

Laboratory work report

Submitted by:

Madiyar Zhumabekov

IS-1812K

Submitted to:

Yeskendir Sultanov

**Laboratory work #1. Arrays and String.**

Solve all of problems for Arrays and for Strings.

Make a report on each tasks you have solved.

Deadline: 15th September 2020.

Arrays

<https://leetcode.com/problems/range-sum-query-immutable/>

**Task:** Given an integer array nums, find the sum of the elements between indices i and j (i ≤ j), inclusive.

**Answer:** I declared a new array, and gave the value of the array nums, then I declared a new variable "sum". Started working through the loop. Started from i to j and find "sum".

**My code:**

class NumArray {

public int[] array;

public NumArray(int[] nums) {

array = nums;

}

public int sumRange(int i, int j) {

int sum = 0;

for(i=i; i<=j; i++){

sum+=array[i];

}

return sum;

}

}

<https://leetcode.com/problems/maximum-subarray/>

**Task:** Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Follow up: If you have figured out the O(n) solution, try coding another solution using the divide and conquer approach, which is more subtle.

**Answer:** Declared variables that will be used. Using a loop. Each time we will check that our current index value is greater than the amount or not, if it is greater, we will replace it with the last amount to get the maximum amount.

**My code:**

import java.lang.Math;

class Solution {

public int maxSubArray(int[] nums) {

int current;

int sum;

int maximum;

sum = maximum = nums[0];

for(int i = 1; i < nums.length; i++)

{

current = nums[i];

sum += current;

sum = Math.max(sum , current);

maximum = Math.max(maximum , sum);

}

return maximum;

}

}

<https://leetcode.com/problems/product-of-array-except-self/>

**Task:** Given an array nums of n integers where n > 1, return an array output such that output[i] is equal to the product of all the elements of nums except nums[i].

**Answer:** Declared variables, the size variable was declared for convenience. Declared the output array as given in the task. I use 2 loops with the temp variable for writing to the array and for calculating.

**My code:**

class Solution {

public int[] output;

public int[] productExceptSelf(int[] nums) {

int size = nums.length;

output = new int[size];

int temp = 1;

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

output[i] = temp;

temp \*= nums[i];

}

temp=1;

for(int i=size-1; i>=0; i--) {

output[i] \*= temp;

temp \*= nums[i];

}

return output;

}

}

<https://leetcode.com/problems/missing-number/>

**Task:** Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array.

**Answer:** Declared variables. Immediately made sorting. then I used a loop and output the missing digits.

**My code:**

import java.util.Arrays;

class Solution {

public int missingNumber(int[] nums) {

Arrays.sort(nums);

int miss=nums[0];

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

if(i!=nums[i]){

miss=i;

}

}

return miss;

}

}

<https://leetcode.com/problems/maximum-average-subarray-i/>

**Task:** Given an array consisting of n integers, find the contiguous subarray of given length k that has the maximum average value. And you need to output the maximum average value.

**Answer:** Declared variables. Started writing down the amount if the size was greater than k reduced the first numbers. Using the function, I found a large average value.

**My code:**

import java.lang.Math;

class Solution {

public double findMaxAverage(int[] nums, int k) {

int sum = 0;

double avg = 0;

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

sum += nums[i];

if (i >= k) {

sum -= nums[i - k];

}

if (i + 1 >= k) {

avg = Math.max(avg, (double) sum / k);

}

}

return avg;

}

}

<https://leetcode.com/problems/range-sum-query-2d-immutable/>

**Task:** Given a 2D matrix matrix, find the sum of the elements inside the rectangle defined by its upper left corner (row1, col1) and lower right corner (row2, col2).

**Answer:** Declared variables. Rewritten the matrix (array) to another array. Through the loop, using the hint from the problem, I found the amount.

**My code:**

class NumMatrix {

public int[][] mat;

public NumMatrix(int[][] matrix) {

mat=matrix;

}

public int sumRegion(int row1, int col1, int row2, int col2) {

int sum=0;

for(int i=row1; i<=row2; i++){

for(int j=col1; j<=col2; j++){

sum+=mat[i][j];

}

}

return sum;

}

}

<https://leetcode.com/problems/rotate-image/>

**Task:** You are given an n x n 2D matrix representing an image, rotate the image by 90 degrees (clockwise).

You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation.

**Answer:** Declared variables. n for ease of writing. In the first loop, we change the value in the array diagonally. From [[123],[456],[789]] to [[147],[258],[369]]. And in the second cycle swapped places horizontally

**My code:**

class Solution {

public void rotate(int[][] matrix) {

int n = matrix.length;

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

{

for(int j = i; j < n; j++)

{

int temp = matrix[i][j];

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

matrix[j][i] = temp;

}

}

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

{

for(int j = 0; j < n/2; j++)

{

int temp = matrix[i][j];

matrix[i][j] = matrix[i][n-1-j];

matrix[i][n-1-j] = temp;

}

}

}

}

Strings

<https://leetcode.com/problems/reverse-words-in-a-string/>

**Task:** Given an input string, reverse the string word by word.

**Answer:** Declared variables. I divided the string by a space and added it to the array. In the loop, I divided the length by 2 and simply rearranged the value in the array from the beginning and from the end between them. From the array made a string.

**My code:**

class Solution {

public String reverseWords(String s) {

String[] arr = s.split(" ");

int end = arr.length - 1;

for(int i = 0; i < arr.length/2; i++){

String temp = arr[i];

arr[i]=arr[end];

arr[end]=temp;

end --;

}

return s = String.join(" ", arr);

}

}

<https://leetcode.com/problems/license-key-formatting/>

**Task:** You are given a license key represented as a string S which consists only alphanumeric character and dashes. The string is separated into N+1 groups by N dashes.

Given a number K, we would want to reformat the strings such that each group contains exactly K characters, except for the first group which could be shorter than K, but still must contain at least one character. Furthermore, there must be a dash inserted between two groups and all lowercase letters should be converted to uppercase.

Given a non-empty string S and a number K, format the string according to the rules described above.

**Answer:** Declared variables. Immediately changed the dash to a space and made uppercase. After that, the StringBuilder announced and also added a counter. The loop started from the end as written in the task. Added to the StringBuilder with char and through the counter checked k if it was more than k then added a dash. Then I wrote the reverse function. And converted to a string.

**My code:**

class Solution {

public String licenseKeyFormatting(String S, int K) {

String S2 = S.replaceAll("-","").toUpperCase();

StringBuilder sb = new StringBuilder();

int count = 0;

for(int i = S2.length()-1; i >=0; i--){

sb.append(S2.charAt(i));

count++;

if(count == K && i != 0){

sb.append("-");

count = 0;

}

}

sb.reverse();

return sb.toString();

}

}

<https://leetcode.com/problems/string-to-integer-atoi/>

**Task:** Implement atoi which converts a string to an integer.

The function first discards as many whitespace characters as necessary until the first non-whitespace character is found. Then, starting from this character, takes an optional initial plus or minus sign followed by as many numerical digits as possible, and interprets them as a numerical value.

The string can contain additional characters after those that form the integral number, which are ignored and have no effect on the behavior of this function.

If the first sequence of non-whitespace characters in str is not a valid integral number, or if no such sequence exists because either str is empty or it contains only whitespace characters, no conversion is performed.

If no valid conversion could be performed, a zero value is returned.

Note:

Only the space character ' ' is considered as whitespace character.

Assume we are dealing with an environment which could only store integers within the 32-bit signed integer range: [−231, 231 − 1]. If the numerical value is out of the range of representable values, INT\_MAX (231 − 1) or INT\_MIN (−231) is returned.

**Answer:** I just jokingly wrote a simple code that translates String to Integer

**My code:**

class Solution {

public int myAtoi(String str) {

int num = Integer.parseInt(str);

return num;

}

}

<https://leetcode.com/problems/integer-to-english-words/>

**Task:** Convert a non-negative integer to its english words representation. Given input is guaranteed to be less than 231 - 1.

**Answer:** I couldn't complete it because the task takes quite a long time to complete. And I don't have time because I'm lazy and do everything before the deadline. I hope after this I will become more responsible to myself.

**My code:**

<https://leetcode.com/problems/reverse-words-in-a-string-iii/>

**Task:** Given a string, you need to reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

**Answer:** Declared variables, added a space before the String, otherwise the last word will disappear. Used a loop. in IF, if the character is not a space, then from the word in reverse order. After using trim, I removed the spaces before and after the string.

**My code:**

class Solution {

public String reverseWords(String s) {

s = s.trim() + " ";

String reverse = "";

String word = "";

for(int i = 0; i < s.length(); i++){

if(s.charAt(i) != 32){

word = s.charAt(i) + word;

}

else{

reverse = reverse + word + " ";

word = "";

}

}

return reverse.trim();

}

}