Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего образование

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Решение задач

Выполнил: студент группы БВТ1902

Гинатуллин Айгиз

Москва

2021

**Задание**

**ЗАДАЧА 1**

Даны две строки: s1 и s2 с одинаковым размером, проверьте, может ли

некоторая перестановка строки s1 “победить” некоторую перестановку

строки s2 или наоборот.

Строка x может “победить” строку y (обе имеют размер n), если x[i]> = y

[i] (в алфавитном порядке) для всех i от 0 до n-1.

Примеры:

Input: s1 = "abc", s2 = "xya"

Output: true

Объяснение: «ayx» - это перестановка строки s2 = «xya»,

которая “побеждает” строку s1 = «abc».

Input: s1 = "abe", s2 = "acd"

Output: false

Объяснение: Все перестановки для s1 = "abe": "abe", "aeb",

"bae", "bea", "eab" и "eba", а все перестановки для s2 =

"acd": "acd", «adc», «cad», «cda», «dac» и «ca». Однако нет

никакой перестановки строки s1, которая может нарушить

некоторую перестановку строки s2 и наоборот.

s1.length == n

s2.length == n

1 <= n <= 10^5

**ЗАДАЧА 2**

Дана строка s, вернуть самую длинную полиндромную подстроку в s.

Примеры:

Input: s = "babad"

Output: "bab"

Note: "aba" is also a valid answer.

Input: s = "cbbd"

Output: "bb"

**ЗАДАЧА 3**

Вернуть количество отдельных непустых подстрок текста, которые могут

быть записаны как конкатенация некоторой строки с самой собой (т.е. она

может быть записана, как a + a, где a - некоторая строка).

Примеры:

Input: text = "abcabcabc"

Output: 3

Explanation: The 3 substrings are "abcabc", "bcabca"

and "cabcab".

Задача 4

На столе стоят 3*n* стопок монет. Вы и ваши друзья Алиса и Боб забираете стопки монет по

следующему алгоритму:

1. Вы выбираете 3 стопки монет из оставшихся на столе.

2. Алиса забирает себе стопку с максимальным количеством монет.

3. Вы забираете одну из двух оставшихся стопок.

4. Боб забирает последнюю стопку.

5. Если еще остались стопки, то действия повторяются с первого шага.

Дан массив целых положительных чисел piles. Напишите функцию, возвращающую максимальное

число монет, которое вы можете получить.

**Пример 1.1:**

**Ввод:** piles = [2,4,1,2,7,8]

**Вывод:** 9

**Пример 1.2:**

**Ввод:** piles = [2,4,5]

**Вывод:** 4

**Пример 1.3:**

**Ввод:** piles = [9,8,7,6,5,1,2,3,4]

**Вывод:** 18

**Ограничения:**

• 3 ⩽ *len*(*piles*) ⩽ 105

• *len*(*piles*) *mod* 3 == 0

• 1 ⩽ *piles*[*i*] ⩽ 104

**Задача 5. «Треугольник с максимальным периметром»**

Массив A состоит из целых положительных чисел длин

отрезков. Составьте из трех отрезков

такой треугольник, чтобы его периметр был максимально возможным. Если невозможно

составить треугольник с положительной площадью функция

возвращает 0.

**Пример 1.1:**

**Ввод:** [2,1,2]

**Вывод:** 5

**Пример 1.2:**

**Ввод:** [1,2,1]

**Вывод:** 0

**Пример 1.3:**

**Ввод:** [3,2,3,4]

**Вывод:** 10

**Пример 1.4:**

**Ввод:** [3,6,2,3]

**Вывод:** 8

**Ограничения:**

• 3 ⩽ *len*(*A*) ⩽ 10000

• 1 ⩽ *A*[*i*] ⩽ 106

**Задача 6. «Максимальное число»**

Дан массив неотрицательных целых чисел nums. Расположите их в таком порядке, чтобы

вместе они образовали максимально возможное число.

**Замечание:** Результат может быть очень большим числом, поэтому представьте его как

string, а не integer.

**Пример 2.1:**

**Ввод:** nums = [10,2]

**Вывод:** ”210”

**Пример 2.2:**

**Ввод:** nums = [3,30,34,5,9]

**Вывод:** ”9534330”

**Пример 2.3:**

**Ввод:** nums = [1]

**Вывод:** ”1”

**Пример 2.4:**

**Ввод:** nums = [10]

**Вывод:** ”10”

**Ограничения:**

• 1 ⩽ *len*(*nums*) ⩽ 100

• 0 ⩽ *nums*[*i*] ⩽ 109

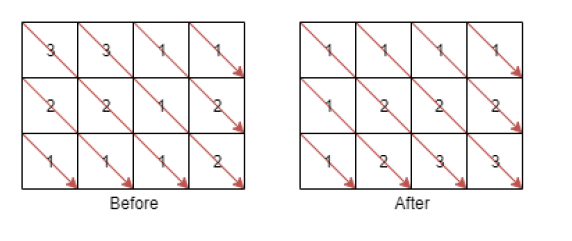
1

**Задача 7. «Сортировка диагоналей в матрице»**

Дана матрица mat размером m \* n, значения целочисленные.

Напишите функцию, сортирующую

каждую диагональ матрицы по возрастанию и возвращающую получившуюся матрицу.



**Пример 3.1:**

**Ввод:** mat = [[3, 3, 1, 1], [2, 2, 1, 2], [1, 1, 1, 2]]

**Вывод:** [[1, 1, 1, 1], [1, 2, 2, 2], [1, 2, 3, 3]]

**Пример 3.2:**

**Ввод:** mat = [[11, 25, 66, 1, 69, 7], [23, 55, 17, 45, 15, 52], [75, 31, 36,

44, 58, 8], [22, 27, 33, 25, 68, 4], [84, 28, 14, 11, 5, 50]]

**Вывод:** [[5, 17, 4, 1, 52, 7], [11, 11, 25, 45, 8, 69], [14, 23, 25, 44, 58,

15], [22, 27, 31, 36, 50, 66], [84, 28, 75, 33, 55, 68]]

**Ограничения:**

• *m* == *len*(*mat*)

• *n* == *len*(*mat*[*i*])

• 1 ⩽ *m, n* ⩽ 100

• 1 ⩽ *mat*[*i*][*j*] ⩽ 100

**Задача 8. «Шарики и стрелы»**

Некоторые сферические шарики распределены по двухмерному пространству. Для каждого

шарика даны *x*координаты

начала и конца его горизонтального диаметра. Так как пространство

двумерно, то *y*координаты

не имеют значения в данной задаче. Координата *xstart* всегда меньше

*xend*.

Стрелу можно выстрелить строго вертикально (вдоль *y*оси)

из разных точек *x*оси.

Шарик

с координатами *xstart* и *xend* уничтожается стрелой, если она была выпущена из такой позиции

*x*, что *xstart* ⩽ *x* ⩽ *xend*. Когда стрела выпущена, она летит в пространстве бесконечное время

(уничтожая все шарики на пути).

Дан массив points, где points[i] = [*xstart*, *xend*]. Напишите функцию, возвращающую

минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

**Пример 1.1:**

**Ввод:** points = [[10,16],[2,8],[1,6],[7,12]]

**Вывод:** 2

**Пример 1.2:**

**Ввод:** points = [[1,2],[3,4],[5,6],[7,8]]

**Вывод:** 4

**Пример 1.3:**

**Ввод:** points = [[1,2],[2,3],[3,4],[4,5]]

**Вывод:** 2

**Пример 1.4:**

**Ввод:** points = [[1,2]]

**Вывод:** 1

**Пример 1.5:**

**Ввод:** points = [[2,3],[2,3]]

**Вывод:** 1

**Ограничения:**

• 0 ⩽ *len*(*points*) ⩽ 104

• *len*(*points*[*i*]) == 2

• *−*231 ⩽ *xstart < xend* ⩽ 231 *−* 1

1

Листинг программы

Tasks

import {strongString} from "./stringTasks/task1.js"  
import {largestPalindrome} from "./stringTasks/task2.js";  
import {largestConcat} from "./stringTasks/task3.js";  
import {triangle} from "./primitiveTasks/task1.js";  
import {maxNumber} from "./primitiveTasks/task2.js";  
import {matrixSort} from "./primitiveTasks/task3.js";  
import {piless} from "./money/piles.js";  
import {point} from "./points/PPoint.js";  
  
***console***.log(strongString("abc", "xya"))  
***console***.log(largestPalindrome("abacaca"))  
***console***.log(largestConcat("acaacaaca"))  
***console***.log(triangle([12,1,123,134]))  
***console***.log(maxNumber([8,99,10]))  
***console***.log(matrixSort([[3, 3, 5, 6], [2, 2, 1, 8], [1, 1, 1, 2], [2, 2, 1, 8]]));  
  
piless([2,4,5])  
  
// point([[1,2],[3,4],[5,6],[7,8]])  
point([[10,16],[2,8],[1,6],[7,12]])  
// point([[1,2],[2,3],[3,4],[4,5]])

task1

export const triangle = (data) =>{  
 data.sort((a, b) => b - a);  
 if(data.length >=3){  
 return {  
 a:data[0],  
 b:data[1],  
 c:data[2]  
 }  
 }  
 return {  
  
 }  
}

task2

const compare = (string1, string2) => {  
 return string1.toString() + string2.toString() > string2.toString()+ string1.toString()  
}  
  
export const maxNumber = (numbersArray) => {  
 if (numbersArray.length === 0) return 0  
 for (let i = 0; i < numbersArray.length; i++) {  
 for (let j = 0; j < numbersArray.length; j++) {  
 if (compare(numbersArray[i], numbersArray[j])) {  
 console.log(numbersArray[i],numbersArray[j]);  
 [numbersArray[i], numbersArray[j]] = [numbersArray[j], numbersArray[i]]  
 }  
 }  
 }  
 let result = ""  
 for(let i of numbersArray){  
 result +=i.toString()  
 }  
 return result  
}

task3

export const matrixSort = (mat) => {  
 let rowLen = mat[0].length, columnLen = mat.length  
 let buff = []  
 let triangleDia = ***Math***.min(rowLen, columnLen) - 1  
 let countMainDia = ***Math***.abs(rowLen - columnLen) + 1  
 let lenMainDia = ***Math***.min(rowLen, columnLen)  
  
 for (let i = 0; i < triangleDia; i++) {  
 for (let j = 0; j < i + 1; j++)  
 buff.push(mat[j][rowLen - 1 - i + j])  
 buff.sort((a, b) => {  
 return a - b  
 })  
  
 for (let j = 0; j < i + 1; j++)  
 mat[j][rowLen - 1 - i + j] = buff[j]  
 buff = []  
 }  
  
 for (let i = 0; i < countMainDia; i++) {  
 if (rowLen > columnLen) {  
 for (let j = 0; j < lenMainDia; j++)  
 buff.push(mat[j][j + i])  
 buff.sort((a, b) => {  
 return a - b  
 })  
  
 for (let j = 0; j < lenMainDia; j++)  
 mat[j][j + i] = buff[j]  
 buff = []  
 } else {  
 for (let j = 0; j < lenMainDia; j++) {  
 buff.push(mat[j + i][j])  
 }  
 buff.sort((a, b) => {  
 return a - b  
 })  
  
 for (let j = 0; j < lenMainDia; j++) {  
 mat[j + i][j] = buff[j]  
 }  
 buff = []  
 }  
 }  
  
 for (let i = 0; i < triangleDia; i++) {  
 for (let j = 0; j < i + 1; j++) {  
 buff.push(mat[columnLen - 1 - i + j][j])  
 }  
 buff.sort((a, b) => {  
 return a - b  
 })  
  
 for (let j = 0; j < i + 1; j++) {  
 mat[columnLen - 1 - i + j][j] = buff[j]  
 }  
 buff = []  
 }  
 return mat  
}

stringTasks/task 1

export const strongString = (string1, string2) => {  
 for(let i = 0;i < string2.length;i++) if(string1[i] > string2[i]) return "second"  
 return "first"  
}

stringTasks/task 2

function isPalindrom (str) {  
 return str === str.split('').reverse().join('');  
}  
  
export const largestPalindrome = (string) =>{  
 let largest = ''  
 for(let i = 0;i <= string.length;i++){  
 for(let j = 0;j <= string.length;j++){  
 const subString = string.slice(i,j)  
 ***console***.log(subString)  
 if(isPalindrom(subString)){  
 if(subString.length>largest.length){  
 largest = subString;  
 }  
 }  
 }  
 }  
 return largest  
}

stringTasks/task 3

const isConcat = (string, subString) => {  
 if (subString.length === 0) {  
 return false  
 }  
 for (let i = 0; i < string.length; i += subString.length)  
 if (string.slice(i, i + subString.length) !== subString)  
 return false  
 return true  
}  
  
export const largestConcat = (string) => {  
 let largest = ''  
 for (let i = 0; i < string.length; i++) {  
 for (let j = 0; j < string.length; j++) {  
 const subString = string.slice(i, j)  
 ***console***.log(subString)  
 if (isConcat(string, subString)) {  
 if (largest.length < subString.length) {  
 largest = subString  
 }  
 }  
 }  
 }  
 return largest  
}

Ppoint

export const point = (array) => {  
 let answer = []  
 let arr1 = []  
 for(let i=0;i<array.length;i++)  
 arr1[i]=0;  
 let k=0  
 let sum=0  
 for(let i=0;i<array.length;i++) {  
 let know=k  
 for (let j = i + 1; j < array.length; j++) {  
 let a = Math.max(array[i][0], array[j][0])  
 let b = Math.min(array[i][1], array[j][1])  
 answer[k] = compare(a, b)  
 if (answer[k] !== undefined) {  
 k++  
 arr1[j] = 1  
 array[j] = []  
 }  
 }  
 if(know!==k) {  
 arr1[i] = 1  
 array[i] = []  
 }  
 }  
 for(let i=0;i<arr1.length;i++)  
 if(arr1[i]!==1)  
 sum++  
 if(answer[answer.length-1]===undefined)  
 console.log("кол-во стрел = "+(answer.length-1+sum))  
 else  
 console.log("кол-во стрел = "+(answer.length+sum))  
}  
const compare = (a, b) => {  
 if(a<b)  
 return [a,b]  
 else if(a===b)  
 return [a,a]  
}

piles

export const piless = (array) => {  
 array.sort();  
 let sum=0  
 for(let i=array.length-2;i>=(array.length)/3;i-=2){  
 sum+=array[i]  
 console.log(array[i])  
 }  
 console.log("max sum = " + sum)  
}

Вывод программы

second  
  
a  
ab  
aba  
abac  
abaca  
abacac  
abacaca  
  
b  
ba  
bac  
baca  
bacac  
bacaca  
  
a  
ac  
aca  
acac  
acaca  
  
c  
ca  
cac  
caca  
  
a  
ac  
aca  
  
c  
ca  
  
a  
  
acaca  
aca  
{ a: 134, b: 123, c: 12 }  
8 10  
99 10  
8 10  
99810  
 [ [ 1, 1, 5, 6 ], [ 1, 2, 2, 8 ], [ 1, 1, 3, 3 ], [ 2, 2, 2, 8 ] ]  
4  
max sum = 4  
кол-во стрел = 2

# Вывод

Я реализовал решение задач

.