

Министерство цифрового развития, связи и массовых коммуникаций
Государственное образовательное учреждение высшего образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Задачи для самостоятельного решения

по дисциплине «Структура и алгоритмы обработки данных»

Выполнил студент группы БФИ-1901:

Бардюк Д.В.

Москва 2021

Задание

Задача 1. «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:

- $3 \leq \text{len}(A) \leq 10000$
- $1 \leq A[i] \leq 10^6$

Задача 2. «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]`

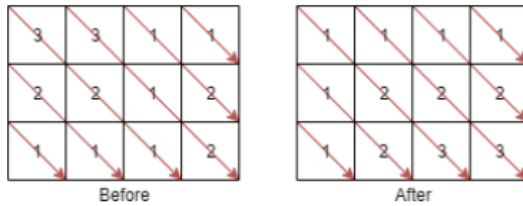
Вывод: `"10"`

Ограничения:

- $1 \leq \text{len}(\text{nums}) \leq 100$
- $0 \leq \text{nums}[i] \leq 10^9$

Задача 3. «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]]`

Выход: `[[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]]`

Выход: `[[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:

- $m == \text{len}(\text{mat})$
- $n == \text{len}(\text{mat}[i])$
- $1 \leq m, n \leq 100$
- $1 \leq \text{mat}[i][j] \leq 100$

Задача 1. «Шарики и стрелы»

Некоторые сферические шарики распределены по двумерному 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 x_{start} всегда меньше x_{end} .

Стрелу можно выстрелить строго вертикально (вдоль y -оси) из разных точек x -оси. Шарик с координатами x_{start} и x_{end} уничтожается стрелой, если она была выпущена из такой позиции x , что $x_{start} \leq x \leq x_{end}$. Когда стрела выпущена, она летит в пространстве бесконечное время (уничтожая все шарики на пути).

Дан массив `points`, где `points[i] = [x_start, x_end]`. 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:

- $0 \leq \text{len}(\text{points}) \leq 10^4$
- $\text{len}(\text{points}[i]) == 2$
- $-2^{31} \leq x_{start} < x_{end} \leq 2^{31} - 1$

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 $x[i] \geq y[i]$ (в алфавитном порядке) для всех i от 0 до $n-1$.

Примеры:

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

```
Output: true
```

Объяснение: «аух» – это перестановка строки $s2 = \text{«хуа»}$, которая “побеждает” строку $s1 = \text{«abc»}$.

```
input: s1 = "abe", s2 = "acd"
```

```
Output: false
```

Объяснение: Все перестановки для $s1 = \text{«abe»}$: “abe”, “aeb”, “bae”, “bea”, “eab” и “eba”, а все перестановки для $s2 = \text{«acd»}$: “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 палиндромную 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 1. «Стопки монет»

На столе стоят $3n$ 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:

- $3 \leq \text{len}(\text{piles}) \leq 10^5$
- $\text{len}(\text{piles}) \bmod 3 == 0$
- $1 \leq \text{piles}[i] \leq 10^4$

Код программы

```
//Контрольные задания 1
function generateArray(lenght){
    let arr = [];
    for(let i = 0; i < lenght; i++){
        arr[i] = 0 + Math.floor(Math.random() * (Math.pow(10,6) - 0 + 1));
    }
    return arr;
}

function thriaAngle(array){
```



```

let per = 0, maxA, maxB, maxC;
for (let i = 0; i < array.length - 2; i++){
    maxA = array[i];

    for (let j = 1; j < array.length - 1; j++){
        maxB = array[j];

        for (let k = 2; k < array.length ; k++){
            maxC = array[k];

            if(i !== j && j !== k && i !== k){
                if ((maxA + maxB > maxC) && (maxB + maxC > maxA) && (maxA + maxC > maxB)){
                    console.log(maxA , maxB, maxC);

                    if(per < maxA + maxB + maxC){
                        per = maxA + maxB + maxC;
                    }
                }
            }
        }
    }
}

return per;
}

let array = generateArray(4);
console.log(array);
console.log(thriaAngle(array));

```

```

let nums = [3,30,34,5,954,1,2972,3,574,5,6,724,8,9,10,11,12,20,30,33,90,900,1000,
354,99,1111,355,324]
// let nums = [0,0,0,1];
function maxNum(array){
    let check = array.every(element => element == 0);
    if(check){
        return 0;
    }
    return array
        .map((item) => item.toString())
        .sort((str1,str2) => (str2+str1) - (str1+str2))
        .join('');
}

console.log(maxNum(nums));

```

```

let mat = [[3, 3, 5, 6], [2, 2, 1, 8], [1, 1, 1, 2]]
mat = [[3, 3, 5, 6], [2, 2, 1, 8], [1, 1, 1, 2],[2, 2, 1, 8],[2, 2, 1, 8]]

```

```

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]]
mat = [[3, 3, 5, 6], [2, 2, 1, 8], [1, 1, 1, 2],[2, 2, 1, 8]]

let rowLen = mat[0].length, columnLen = mat.length
mat.forEach((e)=>{
  console.log(e);
})
console.log('sorting')

let buff = []
let triangleDia = Math.min(rowLen, columnLen) - 1
let countMainDia = Math.abs(rowLen-columnLen)+1
let lenMainDia = Math.min(rowLen, columnLen)

function triangleUp(){
  for(let i = 0; i < triangleDia; i++){
    //console.log(triangleDia);
    for(let j = 0; j < i+1; j++){
      //console.log(i,j, rowLen-1-i+j, mat[j][rowLen-1-i+j])
      buff.push(mat[j][rowLen-1-i+j])
    }
    //console.log('buff',buff );
    buff.sort((a,b)=>{return a-b})

    for(let j = 0; j < i+1; j++){
      //console.log(j,rowLen-1-i, i,buff[j])
      mat[j][rowLen-1-i+j] = buff[j]
    }
    buff = []
  }
}
triangleUp()

function diaMain(){
  //console.log(countMainDia);
  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 = []
    }
}
}
diaMain()
function triangleDown(){
    for(let i = 0; i < triangleDia; i++){
        //console.log(triangleDia);
        for(let j = 0; j < i+1; j++ ){
            buff.push(mat[columnLen-1-i+j][j])
        }
        //console.log('buff',buff );
        buff.sort((a,b)=>{return a-b})

        for(let j = 0; j < i+1; j++ ){
            //console.log(j,rowLen-1-i, i,buff[j])
            mat[columnLen-1-i+j][j] = buff[j]
        }
        buff = []
    }
}
triangleDown()

mat.forEach((e)=>{
    console.log(e);
})

///Задание Шарики и Стрелы
function setup(){
    createCanvas(600, 400)
    frameRate(30)
}

let stages = 0

let scale = 5
let points = [[1,2],[2,3],[3,4],[4,5],[20,30],[30,40],[40,50]]

let spectral=[]
function spectra(){
    for(let i=0; i<100; i++) { spectral.push(0) }
    points.forEach(e=>{
        let buffMax = 0
        for(let i = e[0]; i <= e[1]; i++){

```

```

        spectral[i]++
    }
})
console.log('Spectral: ', spectral)
maxi()
}

function maxi(){
    console.log('-');
    let spectMax = 0
    spectral.forEach(e=>{
        (e > spectMax) ? spectMax = e : e = e
    })
    console.log(spectMax)
    shoot(spectMax)
}

function shoot(max){
    let m = spectral.indexOf(max)
    console.log('m is ',m);
    console.log(points);
    for(let i = 0; i < points.length; i++){
        console.log(i,'try')
        if(points[i][0] <= m && points[i][1] >= m){
            console.log(points[i][0], points[i][1],'spliced')
            points.splice(i,1)
            i--
        }else{
            console.log(points[i][0], points[i][1],'NonSpliced')
        }
    }
    spectral = []
    if(points.length === 0){
        console.log('Success for ', stages+1, ' shots')
    }else{
        console.log(points);
        loop()
        stages++
    }
}

function drawPoints(){
    let k = 10
    points.forEach(e=>{
        let d = e[1]-e[0]
        let r = d/2
        circle((e[0]-r)*scale+100, k*scale, d*scale)
        k +=10
    })
}

```

```

let x = 0

function compare(){
  console.log('+')
  line(x,0,x,600)
  x += scale
  if(x>=300){
    x = 0
    noLoop()
    spectra()
  }
}

function getArrows(){
  let l = points.length
  let last = -1
  peak.forEach(e=>{
    // if(){}
  })
}

function draw(){
  background(220)
  fill(100,200,100)
  noStroke()
  drawPoints()

  stroke(255,0,0)
  compare();
}

```

```

//Задания по строкам
function findWin(str1,str2){
  if(str1.length === 0 || str2.length === 0){
    return `it is empty`;
  }else if (str1.length !== str2.length){
    return `Can't compare it`;
  }
  let arrChar1 = str1.split('').sort();
  let arrChar2 = str2.split('').sort();
  console.log(arrChar1)
  console.log(arrChar2)
  if((arrChar1.every((el, idx) => el <= arrChar2[idx])) || (arrChar2.every((el, idx) => el <= arrChar1[idx]))) {
    return true;
  }
  return false;
}

```

```

}

let str1 = `abe`;
let str2 = 'acd';
console.log(findWin(str1,str2));

function isPolindrome(string){
    return string === string.split('').reverse().join('');
}

function largestPolindrome(string){
    if(string.length ===0){
        return `it is empty`;
    }
    let largest = '';
    for(let i = 0; i<string.length; i++){
        for(let j = 0; j<string.length; j++){
            let substr = string.slice(i,j);
            if(isPolindrome(substr)){
                if(largest.length<substr.length){
                    largest = substr;
                }
            }
        }
    }
    if(largest.length === 0){
        return `Haven't polindrome`;
    }
    return largest;
}

let string = 'babaaabaaa';
console.log(largestPolindrome(string));

function countConcat(string){
    let count = 0;
    let map = new Map();
    for (let i = 0; i < string.length; i++) {
        for (let j = 1; j < string.length; j++) {
            const subString = string.slice(i, j);
            if(subString.length !== 0){
                if(string.indexOf(subString.concat(subString)) !== -1 ){
                    if(map.get(subString) !== 1){
                        map.set(subString,1);
                    }
                }
            }
        }
    }
}

```

```

    }
  }
}
console.log(map);
return count;
}

console.log(countConcat('abcbabc'));

//Задача с монетами
const {DeQueue} = require(`../4lab/queue`)

let array = [9,8,7,6,5,1,2,3,4];
let myCoins = 0;
function coins(array){
  let deq = new DeQueue();
  array.sort().map(item => deq.pushFront(item));
  if (array.length % 3 !== 0){
    return `can't do it`;
  }
  else{
    while(!deq.isEmpty()){
      deq.popFront();
      myCoins+=deq.popFront();
      deq.popBack();
    }
  }
  return myCoins;
}
console.log(`Наша наибольшая сумма: ${coins(array)}`);

```

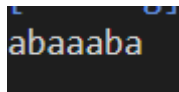
Результат Работы

На рисунке 1 представлен результат работы программы составления наибольшего числа

999954909008724657455355354343333324303029722012111111101000

Рисунок - 1 Результат работы программы составления наибольшего числа

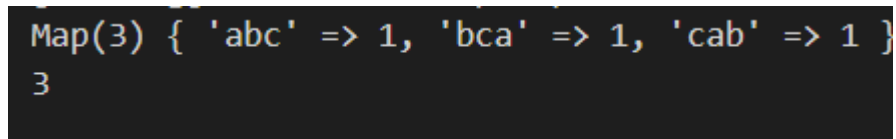
На рисунке 2 представлен результат работы программы поиска палиндрома



```
abaaaba
```

Рисунок - 2 Результат работы программы поиска палиндрома

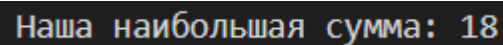
На рисунке 3 представлен результат работы программы поиска количества конкатенаций в строке



```
Map(3) { 'abc' => 1, 'bca' => 1, 'cab' => 1 }  
3
```

Рисунок - 3 Результат работы программы поиска количеств конкатенаций в строке

На рисунке 4 представлен результат работы программы по поиску наибольшей суммы монет в игре из задачи «Стопки монет»



```
Наша наибольшая сумма: 18
```

Рисунок - 4 Результат работы программы проверки голосований

Вывод: в ходе выполнения данных заданий я применил на практике знания полученные в течении курса, разработал программы для решения заданий. Внедрение полученного за курс материала помогло оптимизировать мои алгоритмы.