Laboratory work #1. Arrays and String.

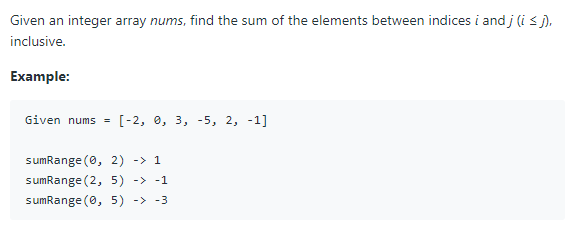
Solve all of problems for Arrays and for Strings.

Make a report on each tasks you have solved.

Deadline: 15th September 2020.

Arrays

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



class NumArray {

private int[] data;

public NumArray(int[] nums) {

data = nums;

}

public int sumRange(int i, int j) {

int sum = 0;

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

sum += data[k];

}

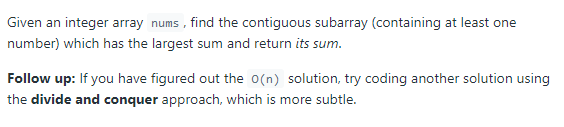
return sum;

}

}

I create the array data, and this array take the elements from nums array. Then the function sumRange go cycle bettwen I and j. Then the variable sum added for each cycle. The final return the sum.

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



class Solution {

public int maxSubArray(int[] nums) {

int maxSub = nums[0], curSum = 0;

for (int n : nums) {

if (curSum < 0)

curSum = 0;

curSum += n;

maxSub = Math.max(maxSub, curSum);

}

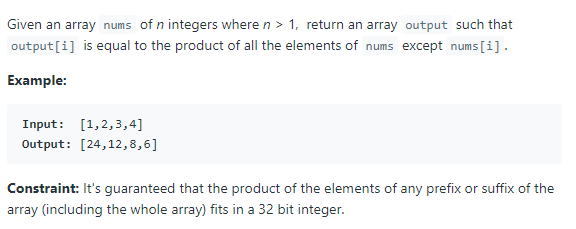
return maxSub;

}

}

maxSub is maximum in the subarray. Firstly, I take the maxSum first element of the array nums. I take variable current sum. I go with cycle from first element to last element in the array nums. I have condition current sum less than 0, then the current sum will equal to 0. In other condition the current sum will summary the n’s element. The cycle will run with elements from the array nums. Then with Math max function I take the maximum element between maximum subarray and current sum. After the cycle the program will show the maximum subarray.

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



class Solution {

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

int length = nums.length;

int[] L = new int[length];

int[] R = new int[length];

int[] answer = new int[length];

L[0] = 1;

for (int i = 1; i < length; i++) {

L[i] = nums[i - 1] \* L[i - 1];

}

R[length - 1] = 1;

for (int i = length - 2; i >= 0; i--) {

R[i] = nums[i + 1] \* R[i + 1];

}

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

answer[i] = L[i] \* R[i];

}

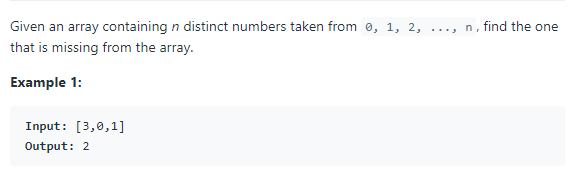
return answer;

}

}

Firstly, create 3 arrays the length equal to the nums length. L is left array. The first element of the L array equal to 1. Then cycle start from 1 to length of the nums array. And left I’s element equal to the nums (i-1)’s element multiply to left (i-1). The next cycle to the right array. Right las t elemet equal to the 1. The cycle start from length -2 to 0. The right I’s element equal nums (i+1) multiply right (i+1)’s element. The third cycle I use to combine the two arrays like r and l. The cycle go, left i’s element multiply to the right I’s element and write to the answer array.

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



Class Solution {

public int missingNumber(int[] nums) {

int sumIfNoneMissing = 0;

int actualSum = 0;

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

sumIfNoneMissing += i+1;

actualSum += nums[i];

}

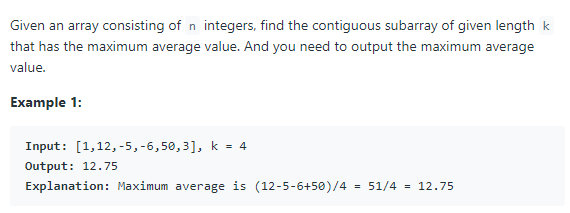
return sumIfNoneMissing - actualSum;

}

}

The take to variable like sumIfNoneMissing and actual sum, both of them equal to the 0. The cycle start from 0 to nums.length. The sumIfNoneMissing add (i+1) because it add only indexes of array but only more than 0. The next variable added for each cycle the element of the array. Then find the difference between sumIfNoneMissing and actualSum.

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



public class Solution {

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

double sum=0;

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

sum+=nums[i];

double res=sum;

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

sum+=nums[i]-nums[i-k];

res=Math.max(res,sum);

}

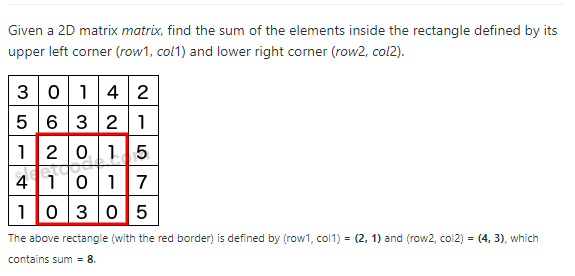
return res/k;

}

}

Firstly, find the sum of the all elements of array nums. Then res variable equal to the sum. Then go with cycle but the cycle start from k to length of nums array. The sum adds difference between nums i’s element and nums (i-k)’s element. Then res equal to the maximum of the res and sum. Result will show res divide by k.

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



class NumMatrix {

private int [][] data;

public NumMatrix(int[][] matrix) {

data = 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 += data[i][j];

}

}

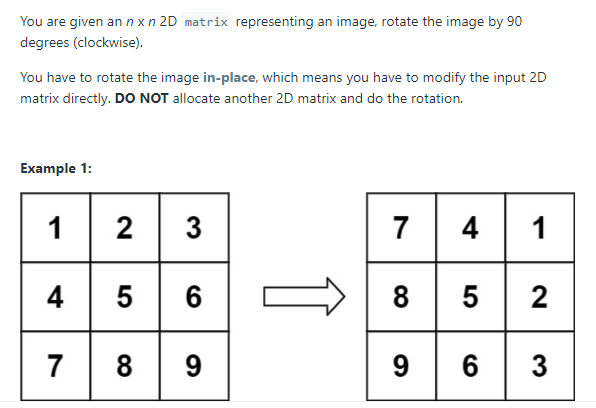
return sum;

}

}

The program runs with 2 dimension array. First cycle starts from row1 and run to the rows2. The next cycle rum from col1 to col2. The sum added data[i][j] element of the array.

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



Class Solution {

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

int n = matrix.length;

int temp;

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

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

temp = matrix[i][j];

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

matrix[j][i] = temp;

}

}

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

int start = 0;

int end = n - 1;

while (start < end) {

int m = matrix[i][start];

matrix[i][start] = matrix[i][end];

matrix[i][end] = m;

start++;

end--;

}

}

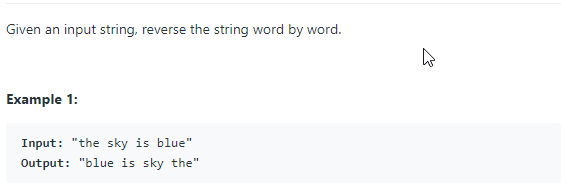
}

}

The program run with two dimensional array. It runs with 2 cycles. And swap the elements, the condition is [i][j] = [j][i]. The next cycle starts from 0 to the n. The while cycle change the elements located on the vertex of the array.

Strings

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



class Solution {

public String reverseWords(String s) {

String str=s.trim();

StringBuffer sBuffer = new StringBuffer();

int counter = 0;

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

if(str.charAt(i) != ' ') counter++;

else{

if(str.charAt(i+1) != ' '){

sBuffer.append(str.substring(i, i + counter + 1));

counter = 0;

}

}

if(i == 0){

sBuffer.append(" ");

sBuffer.append(str.substring(0, counter));

}

}

return sBuffer.toString().trim();

}

}

Trim remove the space from begin and end of string. The cycle run in inverse order. The program run inverse order and run to the space, then substring the from the i’s to (i+counter+1) index. Then append to the new string. If index equal to the 0, it cut the from 0 element to the counter. The append it to the string.

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

Solution {

public String licenseKeyFormatting(String S, int K) {

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

StringBuffer licence = new StringBuffer();

int counter = 0;

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

if(counter == K){

licence.append("-");

licence.append(str.charAt(i));

counter = 1;

}else{

licence.append(str.charAt(i));

counter++;

}

}

return licence.reverse().toString();

}

}

Firstly, replace the – to the “” in the string and all alphabets change to uppercase. The cycle run from length -1 to 0 inverse order. If counter equals to the k, the new string add – the char at string, else it add char at string. In the end the licence string show inverse order.

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

Solution {

public int myAtoi(String str) {

long num=0;

str=str.trim();

String[] sss=str.split(" ");

for (int i=sss[0].length();i>0; i--){

try{

num=Long.parseLong(sss[0].substring(0,i));

if(num<= Integer.MIN\_VALUE) return Integer.MIN\_VALUE;

if(num>= Integer.MAX\_VALUE) return Integer.MAX\_VALUE;

return (int)num;

}catch(NumberFormatException e){

return 0;

}

}

return (int)num;

}

}

Remove the space from begin and end of the string. The program divides the element with space using split function. The cycle run only first element of sss string. And I run from length from sss string to 0; num take element from 0 to the i’s element and change to int.Then condition that given us.

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

class Solution {

public String numberToWords(int num) {

StringBuilder res = new StringBuilder("");

String[] s = {"Hundred", "Thousand", "Million", "Billion"};

String[] w0 = {"Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};

String[] w1 = {"Zero","Ten", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};

String[] w2 = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen" ,"Sixteen", "Seventeen", "Eighteen", "Nineteen"};

int[] numArr = new int[4];

int i = 0;

while(num>0){

numArr[i++] += (num%1000);

num/=1000;

}

for(int i = 3; i>=0 ;i--){

if(numArr[i]!=0){

if(numArr[i] > 99){

res.append(w0[numArr[i]/100]).append(" ").append(s[0]).append(" "); // One Hundred

numArr[i] = numArr[i] - (numArr[i]/100)\*100; // numArr[i] = 23

}if(numArr[i] > 19){

res.append(w1[numArr[i]/10]).append(" "); // Twenty

numArr[i] = numArr[i] - (numArr[i]/10)\*10; // numArr[i] = 3

}if(numArr[i] > 9){

res.append(w2[numArr[i]%10]).append(" ");

numArr[i] = 0;

}if(numArr[i] > 0)

res.append(w0[numArr[i]]).append(" "); // Three

if(i!=0)

res.append(s[i]).append(" "); // when i=1: Thousand

}

}

return res.length() == 0 ? w0[0] : res.substring(0,res.length()-1).toString(); //SB toString, remove last space

}

}

num < 2^31 -> at most 10 digits we want to know how many billions, millions, thousands and the rest we have. so we need 4 numbers when #3 = billions, #2 = millions etc.

Every number is 3 digits, for example - 1234567891 = 1B + 234M + 567T + 891.

numArr[3]\*Billion + numArr[2]\*Million + numArr[1]\*Thusand + numArr[0] = num

example: num = 123,123-> numArr[0] = numArr[1] = 123

if num == 0 nothing was done -> res = "";

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

class Solution {

public String reverseWords(String s) {

s = s.trim();

StringBuffer result = new StringBuffer();

StringBuffer word = new StringBuffer();

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

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

word.append(s.charAt(i));

} else{

result.append(word.reverse());

result.append(" ");

word.setLength(0);

}

}

result.append(word.reverse());

return result.toString();

}

}

The trim removes the space at the begin and end of the string. Then cycle run from 0 to the length of the string. If I’s element equal to the space it appends to the word string. Else it appends the reverse order word string.