**Лабораторная работа 4**

**ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

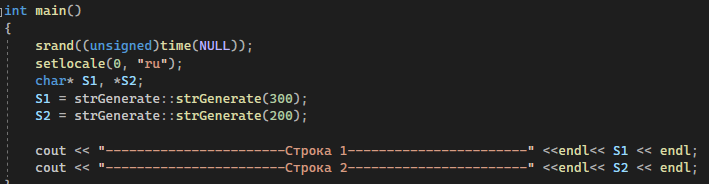
**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

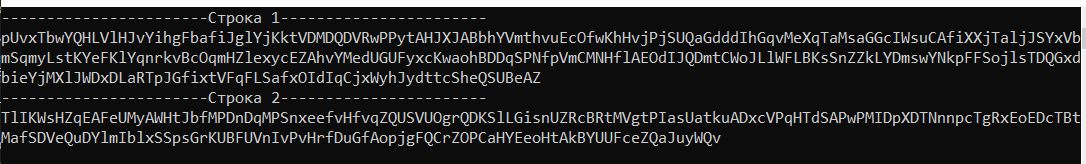
***Задание 1.***

Генератор случайных строк:



Генерирование строк S1(300 символов) и S2(200 символов):

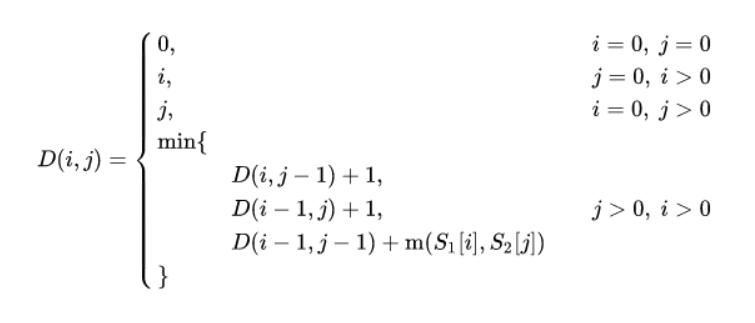




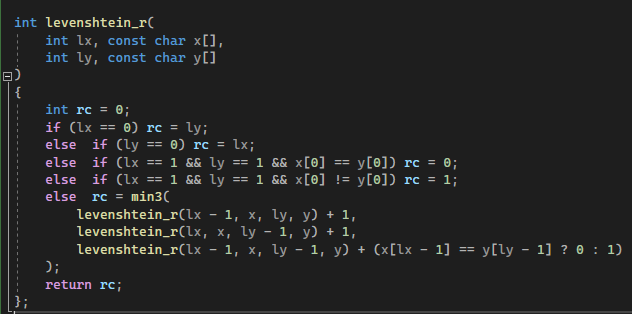
***Задание 2.***

Расстояние Левенштейна (редакционное расстояние, дистанция редактирования) — метрика, измеряющая по модулю разность между двумя последовательностями символов. Это минимальное число односимвольных преобразований (удаления, вставки или замены), необходимых, чтобы превратить одну последовательность в другую.

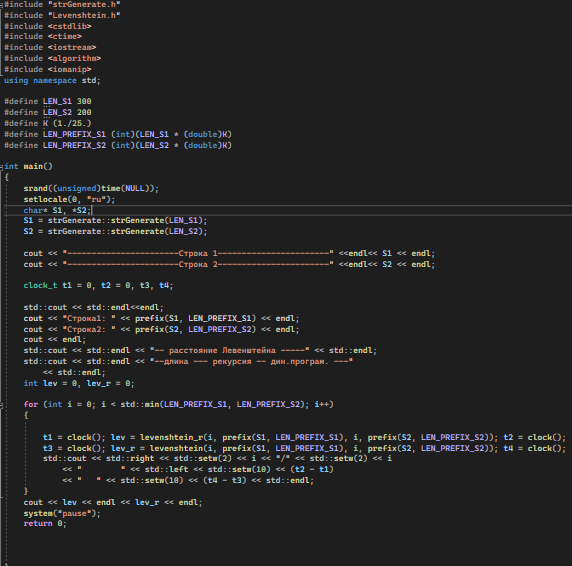
Вычисление каждого преобразования строки вычисляется по формуле Вагнера — Фишера:



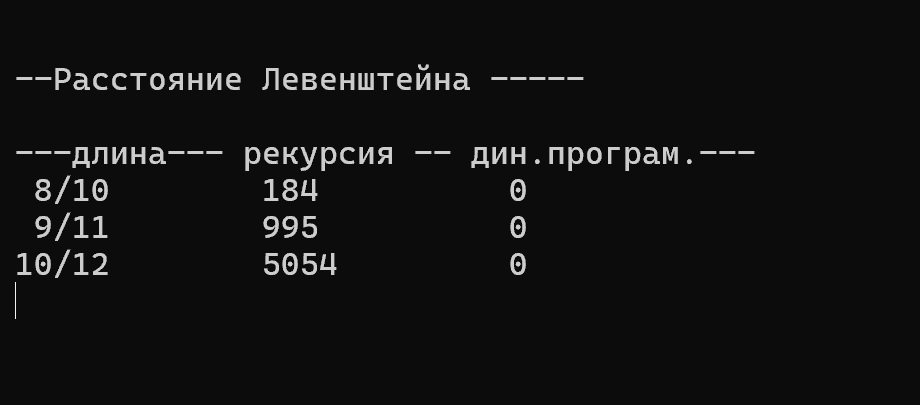
**Вычисление расстояния Левенштейна для строк S1 S2**



Реализация рекурсивного метода вычисления расстояния Левенштейна

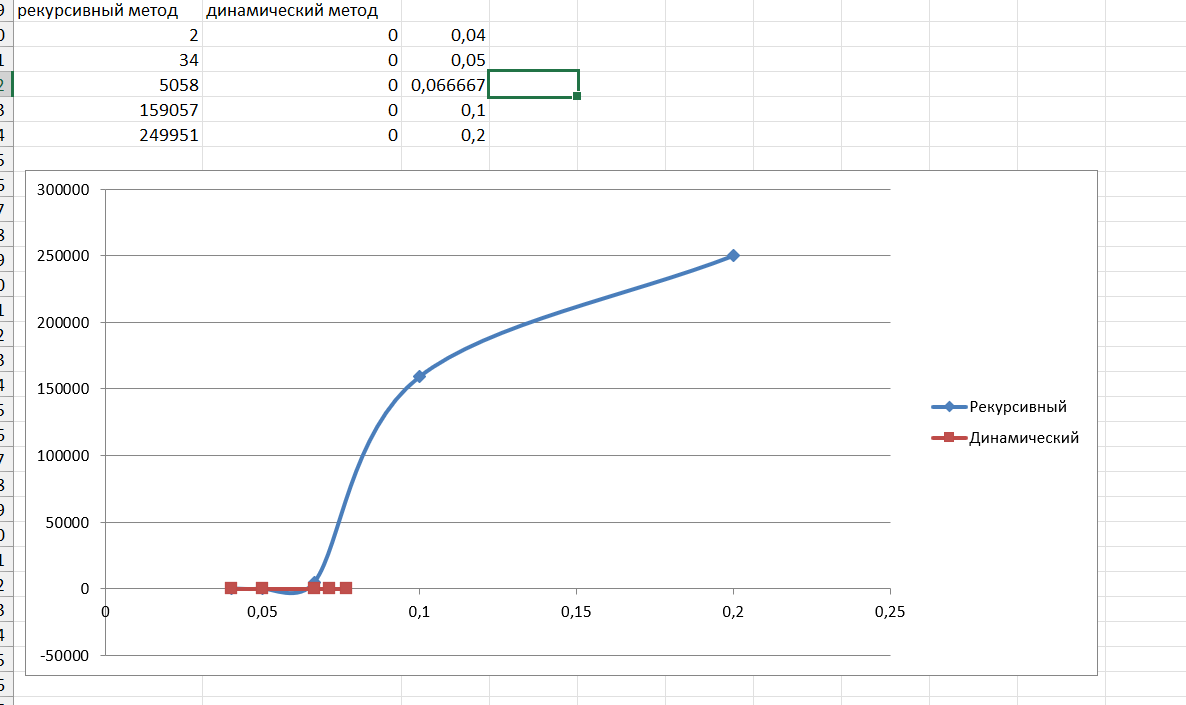


Вычисление



Результат для k = 1/5

***Задание 3.***



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

**Задание 4.** Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом) (каждый шаг алгоритма по примеру из лекции вставить в отчет).

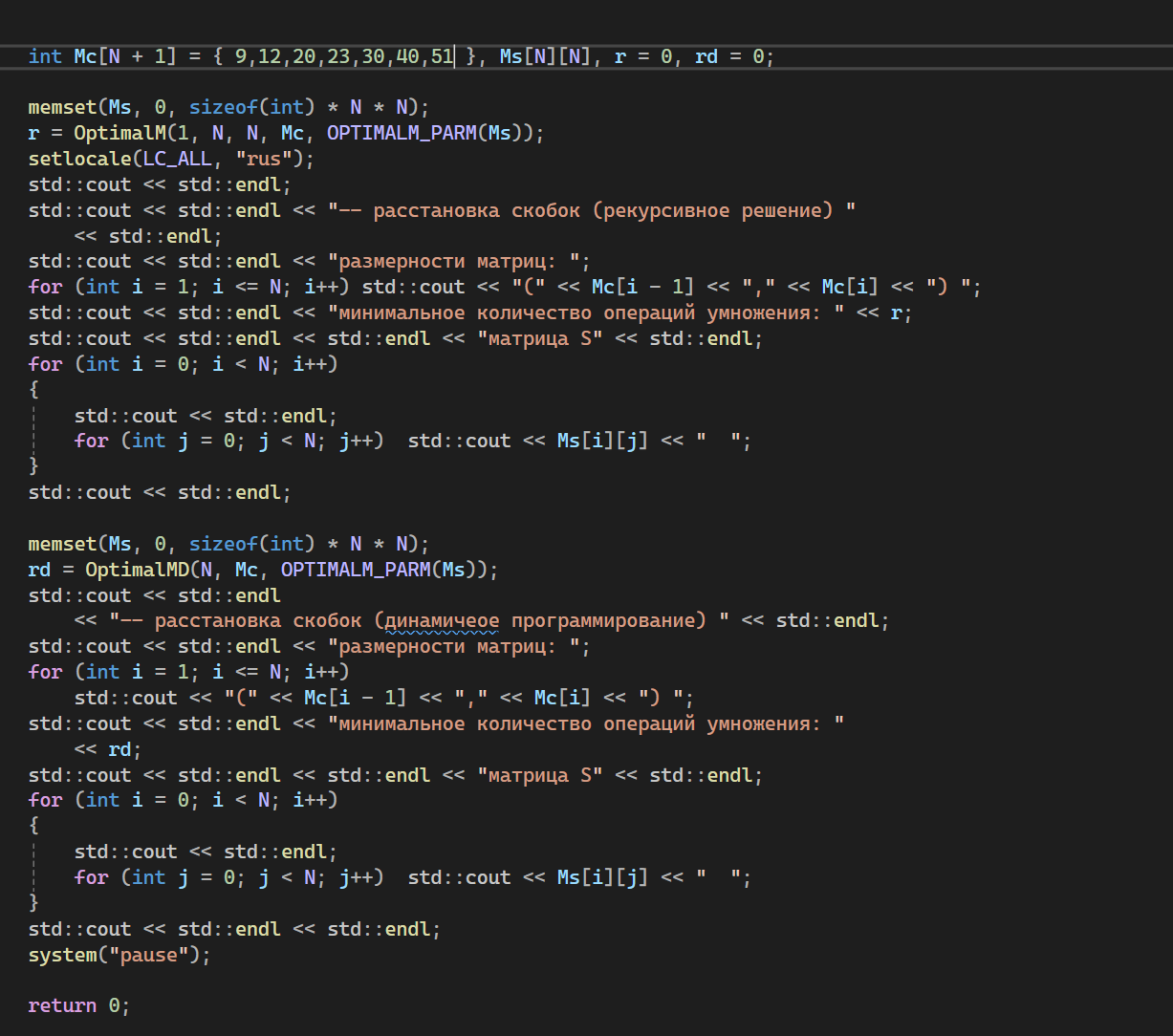
|  |  |
| --- | --- |
| Вариант 6 | |
| Сом | Домик |

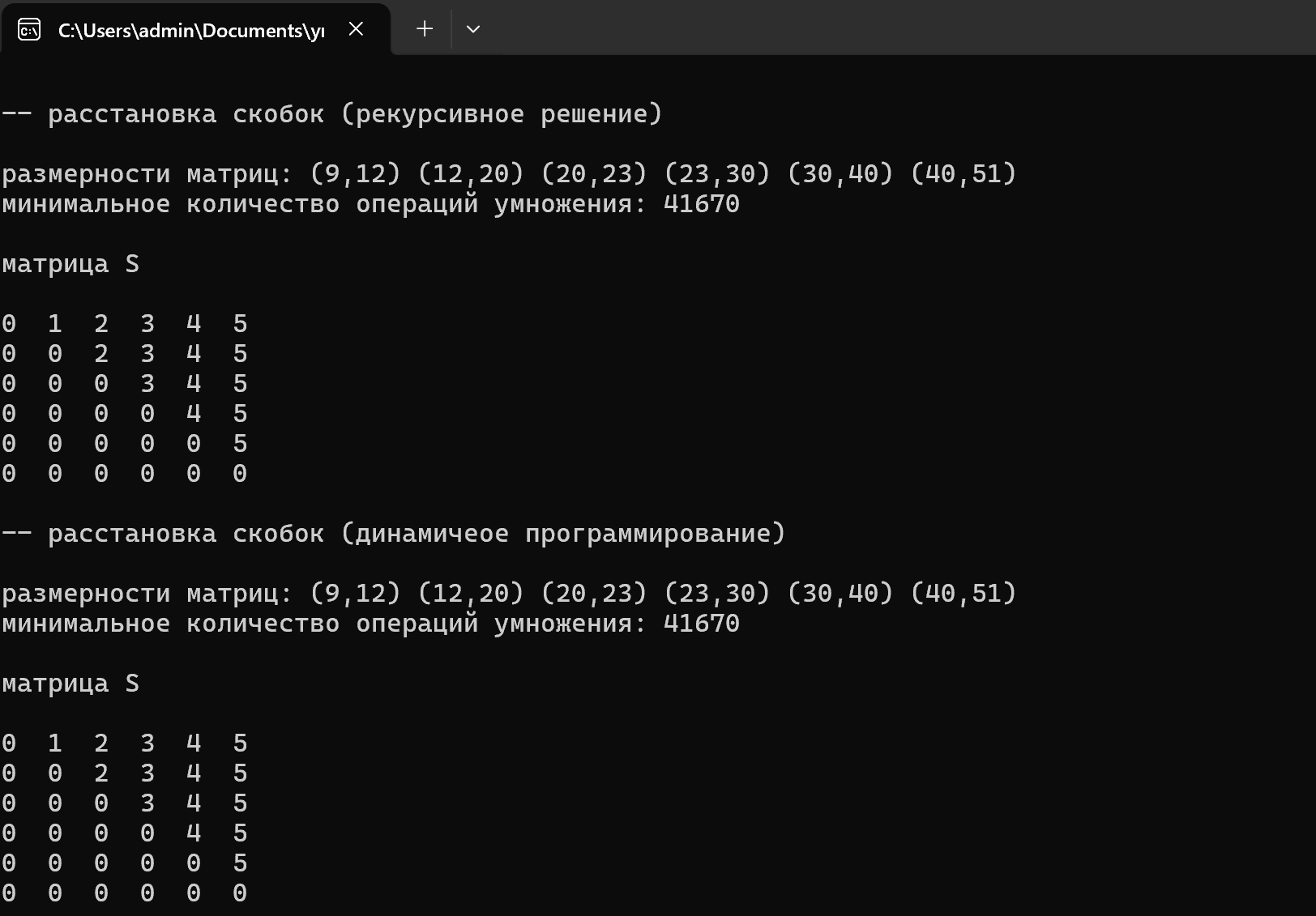
1. Сом, Домик = min(Со, Домик + 1)
   * (Сом, Доми + 1)
   * (Со, Домик + 1)
2. Со, Домик = min(С, Домик + 1)
   * (Со, Доми + 1)
   * (С, Домик + 1)
3. Сом, Доми = min(Со, Доми + 1)
   * (Сом, До + 1)
   * (Со, Доми + 1)
4. Со, Доми = min(С, Доми + 1)
   * (Со, Дон + 1)
   * (С, Доми)
5. С, Домик = min(«», Домик) = 5
   * (С, Доми + 1)
   * (», Домик) = 4
6. С, Доми = min(«», Доми) = 4
   * (С, Дон + 1)
   * (», Дон) = 3
7. Сом, Дон = min(Со, Дон + 1)
   * (Сом, «» + 1)
   * (Со, Дон)
8. Со, Дон = min(С, Дон + 1)
   * (Со, «» + 1)
   * (С, Дон)
9. С, Дон = min(«», Дон) = 3
   * (С, До + 1)
   * (», До) = 2
10. Сом, До = min(Со, До + 1)
    * (Сом, «» + 1)
    * (Со, До)
11. Со, До = min(С, До + 1)
    * (Со, «» + 1)
    * (С, До)
12. С, До = min(«», До) = 2
    * (С, Д + 1)
    * (», Д) = 1
13. Сом, Д = min(Со, Д + 1)
    * (Сом, «»)
    * (Со, «»)
14. Со, Д = min(С, Д + 1)
    * (Со, «»)
    * (С, «»)
15. С, Д = min(«», Д) = 1
    * (С, «»)
    * (», «») = 0
16. С, Д = min(2, 2, 1) = 1
17. Со, Д = min(2, 3, 2) = 2
18. Сом, Д = min(3, 4, 3) = 3
19. С, До = min(3, 2, 2) = 2
20. Со, До = min(3, 3, 1) = 1
21. Сом, До = min(2, 4, 3) = 2
22. С, Дон = min(4, 3, 3) = 3
23. Со, Дон = min(4, 2, 3) = 2
24. Сом, Дон = min(3, 3, 1) = 1
25. С, Доми = min(5, 4, 4) = 4
26. С, Домик = min(6, 5, 5) = 5
27. Со, Доми = min(5, 3, 4) = 3
28. Сом, Домик = min(5, 4, 2) = 2
29. Со, Домик = min(6, 3, 4) = 3
30. Сом, Домик = min(4, 3, 4) **= 3**

**Задание 5.**

Выполнить сравнительный анализ времени, затраченного на решение задачи о наибольшей общей подпоследовательности для двух методов решения (рекурсивное решение, динамическое программирование). Две последовательности взять с соответствии с вариантом. Построить графики зависимости времени вычисления от k. Отобразить ход решения в отчете (по примеру лекции) + код + копии экрана.

|  |
| --- |
| Вариант 6 |
| 9\*12, 12\*20, 20\*23, 23\*30, 30\*40, 40\*51 |

**

**

Принцип расстановки скобок по итоговой матрице:

Скобки расставляются по принципу «сначала внешние – затем внутренние». Имеется 6 матриц, вот их размерность:

A1=9\*12, A2=12\*20, A3=20\*23, A4=23\*30, A5=30\*40, A6=40\*51

Матрица S:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 0 | 1 | 1 | 1 | 1 | 1 |
| **2** | 0 | 0 | 2 | 3 | 4 | 5 |
| **3** | 0 | 0 | 0 | 3 | 4 | 5 |
| **4** | 0 | 0 | 0 | 0 | 4 | 5 |
| **5** | 0 | 0 | 0 | 0 | 0 | 5 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 |

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 5. Следовательно разрыв будет после 5-ой матрицы.

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

Далее берем элемент (2,5) и получаем, что он равен 4. Следовательно получаем:

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

И на последнем шаге мы возьмем элемент(2,4) и он равен 3:

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

Это выражение и есть конечное.

Полученная расстановка скобок позволяет получить минимальное количество операций умножения, равное 41670.