МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет: Информационных технологий

Кафедра: Программной инженерии

Выполнила: студентка 2 курса 5 группы

специальности ПОИТ Ворович К. C.

**Отчёт**

По дисциплине “Математическое программирование”

На тему “Динамическое программирование”

Минск

2024

**Лабораторная работа 4. Динамическое программирование.**

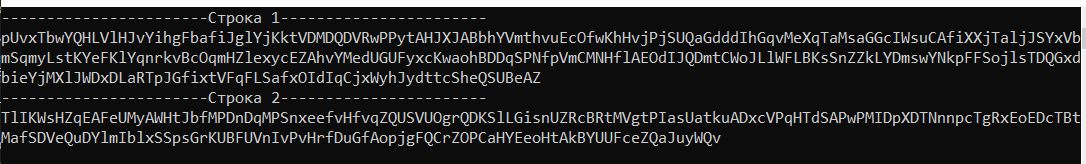
**Цель работы:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

**Ход Работы**

1. **Сгенерировать 2 строки S1 и S2 на С++**



Генератор случайных строк



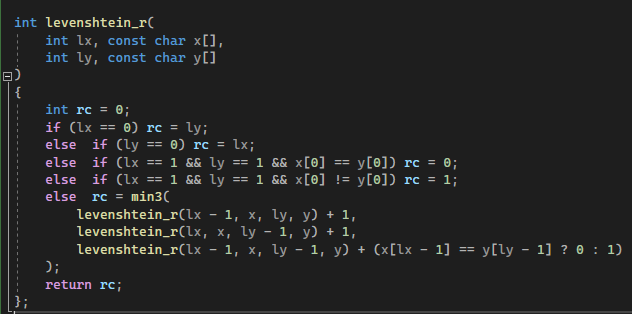
Генерация строк

1. **Вычисление дистанции Левенштейна с помощью рекурсии и динамического программирования.**

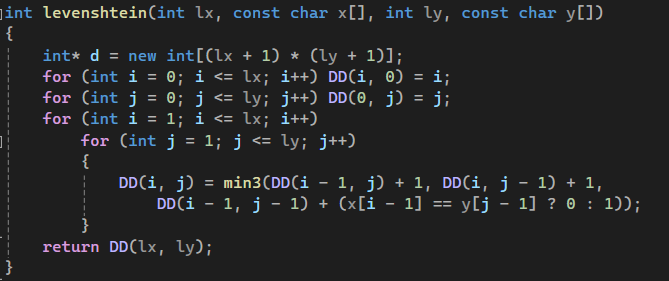
Вычисление каждого преобразования строки вычисляется по формуле Вагнера — Фишера:



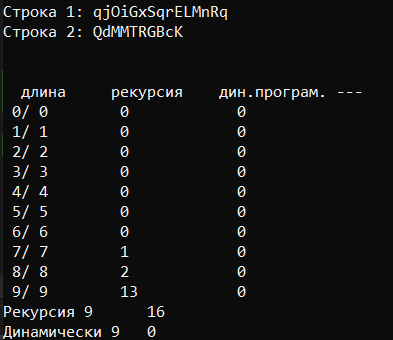
Функция для подсчёта с помощью рекурсии:



Функция для динамической реализации:

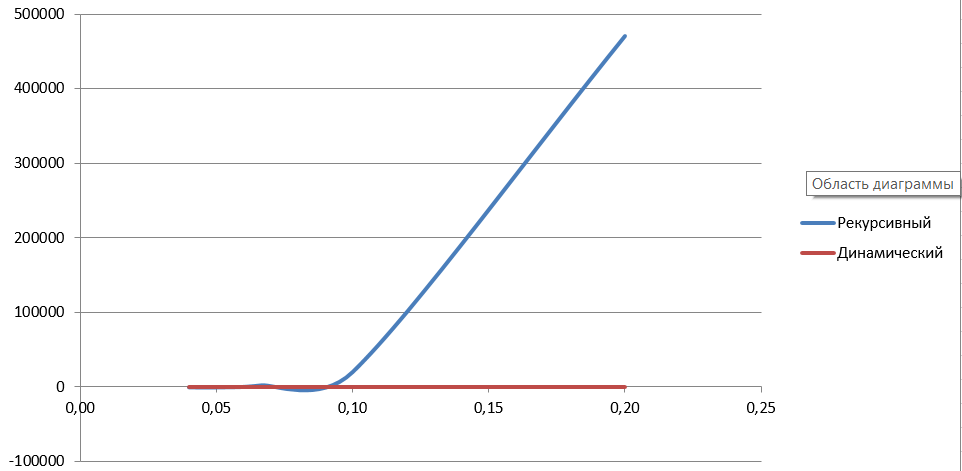


Сравнение:



Для k = 1/20

1. **Сравнительный анализ на графике**



Из графиков видно, что рекурсивный метод крайне неэффективен, по сравнению с методом динамического программирования.

1. **Вычисление дистанции Левенштейна вручную**

акр якорь

1. =1
2. **Решение задачи об оптимальной расстановке скобок при умножении**

**нескольких матриц (рекурсивый метод и динамический метод)**

int OptimalM(int i, int j, int n, const int c[], int\* s)

{

#define OPTIMALM\_S(x1,x2) (s[(x1-1)\*n+x2-1])

int o = INFINITY, bo = INFINITY;

if (i < j)

{

for (int k = i; k < j; k++)

{

bo = OptimalM(i, k, n, c, s) +

OptimalM(k + 1, j, n, c, s) + c[i - 1] \* c[k] \* c[j];

if (bo < o)

{

o = bo;

OPTIMALM\_S(i, j) = k;

}

}

}

else o = 0;

return o;

#undef OPTIMALM\_S

};

// расстановка скобок (динамическое программирование)

int OptimalMD(int n, const int c[], int\* s)

{

#define OPTIMALM\_S(x1,x2) (s[(x1-1)\*n+x2-1])

#define OPTIMALM\_M(x1,x2) (M[(x1-1)\*n+x2-1])

int\* M = new int[n \* n], j = 0, q = 0;

for (int i = 1; i <= n; i++) OPTIMALM\_M(i, i) = 0;

for (int l = 2; l <= n; l++)

{

for (int i = 1; i <= n - l + 1; i++)

{

j = i + l - 1;

OPTIMALM\_M(i, j) = INFINITY;

for (int k = i; k <= j - 1; k++)

{

q = OPTIMALM\_M(i, k) + OPTIMALM\_M(k + 1, j) + c[i - 1] \* c[k] \* c[j];

if (q < OPTIMALM\_M(i, j))

{

OPTIMALM\_M(i, j) = q; OPTIMALM\_S(i, j) = k;

}

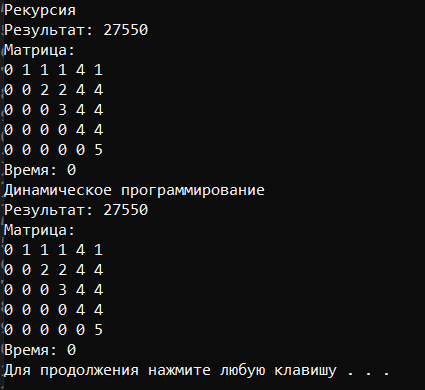
}

}

}

return OPTIMALM\_M(1, n);

Результат программы:



Принцип расстановки скобок по итоговой матрице:

Скобки расставляются по принципу «сначала внешние – затем внутренние». Имеется 6 матриц с размерностями А1=20\*15, А2=15\*30, А3=30\*53, А4 =53\*10, А5 =10\*20, А6 =20\*11.

Матрица S:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 4 | 1 |
| 0 | 0 | 2 | 2 | 4 | 4 |
| 0 | 0 | 0 | 3 | 4 | 4 |
| 0 | 0 | 0 | 0 | 4 | 4 |
| 0 | 0 | 0 | 0 | 0 | 5 |

Найдем элемент (1,6) в матрице S, он равен 1. Это означает, что точка разрыва между 1-ой и 6-ой матрицей находится после 1-ой матрицы. Что позволяет расставить скобки следующим образом:

A1\*(A2\*A3\*A4\*A5\*A6)

Точку разрыва между второй и шестой матрицей определяет элемент (2,6). Он равен 4. Следовательно разрыв будет после 4-ой матрицы.

A1\*((A2\*A3\*A4) \* (A5\*A6)

Далее берем элемент (2,4) и получаем, что он равен 2. Следовательно получаем:

A1\*(((A2\*(A3\*A4)) \* (A5\*A6)

Это выражение и есть конечное.

Полученная расстановка скобок позволяет получить минимальное количество операций умножения, результат равен 27550, матрица (20\*11).

**Вывод:** в результате выполнения лабораторной работы были освоены общие принципы решения задач методом динамического программирования. Были изучены его основные этапы и принципы работы алгоритмов. Были рассмотрены примеры решения задач методом динамического программирования и сравнены с рекурсивным методом.