Striver’s SDE Sheet

Day1: Arrays

**Set Matrix Zeros**

Given an 'N' x 'M' integer matrix, if an element is 0,set its entire row and column to O's, and return the matrix. In particular, your task is to modify it in such a way that if a cell has a value 0 (matrix[i][j] == O), then all the cells of the ith row and jth column should be changed to O.

You must do it in place.

Ans:

Observation and Approach:( Process Based Question )

A simple question with only implementation stuff

Brute :

Travel whole matrix and do the thing that question say set to zero and due its small range it works well on brute force TC : O((N*M)*(N+M)), SC : O(N\*M).

Code:

int n = matrix.size();

    int m = matrix[0].size();

    vector<vector<int>> visited = matrix;

      for(int i=0; i<n; i++){

            for(int j=0; j<m; j++){

                if(matrix[i][j] == 0){

                    for(int k=0; k<m; k++){

                        visited[i][k] = 0;

                    }

                     for(int k=0; k<n; k++){

                        visited[k][j] = 0;

                    }

                }

            }

        }

    for(int i=0; i<n; i++){

            for(int j=0; j<m; j++){

                matrix[i][j] = visited[i][j];

            }

        }

Optimize:

To optimize it we have to use some extra space , like maintain one array for row and column if and element is zero then set zero to that index corresponding index in array

Code:

Time Complexity : O(N\*M),

Space Complexity : O(N+M)

int n = matrix.size();

int m = matrix[0].size();

unordered\_set<int> setRows;

unordered\_set<int> setColumns;

for(int i=0; i<n; i++){

for(int j=0; j<m; j++){

if(matrix[i][j] == 0){

setRows.insert(i);

setColumns.insert(j);

}

}

}

for(int i=0; i<n; i++){

for(int j=0; j<m; j++){

if(setRows.count(i) > 0 || setColumns.count(j) > 0){

matrix[i][j] = 0;

}

}

}

Pascal's Triangle

Given an integer n, return the first n of **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:

<https://upload.wikimedia.org/wikipedia/commons/0/0d/PascalTriangleAnimated2.gif>

Ans:

Observation and Approach: ( Pattern based question )

In first go idea that strike in my mind is that number are generated in a pattern so we have to identify the pattern and get answer

And another thing is that next number is dependent on previous number so we can apply DP

Code:

  vector<vector<long long int>> output(n);

        for(int i=0; i<n; i++){

            output[i].resize(i+1, 1);

            for(int j=1; j<i; j++){

                output[i][j] = output[i-1][j-1] + output[i-1][j];

            }

        }

        return output;

Time Complexity : O(N^2)

Space Complexity : O(1)

Any variation:

1. What if we change the pattern like instead of + change to multiply

(Do not effect)

1. What if we have to find n from give pattern (Maths Stuff)

Next Permutation

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

* For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

* For example, the next permutation of arr = [1,2,3] is [1,3,2].
* Similarly, the next permutation of arr = [2,3,1] is [3,1,2].
* While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, *find the next permutation of* nums.

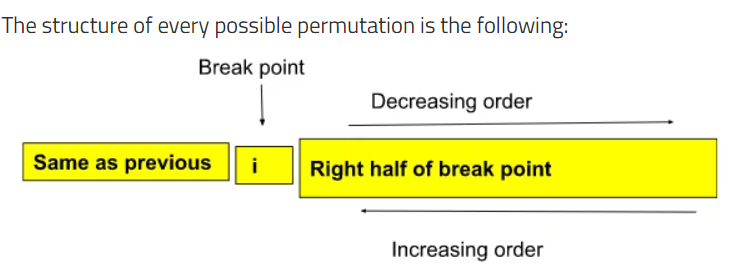
The replacement must be [**in place**](http://en.wikipedia.org/wiki/In-place_algorithm) and use only constant extra memory.

Ans:

Observation and Approach: (Pattern Based Question)

We can solve this question using 3 ways.

1. Find all the permutation of Array(nums) then we can easily find next permutation. ( too Much time 🥲)
2. Solved using Array + Two Pointers.
3. Solved using next\_permutation (inbuilt) function. (sigma Answer 😎)



Code:

void nextPermutation(vector<int>& nums) {

        int n = nums.size(), index = -1;

        for(int i=n-2; i>=0; i--){

            if(nums[i] < nums[i+1]){

                index = i;

                break;

            }

        }

        for(int i=n-1; i>=index && index != -1; i--){

            if(nums[i] > nums[index]){

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

                break;

            }

        }

        reverse(nums.begin() + index + 1, nums.end());

    }

Possible implementation of STL next\_Permutation

bool next\_permutation(BidirIt first, BidirIt last)

{

auto r\_first = [std::make\_reverse\_iterator](https://en.cppreference.com/w/cpp/iterator/make_reverse_iterator)(last);

auto r\_last = [std::make\_reverse\_iterator](https://en.cppreference.com/w/cpp/iterator/make_reverse_iterator)(first);

auto left = [std::is\_sorted\_until](https://en.cppreference.com/w/cpp/algorithm/is_sorted_until)(r\_first, r\_last);

if (left != r\_last)

{

auto right = [std::upper\_bound](https://en.cppreference.com/w/cpp/algorithm/upper_bound)(r\_first, left, \*left);

[std::iter\_swap](https://en.cppreference.com/w/cpp/algorithm/iter_swap)(left, right);

}

[std::reverse](https://en.cppreference.com/w/cpp/algorithm/reverse)(left.base(), last);

return left != r\_last;

}

Possible variation:

* 1. What if it ask for previous Permutation

We can use stl and for basic approach just change the order

(increasing to decreasing)

Maximum Subarray

Given an integer array nums, find the subarray with the largest sum, and return *its sum*.

Ans:

Observation and Approach : (Pattern)

The question is easy because only ask the sum so due this lies on pattern question but if we have to find and show sub array then this question have to do with help of dp

The algo is very simple , if our sum if negative means less than zero then just set it to zero and if our sum is positive than find max from it and this max is maintain by max variable

Code:

 Kadane's Algorithm

long long crntSum=0;

    long long maxSum = INT\_MIN;

    for(int i=0; i<n; i++){

        crntSum+=arr[i];

        if(crntSum < 0){

            crntSum=0;

        }

    maxSum=max(maxSum, crntSum);

    }

    return maxSum;

TC :O(N)

we can also this question by finding all possible sub array and find sum of that and return to answer to but this take expo time

**Sort 0 1 2**

Given an array nums with n objects colored red, white, or blue, sort them [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm)so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Ans:

Observation and Approach : (Pattern)

We have to just sort the array so we can do this in so many different way like:

1. Apply stl (sort)
2. Collect all 1 and 2 then put all this thing in sorted way in new array
3. Or can do like some binary search type things (Dutch National Flag Algorithm)

Code:

let left = 0;

let current = 0;

let right = nums.length - 1;

while( current <= right ){

if(nums[current] === 0){

swap(nums, left, current);

left++;

current++;

}else if (nums[current] === 2){

swap(nums, current, right);

right--;

}else{

current++;

}

}

Best Time to Buy and Sell Stock

(Imp because of its variants)

You are given an array prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.

Ans:

Observation and Approach : (Condition based question)

Brute:

Find profit of all and then return the max

Take O(N^2) time, not feasible

To reduce time we used some space , in this case we use an array that contain max from left so we make decision on each day that buy or not but this take approach take O(N) time and space

So to reduce space we use approach that instead of storing in to array we can calculate the max on air and decide whether to buy or not , it take O(N) time and constant space

Code:

int lsf = INT\_MAX;

        int op = 0;

        int pist = 0;

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

            if(prices[i] < lsf){

                lsf = prices[i];

            }

            pist = prices[i] - lsf;

            if(op < pist){

                op = pist;

            }

        }

        return op;

variants: different type of variants possible like what if 1) multiple time we buy or sell 2) some condition like on this days only we buy and only on this day we can sell 3)find max profit from this array and many more …