Search 1: Binary Search On Array

AGENDA

- Introduction
- Search for an element K
- Search first & last occurence
- Single element in sorted array
- Peak element
- Local minima

★ Mock Interview → (30 days)

↳ DSA ends

↳ 5 Mandatory / 7 Additional

↳ SQL → Mock

↳ LLD → Mock

⇒ Search Story

Lost Boy/Girl ⟶ Police → Target (what to Search)
Police → Search Space (where to Search)

Q where the search will be faster

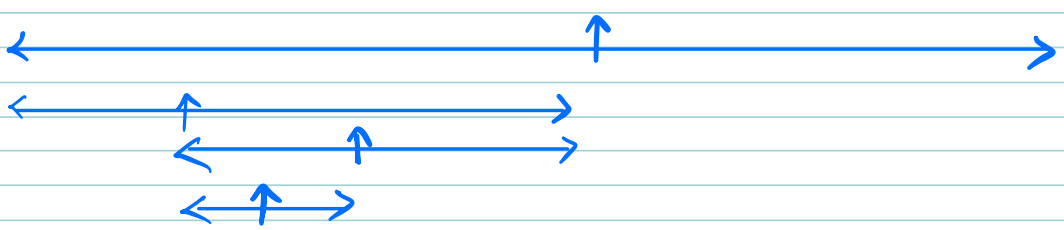
→ word :- {Dictionary, book, Newspaper}

→ Phone no. :- {Telephone directory, diary}

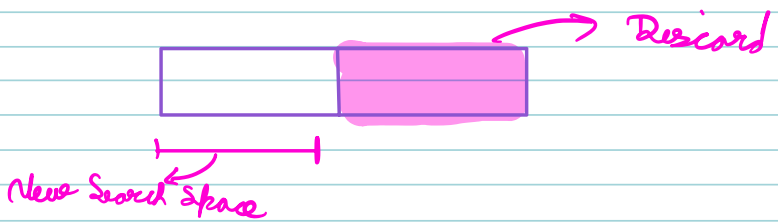
* Learning ⇒ where Search space is sorted, there Searching ~~sorting~~ becomes easy. But we should not says searching is easy only when search space is sorted. I will Learn from examples today

Ex :- Search word (DOG) in Dictionary

A B C D E F .... M N O .... X Y Z



⇒ IDEA of Binary search



Quiz 1 :- In binary search, at each step, the search range is typically :-

↳ Halved

BINARY SEARCH

To apply Binary Search :-

1. Target
2. Search space
3. Some condition to discard 1 half of search space.

Defination :- Divide search space into 2 parts & repeatedly keep on neglecting one half of the search space.

Question :- Given a sorted array with distinct elements, search the index of an element K, if K is not present return -1.

Ex :-

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

if, K = 14 → 4 Ans

if, K = 99 → -1 Ans

Brute force

↳ Linear Search

TC → O(N)

Ex :- A[] = 3 6 9 12 14 19 20 23 25 27 (0 1 2 3 4 5 6 7 8 9)

⇒ OPTIMIZATION

↳ Binary Search

① Target :- K

② Search space :- Entire array

③ Discard Condition.

↳ (i) arr[mid] == K
return mid.

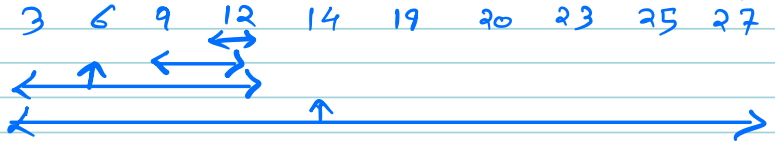
(ii) arr[mid] < K
goto Right

(iii) arr[mid] > K
goto Left

DRY RUN

K = 12

Ex :- A[] = 3 6 9 12 14 19 20 23 25 27 (0 1 2 3 4 5 6 7 8 9)



Left	Right	mid	
0	9	4	goto Left ↳ (Right = mid-1)
0	3	1	goto Right ↳ (Left = mid+1)
2	3	2	goto Right ↳ (Left = mid+1
3	3	3	break;

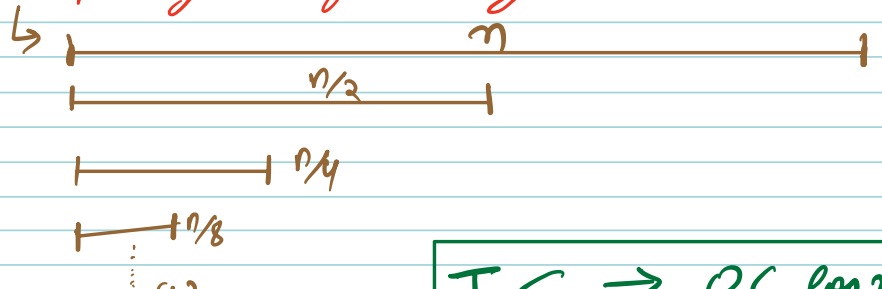
if K = 11

3	2	break	goto Left ↳ Right = mid-1

PSEUDO CODE

```
int Search ( int [] A, int N, int K) {
    L = 0 ;
    R = N-1;
    while ( L <= R) {
        mid = (L+R)/2
        if ( A[mid] == K) {
            return mid;
        } else if (A[mid] < K){
            L = mid + 1;
            // move right;
        } else {
            R = mid-1;
            // move left
        }
    }
    return -1;
}
```

Quiz ?:- If the element 'K' is found in a sorted array, what will be the time complexity of binary search



TC → O(log n)
SC → O(1)

⇒ Best practice

$$mid = \frac{Left + Right}{2} \longrightarrow \boxed{mid = Left + \frac{(Right - Left)}{2}}$$

Mathematically

$$Left + \frac{(Right - Left)}{2}$$

$$\Rightarrow \frac{2Left + Right - Left}{2}$$

$$\Rightarrow \frac{Left + Right}{2}$$

Q why ?

Explanation int $\longrightarrow 2^{31} - 1$

Left assume int can store max 100

if Left = 98 , Right = 99

① then, $mid = \frac{Left + Right}{2} \Rightarrow \frac{98+99}{2}$ → overflow

② while, $mid = Left + \frac{(Right - Left)}{2} \Rightarrow 98 + \frac{(99-98)}{2}$

$\Rightarrow 98 + \frac{1}{2}$ ✓

Question :- Given a sorted array of n elements. Find the index of first occurence of K.

Ex :-

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
-5	-5	-3	0	0	1	1	5	5	5	5	5	5	5	8	10	15	17	18

↳ K = 5 ⟶ 7 **Ans**

Brute force

↳ Linear Search

TC ⟶ O(N)

↳ **OPTIMIZATION**

↳ Binary Search

① Target :- K

② Search Space :- entire array.

③ Conditions to Discard -

↳ (i) A[mid] == k

Quiz :- when searching for the first occurrence of an element in a sorted array, what should you do if the current element matches the target K?

```
ans = mid;
goto Left
Right = mid-1;
```

(ii) A[mid] < k

```
goto right;
Left = mid + 1
```

(iii) A[mid] > K

goto Left;

Right = mid - 1;

DRY RUN

K = 5

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
-5	-5	-3	0	0	1	1	5	5	5	5	5	5	5	8	10	15	17	18

Left	Right	mid	Ans	
0	18	9	9	→ Goto Left, Right = mid-1
0	8	4	(9)	→ Goto Right, Left = mid+1
5	8	6	(9)	→ Goto Right, Left = mid+1
7	8	7	7	→ Goto Left, Right = mid-1
7	6	→	break;	

```
int Search ( int [] A, int N, int k) {
    L = 0 ;
    R = N-1;
    ans = -1;
    while ( L <= R) {
        mid = (L+R)/2

        if ( A[mid] == K) {
            ans = mid
            Right = mid-1;
        } else if (A[mid] < k){
            L = mid + 1;
            // Move right;
        } else {
            R = mid-1;
            // Move Left
        }
    }
    return ans;
}
```

Quiz 4 :- what is the TC of finding the first occurrence of an element in a sorted array using binary search ?

↳

TC → O(log N)

SC → O(1)

Q :- Find the last occurrence ?

↳ H.W.

10:29 — 10:39

Question :- Every element occurs twice except for 1, find the unique element.

Note:- Duplicate elements are adjacent to each other but the array is not sorted.

Ex :-

	0	1	2	3	4	5	6
A =	8	8	5	5	6	2	2

↳ Ans

Brute force

↳ Using XOR over array

TC → O(N)

SC → O(1)

↳ **OPTIMIZATION**

↳ Binary Search.

Ex :-

0	1	2	3	4	5	6	7	8	9	10	11	12
3	3	1	1	8	8	10	10	19	6	6	24	24

(i) Target → unique element

(ii) Search space → entire array

(iii) Conditions to discard

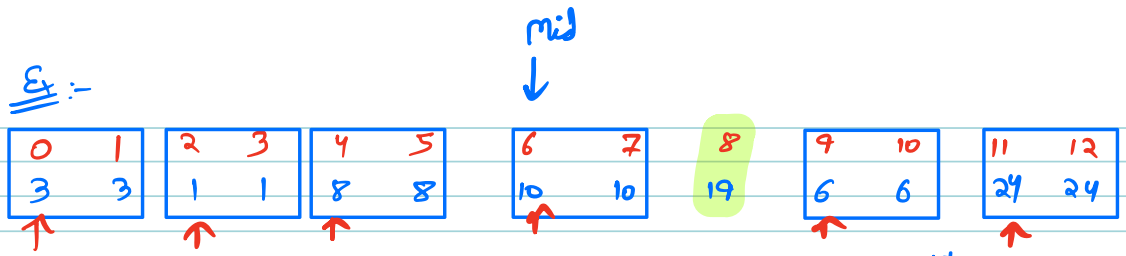
Observations

(1) For repeated elements what you can observe.

(2) Difference b/w the pair on Left & Right of unique?

↳ First occurrence of Repeated on left is even indexed while on Right its odd indexed.

Ex :-



(L) Left	(R) Right	(M) Mid	isUnique	(check Am I not first occurrence) A[mid] == A[mid-1]	mid%2
0	12	6	*	*	yes, Go to Right. ↳ L = M+2
8	12	~~10~~ 9	*	mid = mid-1	No, Go to Left ↳ R = M-1
8	8	8	yes	return;	

PSEUDO CODE

```
find Unique ( int[] A, int N) {
    L = 0
    R = N-1
    if ( N == 1 ) { return 0; }
    if ( A[0] != A[1] ) { return 0; }
    if ( A[N-1] != A[N-2] ) { return N-1; }
    while ( L <= R ) {
        M = L + (R-L)/2
        if ( A[M] != A[M-1] && A[M] != A[M+1] ) {
            return M;
        }
    }
}
```

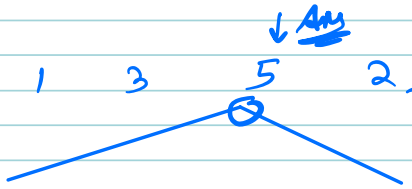
TC → O(log N)

SC → O(1)

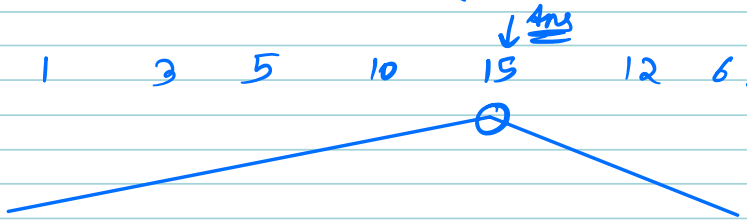
```
if ( A[m] == A[m-1]){
        m = m-1;
}
if ( m % 2 == 0){
        l = m+2;
} else {
        r = m-1;
}
}
```

Question :- Given an increasing decreasing array with distinct elements. Find max element.

Ex1:- A = [1 3 5 2]



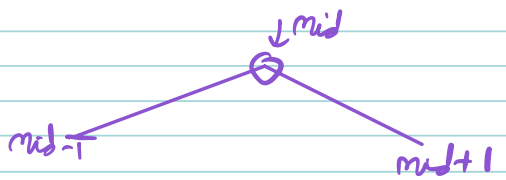
Ex2:- A = [1 3 5 10 15 12 6]



APPROACH

1. Target :- peak element
2. Search space :- array
3. Condition to discard.

case I

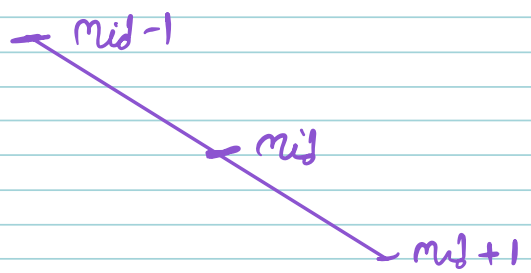


```
if( A[mid] > A[m-1] &&
    A[mid] > A[mid+1]){
    return mid;
}
```

Case II


```
if ( A [mid] > A[m-1] &&
     A[mid] < A[mid+1]){
    L = mid+1;
    // Go right
}
```

Case III



else

```
{ if ( A [mid] < A[ mid-1] &&
       A[mid] > A[mid+1])}
    R = mid-1;
    // Move left;
}
```

Edge Case

① 1 3 5 10

② 15 12 10 8

③ Single element

Question:- Given an array of N distinct elements, find any local minima in the array.

Local Minima :- a no. which is smaller then its adjacent neighbours.

Ex :-

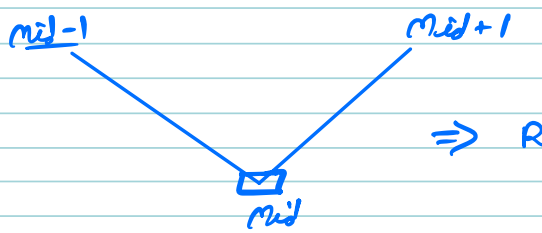
A ⇒ 3 6 1 0 9 15 8

B ⇒ 21 20 19 17 15 9 7

C ⇒ 5 9 15 16 20 21

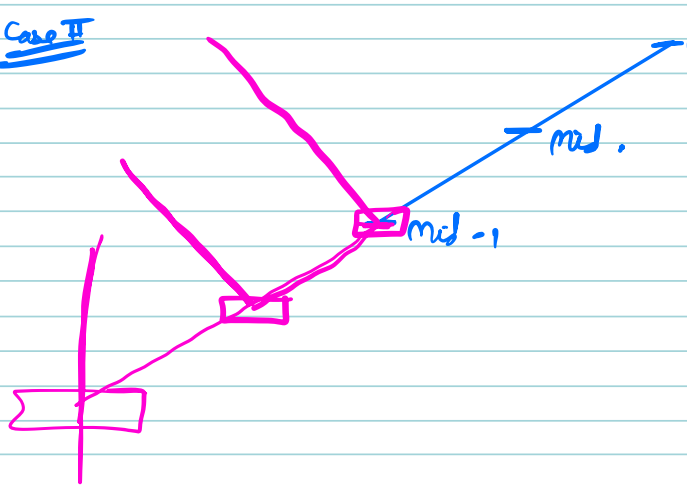
D ⇒ 5 8 12 3

Case I



⇒ Return mid

Case II



Right Side → Not Sure
// Left Side → Sure
Goto Left

Case III

CLUBBING Together

Case IV

=> Left → Not Sure

=> Right → Sure.

=> Left → Sure

=> Right → Sure

[Anywhere]

PSEUDO CODE

```
L = 0 ,        R = N-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] }
while ( L <= R) {
        m = L + (R-L)/2

        if ( A[m] < A[m-1] &&
             A[m] < A[m+1]){
                return A[mid];
        }
        else if ( A[m] < A[m-1]){
                L = mid+1;
        } else {
                R = mid-1;
        }
}
```

3

DRY RUN

idx →	0	1	2	3	4	5	6	7
arr →	4	8	2	7	6	4	1	5

lo	hi	mid	A[mid-1]	A[mid]	A[mid+1]	
0	7	3	2	7	6	→ Go to right
4	7	5	6	4	1	→ Go to right
6	7	6	4	1	5	return 1

TC → O(log N)

SC → O(1)