Agenda

i) search single element in sorted array.

ii) find k in rotated sorted array

iii) find sqrt(N) in $\log_2 N$ complexity

Q-1 Given a sorted array in which all elements are coming twice except for a single element. Find the single element.

Expected TC: $O(\log_2 n)$

	0	1	2	3	4	5	6	7	8	9	10	11	12
A[] =	1	1	3	3	**5**	6	6	7	7	10	10	12	12

	0	1	2	3	4	5	6	7	8	9	10	11	12
A[] =	1	1	3	3	5	5	6	6	8	8	**10**	12	12

i) Brute force idea: XOR of array

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

ii) can we use binary search : Yes

A[] = 1 1 3 3 5 6 6 7 7 10 10 12 12

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

Before single element

first value of pair: even index

second value of pair: odd index

Go to right

After single element

first value of pair: odd index

second value of pair: even index

Go to left

A[] = 1 1 3 3 5 6 6 7 7 10 10 12 12

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

lo m hi

m	fvi	svi	direction
6	5	6	left
2	2	3	right
4			

fvi → odd (go to left)

fvi → even (go to right)

A[] = 1 1 3 3 5 5 6 6 8 8 10 12 12

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

lo hi m

m	fvi	svi	direction
6	6	7	right
9	8	9	right
11	11	12	left
10			

A[] = 1 1 3 3 5 5 6 6 8 8 10 12 12

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

lo m hi

→ corner cases

```
if (A[m] != A[m-1] && A[m] != A[m+1]) {
        return A[m];
}
else if (A[m-1] == A[m]) {
        jvi => m-1
        // take decision based jvi
}
else if (A[m] == A[m+1]) {
        jvi => m
        // take decision based jvi
}
```

jvi → odd (go to left)

jvi → even (go to right)

m	jvi	dir
6	A[m] == A[m+1] jvi = 6	right
9	A[m] == A[m-1] jvi = 8	right
11	A[m] == A[m+1] jvi = 11	left
10	got the ans	

Q.2 Given a **rotated sorted array** containing distinct elements. Search K in array.

Expected TC: $O(\log_2 n)$

	0	1	2	3	4	5	6	7	
A[ ] =	40	50	60	70	80	10	20	25	K = 10

	0	1	2	3	4	5	6	7	
A[ ] =	90	100	10	20	25	30	45	50	K = 45

	0	1	2	3	4	5	6	7
A[ ] =	40	50	60	70	80	10	20	25

	0	1	2	3	4	5	6	7	
A[ ] =	10	20	25	40	50	60	70	80	10 20 25

	0	1	2	3	4	5	6	7
A[ ] =	90	100	10	20	25	30	45	50

	0	1	2	3	4	5	6	7	
A[ ] =	10	20	25	30	45	50	90	100	10 20 25 30 45 50

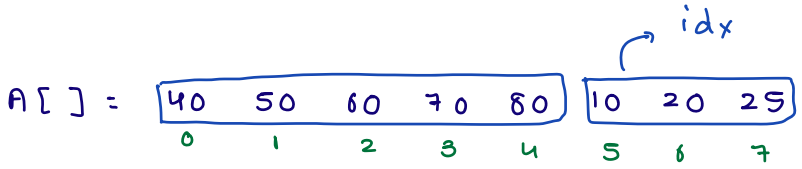
valid rotated sorted array or not

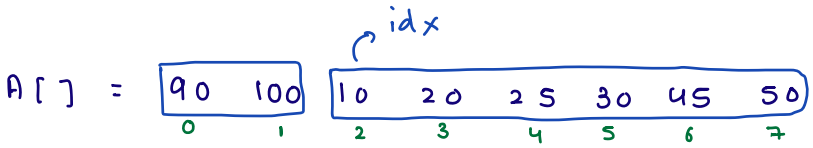
i) 90 100 110 10 25 5 ✗

5 10 25 90 100 110

ii) 90 100 110 10 25 ✓

10 ~~25~~ | 90 100 110 10 25





Idea1 : i) find min element : idx (Binary search : todo)

- binary search → 0 to idx-1
- binary search → idx to n-1

/OR

Idea2 : can we do it in single : yes

array from lo to mid sorted take decision based on that

90 100 110 120 130 | 10 25

array from mid to hi sorted take decision based on that.

90 100 | 10 20 35 45 50

Atleast one of the part (lo to mid | mid to hi) is definitely sorted.

A[ ] = 40 50 60 70 80 10 20 25 K = 10

0 1 2 3 4 5 6 7

lo m hi

A[ ] = 90 100 10 20 25 30 45 50 K = 45

0 1 2 3 4 5 6 7

lo hi

m

A[ ] = 90 100 10 20 25 30 45 50 K = 100

0 1 2 3 4 5 6 7

lo m hi

```
int lo = 0, hi = n-1;

while (lo <= hi) {
    int mid = (lo+hi)/2;
    if( A[mid] == k) {
        return mid;
    }
    else if ( A[lo] < A[mid]) {
        // lo to mid is sorted
        if ( k >= A[lo] && k < A[mid]) {
            hi = mid-1;
        }
        else {
            lo = mid+1;
        }
    }
    else {
        // mid to hi sorted
        if ( k > A[mid] && k <= A[hi]) {
            lo = mid+1;
        }
        else {
            hi = mid-1;
        }
    }
}

return -1;
```

k = 5

A =

90	100	5	12	19	21	25
0	1	2	3	4	5	6

hi
lo
m

m	which part sorted	direction
3	m to hi (3, 6)	left
1	lo to m (0, 1)	
2	got the answer	

Q.3 Given N, find square root of N in $\log_2 N$ complexity.

Note: only integral part of answer is required.

N = 9, ans = 3

N = 11, ans = 3

N = 15, ans = 3

N = 18, ans = 4

N = 16, ans = 4

N

ans: 1 to N/2

can you do it using binary search : Yes

N = 18

1 2 3 4 5 6 7 8 9

lo hi

m

ans = ~~1~~ ~~3~~ 4

m = 5, 5*5 > 18

m = 2, 2*2 <= 18

m = 3, 3*3 <= 18

m = 4, 4*4 <= 18

```
if (mid * mid <= N) {
    ans = mid;
    lo = mid + 1;
}
else {
    hi = mid - 1;
}
```

```
if (mid * mid <= N) {
    ans = mid;
    lo = mid + 1;
}
else {
    hi = mid - 1;
}
```

N = 21

1	2	3	4	5	6	7	8	9	10
			hi	lo					
			m						

mid = 5, 5 * 5 > 21

mid = 2, 2 * 2 <= 21

mid = 3, 3 * 3 <= 21

mid = 4, 4 * 4 <= 21

ans = ~~3~~ 4

```
int sqrt (int N) {
    int lo = 1, hi = N/2;
    int ans = 0;
    while (lo <= hi) {
        int mid = (lo + hi) / 2;
        if (mid * mid <= N) {
            ans = mid;
            lo = mid + 1;
        }
        else {
            hi = mid - 1;
        }
    }
    return ans;
}
```

$1 \le N \le 10^9$

a * b <= c

a <= c / b

If getting issues with big TC then

mid <= N / mid