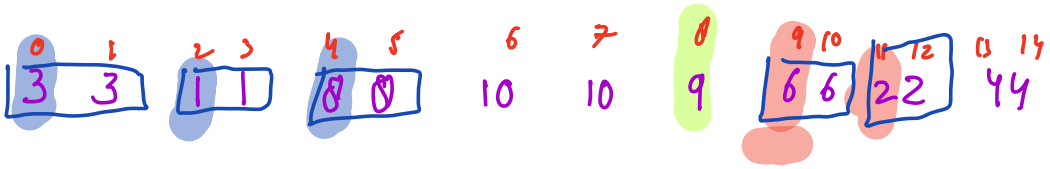
Binary search 2

Q⇒ Every Element occurs twice except for 1, find unique ele.

Note: Duplicate are adj to each other

idea 1: XOR → O(N)

ex1:



1st occurance at even index: go to right

1st occurance at odd index: go to left

target : unique elem?

Search space : given array.

arr[m-1] != arr[m] && arr[m] != arr[m+1]

l	r	m	unique	if(arr[m]=arr[m-1])	m%2==
0	14	7	*	m=m-1	l=m+2
8	14	11	*	—	r=m-1

8 10 9 4 ✓ —— r = m-1

8 8 8

```
find Unique( int arr[] , int N)
{
    l=0, h= N-1;
      if( N==1) return a[0];
    if( arr[0] != arr[1]) return arr[0];
    if( arr[N-1] != arr[N-2]) ret arr[N-1];
     while( l <= r)
     {
       mid = (l + r)/2
       if( a[m-1] != a[m] &&
                        a[m] != a[m+1])
                  return a[m];
       if (a[m-1] == a[m])
                  m = m-1;
       if (m % 2 == 0)
                  l = m+2,
       else
                  h = m-1;
     {
}
```

3, 5, 5

l	r	m	un	if(a[m]==a[m-1])	m%2
0	2	1	x	no change	m-1
0	0	0	✓		

arr[-1] != arr[0]

O(log N)

Q⇒ Given +ve N . Find sqrt (N)

floor

sqrt (25) = 5
sqrt (20) = 4
sqrt (10) = 3

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

O(√N)

N = 30

		ans
i=1 ;	1*1 ≤ 30	1
i=2	2*2 ≤ 30	2
i=3	3*3 ≤ 30	3
i=4	4*4 ≤ 30	4

$i = 5$ $\quad 5*5 \leq 30$ $\quad$ 5 ✓✓

$i = 6$ $\quad 6*6 \leq 30$ ✗ $\quad$ return ans.

$N = 50$

l	h	m		
1	50	25	$m*m > 50$	go left $h = m-1$
1	24	12	$m*m > 50$	go left $h = m-1$
1	11	6	$6*6 \leq 50$	$ans = 6$ $l = m+1$
7	11	9	$9*9 > 50$	$h = m-1$
7	8	7	$7*7 \leq 50$	$ans = 7$ $l = m+1$
8	8	8	$8*8 > 50$	$h = m-1$
8	7	→ return	ans	

T T T T T F F F F

ans

Break: 0:31

T F

(a) $(10+\delta_1)$ $(10+\delta_2)$

(b) (10) $(10+\delta_2)$ $(10-\delta_2)$

Q) Given N array element
Calculate Max subarray sum of len K

	0	1	2	3	4	5	6	7	8	9
arr[10] =	-3	4	-2	5	3	-2	0	2	-1	4

K = 5

[0, 4] : 7
[1, 5] : 8
[2, 6] : 12
[3, 7] : 16
[4, 8] : 10
[5, 9] : 11

Sliding window
TC : O(N)
SC : O(1)

```
int submax(int arr[], int N, int K)
{
    // should return max subarray sum
       of len K
    // TC : O(N), SC: O(1)
}
```

Q ⇒ Given an array +ve integers, find max K such that max subarray sum of len K <= B

B = 20

arr = 3, 2, 5, 4, 6, 3, 7, 2

K	max subarray sum of len = K	ans
1	7 <= 20	1
2	10 <= 20	2
3	16 <= 20	3
4	20 <= 20	4
5	25 <= 20	return 4

T T T T T F F F F

0 1 2 3 4 5 6 7

3, 2, 5, 4, 6, 3, 7, 2 B = 10

l	h	m	max subarray	
0	8	4	20 > 10	left
0	3	1	7 <= 10	ans = 1, goto right
2	3	2	10 <= 10	ans = 2, goto right

3 3 3 14 ≥ 10 go to left

3 2 ↔ return ans

arr(3) = {10, 14, 9}

B = 7

0

⇒ {3, 2, 5} B = 14

ans = 3

0 1 2

3, 2, 5 B = 14

l	r	m	subarray	ans = 1
0	3	1	5 ≤ 14	ans = 2 right
2	3	2	7 ≤ 14	ans = 3 right
3	3	3	10 ≤ 14	

4 3 → break

```
l = 0 ; h = N
while ( l <= h)
{
    mid = (l+h) / 2;
    sum = submax( arr, N, mid); ∈ O(N)
    if ( sum <= B)
    {
      ans = mid;
      l = mid + 1;
    }
    else
    {
       h = m - 1;
    }
}
return ans;
```

O(Log N)

TC: O(n log n)
SC: O(1)

← sqrt(N)

```
l = 1 , h = N
while ( l <= h)
{
    m = (l+h) / 2
    {
```

Case I :
mid * mid == N
return mid

Case 2 :
mid * mid < N

```
        //conditions
    }
}
return ans;
}
```

ans = mid
l = mid + 1

case 3:
mid * mid > N
h = mid - 1;