Лабораторна работа №2

Шифры перестановки

Покрас Илья Михайлович

Содержание

Список иллюстраций

1 Цель работы

Ознакомиться с шифрами перестановки и реализовать программный код маршрутного шифрования, шифрования решеток и шифрования Виженера.

2 Задание

- Создать алгоритм маршрутного шифрования
- Создать алгоритм шифрования с помощью решеток
- Создать алгоритм шифрования Виженера

3 Теоретическое введение

- Маршрутное шифрование - это метод шифрования, при котором символы сообщения переставляются или перестраиваются в соответствии с определенным правилом. Каждый символ сообщения заменяется другим символом или перемещается на определенное количество позиций в алфавите.
- Шифрование с помощью решеток - это метод шифрования, при котором сообщение записывается в ячейки решетки, а затем символы выбираются в определенном порядке для формирования зашифрованного текста. Расшифровка происходит путем восстановления исходного текста из решетки.
- Шифрование Виженера - это метод шифрования, основанный на использовании повторяющегося ключа. Каждая буква ключа соответствует определенной букве алфавита, и при шифровании каждая буква сообщения сдвигается на соответствующее значение ключа. Для расшифровки используется обратная операция сдвига.

4 Выполнение лабораторной работы

4.1 Маршрутное шифрования

Я создал функцию маршрутного шифрования с входными данными: исходным текстом, ключом шифрования и параметрами матрицы - количество строк и столбцов. Данная функция возвращает зашифрованный текст (рис. 4.1).

```
function route_encrypt(message, key, rows, cols)
    message = filter(!isspace, message)
    matrix = fill('_', rows, cols)
    index = 1
    new_message = ""
    for i=1:rows
        for j=1:cols
            if index != rows*cols
                matrix[i, j] = message[index]
                index+=1
            end
        end
    end
    for j in sort(collect(key))
        for i=1:rows
            new_message *= (matrix[i, (findfirst(j, key))])
        end
    end
    return new_message
end
```

Рис. 4.1: Функция маршутного шифровани

Далее я инициализировал переменные, которые содержат исходный текст,

ключ шифрования и данные матрицы(строки, столбцы), после чего использовал эти данные в вызове функции маршутного шифрования. (рис. 4.2).

```
message = "this is a test message!"
rows, cols = 4, 5
key = "water"
route_encrypt(message, key, rows, cols)
```

Рис. 4.2: Инициализация переменных и вызов функций 1

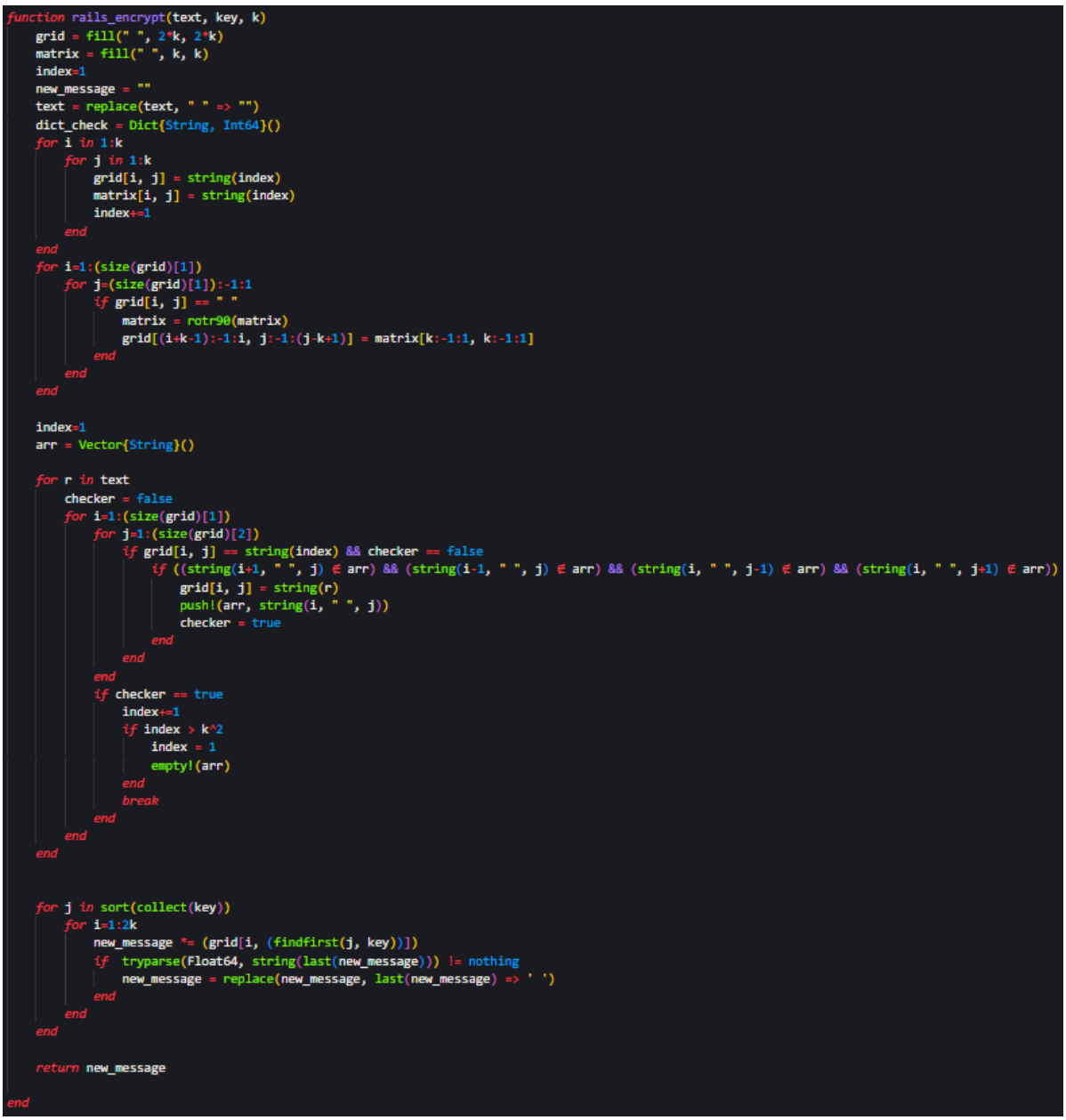
И получил следующий результат (рис. 4.3).

```
"hamgses!iss_iteetsta"
```

Рис. 4.3: Результат программного кода 1

4.2 Шифрование с помощью решеток

Я создал функцию шифрования с помощью решеток принимающую исходный текст, натуральное число k и ключ шифрования, которая создает матрицу с элементами, принимающие значения от 1 до k^2. Далее с помощью циклов я заменил эти числовые значения на символы, содержащиеся в сообщении(кроме пробелов - они удаляются). Далее с помощью ключа я составил новый зашифрованный текст, заменяя те числовые значения, которые остались, на пробелы (рис. 4.4).



```
function rails_encrypt(text, key, k)
    grid = fill(" ", 2*k, 2*k)
    matrix = fill(" ", k, k)
    index=1
    new_message = ""
    text = replace(text, " " => "")
    dict_check = Dict{String, Int64}()
    for i in 1:k
        for j in 1:k
            grid[i, j] = string(index)
            matrix[i, j] = string(index)
            index+=1
        end
    end
    for i=1:(size(grid)[1])
        for j=(size(grid)[1]):-1:1
            if grid[i, j] == " "
                matrix = rotr90(matrix)
                grid[(i+k-1):-1:i, j:-1:(j-k+1)] = matrix[k:-1:1, k:-1:1]
            end
        end
    end

    index=1
    arr = Vector{String}()

    for r in text
        checker = false
        for i=1:(size(grid)[1])
            for j=1:(size(grid)[2])
                if grid[i, j] == string(index) && checker == false
                    if ((string(i+1, " ", j) ∈ arr) && (string(i-1, " ", j) ∈ arr) && (string(i, " ", j-1) ∈ arr) && (string(i, " ", j+1) ∈ arr))
                        grid[i, j] = string(r)
                        push!(arr, string(i, " ", j))
                        checker = true
                    end
                end
            end
            if checker == true
                index+=1
                if index > k^2
                    index = 1
                    empty!(arr)
                end
                break
            end
        end
    end


    for j in sort(collect(key))
        for i=1:2k
            new_message *= (grid[i, (findfirst(j, key))])
            if  tryparse(Float64, string(last(new_message))) != nothing
                new_message = replace(new_message, last(new_message) => ' ')
            end
        end
    end

    return new_message

end
```

Рис. 4.4: Функция шифровани с помощью решеток

Далее я инициализировал переменные, которые содержат исходный текст, натуральное число и ключ шифрования, после чего использовал эти данные в вызове функции шифрования решеток. (рис. 4.5).

```
text = "Hello, New World!"
key = "keys"
k = 2
rails_encrypt(text, key, k)
```

Рис. 4.5: Инициализация переменных и вызов функций 2

И получил следующий результат (рис. 4.6).

```
",lr!HNdwoeolle W"
```

Рис. 4.6: Результат программного кода 2

4.3 Шифрование Виженера

Я создал функцию шифрования Виженера, которая принимает текст и ключ шифрования, и возвращает зашифрованный текст (рис. 4.7).

```
function vigenere_encrypt(text, key)
    alphabet = 'a':'z'
    output = ""
    key_index = 1

    for i in text
        if isletter(i)
            offset = findfirst(isequal(key[key_index]), alphabet) - 1
            index = findfirst(isequal(i), alphabet) + offset
            index > 26 && (index -= 26)
            output *= alphabet[index]
            key_index += 1
            key_index > length(key) && (key_index = 1)
        else
            new_message *= i
        end
    end

    return new_message
end
```

Рис. 4.7: Функция шифровани Виженера

Далее я инициализировал переменные, которые содержат исходный текст, ключ шифрования, после чего использовал эти данные в вызове функции шифрования Виженера. (рис. 4.8).

```
text = "hello world"
key = "key"
encrypted_text = vigenere_encrypt(text, key)
```

Рис. 4.8: Инициализация переменных и вызов функций 3

И получил следующий результат (рис. 4.9).

```
"rijvs uyvjn"
```

Рис. 4.9: Результат программного кода 3

5 Выводы

Я ознакомился с шифрами перестановки и реализовал программный код маршрутного шифрования, шифрования решеток и шифрования Виженера.

Список Литературы

1. Julia - Control Flow
2. Julia - Mathematical Operations
3. Julia - Strings
4. Julia - Arrays
5. Julia - Collections and Data Structures