

Федеральное агентство связи
Ордена трудового красного знамени
Федеральное государственное образовательное
Бюджетное
Учреждение высшего профессионального образования
Московский Технический Университет связи и информатики

Кафедра информатики

Лабораторная работа №3

по дисциплине «СиАОД»
«Методы поиска подстроки в строке»

Выполнил: студ. гр. БСТ1902

Потрываев А.Г

Вариант №15

Проверил: Кутейников И.А.

Москва 2021

Содержание

2	Ход выполнения работы	3
	Задание 1	4
	Задание 2	6
3	Результат работы программы.....	13

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

2 Ход выполнения работы

Задание 1

```
fun main(args: Array<String>){
    val `in` = Scanner(System.`in`)
    val flag = false
    println("Введите исходную строку:")
    var source: String = `in`.nextLine();
    println("Введите строку для поиска:")
    var template: String = `in`.nextLine();
    if(flag){
        source = source.toLowerCase();
        template = template.toLowerCase();
    }
    var m = System.currentTimeMillis()
    var index = KmpSearch(source, template)
    m = System.currentTimeMillis() - m
    println("Алгоритм Кнута-Морриса-Пратта выполнен за $m
милисекунд. Индекс слова = $index")
    m = System.currentTimeMillis()
    index = BMSearch(source, template)
    m = System.currentTimeMillis() - m
    println("Алгоритм Бойера-Мура выполнен за $m миллисекунд.
Индекс слова = $index")
}

fun KmpSearch(source: String, x: String): Int?{
    val d: ArrayList<Int> = arrayListOf(0)
    val template = "$x#$source"
    for (i in 1..template.length) {
        var j = d[i - 1]
        while (j > 0 && template[j] != template[i])
```

```

        j = d[j - 1]
        if (template[j] == template[i])
            j += 1
        d.add(i, j)
        if (j == x.length)
            return i-3
    }
    return null
}

```

```

fun BMSearch(source: String, template: String): Int? {
    val sourceLen = source.length
    val templateLen = template.length
    if (templateLen > sourceLen) {
        return null
    }
    val offsetTable = HashMap<Char, Int>()
    for (i in 0..255) {
        offsetTable[i.toChar()] = templateLen
    }
    for (i in 0 until templateLen - 1) {
        offsetTable[template[i]] = templateLen - i - 1
    }
    var i = templateLen - 1
    var j = i
    var k = i
    while (j >= 0 && i <= sourceLen - 1) {
        j = templateLen - 1
        k = i
        while (j >= 0 && source[k] == template[j]) {

```

```

        k--
        j--
    }
    i += offsetTable[source[i]]!!
}
return if (k >= sourceLen - templateLen) {
    null
} else {
    k + 2
}
}

```

Задание 2

```

fun main(args: Array<String>) {
    val `in` = Scanner(System.`in`)
    val sb = `in`.nextLine() + " " +
        `in`.nextLine() + " " +
        `in`.nextLine() + " " +
        `in`.nextLine()
    val field = Arrays.stream(sb.split(
".toRegex()).toTypedArray()).mapToInt { s: String -> s.toInt() }.toArray()
    if (!checkState(field)) {
        val astar = Astar()
        val res = astar.search(State(field))
        if (res == null) {
            println("Решений не нашлось")
        }
    }
}

```

```

    } else {
        for (s in res) println(s.toString())
    }
} else println("[]")
}

```

```

fun checkState(field: IntArray): Boolean {
    var inv = 0
    for (i in 0..15) if (field[i] != 0) for (j in 0 until i) if (field[j] > field[i]) ++inv
    for (i in 0..15) if (field[i] == 0) inv += 1 + i / 4
    return inv % 2 == 1
}

```

```

class Astar {
    var finalfield = intArrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0)
    private var closedStates = 0
    fun search(startState: State): Collection<State?>? {
        val close = LinkedList<State?>()
        val open = LinkedList<State?>()
        open.add(startState)
        startState.g = 0
        startState.h = getH(startState)
        while (!open.isEmpty()) {
            val x = getStateWithMinF(open)
            if (isTerminate(x)) {
                closedStates = close.size
                return completeSolution(x)
            }
        }
    }
}

```

```

open.remove(x)
close.add(x)
val neighbors = getNeighbors(x)
for (neighbor in neighbors) {
    if (close.contains(neighbor)) {
        continue
    }
    val g = x!!.g + getDistance(x, neighbor)
    var isGBetter: Boolean
    if (!open.contains(neighbor)) {
        neighbor!!.h = getH(neighbor)
        open.add(neighbor)
        isGBetter = true
    } else {
        isGBetter = g < neighbor!!.g
    }
    if (isGBetter) {
        neighbor.parent = x
        neighbor.g = g
    }
}
return null
}

```

```

private fun getDistance(a: State?, b: State?): Int {
    var c = b
    var res: Int = 0
    while (c != null && c != a) {
        c = c.parent
    }
}

```



```

        ++res
    }
    return res
}

```

```

fun getNeighbors(currentState: State?): List<State?> {
    val res: ArrayList<State?> = ArrayList<State?>()
    val field = currentState!!.field
    var zero = 0
    while (zero < 16) {
        if (field[zero] == 0) {
            break
        }
        zero++
    }
    if (zero - 4 >= 0) res.add(State(swap(field, zero, zero - 4), zero - 4))
    if (zero + 4 < 16) res.add(State(swap(field, zero, zero + 4), zero + 4))
    if ((zero + 1) / 4 == zero / 4 && zero + 1 < 16) res.add(State(swap(field,
zero, zero + 1), zero + 1))
    if ((zero - 1) / 4 == zero / 4 && zero - 1 >= 0) res.add(State(swap(field,
zero, zero - 1), zero - 1))
    return res
}

```

```

private fun swap(arr: IntArray, a: Int, b: Int): IntArray {
    val newField = arr.copyOf(16)
    val temp = newField[a]
    newField[a] = newField[b]
    newField[b] = temp
    return newField
}

```

```
}
```

```
private fun completeSolution(terminate: State?): Collection<State?> {  
    val path: LinkedList<State?> = LinkedList<State?>()  
    var c = terminate  
    while (c != null) {  
        path.addFirst(c)  
        c = c.parent  
    }  
    return path  
}
```

```
private fun isTerminate(x: State?): Boolean {  
    return Arrays.equals(x!!.field, finalfield)  
}
```

```
private fun getStateWithMinF(open: Collection<State?>): State? {  
    var res: State? = null  
    var min = Int.MAX_VALUE  
    for (state in open) {  
        if (state!!.f < min) {  
            min = state.f  
            res = state  
        }  
    }  
    return res  
}
```

```
private fun getH(startState: State?): Int {  
    var diff = 0
```

```

    val field = startState!!.field
    for (i in 0..15) {
        if (finalfield[i] != field[i]) diff++
    }
    return diff
}
}

```

```

class State {
    var g = 0
    var h = 0
    var parent: State? = null
    lateinit var field: IntArray
    private val size = 16

    var changed = 0

    val f: Int
        get() = g + h

    constructor(parent: State?) {
        this.parent = parent
    }

    constructor(field: IntArray, changed: Int) {
        this.field = field
        this.changed = changed
    }
}

```

```
val path: Unit
```

```
    get() {  
        if (parent != null) {  
            print(" $changed")  
        }  
    }  
}
```

```
override fun toString(): String {
```

```
    val sb = StringBuilder()  
    for (i in 0..3) {  
        for (j in 0..3) {  
            sb.append(field[j + i * 4]).append("\t")  
        }  
        sb.append("\n")  
    }  
    return sb.toString()  
}
```

```
override fun equals(o: Any?): Boolean {
```

```
    if (this === o) return true  
    if (o == null || javaClass != o.javaClass) return false  
    val state = o as State  
    return field.contentEquals(state.field)  
}
```

```
override fun hashCode(): Int {
```

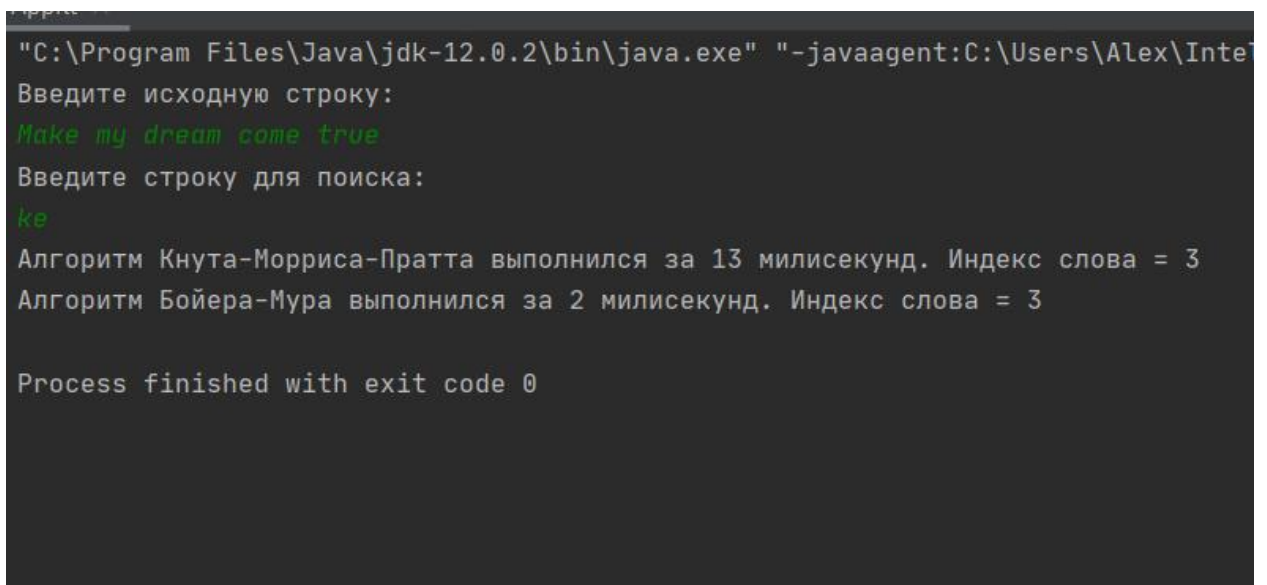
```
    return field.contentHashCode()  
}
```

```
internal constructor(field: IntArray) {
```

```
        this.field = field
    }
}
```

3 Результат работы программы

Результат работы функции поиска элементов в строке изображен на рисунке 1.



```
"C:\Program Files\Java\jdk-12.0.2\bin\java.exe" "-javaagent:C:\Users\Alex\Inte
Введите исходную строку:
Make my dream come true
Введите строку для поиска:
ke
Алгоритм Кнута-Морриса-Пратта выполнен за 13 миллисекунд. Индекс слова = 3
Алгоритм Бойера-Мура выполнен за 2 миллисекунд. Индекс слова = 3

Process finished with exit code 0
```

Рисунок 1 – результат работы функции поиска

Результат работы функции поиска решения пятнашек изображен на рисунке 2.

```
Run: Fifteen x
"C:\Program Files\Java\jdk-12.0.2\bin\java.exe" "-javaagent:C:\Users\A1

5 1 3 4
0 2 6 8
9 10 7 11
13 14 15 12

5 1 3 4
0 2 6 8
9 10 7 11
13 14 15 12

0 1 3 4
5 2 6 8
9 10 7 11
13 14 15 12

1 0 3 4
5 2 6 8
9 10 7 11
13 14 15 12

1 2 3 4
5 0 6 8
9 10 7 11
13 14 15 12
```

```
Run: Fifteen x
7 10 7 11
13 14 15 12

1 2 3 4
5 0 6 8
9 10 7 11
13 14 15 12

1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12

1 2 3 4
5 6 7 8
9 10 0 11
13 14 15 12

1 2 3 4
5 6 7 8
9 10 11 0
13 14 15 12

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0

Process finished with exit code 0
```

Рисунок 2 – результат работы функции решения пятнашек

Вывод: В результате лабораторной работы были созданы функции поиска подстроки в строке и была выполнена задача по решению головоломки пятнашки.