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.

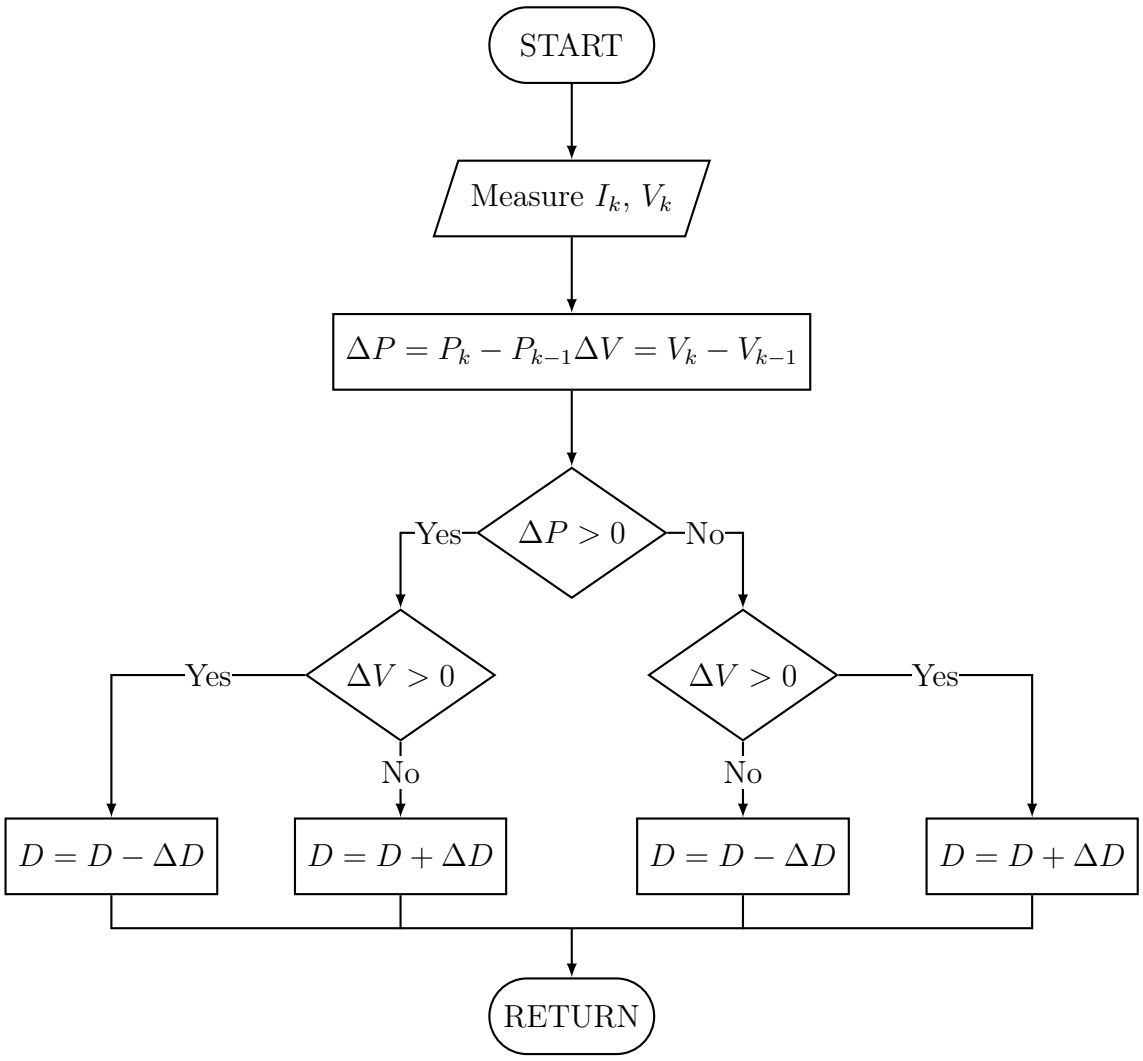
Группа ИУ7-54Б

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

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ


START
Measure I_k, V_k
$\Delta P = P_k - P_{k-1} \Delta V = V_k - V_{k-1}$
$\Delta P > 0$
Yes
No
$\Delta V > 0$
$\Delta V > 0$
Yes
Yes
No
No
$D = D - \Delta D$
$D = D + \Delta D$
$D = D - \Delta D$
$D = D + \Delta D$
RETURN

1 Аналитическая часть

2 Конструкторская часть

3 Технологическая часть

4 Исследовательская часть

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Левенштейн В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов. – М.: «Наука», Доклады АН СССР, 1965. Т. 163. С. 845–848.

[1]

Листинг 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 Итерати
                вный", dli);
84          wprintf(L"%-45ls: %ld\n", L"Дамерау-Левенштейн Рекурси
                вный", dlr);
85          wprintf(L"%-45ls: %ld\n", L"Дамерау-Левенштейн Рекурси
                вный 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, шаг
                изменения длины, и сколько замеров времени требуетс
                я 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 }
```