МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет ИТМО

Лабораторная работа №1

«Алгоритмы хэширования»

Цель работы: Цель работы: изучить алгоритмы хеширования, реализовать алгоритм Рабина-Карпа поиска подстроки в строке с применением хеширования.

Выполнила студентка группы №K3222

Маркозубова Анастасия Кирилловна

Задание

1.Реализовать поиск одинаковых строк.

Дан список строк S[1..n], каждая длиной не более m символов. Требуется найти все повторяющиеся строки и разделить их на группы, чтобы в каждой группе были только одинаковые строки.

2. Реализовать алгоритм Рабина-Карпа поиска подстроки в строке за O (n).

Задание 1. Реализовать поиск одинаковых строк.
Дан список строк S[1..n], каждая длиной не более m символов. Требуется найти все повторяющиеся строки и разделить их на группы, чтобы в каждой группе были только одинаковые строки

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static string[] ReadLines(int n)
    {
        string[] lines = new string[n];
        for (int i = 0; i < n; i++)
        {
            Console.Write($"Введите строку {i+1}: ");
            lines[i] = Console.ReadLine();
        }
        return lines;
    }

    static long CalculateHash(string line, long[] p_pows, long mod)
    {
        long hash = 0;

        for (int i = 0; i < line.Length; i++)
        {
            hash = (hash + (line[i] - 'a' + 1) * p_pows[i]) % mod;
        }

        return hash;
    }

    static long[] CalculatePower(int p, int n, long mod)
    {
        long[] p_pows = new long[n];
        p_pows[0] = 1;

        for (int i = 1; i < n; i++)
        {
            p_pows[i] = (p_pows[i - 1] * p) % mod;
        }

        return p_pows;
    }

    static Dictionary<int, long> HashDictionary(string[] lines, int p, int n,
long mod)
    {
        var p_pows = CalculatePower(p, n, mod);
        var hash = new Dictionary<int, long>();

        for (int i = 0; i < lines.Length; i++)
        {
            hash[i] = CalculateHash(lines[i], p_pows, mod);
        }

        return hash;
    }
```

```
    static void Main()
    {
        int n = 10;
        int p = 31;
        long mod = 1000000007;

        Console.Write("Введите количество строк: ");
        n = int.Parse(Console.ReadLine());

        string[] lines = ReadLines(n);
        var hashes = HashDictionary(lines, p, n, mod);

        hashes = hashes.OrderBy(pair => pair.Value).ToDictionary(pair =>
pair.Key, pair => pair.Value);
        foreach (var item in hashes)
        {
            Console.WriteLine(item.Key + " * * * " + item.Value);
        }

        int group_number = 1;
        var previous_hash = hashes.First().Value;
        Console.WriteLine("Group " + group_number + ":");
        foreach (var hash in hashes)
        {
            if (hash.Value != previous_hash)
            {
                group_number++;
                Console.WriteLine("Group " + group_number + ":");
            }
            Console.WriteLine(hash.Key);
            previous_hash = hash.Value;
        }
    }
```

```
}
Введите количество строк: 10
Введите строку 1: love
Введите строку 2: love
Введите строку 3: sun
Введите строку 4: sun
Введите строку 5: love
Введите строку 6: love
Введите строку 7: dive
Введите строку 8: wind
Введите строку 9: dive
Введите строку 10: dive
2 * * * 14124
3 * * * 14124
7 * * * 132920
6 * * * 170380
8 * * * 170380
9 * * * 170380
0 * * * 170574
1 * * * 170574
4 * * * 170574
5 * * * 170574
Group 1:
2
3
Group 2:
7
Group 3:
6
8
9
Group 4:
0
1
4
5
```

Задание 2. Реализовать алгоритм Рабина-Карпа поиска подстроки в строке за O(n).
Дана строка S и текст T, состоящие из маленьких латинских букв. Требуется найти все вхождения строки S в текст T за время O(|S| + |T|).
using System;

```
class Program
    {
        static void Main(string[] args)
        {
            string s = Console.ReadLine();
            string t = Console.ReadLine();

            int p = 31;
            int n = s.Length;
            int m = t.Length;

            int[] p_pow = new int[Math.Max(n, m)];
            p_pow[0] = 1;

            for (int i = 1; i < p_pow.Length; i++)
            {
                p_pow[i] = p_pow[i - 1] * p;
            }

            int[] hashs = new int[m];
            for (int i = 0; i < m; i++)
            {
                hashs[i] = (t[i] - 'a' + 1) * p_pow[i];
                if (i != 0)
                {
                    hashs[i] += hashs[i - 1];
                }
            }

            int s_hash = 0;
            for (int i = 0; i < n; i++)
            {
                s_hash += (s[i] - 'a' + 1) * p_pow[i];
            }

            int cur_h;
            for (int i = 0; (i + n - 1) < m; i++)
            {
                cur_h = hashs[i + n - 1];
                if (i != 0)
                {
                    cur_h -= hashs[i - 1];
                }
                if (cur_h == s_hash * p_pow[i])
                {
                    Console.WriteLine(i);
                }
            }
        }
    }
```

```
ab
ababccdababghfgbaab
0
2
7
9
17
```

Заключение

В процессе выполнения лабораторной работы были изучены и реализованы алгоритмы хеширования с использованием языка программирования C#. В частности, были рассмотрены следующие методы:

- Полиномиальное хеширование, которое позволяет эффективно вычислять хеш-код строки и быстро находить повторяющиеся элементы.
- Алгоритм Рабина-Карпа, применяемый для поиска подстроки в тексте на основе хеширования, что значительно повышает скорость поиска по сравнению с простым перебором.