SCALER

Searching - 1

TABLE OF CONTENTS

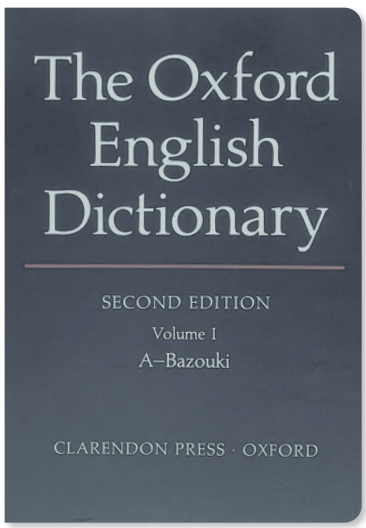
1. Searching
2. Binary Search
3. Problems on Binary Search

Searching

arrangement

Searching a word in newspaper

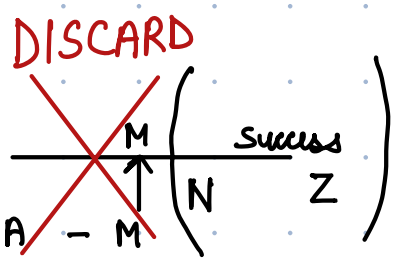
Searching a word in dictionary



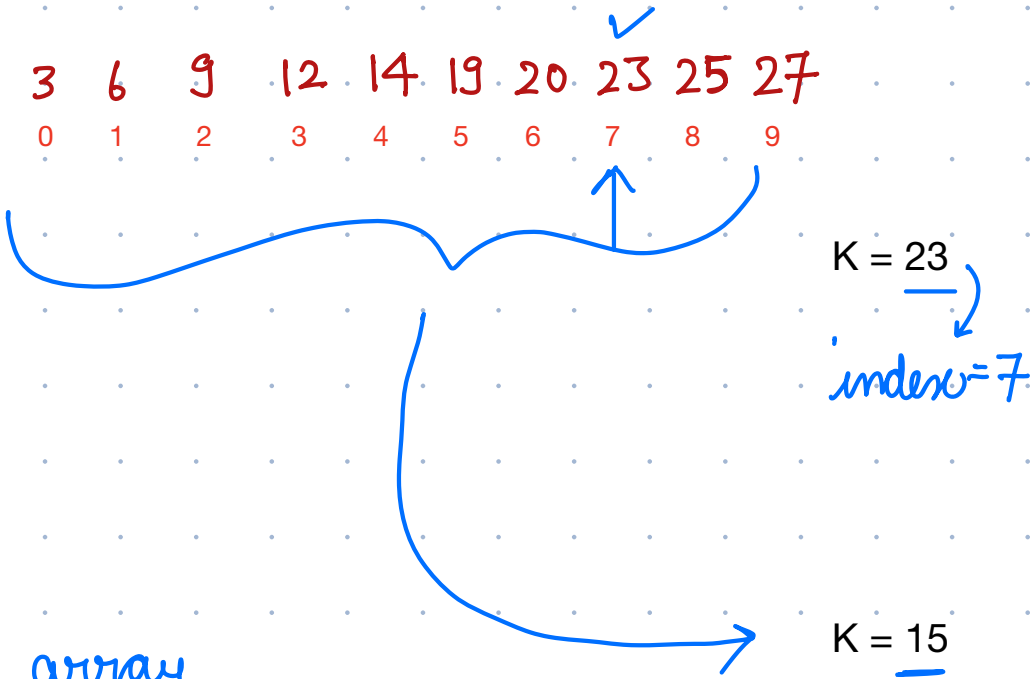
Search Space
Target
Condition ..... discard

Binary Search →

2 parts — Discard 1 part



< Question > : Given a sorted arr[ ]. Search an element K. If K is present return it's index otherwise return -1.



BF Approach

Linear search

Traverse the array

Going through each element and checking if arr[i] == K?

O(N)

Idea -2 Binary Search

Search Space : array

Target => K = 20

Condition ↓

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

↑ mid

l = 0 r = 9

l = 5 r = 9 $\frac{5+9}{2} = 7$

l = 5 r = 6 $\frac{5+6}{2} = 5$

l = 6 r = 6 $\frac{6+6}{2} = 6$ K = 20

K = 10

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

l	r	mid	
0	9	4	14 == 10? Left
0	3	1	6 == 10? Right
2	3	2	9 == 10? R
3	3	3	12 == 10?
3	2	STOP	

</ > Code

```
int binarySearch(int[] arr, int K) {
    N = arr.size()
    l = 0    r = N-1
    while( l <= r ) {
        m = (l + r) / 2
        if(arr[m] == K){return m}
        else if(arr[m] > K){
            r = m-1
        } else {
            l = m+1
        }
    }
    return -1
}
```

SC = O(1)

$m = l + \left(\frac{r-l}{2}\right)$

TC: $N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \rightarrow \frac{N}{8} \rightarrow \ldots\ldots \rightarrow 1$

$1 \quad 2 \quad 3 \quad 4 \quad\quad X$

$O(\log N)$

Range Problem:

-100 ——————————— 100

$l = 78$ $\quad r = 89$

$\frac{(l+r)}{2} = \frac{(78+89)}{2}$

$l + \frac{r-l}{2}$

$78 + \frac{89-78}{2} = 78 + \frac{11}{2} = 78 + 5 = 83$

$\frac{l+r}{2} = \frac{l}{2} + \frac{r}{2}$

$= l - \frac{l}{2} + \frac{r}{2}$

$= l + \frac{r}{2} - \frac{l}{2}$

$= l + \left(\frac{r-l}{2}\right)$

Real life application:

2005 2005 2005 2013 2013 2013 2013 2020 2020 2020 2024

↑

1) all elements are sorted
2) repetitions are there
3) First occurrence of K

< Question > : Given a sorted arr[N]. Find first occurrence of K.

Lot of repetitions

arr[ ] →	-5	-5	-3	0	0	1	5	5	5	5	5	8	10	10	15
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

K = 5

BF Approach Linear search → O(N)

ans = -1

FIRST

Idea -2 Binary Search

K = 5

arr[ ] →	-5	-5	-3	0	0	1	5	5	5	5	5	8	10	10	15
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

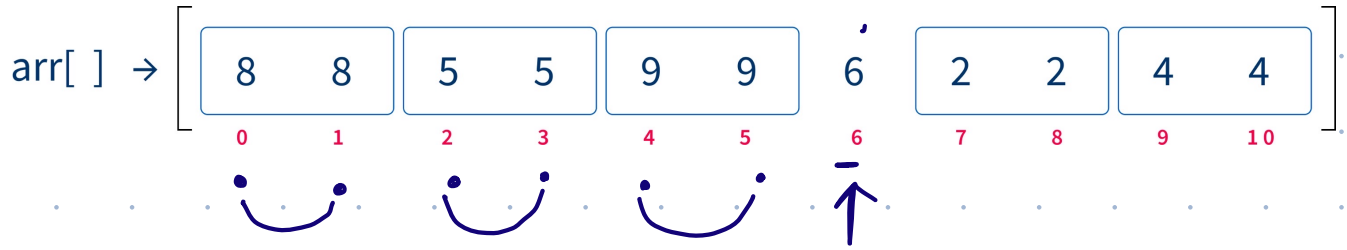
l	r	mid	
0	14	7	left r = 6, ans = 7
0	6	3	right l = 4
4	6	5	right l = m+1 = 6
6	6	6	left r = m-1 = 5, ans = 6
6	5	STOP	

- </> Code

```
ans = -1
while ( l <= r) {
    m = l + (r-l)/2
Case 1: if (arr[m] == K) {
            ans = m
            r = m-1
        }
Case 2: else if(arr[m] < K) {
            l = m+1
        }
Case 3: else { r = m-1 }
}

return ans
```

< *Question* > : Every element occurs twice except for 1. Find that unique element.

Note : Duplicates are adjacent to each other and array isn't necessarily sorted.



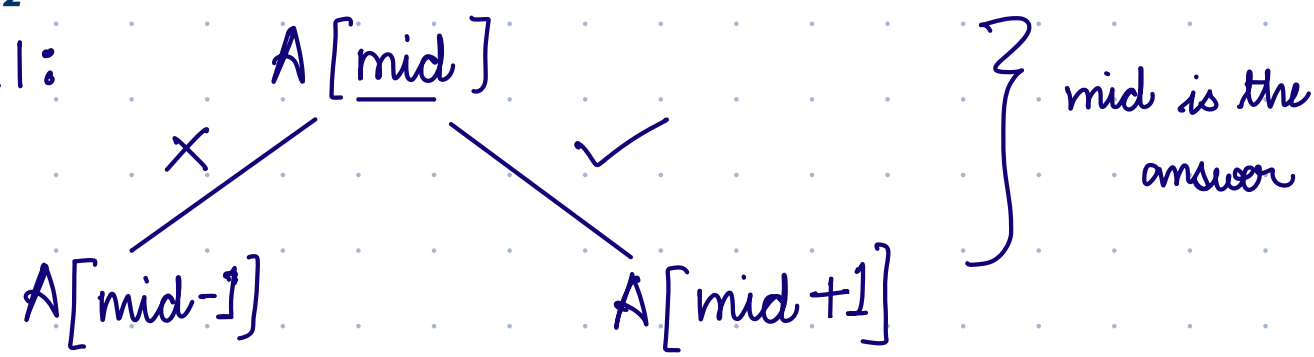
Idea -1

- XOR of all numbers

O(N)

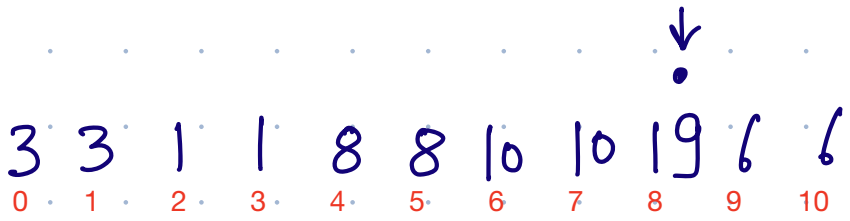
Idea -2

Case 1:



mid == 0 && A[mid] != A[mid+1] → A[mid]

mid == N-1 && A[mid] != A[mid-1] → A[mid]



l	r	mid	
0	10	5	First occ = even Right $l = m+1$
6	10	8 ↓ ans	10 (x) 19 ↑ 8 ans (6 (x))

Observation: Before Unique element ..> F O = even index
after unique element has come, First occurence = odd index

< / > Code

Handle $i = 0 \longrightarrow$ 0th index

Handle $i = N-1 \longrightarrow$ last index

$l = 1 \quad r = N-2$

```
while ( l <= r ) {

    mid = l + (r - l) / 2

    if ( arr[mid] != arr[mid+1]
         ||
         arr[mid] != arr[mid-1] ) {
        return mid
    }
    else if ( arr[mid] == arr[mid+1] ) {
        if ( mid is even index ) {
            l = mid+1
        }
        else {
            r = mid-1
        }
    }
    else {  FO: mid-1
        if (mid-1 is even index) {
            l = mid+1
        }
        else {
            r = mid-1
        }
    }
}
```

TC: $O(\log N)$

SC: $O(1)$

To apply ***Binary Search :***

1. **Target**
2. **Search - space**
3. **Condition on the basis of which search space can be reduced by half**

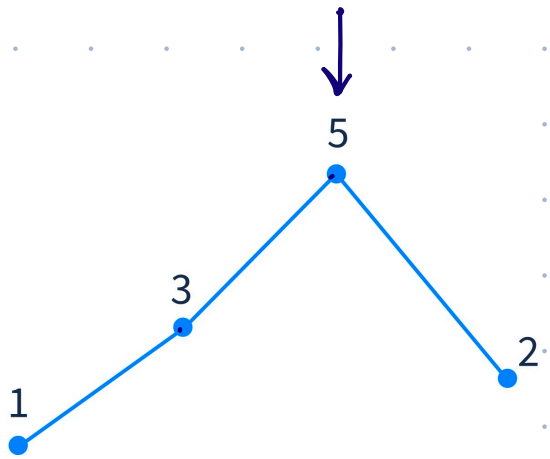
< *Question* > : Given an increasing - decreasing array with distinct elements. Find the max element.

Unique

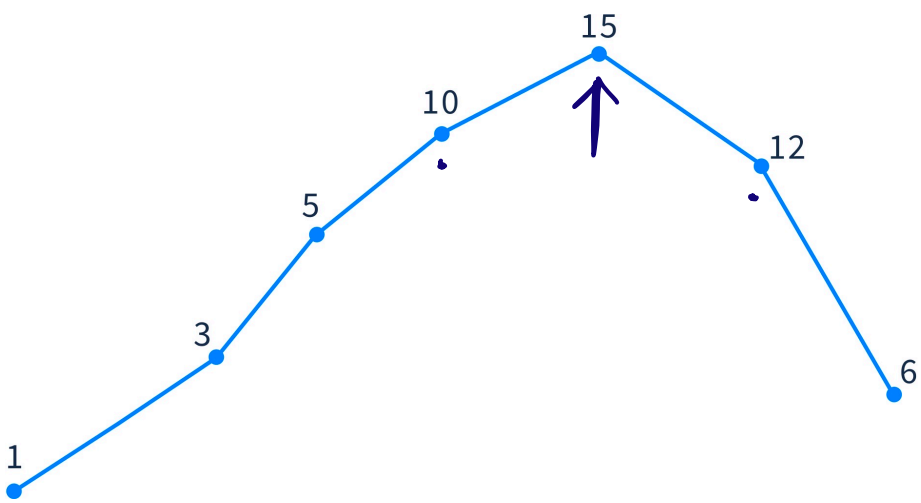
TE > LE

&&

TE > RE



0 1 2 3 4 5 6



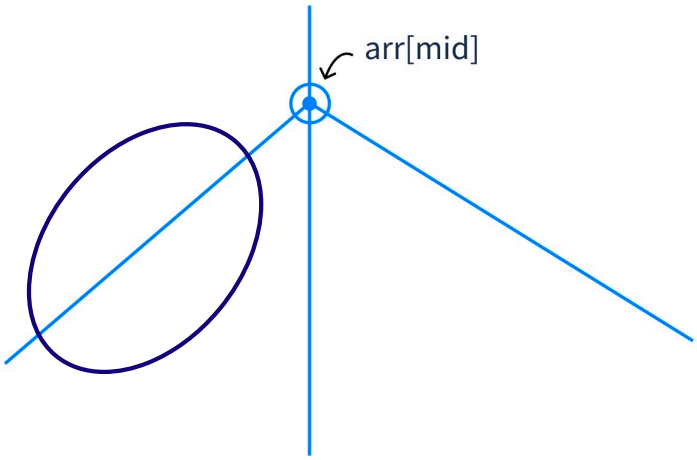
Idea

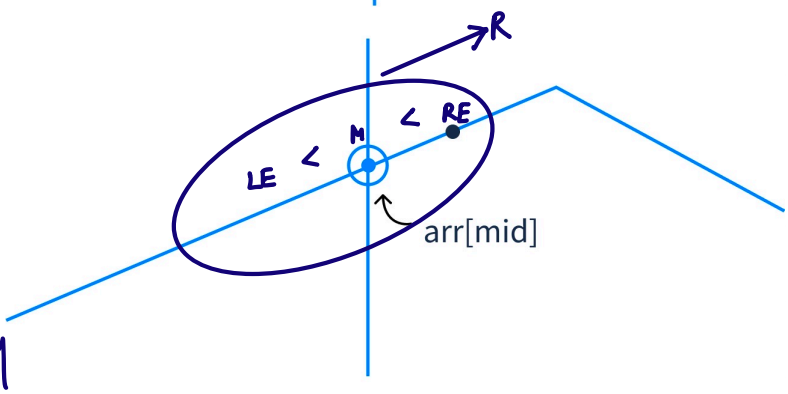
Case 1 :

M > LE

&&

M > RE





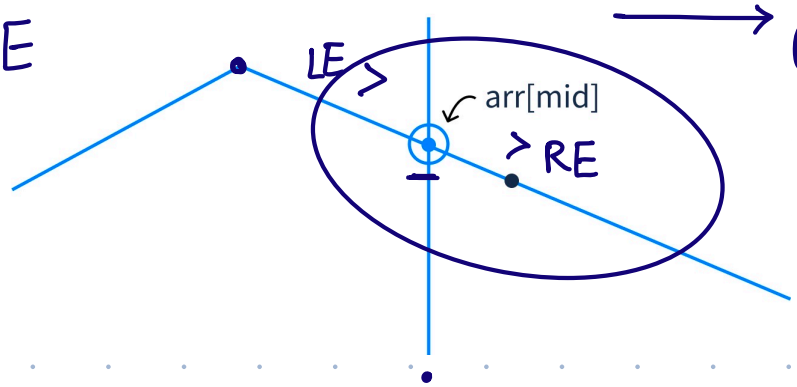
Case 2: RE > M

&&

M > LE

Right

l = m + 1



Case 3

LE > M

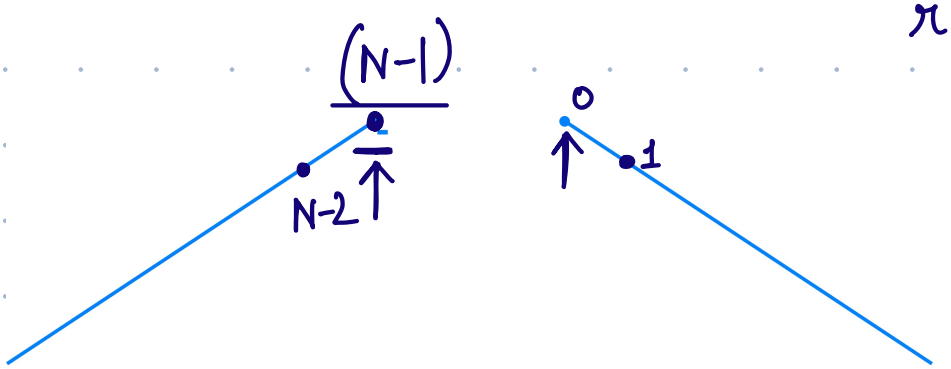
&&

M > RE

Left

r = m - 1

Edge Cases :



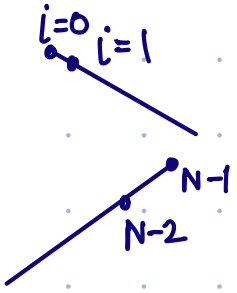
Increasing Array

Decreasing Array

</ > Code

Handle i = 0

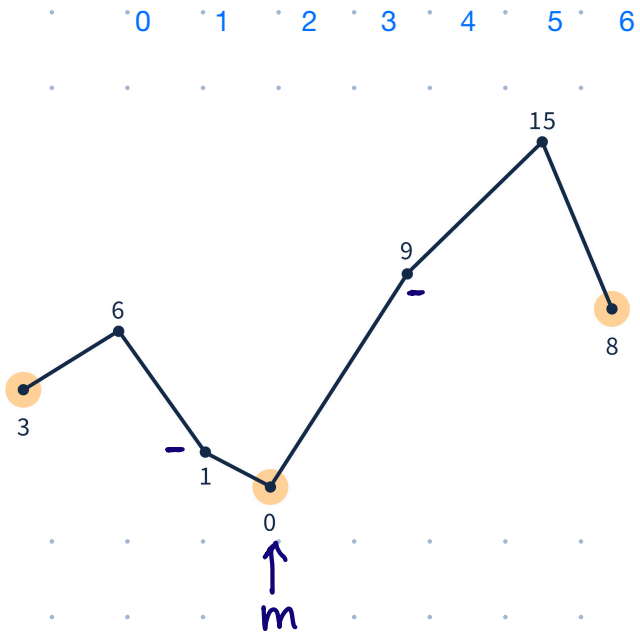
Handle i = N-1

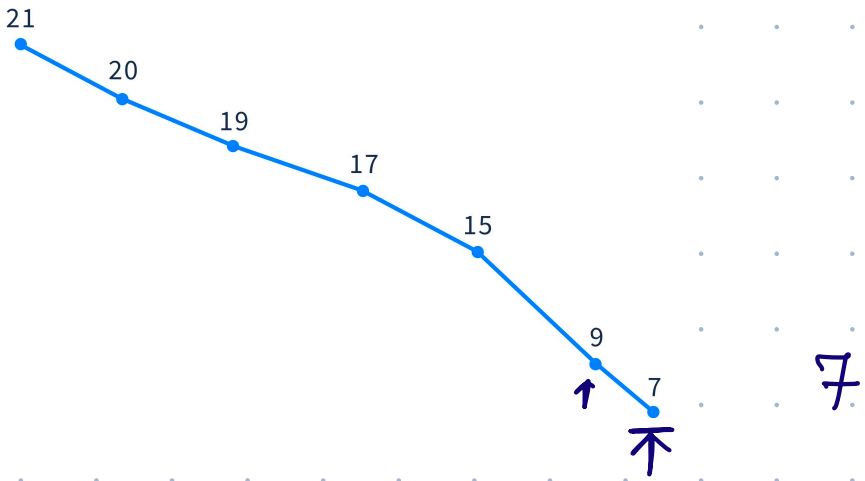



```
l = 1      r = N-2
while ( l <= r ) {

    m = l + (r - l) / 2

Case 1      if ( arr[m] > arr[m+1]
                    &&
                 arr[m] > arr[m-1]) {
                return m
            }
Case 2:     else if ( arr[m+1] > arr[m ]
                    &&
                 arr[m] > arr[m-1]) {
                l = m+1  // right
            }
Case 3:     else {
                r = m-1  // left
            }
}
```

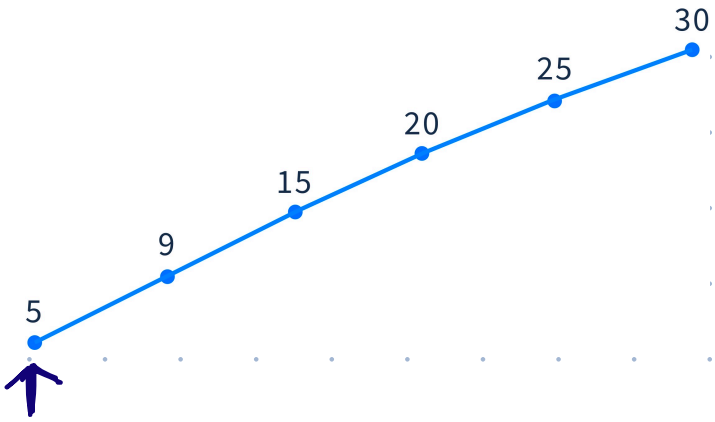
< *Question* > : Given arr[N]. Find any one local minima.

Local Minima : An element which is less than or equals to it's adjacent elements.



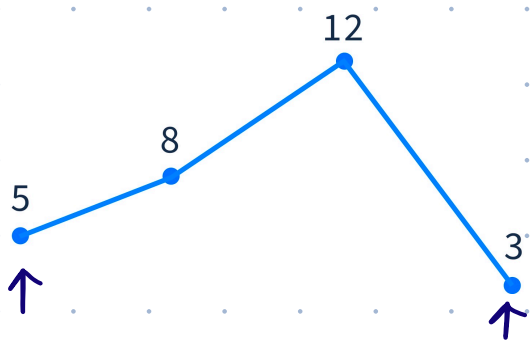


arr[ ] → [5 9 15 20 25 30]



5
ans

arr[ ] → [5 8 12 3]



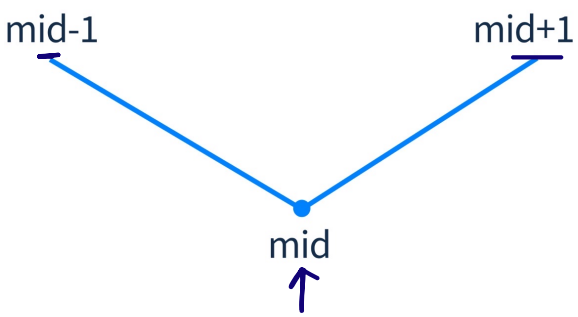
BF Idea

Traverse the array, find i

arr[i-1] > arr[i] < arr[i+1]

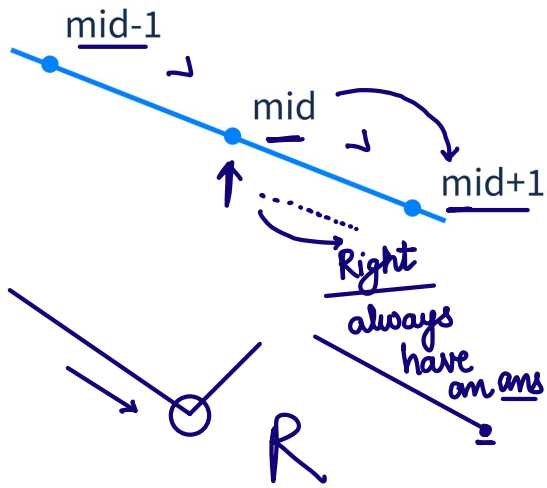
Idea

Case 1

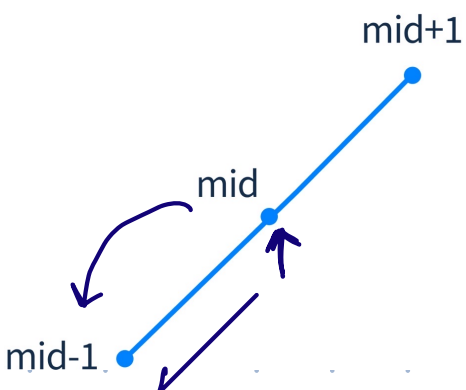


return mid

2



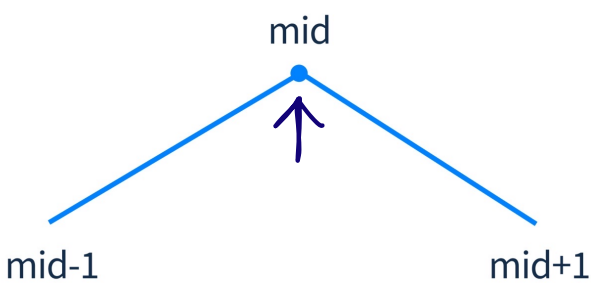
Case 3



l arr[mid -1] < arr[mid]

Left

4



either side

Left side

< / > Code

Handle $i = 0$

Handle $i = N-1$

$l = 1 \quad r = N-2$

```
while (l <= r) {
    m = l + [(r-l)/2]
    Case1 : if(arr[m] < arr[m-1] && arr[m] < arr[m+1]){
                return m
    }
    Case2: else if(arr[m] < arr[m-1] && arr[m] > arr[m+1]){
                l = m+1
    }
    Case3&4: else {
                r = m-1
    }
}
```

TC: $O(\log N)$

SC: $O(1)$

- Can we apply B.S only on sorted array?

No