Day 07

Wednesday, 6 January 2021 9:28 PM

Homework Assignments :

- https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/
- https://leetcode.com/problems/search-in-rotated-sorted-array/
- https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/
- https://leetcode.com/problems/random-pick-with-weight/
- https://leetcode.com/problems/powx-n/

- https://leetcode.com/problems/first-bad-version/
- https://leetcode.com/problems/find-peak-element/
- https://leetcode.com/problems/search-a-2d-matrix-ii/

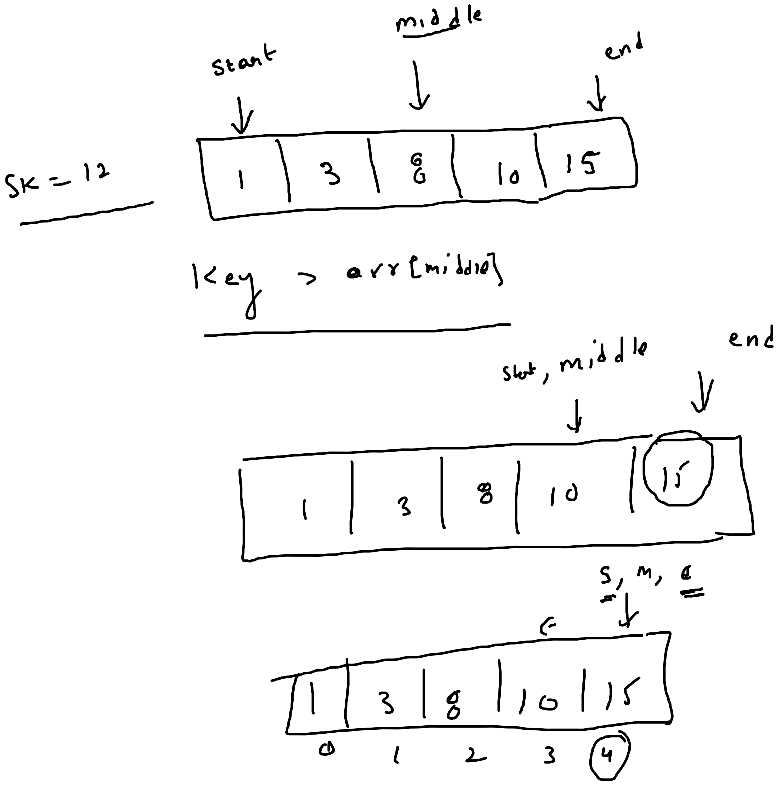
Pattern : Modified Binary Search :

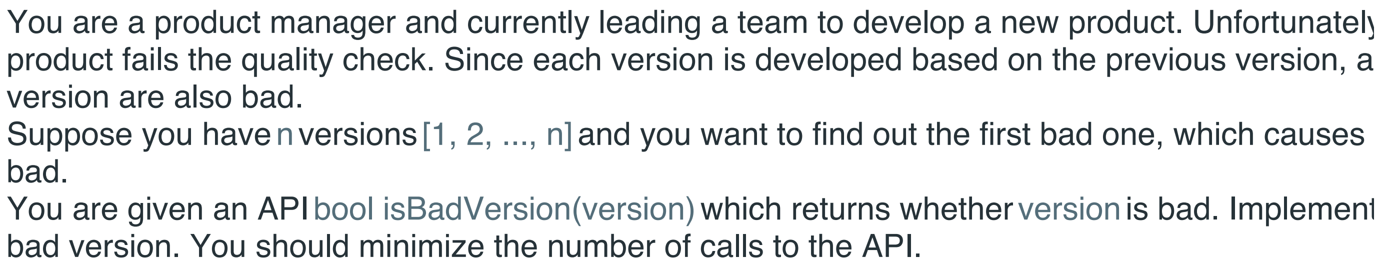
Problem :
Ceiling of a Number : Greater or equal to the key

Input : [4 , 6 , 10] Key : 6
Output : 1

Input : [1 , 3 , 8 , 10 , 15] key = 12
Output :4

```
public int searchCeiling(int[] arr , int key)
{

        If(key > arr[arr.length-1 ]
        {
                Return -1 ;
        }

        Int start = 0 ; end  = arr.length -1 ;
                while(start <=end)
                {

                        int mid = start + (end -start) / 2;
                        If( key < arr[mid])
                        {
                                end = mid -1;

                        }
                        Else (key > arr[mid])
                        {
                                Start = mid + 1;
                        }
                        Else
                        {
                                return mid ;
                        }
                }

        return start ;
        }



}
```





You are a product manager and currently leading a team to develop a new product. Unfortunately product fails the quality check. Since each version is developed based on the previous version, a version are also bad.

Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes bad.

You are given an API bool isBadVersion(version) which returns whether version is bad. Implement bad version. You should minimize the number of calls to the API.

• • • E

Input: n = 5, bad = 4

Practice with your approach .

Find Peak Element:

A peak element is an element that is strictly greater than its neighbour's.
Given an integer array nums, find a peak element, and return its index. If the array contains mu
index to **any of the peaks**.
You may imagine that nums[-1] = nums[n] = -∞.

Input: nums = [1,2,3,1]
Output : 2

Practise with your own solution

Find First and Last Position of Element in Sorted Array

Given an array of integers nums sorted in ascending order, find the starting and ending position
If target is not found in the array, return [-1, -1].
Input: nums = [5,7,7,8,8,10], target = 8

Output: [3,4]

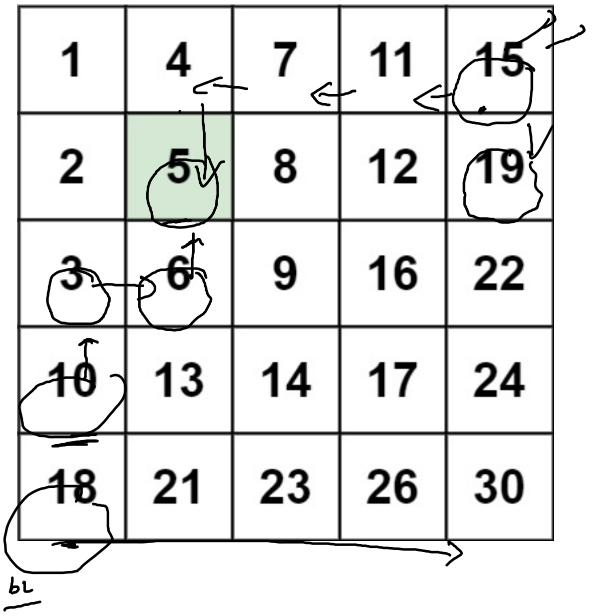
```
Int keyIndex = 0 ;

Mbs (int[] arr , int key , boolean findMaxIndex)
// Find the starting
If(key < arr[mid])
{
    start = mid + 1;
}
Else if(key > arr[mid]
{
    end =mid-1 ;
}
Else
{
      keyIndex = mid ;
    If(findMaxIndex)
    {
      end = mid -1;
     }
    Else
    {
     start = mid+ 1;
    }
}
```

Write an efficient algorithm that searches for a target value in an m x n integer matrix. The matrix
properties:

- Integers in each row are sorted in ascending from left to right.
  Integers in each column are sorted in ascending from top to bottom.

target

Target = 5



1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

row = arr.length-1

col = 0

arr[

~~5 > targ~~

element > tar

matrix[row

row--

else

elk

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        // start our "pointer" in the bottom-left
        int row = matrix.length-1;
        int col = 0;

        while (row >= 0 && col < matrix[0].length) {
            if (matrix[row][col] > target) {
                row--;
            } else if (matrix[row][col] < target) {
                col++;
            } else { // found it
                return true;
            }
        }

        return false;
    }
}
```

Implement pow(*x*, *n*), which calculates *x* raised to the power *n* (i.e. xn).

Example 1:
Input: x = 2.00000, n = 10
Output: 1024.00000

O(logn)

```
class Solution {
public:
    double fastPow(double x, long long n) {
        if (n == 0) {
            return 1.0;
        }
        double half = fastPow(x, n / 2);
        if (n % 2 == 0) {
            return half * half;
```

5 \

5

2

5/2

```
        } else {
            return half * half * x;
        }
    }
    double myPow(double x, int n) {
        long long N = n;
        if (N < 0) {
            x = 1 / x;
            N = -N;
        }
        return fastPow(x, N);
    }
};
```

10 → 5 → 2 →

100 → 50 → 25 →