**Binary Search:**

1. Finding first and last occurrence of a number in an array

Bruteforce: Linear Search

Optimised: Binary Search

Find for first occurrence and find for last occurrence

class Solution {

public:

int firstoccurence (vector<int>& nums, int target){

    int start =0;

    int end = nums.size()-1;

    int res=-1;

    while(start <=end){

        int mid = start + (end - start)/2;

        if(nums[mid]== target){

            end = mid -1; //first occurence

            res=mid;

        }

        else if(nums[mid] > target){

            end = mid -1;

        }

        else{

            start = mid+1;

        }

    }

    return res;

}

int lastoccurence (vector<int>& nums, int target){

    int start =0;

    int end = nums.size()-1;

    int res=-1;

    while(start <=end){

        int mid = start + (end - start)/2;

        if(nums[mid]== target){

            start = mid+1; //last occurrence

            res = mid;

        }

        else if(nums[mid] > target){

            end = mid -1;

        }

        else{

            start = mid+1;

        }

    }

    return res;

}

    vector<int> searchRange(vector<int>& nums, int target) {

        vector<int>ans;

        ans.push\_back(firstoccurence(nums,target));

        ans.push\_back(lastoccurence(nums,target));

        return ans;

    }

};

1. **Count of the target element in array:**

A screenshot of a black and white screen

Description automatically generated

Bruteforce: unordered\_map use

Optimised: last occurrence index - first occurrence index +1

    int count(int arr[], int n, int x) {

        //bruteforce is using unordered\_map

        //optimised: binary search

        //find 1st and last occurence of x in arr

        // then return last occurence - first occurence +1

        //finding first occurence:

        int s =0;

        int e = n-1;

        int first=-1;

        while(s<=e){

            int mid = s + (e-s)/2;

            if(arr[mid] == x){

                e = mid -1;

                first= mid;

            }

            else if(arr[mid] > x){

                e= mid -1;

            }

            else{

                s= mid+1;

            }

        }

        //last occurence ::

        s = 0;

        e=n-1;

        int last=-1;

        while(s<=e){

            int mid = s + (e-s)/2;

            if(arr[mid]==x){

                s= mid +1;

                last = mid;

            }

            else if(arr[mid] < x){

                s= mid+1;

            }

            else{

                e = mid-1;

            }

        }

        // Calculate the count and return

    int res = (first != -1 && last != -1) ? (last - first + 1) : 0;

    return res;

    }

1. Find the rotation count in rotated sorted array:

A screenshot of a black and white screen

Description automatically generated

Return the index of the smallest element in array:

1. Linear Search
2. Binary Search

A computer screen with text on it

Description automatically generated

1. Search in rotated sorted array

