Searching 1 : Binary Search on Array

Search space → e.g. Library

Target → e.g. Book

Condition → helps in finding target by reducing search space.

Binary Search → divide search space into 2 equal parts & keep neglecting one part based on condition.

organised data ⇒ think about binary search

Question 1

Given a sorted array of distinct elements. find index of a given element k, if not present, return -1.

$$A = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline 3 & 6 & 9 & 12 & 14 & 19 & 20 & 23 & 25 & 27 \end{array}$$

Bruteforce

```
for (i=0 to n-1) {                // linear search
    if (A[i] == K)
        return i
}                                 TC = O(N)
return -1                         SC = O(1)
```

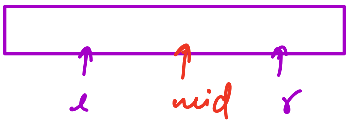
Binary Search → 3 steps

// 1. Define Search Space [0, n-1] // index l to r

```
l=0 , r=n-1

while ( l <= r ) {
```

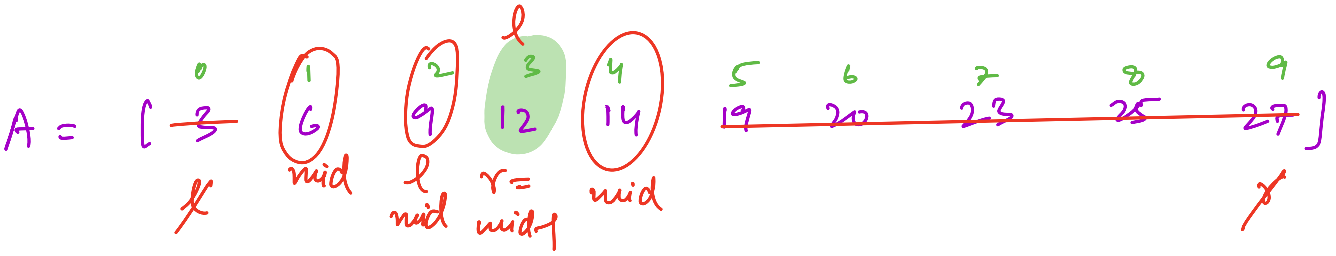
// 2. check if middle element is answer ?

```
    mid = (l+r)/2

    if( A[mid] == K)
        return mid
```



// 3. Decide whether to go left or right.

```
    if( K < A[mid])
        r = mid-1
    else
        l = mid+1
}
return -1
```



K = 12

l	r	mid = (l+r)/2		
0	9	(0+9)/2 = 4	K < A[4]	→ go left
0	3	1	K > 6	→ go right
2	3	2	K > 9	→ go right
3	3	3	K = 12	

TC: $N \to N/2 \to N/4 \to \ldots\ldots \to 1 \to 0$
↑ stop

iterations = $\log_2 N$

TC = $O(\log N)$

SC = $O(1)$

Best Practice

use used $mid = \frac{(l+r)}{2}$

assume INT_MAX = 100

$l = 80,\ r = 90 \qquad \frac{(l+r)}{2} = \frac{(80+90)}{2} = \frac{\boxed{170}}{2}$ overflow

$$\frac{(l+r)}{2} \Rightarrow l + \frac{l}{2} - l + \frac{r}{2}$$

$$\Rightarrow l + \frac{r}{2} - \frac{l}{2} \Rightarrow l + \frac{(r-l)}{2}$$

$$l = 80,\ r = 90 \qquad \Rightarrow l + \frac{(r-l)}{2} \Rightarrow 80 + \frac{(90-80)}{2}$$

$$\Rightarrow 80 + \frac{10}{2} = 80 + 5$$

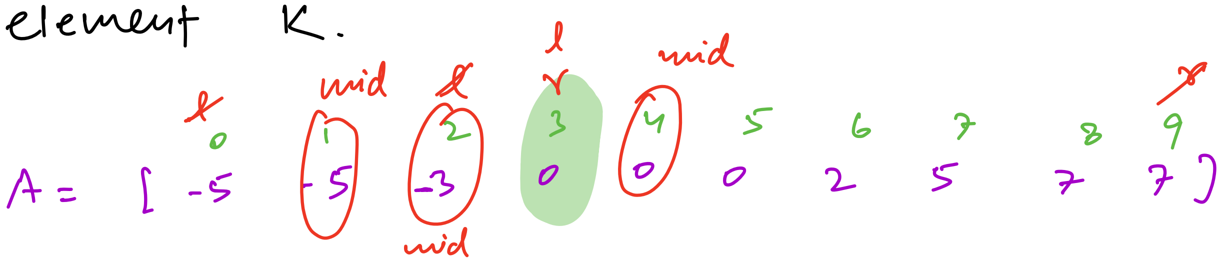
$$= 85$$

Question

Given array of email years, return the index of first email of a given year.

$$A = [\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2005 & 2005 & 2013 & 2018 & 2018 & \boxed{2020} & 2020 & 2023 \end{array}]$$

Given a sorted array, find first index of given element K.



K=0 ans=3

K=1 ans=-1

//1. Define Search Space [0, n-1] // index l to r

```
l=0 , r=n-1
while ( l <= r ) {
//2. check if middle element is answer ?
        mid = l + (r-l)/2
        if( A[mid] == K && ( mid==0 || A[mid-1] != K))
            return mid

//3. Decide whether to go left or right.
        if( K <= A[mid])
            r = mid-1
        else
            l = mid+1
}
```

K < mid → go left

K > mid → go right

K = mid → go left

return -1

TC = O(logN)
SC = O(1)

Question 3

Given an array where every element occurs twice except for 1 element that appear once. Find the unique element. All equal pairs of elements are together. (unsorted but organized)

not equal

$$\begin{array}{lccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \\ A = [& 8 & 8 & 2 & 2 & \boxed{6} & 5 & 5 &] \end{array}$$

ans = 6

Idea 1: ans = XOR of all elements

TC = O(N) SC = O(1)

```
//1. Define Search Space   [0, n-1]  // index l to r

    l=0 , r=n-1

    while ( l <= r ) {
//2. Check if middle element is answer ?
        mid = l + (r-l)/2
```

```
if((mid==0 || A[mid-1] != A[mid]) &&
   (mid == n-1 || A[mid+1] != A[mid]))
        return A[mid]
```

//3. Decide whether to go left or right

$$A = [\underbrace{\overset{l\ 0}{8}\ \ \overset{1}{8}}_{\text{even odd}}\ \ \overset{2}{2}\ \ \underbrace{\boxed{\overset{3}{2}}}_{\text{mid}}\ \ \overset{4}{6}\ \ \underbrace{\overset{5}{5}\ \ \overset{6}{5}}_{\text{odd even}}\ \overset{r}{]}$$

8, 8 → (0,1)

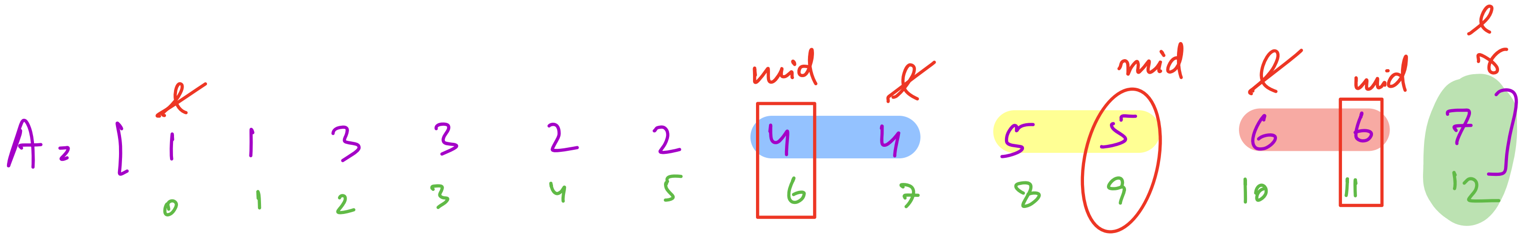
2, 2 → (2,3)

5, 5 → (5,6)

```
if ( mid == 0 || A[mid-1] != A[mid] ) { // (mid, mid+1) pair

      if (mid % 2 ==0) // (even-odd) pair
            l = mid+1

      else       // (odd-even) pair
           r = mid-1

}
else {   // (mid-1 , mid) pair

    if (mid % 2  ==0)  // (odd-even) pair
         r = mid-1
    else
        l = mid+1
    }
}
```

TC = O(logN)

SC = O(1)

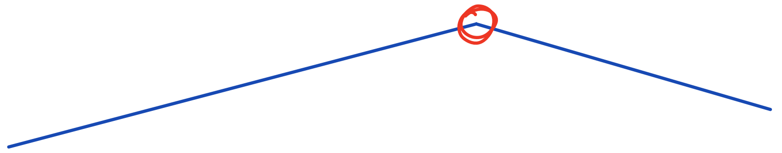


Question 4

Given an increasing - descreasing array. find the max element. (Peak element)

	0	1	2	3	4	5	6
A =	[1	3	5	10	15	12	6]

ans = 15

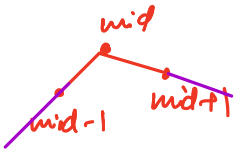


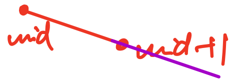
//1. Define Search Space [0, n-1] // index l to r

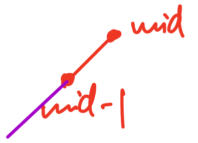
```
l=0 , r=n-1
while ( l <= r ) {
```

//2. check if middle element is answer ?

```
mid = l + (r-l)/2

if( (mid==0 || A[mid-1]  <  A[mid]) &&
    (mid == n-1 || A[mid+1]  <  A[mid]))
        return  A[mid]
```

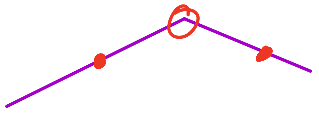


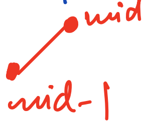


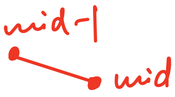


//3. Decide whether to go left or right

```
if( mid ==0 || A[mid-1] < A[mid] )
        l = mid+1

else
        r = mid-1

}
```







TC = O(log N)

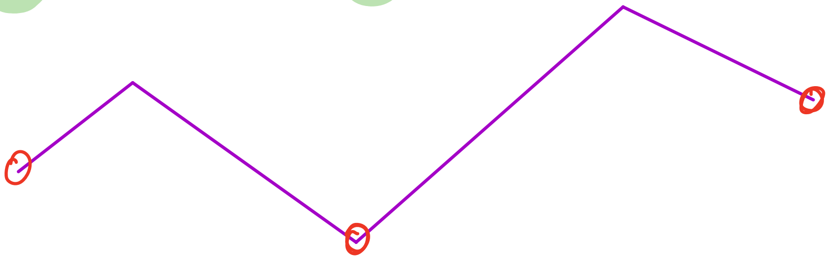
SC = O(1)

Question 5

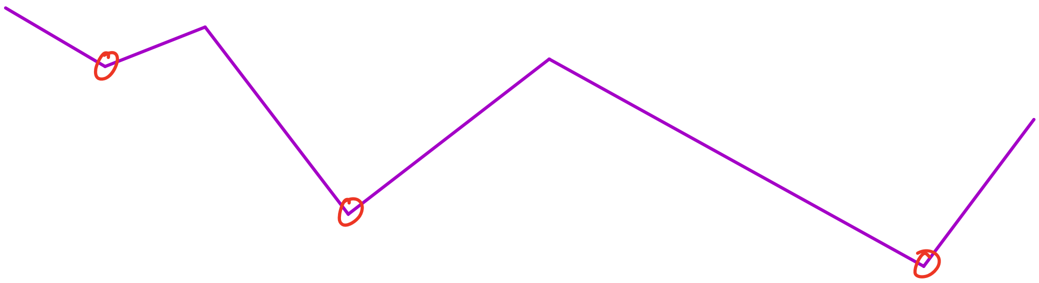
Given an array of distinct elements, find any one local minima. i.e,

$$\boxed{A[i-1] > A[i] < A[i+1]}$$

	0	1	2	3	4	5	6
A= [	3	6	1	0	9	15	8]



A= [9 7 8 3 5 6 2 1 0 4]



Brutforce :

$\forall i$, check if A[i] is minima

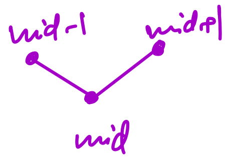
TC = O(N)   SC = O(1)

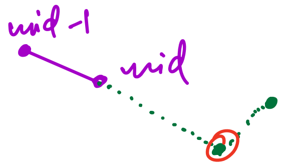
search in TC < O(N) ?

↙

Binary Search

```
//1. Define Search Space  [0,n-1]  // index l to r

    l=0 ,  r=n-1

    while ( l <= r ) {
//2. check if middle element is answer ?
          mid = l + (r-l)/2

          if((mid==0 || A[mid-1] > A[mid]) &&
             (mid == n-1 || A[mid+1] > A[mid]))
                return A[mid]

//3. Decide whether to go left or right

          if( mid==0 || A[mid-1] > A[mid]) {
                l = mid+1
          }
          else {
                r = mid-1
          }
    }
```





TC = O(log N)

SC = O(1)