Федеральное агентство связи
Ордена трудового красного знамени
Федеральное государственное бюджетное
Образовательное учреждение высшего образования
Московский Технический Университет связи и информатики

Кафедра МКиИТ

Курсовая работа по дисциплине

«Структуры и алгоритмы обработки данных»

Выполнил студент

Группы БСТ1902

Бадмаев А.Д.

Москва 2021

Оглавление

Шарики и стрелы 3
Стопки монет 4
Объединение отрезков 5
Треугольник с максимальным периметром 6
Максимальное число 7
Сортировка диагоналей в матрице 8
Задание №1 со строками 9
Задание №2 со строками 10
Задание №3 со строками 11
Вывод 12

Шарики и стрелы

Некоторые сферические шарики распределены по двухмерному пространству. Для каждого шарика даны xкоординаты начала и конца его горизонтального диаметра. Так как пространство двумерно, то укоординаты не имеют значения в данной задаче. Координата xstart всегда меньше xend. Стрелу можно выстрелить строго вертикально (вдоль уоси) из разных точек хоси. 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.

Код программы

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

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

    console.log(res);
}

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]])
```

Результаты работы:

```
PS D:\Доки\Алгоритмы\lab1> node .\problem_baloons.js
2
4
2
1
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.

Код программы:

```
var maxCoins = function(piles) {
    const start = piles.length / 3;
    piles.sort((a, b) => a - b);
    let res = 0;
    for(let i = start; i < piles.length; i+=2) res += piles[i];
    console.log(res);
};


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

Результаты работы:

```
PS D:\Доки\Алгоритмы\lab1> node .\problem_coins.js
9
4
18
PS D:\Доки\Алгоритмы\lab1>
```

Объединение отрезков

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

Код программы:

```
var merge = function(intervals) {
    if (intervals.length <= 1) return intervals;
    intervals.sort((a, b) => a[0] - b[0])
    let res = [];
    let currentInterval = intervals[0];
    res.push(currentInterval);

    for (let interval of intervals) {
        let currentIntervalEnd = currentInterval[1];
        let nextIntervalBeg = interval[0];
        let nextIntervalEnd = interval[1];
        if (currentIntervalEnd >= nextIntervalBeg) {
            currentInterval[1] = Math.max(currentIntervalEnd, nextIntervalEnd);
        } else {
            currentInterval = interval;
            res.push(currentInterval)
        }
    }
   console.log(res);
};

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

Результаты работы:

```
PS D:\Доки\Алгоритмы\lab1> node .\problem_intervals.js
[ [ 1, 6 ], [ 8, 10 ], [ 15, 18 ] ]
[ [ 1, 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.

Код программы:

```
function maxP(arr) {
    arr.sort((a, b) => b - a);
    console.log(arr)
    for (let i = 0; i < arr.length - 2; i++) {
        let maxPerim = 0;
        let p = (arr[i] + arr[i + 1] + arr[i + 2]) / 2;
        let S = Math.sqrt(p * (p - arr[i]) * (p - arr[i + 1]) * (p - arr[i + 2]))
;
        if (S > 0)
            maxPerim = arr[i] + arr[i + 1] + arr[i + 2];
        if (maxPerim) {
            console.log(`Максимальный периметр: ${maxPerim}`);
            return;
        }
    }
    console.log(`Треугольника нет`);
}

const arr = genArray(5, 1, 10);
console.log(arr)
maxP(arr)
```

Результаты работы:

```
[ 10, 8, 1, 7, 9 ]
[ 10, 9, 8, 7, 1 ]
Максимальный периметр: 27
```

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.

Код программы:

```
function getLargest(arr) {
    return arr
    .map(String)
    .sort((a, b) => (b + a) - (a + b))
    .join('');
}
console.log(arr);
console.log(getLargest(arr));
```

Результаты работы:

```
[ 10, 9, 8, 7, 1 ]
987110
```

Сортировка диагоналей в матрице

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

Код программы:

```
function diagonalSort(arr) {
    const columnLength = arr.length;
    const rowLength = arr[0].length;

    let countDiag = 0;

    while (++countDiag !== columnLength) {
        for (let i = 0; i < columnLength; i++) {
            for (let j = 0; j < rowLength; j++) {
                if (i + 1 < columnLength && j + 1 < rowLength && 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;
                }
            }
        }
    }
    console.log(M)
}
console.log(M)
diagonalSort(M)
```

Результаты работы:

```
[
  [ 6, 8, 6, 10, 8 ],
  [ 3, 2, 3, 4, 6 ],
  [ 1, 10, 7, 6, 7 ],
  [ 6, 2, 8, 2, 3 ],
  [ 4, 3, 9, 1, 2 ]
]
[
  [ 2, 3, 4, 6, 8 ],
  [ 1, 2, 3, 6, 10 ],
  [ 1, 3, 2, 6, 7 ],
  [ 3, 2, 8, 6, 8 ],
  [ 4, 6, 9, 10, 7 ]
]
```

Задание №1 со строками

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'))
```

Результаты работы:

```
true
false
```

Задание №2 со строками

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

Код программы:

```
var longestPalindrome = function (s) {
    for (let j = s.length - 1; j >= 0; j--) {
        let i = 0,
            k = j;
        while (k < s.length) {
            let substr = s.substring(i, k + 1);
            if (isPalindrome(substr)) return substr;
            i++, k++;
        }
    }
    return "";
};

function isPalindrome(str) {
    let l = 0,
        r = str.length - 1;
    while (l < r) {
        if (str[l] !== str[r]) return false;
        l++, r--;
    }
    return true;
}
```

Результаты работы:

```
bab
bb
```

Задание №3 со строками

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:

```
var concatSubstr = function(text) {
    const string = new Set();
    for (let i = 0; i < text.length; i++) {
        for (let j = i + 1; j < text.length; j++) {
            const left = text.substring(i,j);
            const right = text.substring(j, j + j - i);
            if (left === right) string.add(left);
        }
    }
    return string.size;
};

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

Результаты работы: 3

Вывод

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