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

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Лабораторная работа № 3

по дисциплине «Структура и алгоритмы обработки данных»

Выполнил студент группы БФИ-1901:

Бардюк Д. В.

Москва 2021

Задание

Задание 1

Реализовать методы поиска подстроки в строке. Добавить возможность ввода строки и подстроки с клавиатуры. Предусмотреть возможность существования пробела. Реализовать возможность выбора опции чувствительности или нечувствительности к регистру. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

Алгоритмы:

1. Кнута-Морриса-Пратта
2. Упрощенный Бойера-Мура

Задание 2 «Пятнашки»

Игра в 15, пятнашки, такен — популярная головоломка, придуманная в 1878 году Ноем Чепмэном. Она представляет собой набор одинаковых квадратных костяшек с нанесёнными числами, заключённых в квадратную коробку. Длина стороны коробки в четыре раза больше длины стороны костяшек для набора из 15 элементов, соответственно в коробке остаётся незаполненным одно квадратное поле. Цель игры — перемещая костяшки по коробке, добиться упорядочивания их по номерам, желательно сделав как можно меньше перемещений.

Код программы

Задание 1

```
function boyerFind(str, substr){
    let library = {},
        subLength = substr.length-1,
        strLength = str.length,
        resultArr = [],
        j, defaultLetter;

    for(let i = 0; i < subLength+1; i++){
        library[substr.charAt(i)] = subLength - i;

    }
    console.log(library);

    let i = 0;
       while (i<strLength){
        for(j = subLength; j >= 0; j--){
            if(str.charAt(j+i) != substr.charAt(j)){
```

```
                break;
            }
        }
            if(j<0){
                resultArr.push(++i);
            }
            else {
                defaultLetter = library[str.charAt(j+i)];
                if(!defaultLetter){
                    defaultLetter = subLength + 1;
                }
                defaultLetter+= j - subLength;
                if(defaultLetter < 0){
                    defaultLetter = 1;
                }
                i+= defaultLetter;
            }

        }
    if(resultArr.length == 0){
        return `Sorry ${substr} is not found`;
    }
    return resultArr;
}


console.log(boyerFind('Mnuci tuci', 'tuci'));
```

```
function prefixFind(str, substr){
    const strFound = substr.concat('#', str);
    let pi = [];
    pi[0] = 0;

    for(let i = 1; i<strFound.length; i++){
        let j = pi[i - 1];

        while ((j > 0) && (strFound[i] != strFound[j])){
            j = pi[j-1];
        }
        //console.log(strFound[i], strFound[j],i,j,pi);
            if (strFound[i] == strFound[j]){
                pi[i] = ++j;

            } else{
                pi[i] = j;
            }

        }

    return pi;
```

```
}

function find(str, substr){
   const pi = prefixFind(str, substr);
   let count = 0, length = substr.length;
   for(let i = 0; i < pi.length; i++ ){
        if (pi[i] == length){
            count++;
        }
   }console.log(pi);
   return `${substr} встречается в ${str} ${count} раз(а)`
}

console.log(find('aabaabaaaabaabaaab', 'aabaa'));
```

Задание 2

```
function graphSearch (array) {
    let queue = [],chekPosition=[];
    const answer = [[1, 2, 3, 4],
        [5, 6, 7, 8],
        [9, 10, 11, 12],
        [13, 14, 15, 0]];

    queue.push(
        {
            array: array,
            path: [],
            opt: 0
        }
    );
    while (queue.length > 0) {
        const current = queue.shift();

        chekPosition.push(current.array);
        if (JSON.stringify(current.array) === JSON.stringify(answer)) {

            return current.path;
        }

        let indexOfZeros;
        for (let i = 0; i < 4; i++) {
            for (let j = 0; j < 4; j++){
                if (current.array[i][j] === 0) {
                    indexOfZeros = [i, j];
                    break;
                }
            }
        }
```

```
        if (indexOfZeros[0] < 3 && current.opt !== 2) {
            console.log(current.opt)
            let newArray = JSON.parse(JSON.stringify(current.array))
            newArray[indexOfZeros[0]][indexOfZeros[1]] = newArray[indexOfZeros[0]
 + 1][indexOfZeros[1]]
            newArray[indexOfZeros[0] + 1][indexOfZeros[1]] = 0
            const action = newArray[indexOfZeros[0]][indexOfZeros[1]];
            let newPath = JSON.parse(JSON.stringify(current.path))
            newPath.push(action)
            if (finder(chekPosition, newArray)) {
                queue.push(
                    {
                        array: newArray,
                        path: newPath,
                        opt: optimal(newArray)
                    }
                )
            }

        }
        if (indexOfZeros[0] > 0 && current.opt !== 1) {
            console.log(current.opt)
            let newArray = JSON.parse(JSON.stringify(current.array))
            newArray[indexOfZeros[0]][indexOfZeros[1]] = newArray[indexOfZeros[0]
 - 1][indexOfZeros[1]]
            newArray[indexOfZeros[0] - 1][indexOfZeros[1]] = 0
            const action = newArray[indexOfZeros[0]][indexOfZeros[1]];
            let newPath = JSON.parse(JSON.stringify(current.path))
            newPath.push(action)
            if (finder(chekPosition, newArray)) {
                queue.push(
                    {
                        array: newArray,
                        path: newPath,
                        opt: optimal(newArray),
                    }
                )
            }
            //console.log("2  ", current.path)
        }
        if (indexOfZeros[1] < 3 && current.opt !== 4) {
            console.log(current.opt)
            let newArray = JSON.parse(JSON.stringify(current.array))
            newArray[indexOfZeros[0]][indexOfZeros[1]] = newArray[indexOfZeros[0]
][indexOfZeros[1] + 1]
            newArray[indexOfZeros[0]][indexOfZeros[1] + 1] = 0
            const action = newArray[indexOfZeros[0]][indexOfZeros[1]];
            let newPath = JSON.parse(JSON.stringify(current.path))
            newPath.push(action)

            if (finder(chekPosition, newArray)) {
```

```
                queue.push(
                    {
                        array: newArray,
                        path: newPath,
                        opt: optimal(newArray),
                    }
                )
            }
            //console.log("3  ", current.path)
        }
        if (indexOfZeros[1] > 0 && current.opt !== 3) {
            console.log(current.opt)
            let newArray = JSON.parse(JSON.stringify(current.array))
            newArray[indexOfZeros[0]][indexOfZeros[1]] = newArray[indexOfZeros[0]
][indexOfZeros[1] - 1]
            newArray[indexOfZeros[0]][indexOfZeros[1] - 1] = 0
            let action = newArray[indexOfZeros[0]][indexOfZeros[1]];
            let newPath = JSON.parse(JSON.stringify(current.path))
            newPath.push(action)
            if (finder(chekPosition, newArray)) {
                queue.push(
                    {
                        array: newArray,
                        path: newPath,
                        opt: optimal(newArray),
                    }
                )
            }
        }
        queue.sort((a, b) => {
            return a.opt - b.opt
        })
    }


}
const finder = (array, sought) => {
    let k=0
    array.map(item => {
        if (JSON.stringify(item) === JSON.stringify(sought)) {
            k++
            return false;
        }
    })
    return k === 0;
}

const optimal = (array) => {
    let counter = 0

    for (let i = 0; i < 4; i++) {
```

```
        for (let j = 0; j < 4; j++) {
            for(let o = 0;o < 4;o++){
                if (array[o].indexOf(4 * i + j + 1) !== -1) {
                    counter += Math.abs(i - o)
                        + Math.abs(j - array[o].indexOf(4 * i + j + 1))
                }
            }
        }
    }

    for (let i = 0; i < 4; i++) {
        for (let j = 0; j < 3; j++) {
            if (array[i][j] > array[i][j + 1] && array[i][j]!==0 && array[i][j+1]
!==0) {
                counter += 2
            }
        }
    }


    if(array[3][3]!==12||array[3][3]!==15)
        counter+=2
    return counter
}

let inv = 0;
// let arr = [5,9,8,14,0,6,12,3,13,11,1,10,15,2,7,4]
let arr = [1,2,3,4,5,6,7,8,13,9,11,12,10,14,15,0]
for (let i = 0; i < 16; i++) {
    if (arr[i])
        for (let j = 0; j < i; ++j)
            if (arr[j] > arr[i])
                inv++;
}
for (let i = 0; i < 16; ++i) {
    if (arr[i] === 0)
        inv += 1 + i / 4;
}

let arr1 = Array();
let k = 0;
for (let i = 0; i < 4; i++) {
    arr1[i] = Array();
    for (let j = 0; j < 4; j++) {
        arr1[i][j] = arr[k];
        k++;
    }
}


if (inv & 1) {
```

```
    console.log("Решения нет")
} else {
    console.log("Решение есть")
    console.log(graphSearch(arr1).join(","));
}
```

Результат работы

На рисунке 1 представлен результат работы программ задания 1

```
{ t: 3, u: 2, c: 1, i: 0 }
[ 7 ]
```

```
aabaa встречается в aabaabaaaabaabaaab 4 раз(а)
```

Рисунок 1 – Результат работы программ поиска подстроки в строке

На рисунке 2 представлен результат работы программ задания 2

```
15,14,10,13,9,11,14,15,12,14,11,10,15,11,14,12,11,14,10,15,14,10,15,14,10,11,12,15,11,10,14,11,15,12,10,15,11,14,15,10,12,11,10,15,14,10,11,12
```

Рисунок 2 – Последовательность шагов для сбора пятнашек

Вывод: в ходе выполнения данной работы я познакомился с работой популярных алгоритмов поиска подстроки в строке, мною были реализованы данные алгоритмы. Также мною была реализована программа для поиска оптимального пути на примере игры в пятнашки.