**Linear Search**

1. **Count the zeroes in an Array**

pair<int,int> count\_zeroes\_and\_ones(int arr[],int size){

    int countZeroes = 0 , countOnes = 0;

    for(int i=0;i<size;i++){

        if(arr[i] == 0){

            countZeroes++;

        }else{

            countOnes++;

        }

    }

    return {countZeroes,countOnes};

}

1. **Maximum number in array**

int find\_max(int arr[],int size){

    int max\_elem = INT\_MIN;

    for(int i=0;i<size;i++){

        if(arr[i] > max\_elem){

            max\_elem = arr[i];

        }

    }

    return max\_elem;

}

1. **Extreme print in Array**

void extremePrintArray(int arr[], int size)

{

    int start = 0;

    int end = size - 1;

    while (start <= end)

    {

        if (start == end)

        {

            cout << arr[start++] << " ";

        }

        else

        {

            cout << arr[start++] << " ";

            cout << arr[end--] << " ";

        }

    }

}

1. **Reverse an Array**

void reverse(int arr[], int size)

{

    int start = 0;

    int end = size - 1;

    while (start <= end)

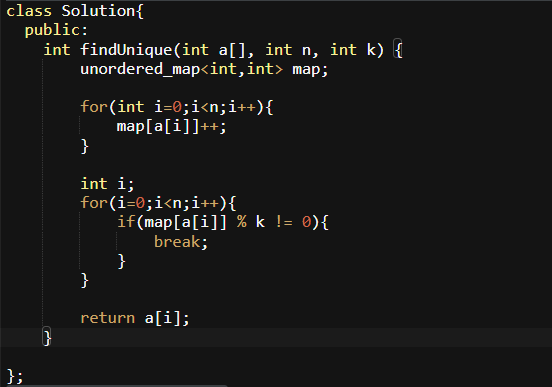
    {

        swap(arr[start++], arr[end--]);

    }

}

1. **Find unique Element**

****

1. **Print pairs**

void print\_pairs(int arr[], int size)

{

    for (int i = 0; i < size; i++)

    {

        for (int j = 0; j < size; j++)

        {

            cout << "(" << arr[i] << "," << arr[j] << ")";

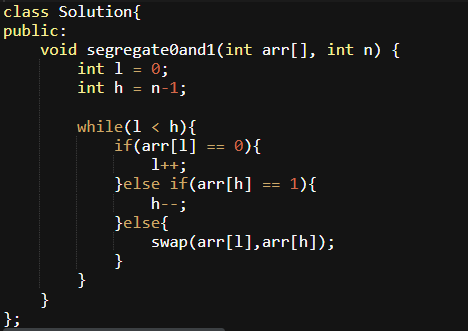
        }

        cout<<endl;

    }

}

1. **Sort 0’s and 1’s**

****

1. **Right shift array elem by 1**

void shift\_array\_by\_one(int arr[], int size)

{

    int temp = arr[size - 1];

    for (int i = size - 1; i >= 1; i--)

    {

        arr[i] = arr[i - 1];

    }

    arr[0] = temp;

}

1. **Shift negative elem to one side**

void shiftNegativeOneSide(int arr[], int size)

{

    int l = 0;

    int h = size - 1;

    while (l < h)

    {

        if (arr[l] < 0)

        {

            l++;

        }

        else if (arr[h] > 0)

        {

            h--;

        }

        else

        {

            swap(arr[l], arr[h]);

        }

    }

}

**For maintaining order either we need to use extra auxillary space or T:C will be O(n^2)**

1. **Sort Colors**

We don’t do index++ when nums[index] == 2 because we don’t know after swapping at index will it be 0/1/2 but in case of nums[index] == 0 /1 we are sure that it will be 0 or 1

class Solution {

public:

    void sortColors(vector<int>& nums) {

        int n = nums.size();

        int index = 0; // looping variable

        int left = 0;

        int right = n-1;

        while(index <= right) { // imp!!!!!!!

            if(nums[index] == 0) {

                swap(nums[index], nums[left]);

                left++;

                index++;

            }

            else if(nums[index] == 2) {

                swap(nums[index], nums[right]);

                right--;

                //catch -> no need of index++

                //index++

            }

            else {

                index++;

            }

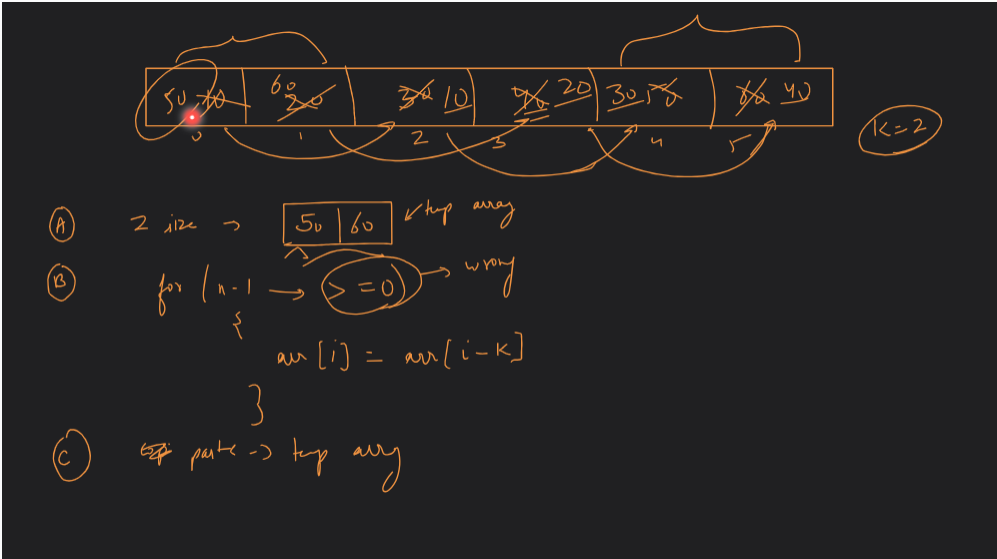
        }

    }

};

1. **Rotate array to the right by k steps**

**First approach**

****

class Solution {

public:

    void rotate(vector<int>& nums, int k) {

        int n = nums.size();

        vector<int> temp(k);

        k = k % n;

        int j=0;

        for(int i=n-k;i<n;i++){

            temp[j++] = nums[i];

        }

        for(int i=n-1;i>=k;i--){

            nums[i] = nums[i-k];

        }

        for(int i=0;i<k;i++){

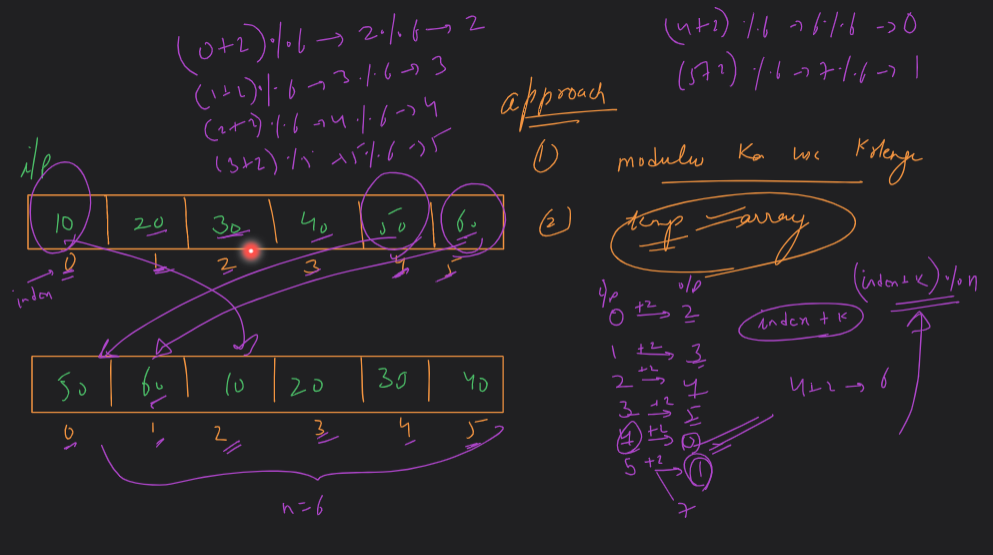
            nums[i] = temp[i];

        }

    }

};

**Second approach**

****

class Solution {

public:

    void rotate(vector<int>& nums, int k) {

        vector<int> result(nums.size());

        for(int i=0;i<nums.size();i++){

            int new\_ = (i + k) % nums.size();

            result[new\_] = nums[i];

        }

        nums = result;

    }

};

**Third approach**

class Solution {

public:

    void reverse(vector<int> &nums, int start, int end){

        while(start <= end)

            swap(nums[start++],nums[end--]);

    }

    void rotate(vector<int>& nums, int k) {

        int n = nums.size();

        k %= n;

        reverse(nums,0,n-1);

        reverse(nums,0,k-1);

        reverse(nums,k,n-1);

    }

};

1. **Row with maximum ones**

class Solution {

public:

    vector<int> rowAndMaximumOnes(vector<vector<int>>& mat) {

        int idx = 0;

        int countOnes = 0;

        for(int row=0;row<mat.size();row++){

            int tempCount = 0;

            for(int col=0;col<mat[row].size();col++){

                if(mat[row][col] == 1){

                    tempCount++;

                    if(tempCount > countOnes){

                        countOnes = tempCount;

                        idx = row;

                    }

                }

            }

        }

        return {idx,countOnes};

    }

};

1. **Rearrange Elements by sign (lc-2149)**

class Solution {

public:

    vector<int> rearrangeArray(vector<int>& nums) {

        int n = nums.size();

        int index = 0;

        int evenIndex = 0;

        int oddIndex = 1;

        vector<int> ans(n);

       while(index < n ) {

           if(nums[index] > 0) {

            ans[evenIndex] = nums[index];

            evenIndex += 2;

           }

           index++;

       }

       index = 0;

        while(index < n ) {

           if(nums[index] < 0) {

            ans[oddIndex] = nums[index];

            oddIndex += 2;

           }

           index++;

       }

       return ans;

    }

};

1. **Transpose of a matrix**

void transpose(int arr[][4], int row, int col)

{

    for (int i = 0; i < row; i++)

    {

        for (int j = i; j < col; j++)

        {

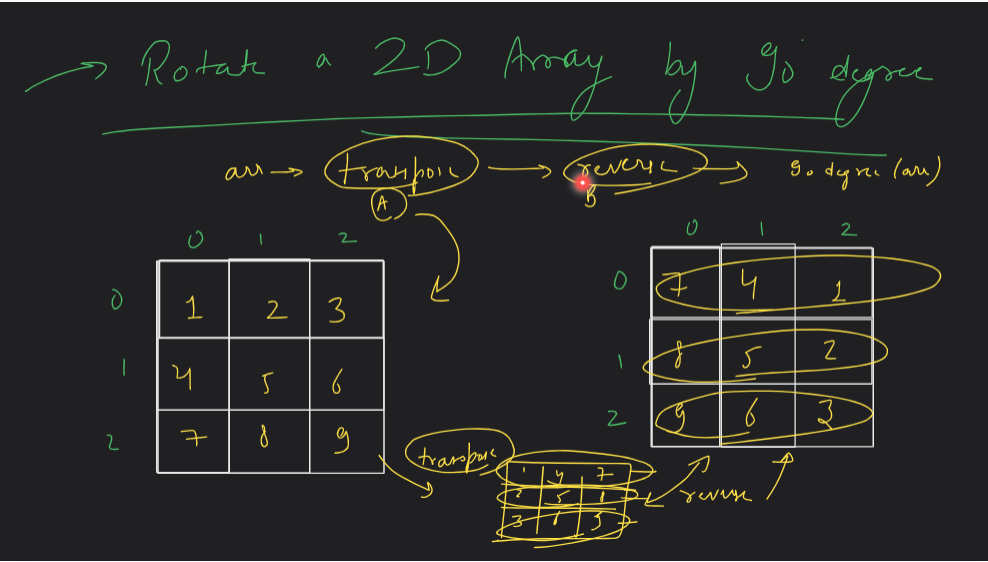
            swap(arr[i][j], arr[j][i]);

        }

    }

}

1. **Rotate Image (verry imp) (lc -> 48)**

****

class Solution {

public:

    void rotate(vector<vector<int>>& matrix) {

        // take transpose

        for (int row = 0; row < matrix.size(); row++) {

            for (int col = row; col < matrix.size(); col++) {

                swap(matrix[row][col], matrix[col][row]);

            }

        }

        // reverse

        for(int row = 0;row<matrix.size();row++){

            reverse(matrix[row].begin(),matrix[row].end());

        }

    }

};