7 * 7 < 50$	ans = 7 l = m + 1
8	8	8	$8 * 8 > 50$	: hi = m − 1
8	7		→ Stop	, Ans = 7

$n = 4$

hi → n/2

lo	hi	m	m*m & n	
1	2	1	$1 * 1 < 4$ →	ans = 1 lo = m + 1
2	2	2	$2 * 2 == 4$	True

* It is going to fail for n = 5 & n = 1 ⇒ {TODO}

* Code → {TODO}

lo = 1   hi = n , ans = 0

```
while (lo ≤ hi)
{
    int m = (lo + hi) / 2

    if (m * m == n) {
        return m;
    }
    else if (m*m < n) {
        ans = m
        lo = m+1
    }
    else {
        hi = m - 1
    }
}
return ans;
```

* Sorted arr[] = { 3, 9, 14, 16, 20, 28, 35 }

Given k, can k be present in arr[] or not ?

k = 16 → Yes

k = 25 → Yes

k = 60 → NO

Obs → If we have to search for k, then sorted array can help

* Given an input ar[], formed by rotating a distinct sorted array right to left.
Search ele & return index if it is present else return -1.

ar[] = {	10	11	12	13	17	20	23	25	26	1	3	5	6	8	}
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	

k = 17

Idea 1 → Linear search on array & check if ele == k

TC: O(n)

SC: O(1)

ar[ ] = { 10 11 12 13 17 20 23 25 26 1 3 5 6 8 }

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

K = 17

lo m hi

lo	hi	m
0	13	6
0	5	2
3	5	4

∴ Left half is sorted

hi = m − 1

∴ Left half is sorted

K is not present in lo....m

lo = m+1

3 5 4 → return mid

ar[ ] = { 10 11 12 13 17 20 23 25 26 1 3 5 6 8 }

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

K=5

m lo h

lo	hi	m
0	13	6
7	13	10

check which half is sorted

if (ar[lo] ≤ ar[m]) → Yes

if (tar is lying b/w lo to m)

else : lo = m+1

if (ar [m] ≤ ar[hi]){

if (tar is lying b/w m to hi)

} lo = m+1

11 13 12

if (ar[lo] ≤ ar[m]) → Yes

if (tar is lying b/w lo to m)

hi = m - 1

11 11 11

if (ar[m] == tar)

return m

ar[] = { 1 2 3 4 5 6 }

Rotate 1 = { 6 1 2 3 4 5 }

0 1 2 3 4 5

Rotate 2 = { 5 6 1 2 3 4 }

Rotate 3 = { 4 5 6 1 2 3 }

Rotate 4 = { 3 4 5 6 1 2 }

m

```
lo = 0        hi = n-1

while (lo ≤ hi)
    m = (lo+hi)/2;

    if (ar[m] == tar) return m

    else if ( ar[lo] ≤ ar[m]) { lo... m] sorted
        if ( ar[lo] ≤ tar  &&  tar < ar[m]) hi=m-1
        else  lo = m+1

    } else {  // [m.....hi] sorted
        if ( tar > ar[m]  &&  tar ≤ ar[hi]) lo=m+1
        else  h = m-1;
    }
}
return -1.
```

10:06 pm → 10:16 pm

Q1. Find A^{th} magical no.

A number is magical if it is divisible by B or C.

Eg:- B = 2 , C = 3 , A = 8

Find the 8^{th} magical no. for B = 2 & C = 3

1 2 3 4 5 6 7 8 9 10 11 12 → Ans = 12

B = 3 C = 17 find 4^{th} magical no.

1 2 3 4 5 6 7 8 9 10 11 12 → Ans = 4

count → 1 2 3 4

Brute force : Consider all the numbers from 1 & check if a number is magical or not until unless my count == A

No. of iterations = A * min(B, C)

Max possible no. = A * min (B, C)

Search space = [1 → A * min(B, C)]

Target = A^{th} magical no

Q $B = 3$, $C = 5$ & find the count of magical no. from 1 to 35

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

Mul of 3 from [1 - 35] $= \frac{35}{3} = 11$

Mul of 5 from [1 - 35] $= \frac{35}{5} = 7$

Multiples of 3 * 5 from [1 - 35] $= \frac{35}{15} = 2$

Ans $= 11 + 7 - 2 = 16$

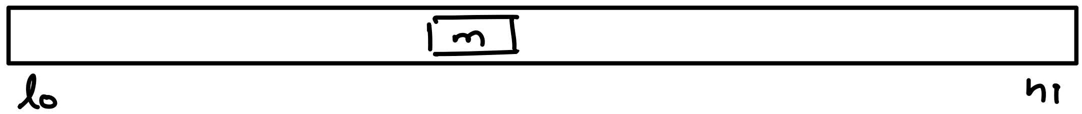
Q count of magical no. from [1 - 100] $B = 9$ $C = 12$

$$= \frac{100}{9} + \frac{100}{12} - \frac{100}{LCM(9,12)}$$

$$= \frac{100}{9} + \frac{100}{12} - \frac{100}{LCM(9,12)} \Longrightarrow$$

$LCM(a,b) * GCD(a,b) = a * b$

Back to original ques



For this mid value,

count of magical no. $\leq m$

count < A

//go to RHS

$lo = m+1$

$== A$

$ans = m$

$hi = m - 1$

count > A

//go to left

$hi = m - 1$

$B = 5$, $C = 7$, we want to find 3^{rd} magical no.

Range = [1 to 15]

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

lo	hi	mid	Count of magical no.
1	15	8	$\frac{8}{5} + \frac{8}{7} - \frac{8}{35} = 2$ lo = mid + 1
9	15	12	$\frac{12}{5} + \frac{12}{7} - \frac{12}{35} = 3$ ans = 12 hi = m − 1
9	11	10	$\frac{10}{5} + \frac{10}{7} - \frac{10}{35} = 3$ ans = 10 hi = m − 1
9	9	9	$\frac{9}{5} + \frac{9}{7} - \frac{9}{35} = 2$ lo = m + 1

10 9 $\longrightarrow$ Stop

```
lo = 1
hi = A * min(B, C)

while (lo <= hi)
    mid = (lo + hi) / 2

    int count = countmagical(mid, B, C);

    if (count < A) {
        lo = mid + 1
    } else if (count > A) {
        hi = mid - 1
    }
    else {
        ans = mid;
        hi = mid - 1;
    }
}
```

$* \text{ int } lcm = \dfrac{(a * b)}{gcd(a, b)}$

TC: O(log(search space) * LCM part)

TC: O(log(A * min(B, C)) * log(min(B, C)))

SC: O(1)

Note :- we can optimise the TC by calculating LCM & passing it in countmagical function.

* Median of Array → Middle element of sorted arr.

Median

arr[ ]= {1, 2, 3, 4, 5}

arr[ ]= {1, 2, 3, 4}

$\frac{2+3}{2} = 2.5$

A[ ] = {1, 4, 5}
B[ ] = {2, 3}

Ans = 3

A[ ] = {1, 2, 3}
B[ ] = {4}

Ans = 2.5

Given with two sorted arrays & you need to figure out the median

Idea 1 → Merge 2 sorted arrays & return the middle element.

TC : O(n+m)

Eg:-

A[ ] = {1, 3, 4, 7, 10, 12}

B[ ] = {2, 3, 6, 15}

size = 10

largest ele of first

smallest ele of second

5 ele 5 ele

$$\text{median} = \frac{(l+s)}{2}$$

01. Scenario 1 → 4 ele from first arr

1 3 4 7 10 12

2 3 6 15

02 Scenario 2 → 2 ele from first arr

l_1 r_1

1 3 4 7 10 12

2 3 6 15

l_2 r_2

$$\left\{ \begin{array}{l} l_1 \leq r_1 \\ l_2 \leq r_2 \end{array} \right.$$

$$\left. \boxed{\begin{array}{l} l_1 \leq r_2 \\ l_2 \leq r_1 \end{array}} \right\} \text{correct splitting}$$

A[ ] = { 1, 3, 4, 7, 10, 12 }

B[ ] = { 2, 3, 6, 15 }

Scenario 3 → Pick 3 ele from first arr

l_1

1 3 4

2 3

l_2

r_1

7 10 12

6 15

r_2

$$\text{Ans} = \frac{4+6}{2} = 5$$

* Idea → Apply Binary search on array A to figure out how many ele we are going to pick from A

Eg:

l_1 r_1

$A[\,] = \{7, \mid 12, 14, 15\}$

l_2 r_2

$B[\,] = \{1, 2, 3, 4, \mid 9, 11\}$

Total no. of ele = 10

no. of ele on one side = 5

Search space = [0 4]

lo	hi	mid
0	4	2
0	1	0
1	1	1

0, 4, 2 → choose 2 ele from arr1 & 3 ele from arr2

$l_1 \le r_2$ ✗ : go to left

$hi = mid - 1$

0, 1, 0 → choose 0 ele from arr1

$l_2 \le r_1$ ✗

$lo = mid + 1$

1, 1, 1 → valid distribution

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

* For odd size combined arr → Dry run by yourself

$lo = 0$

$hi = \min\left(\frac{n+m}{2}, \text{length of } A\right)$

```
while ( lo <= hi )
|
|    int  cut1 = (lo + hi) / 2
|
|    int  cut2 = ((n + m + 1) / 2) - cut1
|
|       int  l1 = (cut1 == 0) ? -∞ : ar[cut1 - 1]
|       int  l2 = (cut2 == 0) ? -∞ : ar[cut2 - 1]
|
|       int  r1 = (cut1 == n) ? ∞ : ar[cut1]
|       int  r2 = (cut2 == m) ? ∞ : ar[cut2]
|
|    if ( l1 <= r2 && l2 <= r1 ){
|    |
|    |   if ((n+m) % 2 == 0) return (max(l1, l2) + min(r1, r2)) / 2
|    |
|    |   else  return  max(l1, l2);
|    }
|    else if ( l1 > r2 )  hi = cut1 - 1
|
|    else  lo = cut1 + 1;
}
```

TC : O(log n)

SC : O(1)