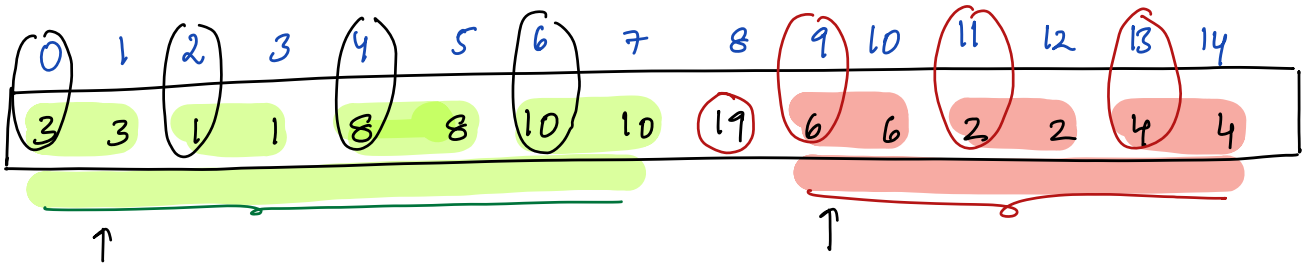
→ Recap → Target
→ Search Space

When BS? → Divide search space into 2 parts & discard 1 half using condition.

Every element occurs twice except for 1, find unique element

Note: Duplicates are adjacent to each other.

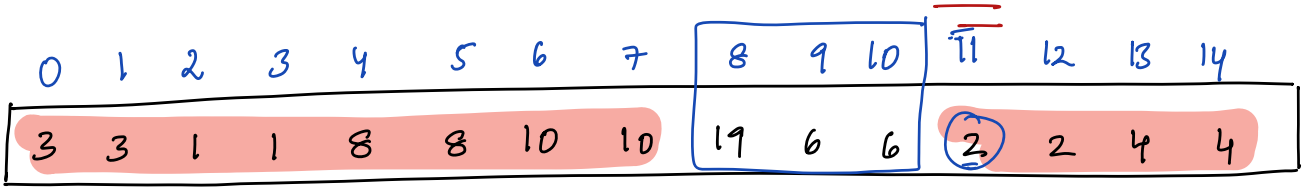
// BF: XOR of all elements → TC: O(N) SC: O(1)



left side:
- goto right side
- l = m+1
- All first occurence are in even index

right side:
- goto left side
- h = m-1
- All first occurence are in odd index

l=0, h=14, m=7 → 2nd occurence } → goto 1st occurence
m = m-1



l=0, h=14 → m=7 → 6 : m first occurence
goto right : l = m+2

l = 8 h = 14 → m = 11 → 11 :
goto left : h = m-1

Pseudo Code

TC : $O(\log N)$

SC : $O(1)$

```
if ( N == 1 ) { return A[0] }

if ( A[0] != A[1] ) return A[0]

if ( A[N-1] != A[N-2] ) return A[N-1]

l = 0,  h = N-1
while ( l <= h ) {
        m = (l + h) / 2
        if ( ar[m] != ar[m-1] && ar[m] != ar[m+1] ) {
            return ar[m]
        }
        if ( ar[m] == ar[m-1] ) {
            // mid is landing on 2nd occurence
            m = m - 1
        }
        if ( m % 2 == 0 ) {
            // -> on left
            l = m + 2
        }
        else {
            h = m - 1
        }
}
```

68 Given an array which is formed by rotating a Distinct sorted array by k times, Search for a given a Elem in rotated array

Inc (Back - front)

	0	1	2	3	4	5	6	7	8	9	10	11	12
ar[13] =	-20	-14	-8	-4	1	2	4	7	11	14	19	23	27

k: no rotation

	0	1	2	3	4	5	6	7	8	9	10	11	12
rar[13] =	11	14	19	23	27	-20	-14	-8	-4	1	2	4	7

k = 5

k Elements : [0, k-1]

N-k Elements [k N-1]

19 : ✓

7 : ✓

24 : ✗

Sol:

1) Rotate & Search
   - Rotate: O(N)
   - Search: O(log N)

2) Iterate & Search
   - O(N)

Sol3) : obs1 : 1st Sorted array : [0, k-1]
l = 0, h = k-1
2nd Sorted array : [k, N-1]
l = k, h = N-1
→ TC : O(log N + log N) SC : O(1)

Sol4) : obs1 :
All Elements in 2nd Sorted < ar[0]

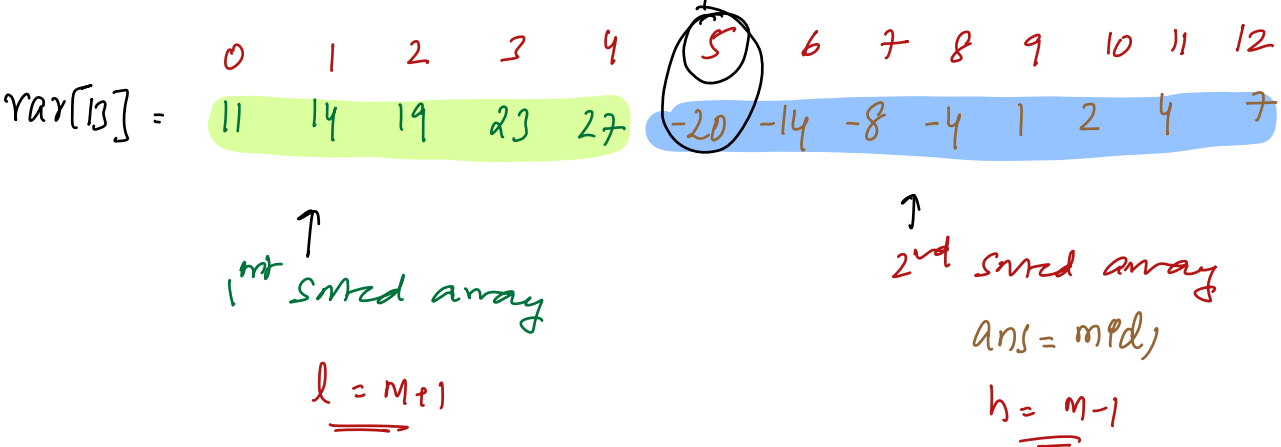
TC : O(log N)

```
if (ele < ar[0]) {
    // 2nd Sorted [Apply BS [k, N-1]
} else
    // 1st Sorted [Apply BS [0, k-1]
```

// Every Same :

Rotation factor k is not given ?

Scan & get k



// k = 0

```
l = 0, h = N-1
while( l <= h){
    m = (l + h)/2
    if (ar[m] < ar[0]){
        // 2nd sorted array
        k = m
        h = m-1;
    }
    else { l = m+1}
}
return k;
```

↳ rotation factor

TC : O(log N)

SC : O(1)

Eg:

	0	1	2	3	4	5
ar[] :	3	4	5	8	10	11

Apply BS to get k

l	h	m	ar[m] < ar[0]
0	5	2	ar[2] < ar[0]
3	5	4	ar[4] < ar[0]
5	5	5	ar[5] < ar[0]
6	5	} = Break	

In this case k = 0

10:30 break

→ No inbuilt functions

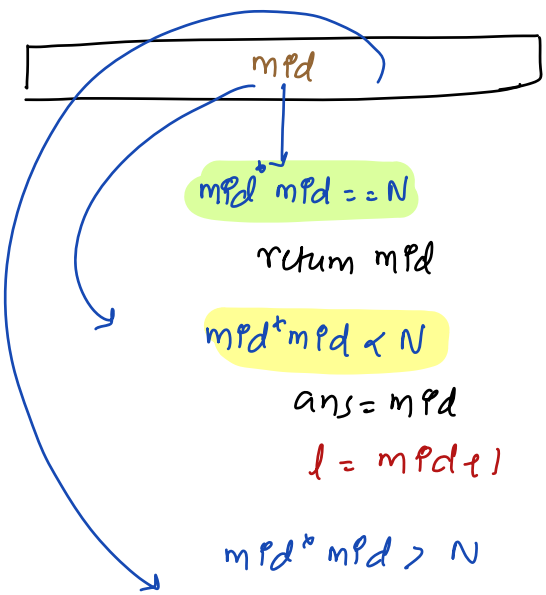
→ Given N (≥ 0) find SQRT(N)

→ Target: (√N)

N	SQRT(N)
16	4
25	5
30	5
39	6

BS: → Search Space: [1 N]

↳ Discard left & right



i	i*i < 39	ans
1	1*1 < 39	1
2	2*2 < 39	2
3	9 < 39	3
4	16 < 39	4
5	25 < 39	5
6	36 < 39	6 } →
7	49 < 39 ✗	(break)

(8, 9, . .) cannot be ans

TC: O(SQRT(N))

```
i = 1; ans = __
while (i*i <= N)
{
    ans = i;
    i++;
}
return ans;
```

TC: O(√N)

SC: O(1)

N = 39 — ans = 6

l	h	m	
1	39	20*20 > 39	h = m-1
1	19	10*10 > 39	h = m-1
1	9	5*5 < 39	a = m, l = m+1
6	9	7*7 > 39	h = m-1
6	6	6*6 < 39	a = m, l = m+1
7	6		{ break }

// Pseudo code

// Constraints →

$1 <= N <= 10^9$

$1 <= N <= 10^{18}$ → $\{N = 10^{18}\}$ $l = 0$ $h = 10^{18}$

$m = \frac{(l+h)}{2} \rightarrow \frac{10^{18}}{2}$

$m * m = x \{10^{36}\}$

$\underbrace{10}_{9}$

```
— sqrt(N) {
    l = 1, h = N;   ans = —
    while (l <= h) {
        long m = (l+h)/2
        if (m*m == N) {
            return mid
        }
        if (m*m > N) {
            h = m-1
        }
        else {
            l = m+1
            ans = m
        }
    }
}
```

$1 <= N <= 10^{18}$

$1 \quad <= \quad N \quad <= \quad 10^{18}$

1 (l) $<= N <=$ 10^9 (hi)

sqrt() $[1 \quad 10^9]$: $[1, \quad 10^{18}]$ Count

$1 <= N <= 10^{18}$

// cuberoot()

1 (l) $<= N <=$ 10^6 (h)

cubrt $[1 \quad 10^6]$: $[1 \quad 10^{18}]$ Count

Q] Given an array of +ve Integers, find maximum k such that, { max subarray sum of len k } $\leq$ B } ← given in Input

	0	1	2	3	4	5	6	7
AR[8] =	3	2	5	4	6	3	7	2

Given B = 20

k = 5 [0 4] = 20 [2 6] = 25 [1 5] = 20 [3 7] = 22 } max = 25 $\leq$ 20 ✗

(Max subarray sum of len)

			ans
k = 1 :	7	$\leq$ 20	1
k = 2 :	10	$\leq$ 20	2
k = 3 :	16	$\leq$ 20	3
k = 4 :	20	$\leq$ 20	4
k = 5 :	25	> 20	✗
k = 6 :			✗
k = 7			✗
k = 8			✗

val: ✓ ✓ ✓ ✓ ✗ ✗ ✗ ✗ ✗

Target : k → { Subarray Sum }

Search : [1 . N]

Ex: AR[4] : 13 14 20 8 → B = 5

k = 1 : { max subsum : 20 > 8 } ✗

```
l = 1, h = N, ans = 0
while( l <= h) {
    k = (l + h)/2
    if(Msubsum(ar[], N, k) >= B){      // TC: O(N)
        ans = k;
        l = k + 1
    }
    else { h = k - 1 }
}
return ans;
```

TC: (logN) * {N}

TC: O(NlogN)

SC: O(1)

given size k

3 mins

Binary Search

- Sorted Array
- Array is present
- Your own Search Space

→ 1) Revise notes

2) Issues in reaching

- → Question / Engli
- → Intuition
- → Examples
- → Fast
- → Slow
- → Getting Break

{ Stuck, } way

// sqrt()

$1 <= N <= 10^{18}$

$l = 1, \quad h = 10^9$

$m * m$

| cuberoot()

$1 <= N <= 10^{18}$

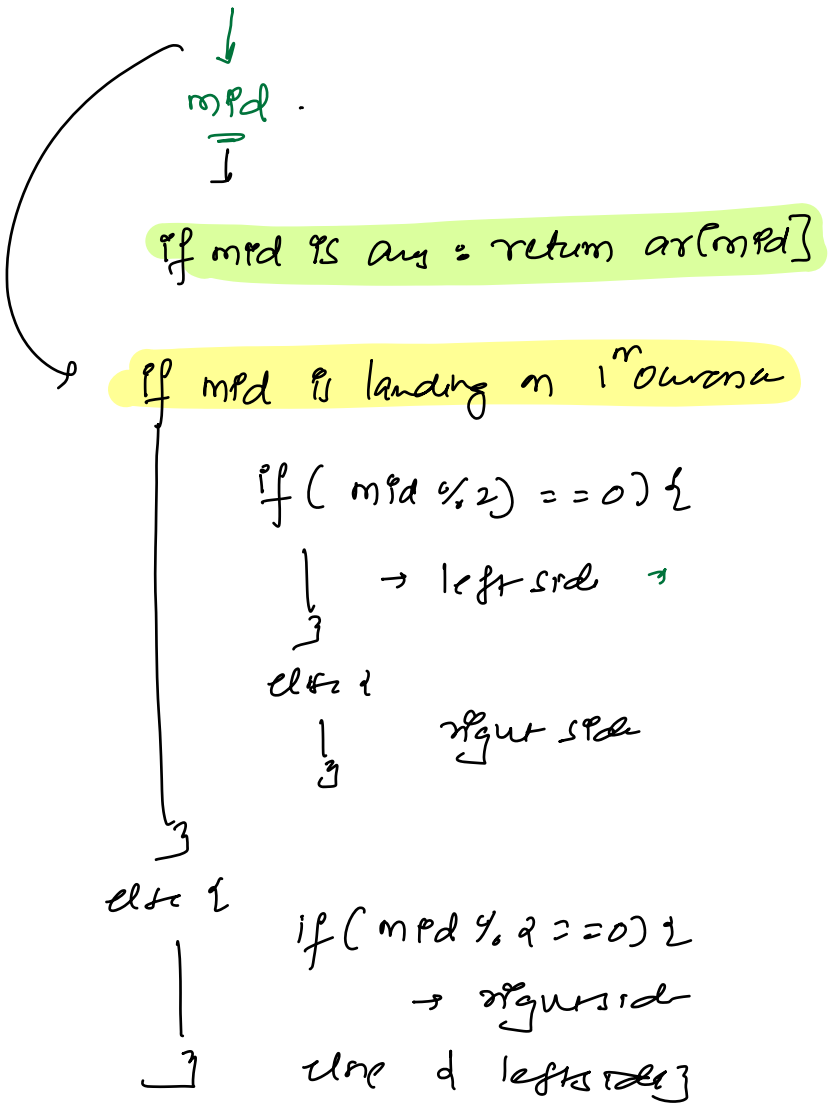
$l = 1, \quad h = 10^6$

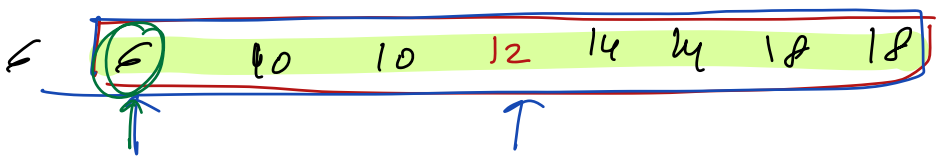
$m * m * m$

| 4^{th} root

$m * m * m * m$

// 1 Problem





l = mid

l	h
0	1
1	2

l	h	m	
0	2	1	→ 0
1	2	1	→ 0
1	2	1	→ 0

1st Element

```
if ( N == 1 ) { return A[0] }

if ( A[0] != A[1] ) return A[0]

if ( A[N-1] != A[N-2] ) return A[N-1]

l = 0,  h = N-1
while ( l <= h ) {
        m = (l + h) / 2
        if ( ar[m] != ar[m-1] && ar[m] != ar[m+1] ) {
            return ar[m]
        }
        x = m
        if ( ar[m] == ar[m-1] ) {
            // mid is landing in 2nd occurence
            x = x - 1
        }
        if ( x % 2 == 0 ) {
            // -> on left
            l = m + 1
        }
        else {
            h = m - 1
        }
}
```