Todays Content

1. Search in rotated sorted array
2. First missing +ve Integer

Pre-ReQuists

Given ar[] = {3 9 14 16 20 28 35 40 49}

Given k, Can k be present in ar[] or not?

k chana

16 : Yes : {3 ... 16 ... 49}

25 : Yes : {3 ... 25 ... 49}

60 : No : 60 > 49.

k : if (ar[0] <= k && k <= ar[N-1]) k can be present.

Rotate ar[] Revisim:

Given ar[N] rotate ar[] right to left by k times

	0	1	2	3	4	5	6
ar[7] = {	-2	1	3	4	6	8	9 }

k = 3

Rotate 3 : { 6 8 9 -2 1 3 4 }

	0	1	2	3	4	5	6	7	8
ar[9] = {	3	6	9	10	11	14	20	23	30 }

k = 4

Rotate 4: { 14 20 23 30 3 6 9 10 11 }

	0	1	2	3	4	5	6	7	8	9
ar[10] = {	2	4	6	8	12	15	19	21	26	30 }

k = 4

	0	1	2	3	4	5	6	7	8	9
Rotate 4 : {	19	21	26	30	2	4	6	8	12	15 }

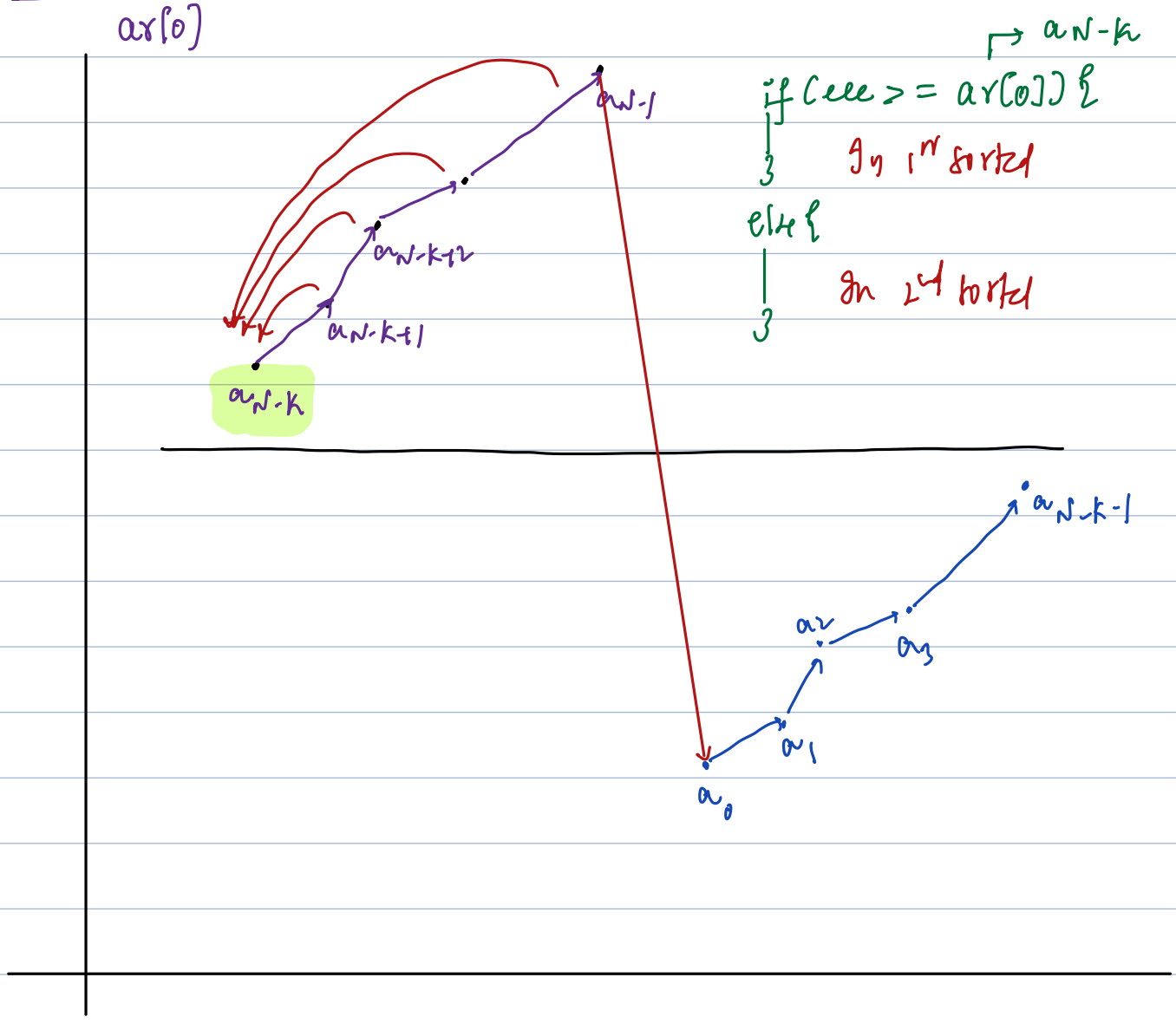
#Sorted arr[];

Rotate k times

arr[]: $a_0 < a_1 < a_2 < a_3 < \dots < a_{N-K-1} < a_{N-K} < a_{N-K+1} \dots < a_{N-1}$

arr[]: $a_{N-K}\ a_{N-K+1} \dots a_{N-1}\ \ a_0\ a_1\ a_2\ a_3 \dots a_{N-K-1}$

arr[0]



↳ a_{N-K}

if (eee >= arr[0]) {

} go in 1st sorted

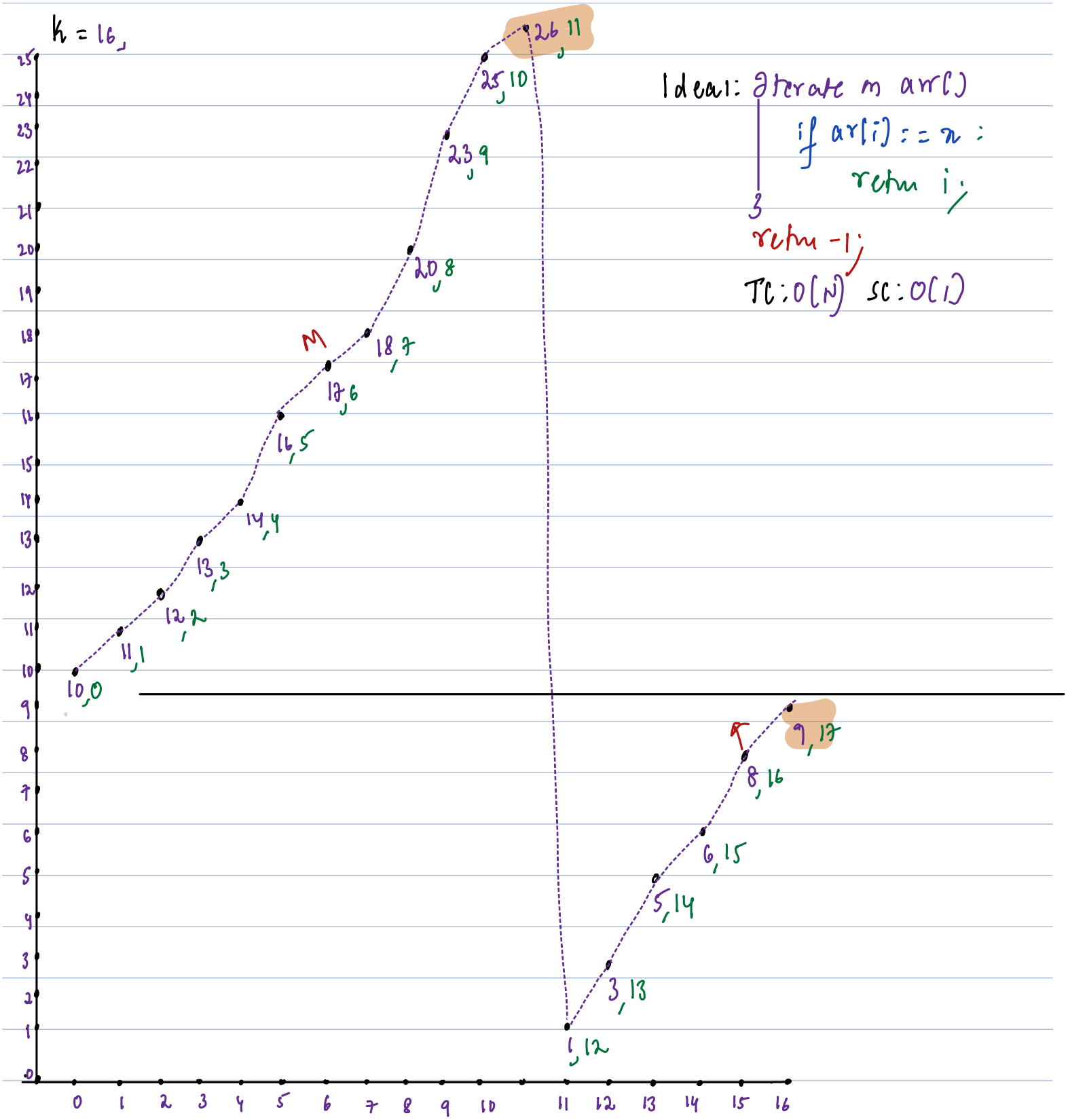
else {

} go in 2nd sorted

38. Given an input arr[], formed by rotating a distinct sorted array right to left by some no: of times.

Search ele & return index in input arr[] if ele is not present return -1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
arr[]: {	10	11	12	13	14	16	17	18	20	23	25	26	1	3	5	6	8	9 }



Ideal: Iterate in arr[]

```
if arr[i] == x :
    return i;
return -1;
```

TC: O(N) SC: O(1)

Q: Given ar[N], search for k.

Idea2: TC: $O(\log N_1 \log N) = O(\log \frac{N}{2})$ SC: O(1)

Q How to calculate max ele index?

Appl: Using peak of arr()

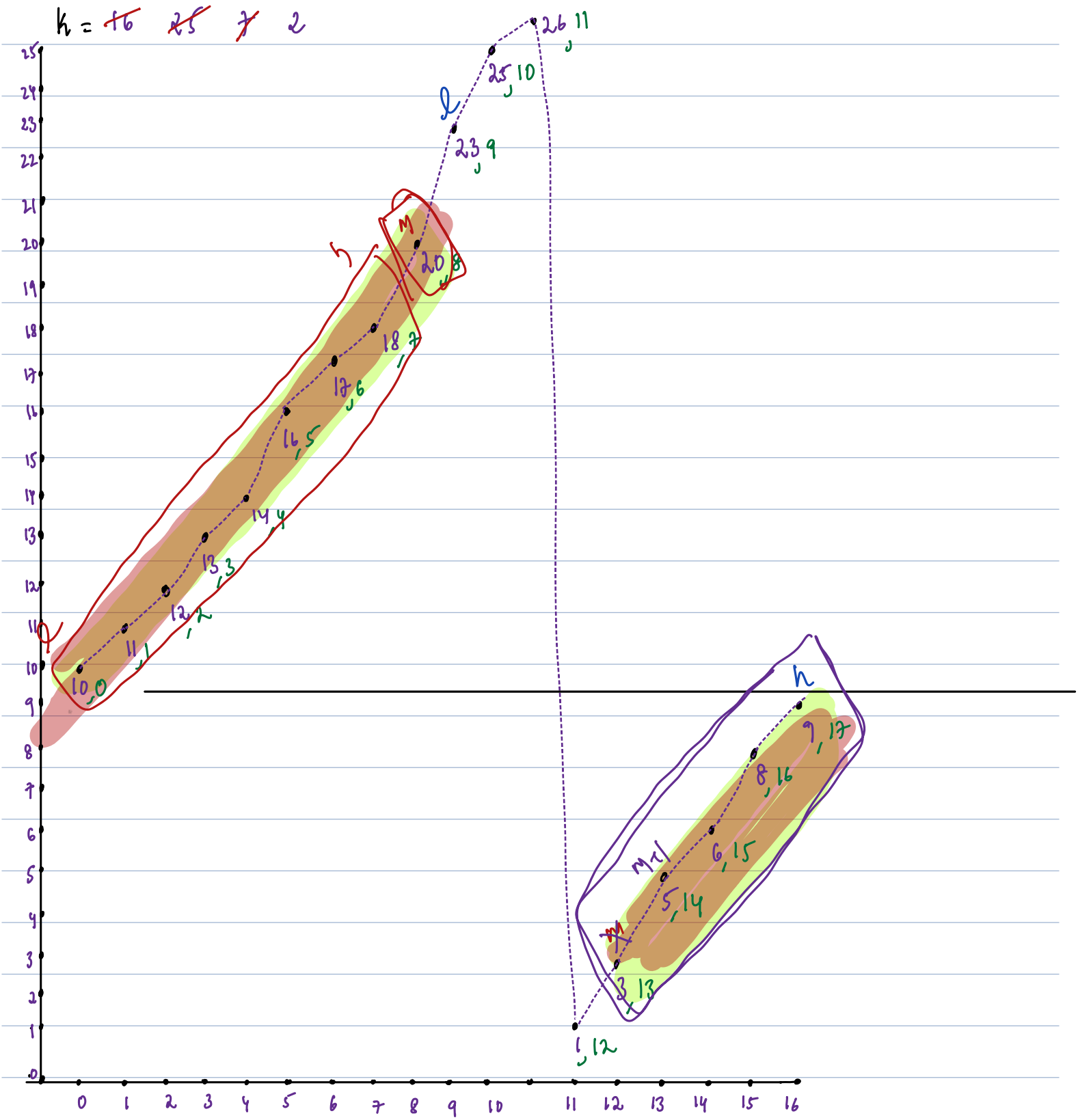
Issu: Abve logic will not work, Because arr() can contain 2 peaks, we cannot guarantee it will return max of arr().

App2: Apply BS to find max ele.

```
int l=0, h=N-1, ans=0;
while(l<=h){
    if(ar[m]>=ar[0]){ #1st part
        ans=m;
        l=m+1;
    }else{ #2nd part
        h=m-1;
    }
}
k=ans; # k is index of max element.
1st sorted array: [0..k]
2nd sorted array: [k+1..N-1]
#Search x.
if(x>=ar[0]){
    #Might be in 1st sorted array.
    Apply BS [0..k]
}else{
    #Might be in 2nd sorted array
    Apply BS [k+1..N-1]
}
```

Ideas:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ar[]: {	10	11	12	13	14	16	17	18	20	23	25	26	1	3	5	6	8	9 }



```
int searchRotated(vector<int> &ar, int ele){

    int N = ar.size();
    int l = 0, h = N-1;
    while (l <= h){
        int m = (l+h)/2;
        if(ar[m] == ele){ return m;}
        if(ar[m] >= ar[0]){ #1st part
            #obs1: {0..m} is sorted
            if(ele >= ar[0] && ele <= ar[m]){
                h = m-1;
            }else{
                l = m+1;
            }
        }else{ #2nd part
            #obs2: {m..N-1} is sorted
            if(ele >= ar[m] && ele <= ar[N-1]){
                l = m+1;
            }else{
                h = m-1;
            }
        }
    return -1;
}
```

28 Given a distinct sorted array the elements
Return 1st Missing natural number not in array.

ar[] = { 1 2 3 5 6 7 9 10 } ans =

ar[] = { 1 2 3 4 5 7 8 10 12 } ans =

ar[] = { 2 4 7 8 10 } ans =

ar[] = { 1 2 3 4 5 } ans =

#Idea:

	0	1	2	3	4	5	6	7
ar[] = {	1	2	3	5	6	7	9	10 }

ele:

```
int firstpositive(vector<int> &arr){
```

3

Idea 2:

	0	1	2	3	4	5	6	7	8	9	10	11	12
ar[] = {	1	2	3	4	5	7	8	10	12	14	15	16	18 }

```
int firstpositive(vector<int> &arr){

}
```

#2nd versin

Given a distinct sorted array elements

Return 1st missing natural number not in array

Ex:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
ar[]: {	-4	-3	-1	0	1	2	3	4	5	7	8	10	11	12 }

#Ideal

```
int firstpositive(vector<int> &arr){
```

3