Binary Search

30 November 2024 15:09

```
int mid = low + (high - low) / 2
```

```
        if (array[mid] == key)
            return mid


            if (array[mid] < key)
                low = mid + 1

            else
                high = mid - 1
```

Binary Search first occurrence

30 November 2024 15:09

```
int mid = low + (high - low) / 2
```

```
        if (array[mid] == key)
            result=mid
            high=mid-1  //for first occurrence
            low=mid+1   //for last occurrence

           if (array[mid] < key)
               low = mid + 1

           else
               high = mid - 1
```

Peak Element

30 November 2024 15:09

A peak element is an element that is strictly greater than its neighbors.
array should be mountain shaped. (bitonic sequence array)

A mountain-shaped array is an array that increases monotonically up to a peak and then decreases monotonically.

input=7
1 1 2 3 6 5 4

output=
Peak element index: 4

input=5
1 2 1 0 1

output=Peak element index: 1

```
int low = 0
int high = n - 1

while (low < high)
    int mid = (low + high) / 2

    if (nums[mid] > nums[mid + 1])
        high = mid      // Continue searching in the left half
    else
        low = mid + 1  // Continue searching in the right half
End while

return low  //both low or high can be returnd
```

Peak element will be always be at the index where low and high can meet.
so loop should continue until that.

Pivot Element

30 November 2024 15:09

Pivot element is the only element in input array which is smaller than it's previous element.

A pivot element divides a sorted rotated array into two monotonically increasing array.

input should be a sorted rotated array is an array that was originally sorted in ascending order
but has been "rotated" around a pivot point. This means that some of the elements
from the beginning of the array have been moved to the end.

input=7
4 5 6 7 1 2 3
output=
Pivot element is: 1

input=8
6 7 8 1 2 3 4 5
Pivot Element is: 1

```
              low=0
              high=n-1

    if (high < low)
            return arr[0]             // Array not rotated, return first element

        if (high == low)
            return arr[high]                  // Only one element left

        int mid = (low + high) / 2
                                              // Check if mid+1 is the pivot
        if (mid < high && arr[mid] > arr[mid + 1])
            return arr[mid + 1]

                                                 // Check if mid is the pivot
        if (mid > low && arr[mid] < arr[mid - 1])
            return arr[mid]


         // Adjust low and high pointers
       if (arr[low] <= arr[mid])
            return findPivot(arr, mid + 1, high); // Search in the right half
        else
            return findPivot(arr, low, mid - 1); // Search in the left half
```

The condition arr[low] <= arr[mid] means that the left half of the array (from low to mid) is sorted in ascending order. so we need to look for pivot from mid+1 to high. This is because in a sorted segment, the elements are increasing. The pivot being the smallest element cant be in this sorted portion.

if its false, it means that right half of the array is sorted from mid+1 to high. hence we need to look for pivot from low to mid-1.

another example:

Input: arr=[3,4,5,6,7,8,1,2]

for above conditions.