Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №4**

**Динамическое программирование**

Выполнил:

Студент 2 курса 7 группы ФИТ

Володькин Никифор Дмитриевич

**ЦЕЛЬ РАБОТЫ: освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.**

***Задание 1***. На языке С++ сгенерировать случайным образом строку букв латинского алфавита S1 длиной 300 символов и S2 длиной 200.

Была реализована программа на языке C++ для генерация случайных строк.

#include <iostream>

#include <string>

int main()

{

srand(time(NULL));

std::string S1 = "";

for (int i = 0; i < 300; i++)

{

S1 += ('a' + rand() % 26);

}

std::string S2 = "";

for (int i = 0; i < 200; i++)

{

S2 += ('a' + rand() % 26);

}

std::cout << "S1: " << S1 << std::endl;

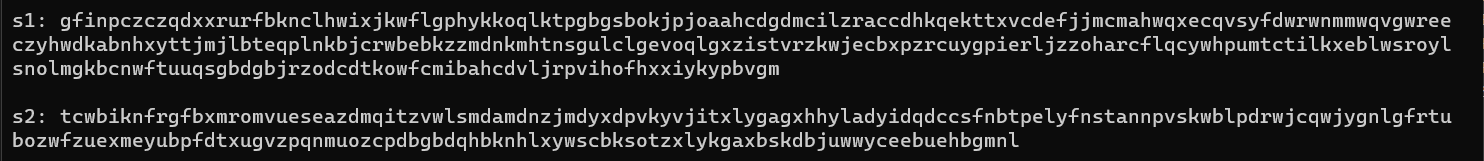
std::cout << std::endl;

std::cout << "S2: " << S2 << std::endl;

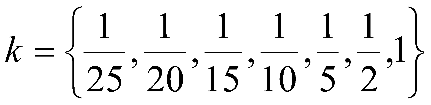
std::cout << std::endl;

}

Генерация строк на языке C++



Результат выполнения программы

***Задание 2***. Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки .

Были разработаны функции для вычисления дистанции Левенштейна.

#include <iostream>

#include <iomanip>

#include <algorithm>

#include <string>

#include "Levenshtein.h"

#define DD(i,j) d[(i)\*(ly+1)+(j)]

int min3(int x1, int x2, int x3)

{

return std::min(std::min(x1, x2), x3);

}

int levenshtein(int lx, std::string x, int ly, std::string y)

{

int\* d = new int[(lx + 1) \* (ly + 1)];

for (int i = 0; i <= lx; i++) DD(i, 0) = i;

for (int j = 0; j <= ly; j++) DD(0, j) = j;

for (int i = 1; i <= lx; i++)

{

for (int j = 1; j <= ly; j++)

{

DD(i, j) = min3(

DD(i - 1, j) + 1,

DD(i, j - 1) + 1,

DD(i - 1, j - 1) + (x[i - 1] == y[j - 1] ? 0 : 1)

);

}

}

return DD(lx, ly);

}

int levenshtein\_r(int lx, std::string x, int ly, std::string y)

{

int rc = 0;

if (lx == 0) rc = ly;

else if (ly == 0) rc = lx;

else if (lx == 1 && ly == 1 && x[0] == y[0]) rc = 0;

else if (lx == 1 && ly == 1 && x[0] != y[0]) rc = 1;

else rc = min3(

levenshtein\_r(lx - 1, x, ly, y) + 1,

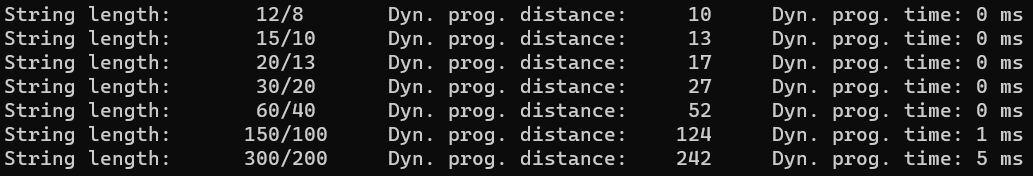
levenshtein\_r(lx, x, ly - 1, y) + 1,

levenshtein\_r(lx - 1, x, ly - 1, y) + (x[lx - 1] == y[ly - 1] ? 0 : 1)

);

return rc;

};

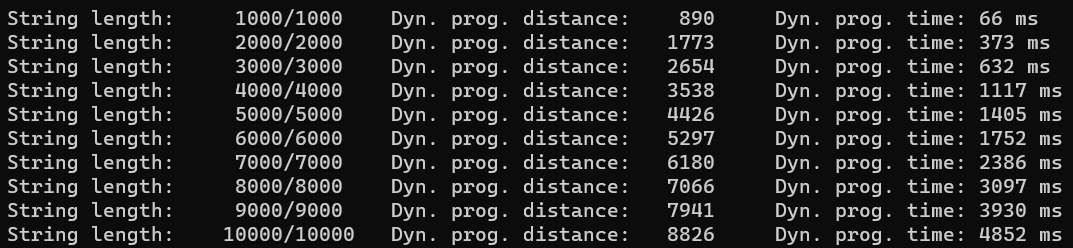


Результат вычисления дистанции Левенштейна

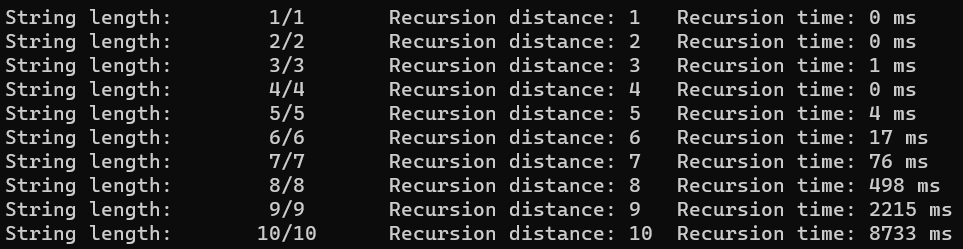
Из-за большой ресурсоемкости не удается вычислить дистанцию Левенштейна для строк такой длины рекурсивным методом.

***Задание 3.*** Выполнить сравнительный анализ времени затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . (копии экрана и график вставить в отчет).

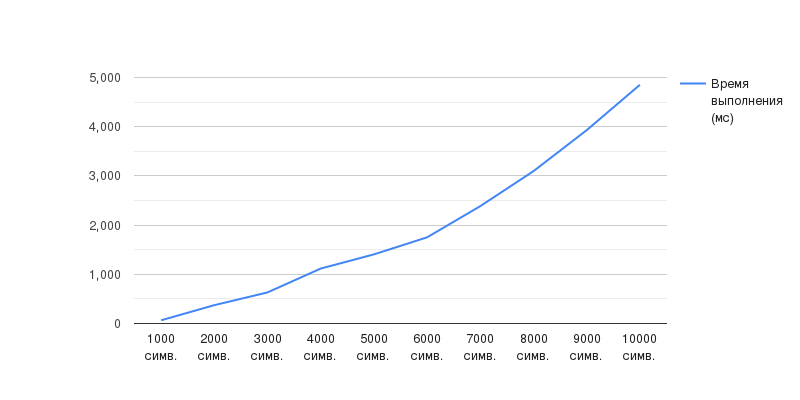
Из-за большой разности в производительности были использованы различные длины строк для демонстрации разницы производительности двух алгоритмов.



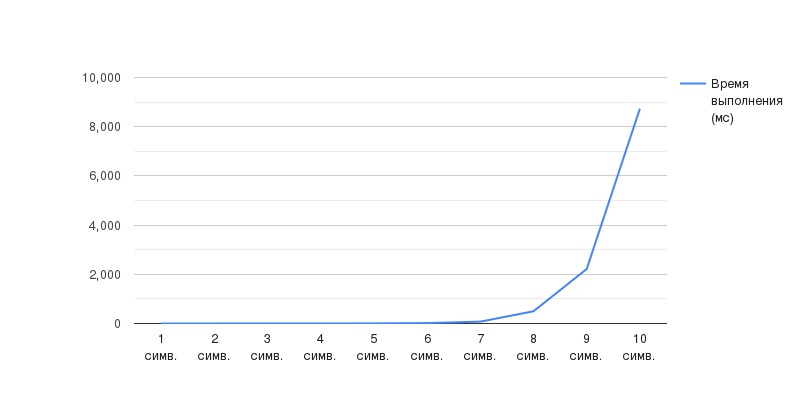
Результат вычисления с помощью динамического программирования



Результат вычисления с помощью рекурсии



Зависимость времени выполнения алгоритма динамического программирования от количества символов



Зависимость времени выполнения алгоритма динамического программирования от количества символов

**ВЫВОД: Освоены общие принципы решения задач методом динамического программирования, полученные решения задач сравнены с рекурсивным методом.**