Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ОТЧЕТ

по дисциплине Алгоритмы и структуры данных
Практическая работа №7с — Дополнительные задания. Подходы к алгоритмизации

Преподаватель _____ Матковский И. В.
подпись, дата инициалы, фамилия

Студент КИ19-07Б, 031941597 _____ Горбацевич А. А.
номер группы, зачётной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

1. Задание на работу

1.1 Для заданной проблемы выбрать наиболее эффективный алгоритм и реализовать его в виде программы.

2. Задание на вариант

2.1 Даны две строки. За одно действие вы можете убрать из строки один произвольный символ или добавить в строку новый произвольный символ. Сколько действий вам нужно, чтобы превратить одну строку в другую?

3. Исходный код программы

```
// dsaa_07.cpp
// Горбацевич Андрей
#include <iostream>
#include <chrono>
#include <valarray>


inline void time_passed(std::chrono::high_resolution_clock::time_point start, double& holder) {
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(stop - start);
    holder = duration.count();
}

class matrix {
private:
    int _rows = 0;
    int _cols = 0;
    std::valarray<int> data;

public:
    matrix()  = default;
    matrix(int32_t rows, int32_t cols) : _rows(rows), _cols(cols), data(rows*cols) {}

    int& operator()(int i, int j) {
        return this->data[i * this->_cols + j];
    }

    int operator()(int i, int j) const {
        return this->data[i * this->_cols + j];
    }

    [[nodiscard]] int cols() const {
        return this->_cols;
    }

    [[nodiscard]] int rows() const {
        return this->_rows;
    }
};

class levenstein{
private:
    std::string ls;
    std::string rs;

    matrix D;

public:
    levenstein(const std::string &ls, const std::string &rs) {
        this->ls = ls;
        this->rs = rs;
        this->D = matrix(ls.length() + 1, rs.length() + 1);

        D(0, 0) = 0;
        for (int j = 1; j <= D.cols(); j++) {
```

```
            D(0, j) = D(0, j - 1) + 1;
        }
        for (int i = 1; i <= D.rows(); i++) {
            D(i, 0) = D(i - 1, 0) + 1;
            for (int j = 1; j <= D.cols(); j++) {
                D(i, j) = std::min({
                    D(i - 1, j) + 1,
                    D(i, j - 1) + 1,
                    D(i - 1, j - 1) + (ls[i] == rs[j]? 0 : 2)
                });
            }
        }
    }

    int distance() {
        return this->D(ls.length(), rs.length());
    }

    void print() {
        printf("\t\t");
        for (int j = 0; j < D.cols(); j++) {
            printf("%c\t", rs[j]);
        }
        printf("\n");

        for (int i = 0; i < D.rows(); i++) {
            for (int j = 0; j < D.cols(); j++) {
                if (j == 0) {
                    if (i > 0) {
                        printf("%c\t", ls[i - 1]);
                    }
                    else {
                        printf("\t");
                    }
                }
                printf("%d\t", D(i, j));
            }
            printf("\n");
        }

        printf("\n");
    }
};

int main() {
    std::string f_word, s_word;
    std::cout << "Enter two words, separated by space symbol:\n";
    std::cin >> f_word >> s_word;

    double ellapsed_time;
    auto start = std::chrono::high_resolution_clock::now();
    auto l = levenstein(f_word, s_word);
    time_passed(start, ellapsed_time);

    l.print();
    std::cout << l.distance() << " action(s) need to be done, found in " << ellapsed_time << "microseconds\n";

    return 0;
}
```

4. Теоретические оценки временной сложности алгоритмов

4.1 Расстояние Левенштейна (а именно - Алгоритм Вагнера — Фишера) позволяет рассчитывать нам на O(NM), где N и M – длины рассматриваемых строк.

5. Экспериментальные оценки временной и пространственной сложности программы

Первое слово	Второе слово	Время, микросекунды
brooklyn	bekley	0
java	kotlin	0
calamity	cataclysm	0

Приложение А
Результаты работы программы

```
Enter two words, separated by space symbol:
brooklyn berkley
                b       e       r       k       l       e       y
        0       1       2       3       4       5       6       7
b       1       2       1       2       3       4       5       6
r       2       3       2       3       4       5       6       7
o       3       4       3       4       5       6       7       8
o       4       5       4       3       4       5       6       7
k       5       6       5       4       3       4       5       6
l       6       7       6       5       4       5       4       5
y       7       8       7       6       5       6       5       6
n       8       9       8       7       6       7       6       5

5 action(s) need to be done, found in 0 microseconds
```

Рисунок 1: результат работы программы

```
Enter two words, separated by space symbol:
java kotlin
                k       o       t       l       i       n
        0       1       2       3       4       5       6
j       1       2       3       4       5       6       7
a       2       3       4       5       6       7       8
v       3       4       5       6       7       8       9
a       4       5       6       7       8       9       8

8 action(s) need to be done, found in 0 microseconds
```

Рисунок 2: результат работы программы