Oct 12th, 2023

Given that an array with increasing and decreasing sequence with a peak element and a target and we've to find that target in that given list with min index which element can have duplicate at most 2 time. No of calls can be made <= 100

We've to return the minimum index therefore there can be following case :-

i) Element present at left side of peak element
ii) Element present at right side of peak element
iii) Element may present on both side
iv) Element is not present.

Ex : 1 2 3 4 5 2 1 target = 1
0 1 2 3 4 5 6

ans = 0

there are 2 index which having 1 i.e 0 & 6 but the min is 0 So, 0 is our answer

* To solve this problem there are 2 ways to do :

i) Linear search

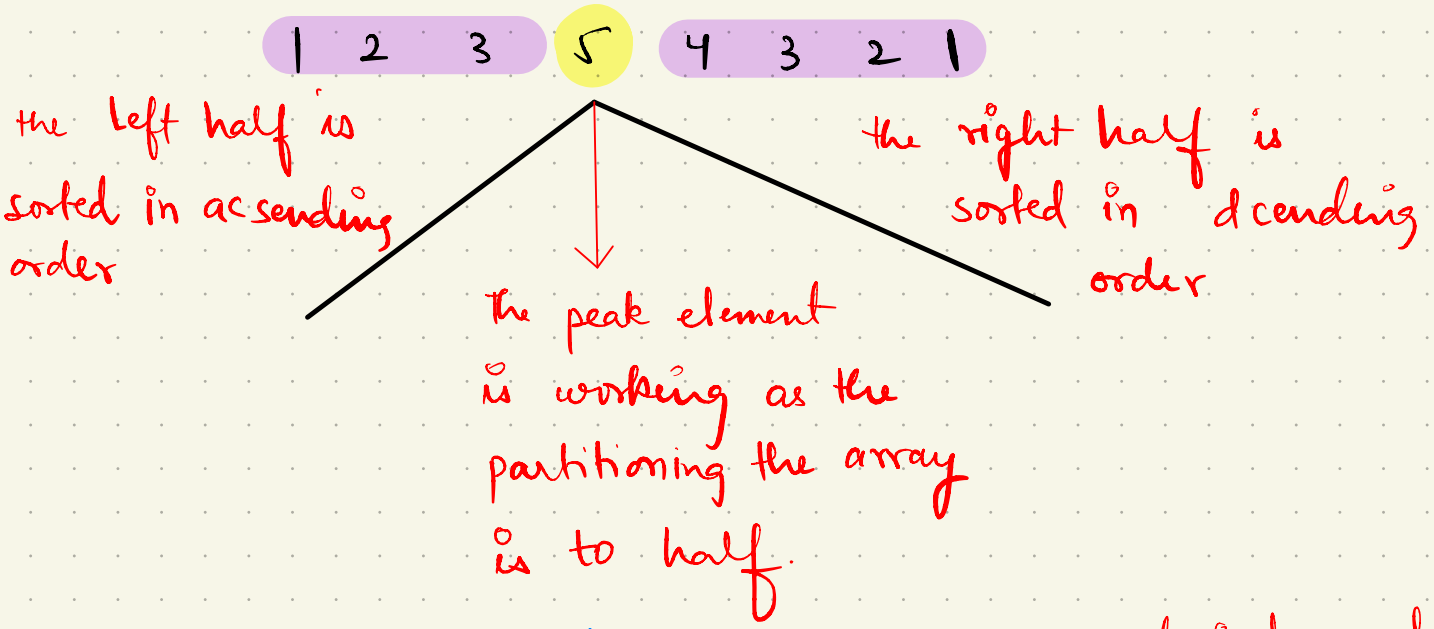
↳ this can be solution but it's not satisfying the constaints i.e No of calls <= 100

ii) Binary search

The requirements of the Binary search are statisfying i.e sorted search space.

But how we gonna apply on the List, we need a defined a search space to apply. And we must also know how to eliminate the remaining non-required search space.

So. first we gonna define search space



So, first we gonna find the peak element index and afterwards we'll apply binary search on the two list and return ans.

psuedo code :-

```
find_peak_El (arr, n) {
  l=0, h=n-1;
  while (l<=h) {
    mid = l + (h-l)/2;
    if (arr[mid] < arr[mid+1]) l= mid+1;
    else h = mid-1;
  }
  return l;
}
```

```
find_left_half(arr, l, h, target){
    while(l<=h){
        mid = l + (h-1)/2;
        if(arr[mid]== target){
            return mid;
        }
        else if(arr[mid] < target){
            l = mid+1;
        }else   h = mid -1;
    }
    return -1;
}
find_right_half(arr, l, h, target){
    while(l<=h){
        mid = l + (h-1)/2;
        if(arr[mid]== target){
            return mid;
        }
        else if(arr[mid] < target){
            l = mid+1;
        }else   h = mid -1;
    }
    return -1;
}
```