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

→ Brute force idea : applying XOR on entire array

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

→ can we apply search :

	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

Before the single element

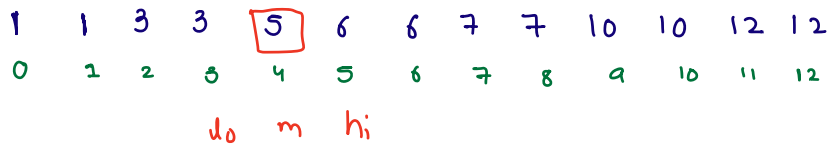
1st value of pair = even

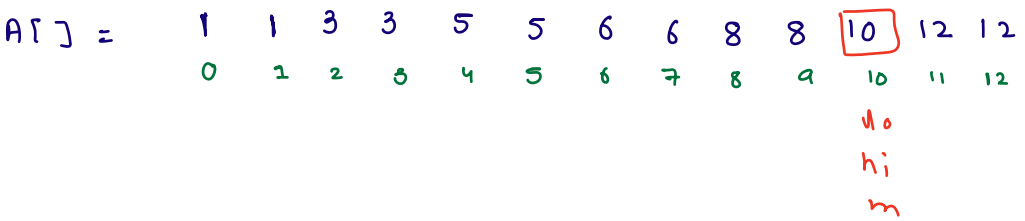
2nd value of pair = odd

After the single element

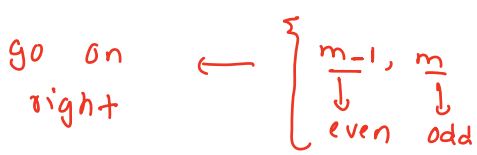
1st value of pair = odd

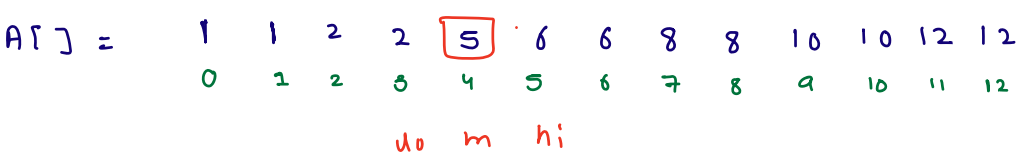
2nd value of pair = even


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


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


Go on right ←
m, m+1
even odd
Go to left →
m, m+1
odd even


go on right ←
m-1, m
even odd


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

```java
if(A.length == 1) {
    return A[0];
}
if(A[0] != A[1]) {
    return A[0];
}
if(A[n-1] != A[n-2]) {
    return A[n-1];
}


int lo = 0, hi = n-1;
while(lo <= hi) {
    int m = (lo + hi)/2;

    if(A[m] != A[m-1] && A[m] != A[m+1]) {
        return A[m];
    }
    else if((m % 2 == 0 && A[m-1] == A[m]) || (m % 2 != 0 && A[m] == A[m+1])) {
        //go on left
        hi = m-1;
    }
    else {
        //go on right
        lo = m+1;
    }
}
```

m-1, m → odd

m, m+1 → odd

} ⇒ if index of 1st value in pair is odd then go on left.

TC: $O(\log_2 n)$

SC: $O(1)$

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

ji = index of first value in pair

si = index of second value in pair

m	
6	A[m] = A[m+1] 6 7 ji = even si = odd
9	A[m-1] = A[m] 8 9 ji = even si = odd
11	A[m] = A[m+1] 11 12 ji = odd si = even
10	Got 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

→ Brute force: linear search TC: O(n)

→ can we apply binary search : ?

simple idea:

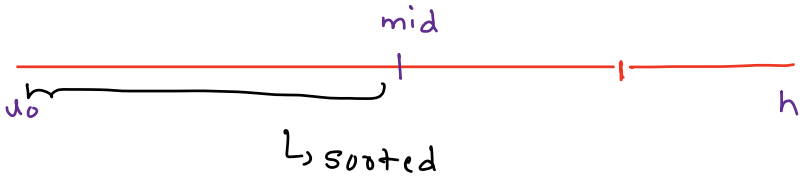
i) find smallest element : idx [todo]

- └ binary search => 0 to idx-1
- └ binary search => idx to n-1

next idea:

i) can we do it in single traversal : yes

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



50 60 90 100 | 10 20

↳ sorted

(take decision based on lo to mid)

50 60 | 10 12 15 20

↳ sorted

(take decision based on mid to hi)

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

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

k = 5

hi

lo

m

if (A[mid] == k) {

return ans;

}

else if (A[lo] < A[mid]) {

// lo to mid is sorted

}

else {

// mid to hi is sorted

}

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

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

Note: only integral part of answer is required.

can we apply binary search : Yes

N = 9 ans = 3

N = 11 ans = 3

N = 15 ans = 3

N = 16 ans = 4

sqrt(N) ⇒ 1 to N/2

N = 20

1 2 3 4 5 6 7 8 9 10

hi lo

ans = ~~2~~ ~~3~~ 4

m = 5, 5 * 5 > 20

m = 2, 2 * 2 <= 20

m = 3, 3 * 3 <= 20

m = 4, 4 * 4 <= 20

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

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

$0 \leq N \leq 10^9$

$a * b <= c$

$\Rightarrow a <= \frac{c}{b}$ ✓

Doubts

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 in pair: even

2nd value in pair: odd

go on right

After single element

first value in pair: odd

2nd value in pair: even

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

```
if (A[m] != A[m-1] && A[m] != A[m+1]){
        return A[m];
}
else if( A[m-1] == A[m]) {
        int ji= m-1;
        // based on ji take decision
}
else if ( A[m] == A[m+1]) {
        int ji = m;
        // base on ji take decision
}
```

m-1, m

ji = m-1

↳ odd

m, m+1

ji = m

↳ even

matrix Search

n=3 m=4

idx = cell no.

	0	1	2	3
0	10 (0)	18 (1)	20 (2)	22 (3)
1	24 (4)	28 (5)	29 (6)	31 (7)
2	32 (8)	48 (9)	59 (10)	64 (11)

i = idx / m ;

j = idx % m ;

idx => 0 to n*m-1

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

lo mid hi

n -> rows

m -> cols

mid=5, i = 1 , j=1

```
int lo = 0, hi = n * m-1 ;
while (lo <= hi ) {
    int mid = (lo + hi ) /2 ;
    int i = mid / m ;
    int j = mid % m ;
    if(A[i][j] == k ) -> got the ans
    else if (A[i][j] < k) -> go on right, lo = mid+1
    else -> go on left, hi = mid - 1 ;
}
```