



Assignment Solutions | Binary search - 2 | Week 10

1. Write a program to apply binary search in array sorted in decreasing order.

Solution :

```
#include<iostream>
using namespace std;

int main(){
    int a[] = {10,7,6,4,2,1};
    int n = 6;
    int tgt = 5;
    int lo = 0 , hi = n - 1;
    bool flag = false;
    while(lo <= hi){
        int mid = lo + (hi - lo)/2;

        if(a[mid] == tgt){
            flag = true;
            break;
        }

        else if(a[mid] > tgt)lo = mid + 1;

        else hi = mid - 1;
    }

    if(flag)cout<<"Element exists"<<endl;
```

2. You have a sorted array of infinite numbers, how would you search an element in the array?

Solution :

```

#include<iostream>
using namespace std;

int main(){
    int a[] = {1,2,4,7,10,12,15,18};
    int lo = 0 , hi = 1;
    int tgt = 10;
    while(a[hi] < tgt){
        lo = hi;
        hi *= 2;
    }
    cout<<lo<<" "<<hi<<endl;
    bool flag = false;
    while(lo <= hi){
        int mid = lo + (hi - lo)/2;

        if(a[mid] ==tgt){
            flag = true;
            break;
        }
        else if(a[mid] > tgt){
            hi = mid - 1;
        }
    }
}

```

3. You are given an $m \times n$ integer matrix `matrix` with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer `target`, return `true` if `target` is in `matrix` or `false` otherwise.

You must write a solution in $O(\log(m * n))$ time complexity. [Leetcode 74]

Example 1:

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

Output: `true`

Example 2:

Input: `matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]]`, `target = 13`

Output: `false`

Solution :

```

bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int m = matrix.size();
    int n = matrix[0].size();

    int lo = 0 , hi = n*m - 1;

    while(lo <= hi){
        int mid = lo + (hi - lo)/2;

        if(matrix[mid/n][mid%n] == target)return true;
        else if(matrix[mid/n][mid%n] > target)hi = mid - 1;
        else lo = mid + 1;
    }
    return false;
}

```

4. There is an integer array `nums` sorted in non-decreasing order (not necessarily with **distinct** values).

Before being passed to your function, `nums` is **rotated** at an unknown pivot index `k` ($0 \leq k < \text{nums.length}$) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,4,4,5,6,6,7]` might be rotated at pivot index `5` and become `[4,5,6,6,7,0,1,2,4,4]`.

Given the array `nums` **after** the rotation and an integer `target`, return `true` if `target` is in `nums`, or `false` if it is not in `nums`.

You must decrease the overall operation steps as much as possible.

[Leetcode 81]

Example 1:

Input: `nums = [2,5,6,0,0,1,2]`, `target = 0`

Output: `true`

Example 2:

Input: `nums = [2,5,6,0,0,1,2]`, `target = 3`

Output: `false`

Solution :

```
bool search(vector<int>& a, int tgt) {
    int low = 0 , hi = a.size() - 1;

    while(low <= hi){
        int mid = low + (hi - low)/2;

        if(a[mid] == tgt)return true;
        else if(a[low] == a[mid] and a[mid] == a[hi]){
            low++;
            hi--;
        }
        else if(a[low] <= a[mid]){
            if(a[low] <= tgt and tgt <= a[mid])hi = mid - 1;
            else low = mid + 1;
        }

        else {
            if(a[mid] <= tgt and tgt <= a[hi])low = mid + 1;
            else hi = mid - 1;
        }
    }
    return false;
}
```
