Федеральное агентство связи

ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра МКиТ

Курсовая работа

по дисциплине «СиАОД»

Выполнил: студент

Группы БСТ1902

Игнатов В.С.

Вариант №5

Москва 2021

Оглавление

Задача «Треугольник с максимальным периметром»

Описание

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.

Код программы

```
let arr1 = [2, 1, 2];
let arr2 = [1, 2, 1];
let arr3 = [3, 2, 3, 4];
let arr4 = [3, 6, 2, 3];

maxPerimeter(arr1, arr2, arr3, arr4);

function maxPerimeter(...arr) {
  arr.forEach((arrItem) => {
    arrItem.sort((a, b) => b - a);

    let max = 0;

    for (let i = 0; i < arrItem.length - 2; i++) {
      if (arrItem[i] < arrItem[i + 1] + arrItem[i + 2]) {
        max = Math.max(max, arrItem[i] + arrItem[i + 1] + arrItem[i + 2]);
        break;
      }
    }

    if (max) {
      console.log(`Максимальный периметр: ${max}`);
    } else {
      console.log(`Треугольника нет`);
    }
  });
}
```

Результат

```
Максимальный периметр: 5
Треугольника нет
Максимальный периметр: 10
```

Рисунок 1 – Результат работы программы

Задача «Максимальное число»

Описание

Дан массив неотрицательных целых чисел nums. Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число.

Код программы

```
let numbers = [3, 30, 34, 5, 9];
let number2 = [10,  2];
let number3 = [10];

const concatMax = (arr) => {
  let result = arr.map(String).sort((a, b) => (b + a) - (a + b)).join("");
  return result;
}

console.log(concatMax(numbers), concatMax(number2), concatMax(number3));
```

Результат

```
9534330 210 10
```

Рисунок 2 – Результат работы программы

Задача «Сортировка диагоналей в матрице»

Описание

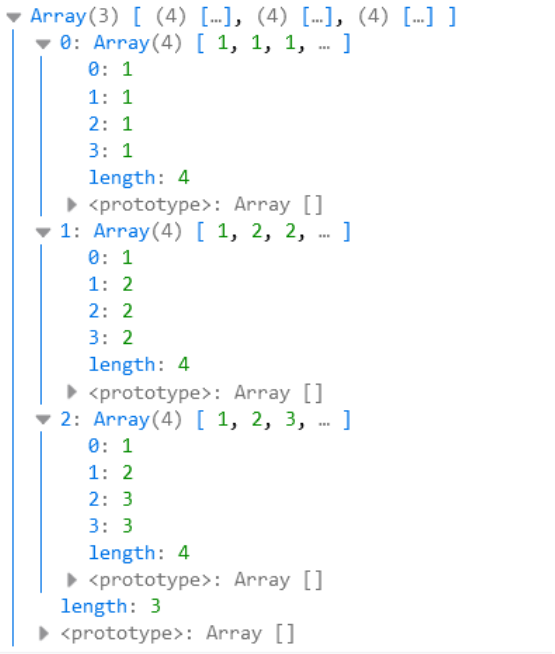
Дана матрица mat размером m * n, значения - целочисленные. Напишите функцию, сортирующую каждую диагональ матрицы по возрастанию и возвращающую получившуюся матрицу.

Код программы

```
let mat1 = [
  [3, 3, 1, 1],
  [2, 2, 1, 2],
  [1, 1, 1, 2],
];
let mat2 = [
  [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],
];

sortDiagonals(mat1);
sortDiagonals(mat2);
```

```
function sortDiagonals(arr) {
  const m = arr.length;
  const n = arr[0].length;

  let c = 1;

  while (c !== m) {
    sorting(m, n);
    c++;
  }

  console.log(arr);

  function sorting(m, n) {
    for (let i = 0; i < m; i++) {
      let a = (b = i);
      for (let j = 0; j < n; j++) {
        if (i + 1 < m && j + 1 < n && arr[i + 1][j + 1] < arr[i][j]) {
          let swap = arr[i + 1][j + 1];
          arr[i + 1][j + 1] = arr[i][j];
          arr[i][j] = swap;
        }
      }
    }
  }
}
```

Результат



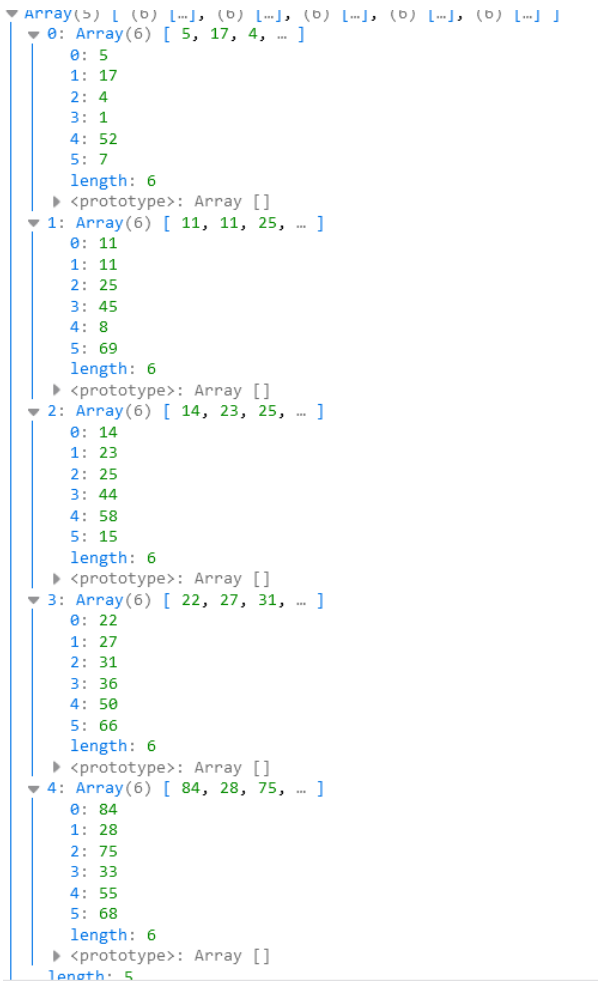


Рисунок 3 – Результат работы программы

Задача «Объединение отрезков»

Описание

Дан массив отрезков intervals, в котором intervals[i] = [start i, end i], некоторые отрезки могут пересекаться. Напишите функцию, которая объединяет все пересекающиеся отрезки в один и возвращает новый массив непересекающихся отрезков.

Код программы

```
const sumIntervals = (intervals) => {
    /* Сортировка */
    intervals.sort((a, b) => {
        return a[0] - b[0]
    })

    /* Поиск пересекающихся интервалов */
    intervals.forEach((firstInterval, firstIndex) => {
        intervals.forEach((secondInterval, secondIndex) => {
            if (firstIndex !== secondIndex) {
```

```
                if (secondInterval[0] <= firstInterval[1] &&
secondInterval[0] >= firstInterval[0]) {
                    if (secondInterval[1] > firstInterval[1])
                        firstInterval[1] = secondInterval[1];

                    secondInterval[0] = secondInterval[1] = -1
                }
            }

        })
    });

    /* Удаление отрицательных */
    intervals = intervals.filter(elem => elem[0] !== -1 && elem[1] !== -1);

    /* Вывод в консоль массива */
    console.log(intervals)
}

sumIntervals([[8, 10], [1, 3], [15, 18], [2, 6]])
sumIntervals([[4, 5], [1, 4]])
```

Результат

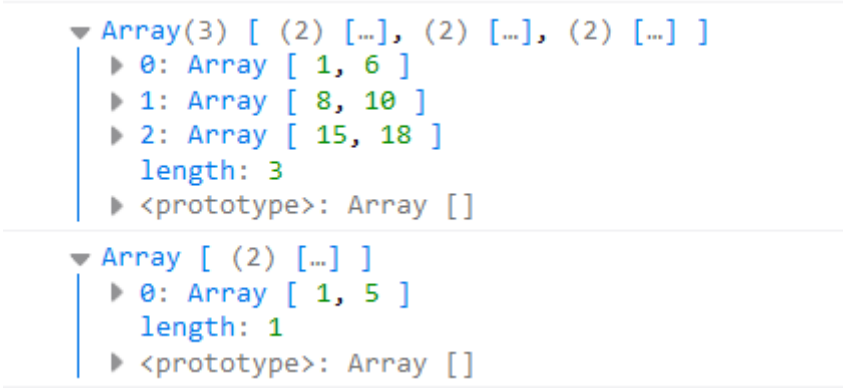


Рисунок 4 – Результат работы программы

Задача «Стопки монет»

Описание

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.

Код программы

```
const coins = (piles) => {
    piles.sort((a, b) => a - b);

    const n = piles.length;
    let yourCoins = 0;

    for (let i = n / 3; i < n; i += 2) {
        console.log(`Массив: ${piles}, Элемент: ${piles[i]}`);
        yourCoins += piles[i];
    }

    console.log(yourCoins);
}

coins([2, 4, 1, 2, 7, 8])
coins([2, 4, 5])
coins([9, 8, 7, 6, 5, 1, 2, 3, 4])
```

Результат

```
9
4
18
```

Рисунок 5 – Результат работы программы

Задача «Шарики и стрелы»

Описание

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 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 ⩽ x ⩽ 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.

Код программы

```
const balloons = (points) => {
    points.sort((a, b) => a[0] - b[0]);
    let prev = null,
        count = 0;

    for(let [start, end] of points) {
        if (prev == null || prev < start) {
            count++;
            prev = end;
        } else prev = Math.min(prev, end);
    }

    console.log(count);
}

balloons([[10,16],[2,8],[1,6],[7,12]])
balloons([[1,2],[3,4],[5,6],[7,8]])
balloons([[1,2],[2,3],[3,4],[4,5]])
balloons([[1,2]])
balloons([[2,3],[2,3]])
```

Результат

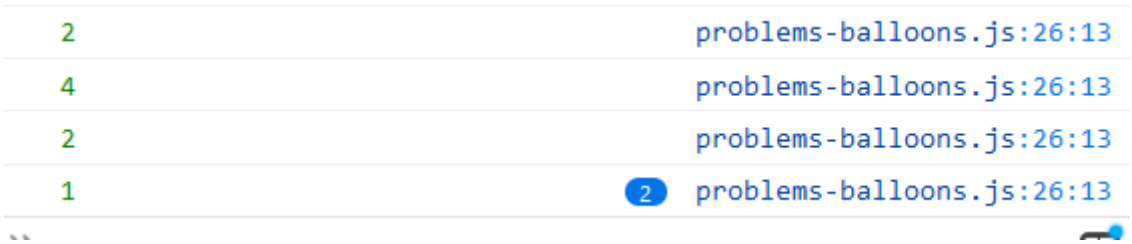


Рисунок 6 – Результат работы программы

Задача №1 со строками

Описание

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.

Код программы

```
const defeatStrings = (s1, s2) => {
    s1 = s1.split("").sort();
    s2 = s2.split("").sort();
    let bool1 = true, bool2 = true;

    for (let i = 0; i < s1.length; i++) {
        if (s1[i] > s2[i]) bool1 = false;
        if (s1[i] < s2[i]) bool2 = false;
    }

    return bool1 || bool2;
}

console.log(defeatStrings('abc', 'xya'))
console.log(defeatStrings('abe', 'acd'))
```

Результат

```
true                                problems-strings.js:23:9
false                               problems-strings.js:24:9
```

Рисунок 7 – Результат работы программы

Задача №2 со строками

Описание

Дана строка s, вернуть самую длинную полиндромную подстроку в s

Код программы

```
const isPalindrome = (str, i, j) => {
    let start = i, end = j;
    while (start < end) {
        if (str[start] !== str[end]) {
            return false;
        }
```

```
        start++;
        end--;
    }
    return true;
}

const longestPalindrome = (s) => {
    for (let i = s.length - 1; i >= 0; i--) {
        let start = 0;
        let end = i;
        while (end < s.length) {
            if (isPalindrome(s, start, end)) {
                return s.substring(start, end + 1);
            }
            start++;
            end++;
        }
    }
    return "";
}

console.log(longestPalindrome('babad'));
console.log(longestPalindrome('cbbd'));
```

Результат

```
bab                                  problems-strings.js:58:9
bb                                   problems-strings.js:59:9
```

Рисунок 8 – Результат работы программы

Задача №3 со строками

Описание

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).

Код программы

```
const countNotEmpty = (text) => {
    const strings = new Set();
    for (let i = 0; i < text.length; i++) {
        for (let j = i + 1; j < text.length; j++) {
            const left = text.slice(i,j);
            const right = text.slice(j, j + j - i);
            // console.log(left, right);
            if (left === right) strings.add(left);
        }
    }
    return strings.size;
}
```

```
console.log(countNotEmpty("abcabcabc"));
```

Результат

Рисунок 9 – Результат работы программы

Вывод

В ходе выполнения курсовой работы, был решен ряд задач, в которых были использованы необходимые знания по алгоритмам и структурам данных.