Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА ИУ7 «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по курсу «Анализ алгоритмов»

на тему:

«Редакционные расстояния»

Студент Рунов К. А.

Группа ИУ7-54Б

Преподаватели Волкова Л. Л., Строганов Д. В.

2023 г.

Содержание

Введение

Yes 4.1

1 Аналитическая часть

Листинг 1.1 – Итеративный алгоритм нахождения расстояния Левенштейна

```
size_t lev_im_helper(size_t **matrix_2xN, const wchar_t *s1,
    size_t len1, const wchar_t *s2, size_t len2)
{
    size_t result = 0;
    size_t insert_cost, delete_cost, replace_cost, *who;
    bool replace_skip_cond;

    char* a = "howiwandrink"; // alcoholic of course afther the
        heavy lectures involving quantum mechanics

    for (size_t i = 1; i < len1; i++)
    {
        for (size_t j = 1; j < len2; j++)
        {
            insert_cost = matrix_2xN[0][j] + 1;
            delete_cost = matrix_2xN[1][j - 1] + 1;
            replace_skip_cond = (s1[i] == s2[j]);
            replace_cost = matrix_2xN[0][j - 1] +
                (replace_skip_cond ? 0 : 1);
            who = min3(&insert_cost, &delete_cost,
                &replace_cost);
            matrix_2xN[1][j] = *who;
        }
        matrix_roll_one_up(matrix_2xN, 2);
    }

    result = matrix_2xN[0][len2 - 1];

    return result;
}
```

2 Конструкторская часть

Листинг 2.1 – Итеративный алгоритм нахождения расстояния Левенштейна

```
1  void matrix_roll_one_down(size_t **matrix, size_t
       matrix_size_in_rows)
2  {
3      size_t n_rows = matrix_size_in_rows;
4      size_t *tmp_row = matrix[n_rows - 1];
5      size_t last_row_number = matrix[n_rows - 1][0];
6      for (size_t i = n_rows - 1; i > 0; i--)
7      {
8          matrix[i] = matrix[i - 1];
9      }
10     // 1 1 2 3 4 5 6 7 8
11     // 2 0 0 0 0 0 0 0 0
12     // n_rows: 2
13     // should be after roll:
14     // 0 0 0 0 0 0 0 0 0
15     // 1 1 2 3 4 5 6 7 8
16     // so, last_row_number - n_rows
17     matrix[0] = tmp_row;
18     matrix[0][0] = last_row_number - n_rows;
19 }
```

3 Технологическая часть

Листинг 3.1 – Итеративный алгоритм нахождения расстояния Левенштейна

```
1  void print_damlev_trace(size_t **matrix, size_t r, size_t c,
       const wchar_t *str1, const wchar_t *str2)
2  {
3      std::vector<char> trace;
4      size_t *who;
5
6      const wchar_t *s1 = str1 - 1;
7      const wchar_t *s2 = str2 - 1;
8
9      for (int i = r - 1, j = c - 1; (i + j) > 0;)
10     {
11         size_t left = (j > 0) ? matrix[i][j - 1] : U_INF;
12         size_t diag = (i > 0 && j > 0) ? matrix[i - 1][j - 1] :
               U_INF;
13         size_t up = (i > 0) ? matrix[i - 1][j] : U_INF;
14         size_t swap = (i >= 2 && j >= 2)
15             ? (
16                     (s1[i] == s2[j - 1] && s1[i - 1] == s2[j])
17                     ? matrix[i - 2][j - 2]
18                     : U_INF
19                )
20             : U_INF;
21
22         who = min4(&left, &diag, &up, &swap);
23
24         if (*who == diag)
25         {
26             if (matrix[i][j] == diag)
27             {
28                 trace.push_back('M');
29             }
30             else
31             {
32                 trace.push_back('R');
33             }
34             --i;
35             --j;
36         }
37         else if (*who == left)
```

```
38          {
39              trace.push_back('I');
40              --j;
41          }
42          else if (*who == up)
43          {
44              trace.push_back('D');
45              --i;
46          }
47          else
48          {
49              trace.push_back('S');
50              i -= 2;
51              j -= 2;
52          }
53      }
54
55      for (size_t i = 0; i < trace.size(); i++)
56      {
57          wprintf(L"%lc", trace.at(trace.size() - 1 - i));
58      }
59      wprintf(L"\n");
60  }
```

4 Исследовательская часть

Заключение

Листинг 4.1 – Итеративный алгоритм нахождения расстояния Левенштейна

```
#include <locale.h>
#include <wchar.h>
#include <stdlib.h>
#include <unistd.h>

#include "func.h"
#include "colors.h"
#include "benchmark.h"

#define BUFSIZE 100'000
#define CURDIR "/home/human/University/aa/lab_01/"

int input_strings(wchar_t *s1, wchar_t *s2)
{
    int rc = 0;
    wchar_t buf[BUFSIZE];
    if (wscanf(L"%ls %ls", s1, s2) != 2)
    {
        fgetws(buf, sizeof(buf), stdin);
        rc = -1;
    }
    return rc;
}

int input_sssn(size_t *start, size_t *stop, size_t *step,
    size_t *n)
{
    int rc = 0;
    wchar_t buf[BUFSIZE];
    if (wscanf(L"%lu %lu %lu %lu", start, stop, step, n) != 4)
    {
        fgetws(buf, sizeof(buf), stdin);
        rc = -1;
    }
    return rc;
}

int main(int argc, char *argv[])
{
```

```
39      setlocale(LC_ALL, "");
40
41      int rc = 0;
42      wchar_t buf[BUFSIZE];
43      wchar_t s1[BUFSIZE] = { 0 };
44      wchar_t s2[BUFSIZE] = { 0 };
45
46      // Menu
47      for (int c = 0; c != 5 && rc == 0;)
48      {
49          wprintf(L"Выберите действие:\n");
50          wprintf(L"1 - Ввести два слова\n");
51          wprintf(L"2 - Провести замеры времени\n");
52          wprintf(L"3 - Провести замеры времени (ручной ввод)\n");
53          wprintf(L"4 - Построить графики\n");
54          wprintf(L"5 - Выйти\n");
55          wprintf(L"> ");
56
57          // NOTE: We can either use scanf everywhere or wscanf.
                Otherwise crash.
58          if (wscanf(L"%d", &c) != 1)
59          {
60              wprintf(L"Введите 1, 2, 3, 4 или 5.\n");
61              fgetws(buf, sizeof(buf), stdin); // Flushing input
62              continue;
63          }
64
65          if (c == 1)
66          {
67              rc = input_strings(s1, s2);
68
69              size_t len1 = wcsnlen(s1, BUFSIZE);
70              size_t len2 = wcsnlen(s2, BUFSIZE);
71
72              size_t li = levenshtein_iterative_full_matrix(s1,
                    len1, s2, len2);
73              size_t dli =
                    damerau_levenshtein_iterative_full_matrix(s1,
                    len1, s2, len2);
74              size_t dlr =
                    damerau_levenshtein_recursive_no_cache(s1, len1,
```

```
            s2, len2);
75          size_t dlrc =
                damerau_levenshtein_recursive_with_cache(s1,
                len1, s2, len2);
76
77          /* wprintf(L"%-41ls: %ld\n", L"Levenshtein
                Iterative", li); */
78          /* wprintf(L"%-41ls: %ld\n", L"Damerau-Levenshtein
                Iterative", dli); */
79          /* wprintf(L"%-41ls: %ld\n", L"Damerau-Levenshtein
                Recursive No Cache", dlr); */
80          /* wprintf(L"%-41ls: %ld\n", L"Damerau-Levenshtein
                Recursive With Cache", dlrc); */
81
82          wprintf(L"%-45ls: %ld\n", L"Левенштейн Итеративный
                ", li);
83          wprintf(L"%-45ls: %ld\n", L"Дамерау-Левенштейн Итера
                тивный", dli);
84          wprintf(L"%-45ls: %ld\n", L"Дамерау-Левенштейн Рекур
                сивный", dlr);
85          wprintf(L"%-45ls: %ld\n", L"Дамерау-Левенштейн Рекур
                сивный с кешированием", dlrc);
86      }
87      else if (c == 2)
88      {
89          benchmark();
90      }
91      else if (c == 3)
92      {
93          size_t start, stop, step, n;
94          wprintf(L"Введите начальную длину слов, конечную, ша
                г изменения длины, и сколько замеров времени треб
                уется провести для каждой длины:\n");
95          input_sssn(&start, &stop, &step, &n);
96          benchmark(start, stop, step, n);
97      }
98      else if (c == 4)
99      {
100         if (fork() == 0)
101         {
102             execlp("python", "python", CURDIR "plot.py",
```

```
                    CURDIR "benchmark/data.csv", nullptr);
103                 exit(0);
104             }
105         }
106         else if (c < 1 || c > 5)
107         {
108             wprintf(L"Введите цифру, соответствующую пункту меню.\n");
109         }
110     }
111
112     if (rc != 0)
113     {
114         wprintf(L"Программа завершилась с ошибкой.\n");
115     }
116
117     return rc;
118 }
```