Lesson:

Java

Problems on Binary Search - 2

Pre-Requisites

- Binary Search

Pattern: Binary Search to find Peak Element

Problem 1: Given a mountain array ‘a’ of length greater than 3, return the index ‘i’ such that arr[0] < arr[1] < ... < arr[i - 1] < arr[i] > arr[i + 1] > ... > arr[arr.length - 1]. This index is known as the peak index.

Input :
```
Array = [0,4,1,0]
```

Output :
```
1
```

Code link - https://pastebin.com/DH7xKiTT

Explanation:
1. For a rising curve, a[i] > a[i - 1] for all i belonging to 1 to n - 1, and for a falling curve the condition is reversed, a[i] < a[i - 1] for all i belonging to 1 to n - 1. We need to find the last index for which a[i] > a[i - 1] evaluates to true.
2. To find that value, apply binary search-
   a. Initialise low pointer with index ‘1’(it ensures that we always have a left element to compare with our current element) and high pointer with index ‘n - 1’.
   b. If the mid element is greater than the value before it then mark it as a possible answer and move to the latter half of the array.
   c. Else, move to the first half of the array.

Problem 2: A peak element is an element that is strictly greater than its neighbors.
Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks.
You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

Input :
```
Array = [1,2,1,2,6,10,3]
```

Output :
```
Either index 1 or index 5 are the correct output. At index 1, 2 is the peak element and at
index 5, 10 is the peak element.
```

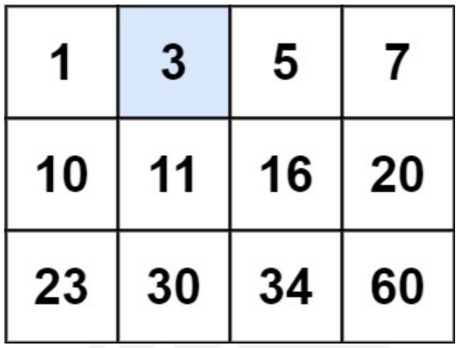
Code link - https://pastebin.com/JXtVbjeM

Explanation:

1. At every iteration-
   a. We check the edge cases, that is if the ‘mid’ index is either the first or the last index. This way we can return if either of them is peak index, if they are not then move to the next else statement.
   b. Here we check if the middle index satisfies the peak element property then we return this index, else we move towards the side where the peak element is more likely to occur.
   c. Let's say if the value at the current index is less than the previous index, then we move towards the right side as there are higher chances of finding the peak element.

Pattern: Binary Search on 2d Matrix

Problem 3: Search the ‘target’ value in a 2d integer matrix of dimensions n x m and return true if found, else return false.
This matrix has the following properties:
1. Integers in each row are sorted from left to right.
2. The first integer of each row is greater than the last integer of the previous row.



1	3	5	7
10	11	16	20
23	30	34	60

Input :
```
Matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
```

Output :
```
true
```

Code link - https://pastebin.com/xdx9gf7y

Explanation:
Note the fact that in matrix, the elements are sorted row wise and the first integer of each row is greater than the last integer of the previous row.

1. Consider the 2d matrix to be a linear array with starting index as ‘0’ and last index as ‘n x m - 1’ (0-based indexing).
2. Apply binary search as the formed array will be sorted.
3. Find the value of mid at every point and convert it to corresponding x and y points on the grid. This can be easily done by dividing the middle index by m and taking remainder with respect to m respectively.

This way it becomes a standard binary search problem.

Upcoming Class Teasers

- Problems on Binary Search - 3