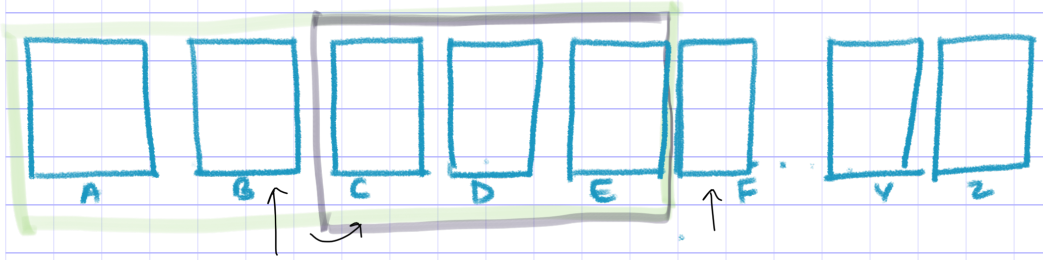
SEARCHING 1

Say u're given a collection of elements & u're asked to tell whether a particular ele is present or not in mat collection.

How will you search for it?

— LINEAR SEARCH

$O(n)$



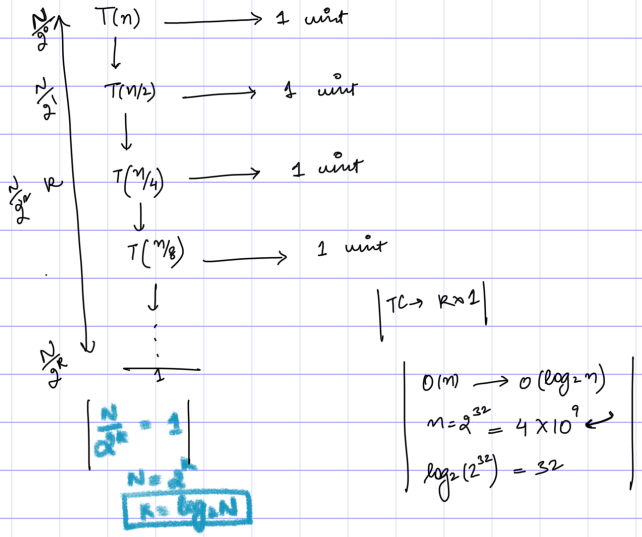
word → Den

Because dictionary is sorted, we're able to go like that

Sorted Array

will you randomly land on any index or where?

Search space: Area where we are applying binary search.



$T(n) = T\left(\frac{n}{2}\right) + 1$

$T(n) = T\left(\frac{n}{4}\right) + 1 + 1$

$T(n) = T\left(\frac{n}{8}\right) + 1 + 1 + 1$

$\vdots$

$T(n) = T\left(\frac{n}{2^k}\right) + k$

$T(n) = 1 + k$

$T(n) = k$

Each step takes 1 unit of time

Steps → R

Time = R×1

	0	1	2	3	4	5	6	7	8	9
ar[] :	3	6	8	12	14	19	20	23	25	27

K = 12

Search space changed
Therefore it needs to be updated
How can I keep track of my search space?
↳ defined by start & end pt.

So keep two pointers start & end

0	1	2	3	4	5	6	7	8	9
3	6	8	12	14	19	20	23	25	27

s	e	mid $\left[\frac{s+e}{2}\right]$			
0	9	4	14 ≠ 12	14 > 12	e = mid − 1
0	3	1	6 ≠ 12	6 < 12	s = mid + 1
2	3	2	8 ≠ 12	8 < 12	s = mid + 1
3	3	3	12 = 12	True.	

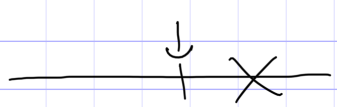
When both are same, will you stop & return?
– NO
Because we don't know what that ele is

Code

```
int binarySearch ( ar[] , int n , int K) {
    s=0 , e = n-1
    while ( s <= e) {
        mid = (s+e)/2
        if(ar[mid] == K) {
            return mid
        }
        else if (ar[mid] < K){
            s = mid +1
        }
        else { e = mid - 1}
    }
    return -1
}
```

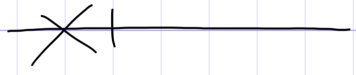
TC → O(log n)
SC → O(1)

Why mid?



Best 50
Worst 50

Best 70
Worst 30



Best 80
Worst 20

Q1. Find first occurrence of an element in sorted array.

0	1	2	3	4	5	6	7	8
2	4	10	10	10	18	20	20	27

K=4 → 1

K=10 → 2

K=20 → 6

K=19 → -1

```
int BS ( arr, n, K) {
    s = 0    e = n-1    ans = -1
    while( s <= e) {
        m = (s+e)/2
        if( K == arr[m]) {
            ans = m
            e = m-1
        }
        else if( K < arr[m]){
            e = m-1
        }
        else s = m+1
    }
    return ans
}
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	2	2	10	10	10	18	18	20	20	20	20	20	28

K=20

s e m

0	13	7	20 > 18	s = m + 1
8	13	10	20 == 20	store the answer & move to left
8	9	8	20 == 20	store & move to left

[Rule of BS says to discard one of the half everytime]

TC → $O(\log_2 n)$
SC → $O(1)$

Q. Find no. of occurrences of an element.

[2 3 3 4 4 4 5 6 7 7 7 7]

K = 4 → 3
K = 6 → 1
K = 7 → 4
K = 1 → 0

just like above question,
find first as well as last occurrence
Then last − first + 1 is your answer

Q. Search an ele in sorted but rotated array

Rotated sorted array is given with unique elements.
and K is given i.e the no. of times array has been rotated

Eg.

	0	1	2	3	4	5	6	7	8	9	10
ar[ ] = {	-15	-9	-6	0	2	5	8	10	14	18	23 }

K = 6

23	-15	-9	-6	0	2	5	8	10	14	18
18	23	-15	-9	-6	0	2	5	8	10	14
14	18	23	-15	-9	-6	0	2	5	8	10
⋮										
5	8	10	14	18	23	-15	-9	-6	0	2

K

Solution

1) linear search?

TC O(n)
SC O(1)

2)

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

s=0 e=K-1 Apply BS

s=K e=n-1 Apply BS

TC $O(\log n) + O(\log n) = O(\log n)$

SC O(1)

If K = 11 Rotations 0
K = 15 Rotations 4
K = 24 Rotations 3

Actual rotations → K%N.

K = K%N

Actual Question

K isn't given

	0	1	2	3	4	5	6	7	8	9	10
arr[] =	5	8	10	14	18	23	-15	-9	-6	0	2

Observation: K is nothing but the least val of array [pivot]
How to find it?

	0	1	2	3	4	5	6	7	8	9	10
arr[] = {	5	8	10	14	18	23	-15	-9	-6	0	2 }

If mid lands somewhere here, goto right
s = mid + 1

If mid lands somewhere here, store answer go to left for better answer.

How to know where mid lies?

After rotation

left sorted array > right sorted array

first ele of left sorted array > all ele of right sorted array.

```
ans = -1
if (arr[0] > arr[mid]) { // mid is on right side
    ans = mid
    e = mid - 1
}
else {
    s = mid + 1
}
```

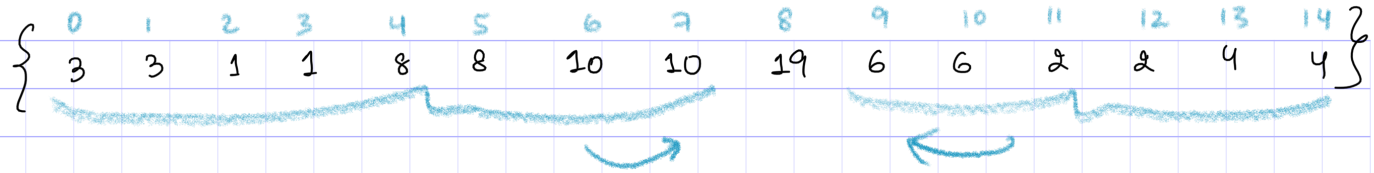
Q: Given N array elements, all elements repeats twice except one repeated elements are adjacent to each other.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
{	3	3	1	1	8	8	10	10	19	6	6	2	2	4	4 }

Brute Force

1) Linearly iterate over the array
   O(n) SC O(1)

2) XOR of all ele $a \wedge a = 0$
   O(n)
   SC O(1)



BS in general

Look at an ele

// go to right if answer is on right

// go to left if answer is on left

For what all indices you will go to right?

0 – 7

For what all indices will you go to left?

9 – 14

How to know where is mid?

Hint: look at index val of first occurrence before & after unique element

Observation:

Before: 1st Occ → Even

After: 1st occ → Odd

```
if(mid % 2 == 0){
    // move right
}
```

```
if (mid % 2 == 1){
    // move left
}
```

Issue: Is there a guarantee that mid will lie on first occurrence?

```
if( arr[mid] == arr[mid - 1]){
    mid = mid - 1
}
```

```
int nonRepeated(A, n){
    if(n==1) return A[0]
    if(A[s] != A[s+1]) return A[s]
    if(A[e] != A[e-1]) return A[e]
    s=0, e=n-1
    while(s<=e){
        mid = (s+e)/2
        if(A[mid] != A[mid-1] && A[mid] != A[mid+1]){
            return mid
        }
        if(mid!=0 && A[mid] == A[mid-1]) mid = mid-1
        if(mid %2 ==0) s = mid+1
        else e = mid-1
    }
    return -1
}
```

0	1	2	3	4	5	6	7	8	9	10
2	5	8	10	14	18	23	-15	-9	-6	0

s	e	m	A[m]	first ele (2)	
0	10	5	18	18 > 2	move to right s = m+1
6	10	8	-9	-9 < 2	store as ans ans = 8 move left e = m-1
6	7	6	23	23 > 2	move right s = m+1
7	7	7	-15	-15 < 2	ans = 7 e = m-1
8	7				

over