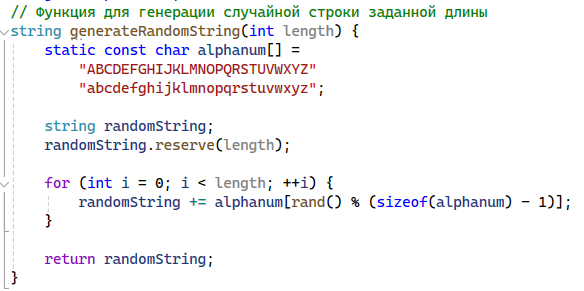
**Лабораторная работа №4. Динамическое программирование**

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

**Ход работы**

## **1. Сгенерировать две строки размером 300 и 250 символов латинского алфавита.**

Код программы:



Листинг 1. Функция генерации случайной строки

Результат:

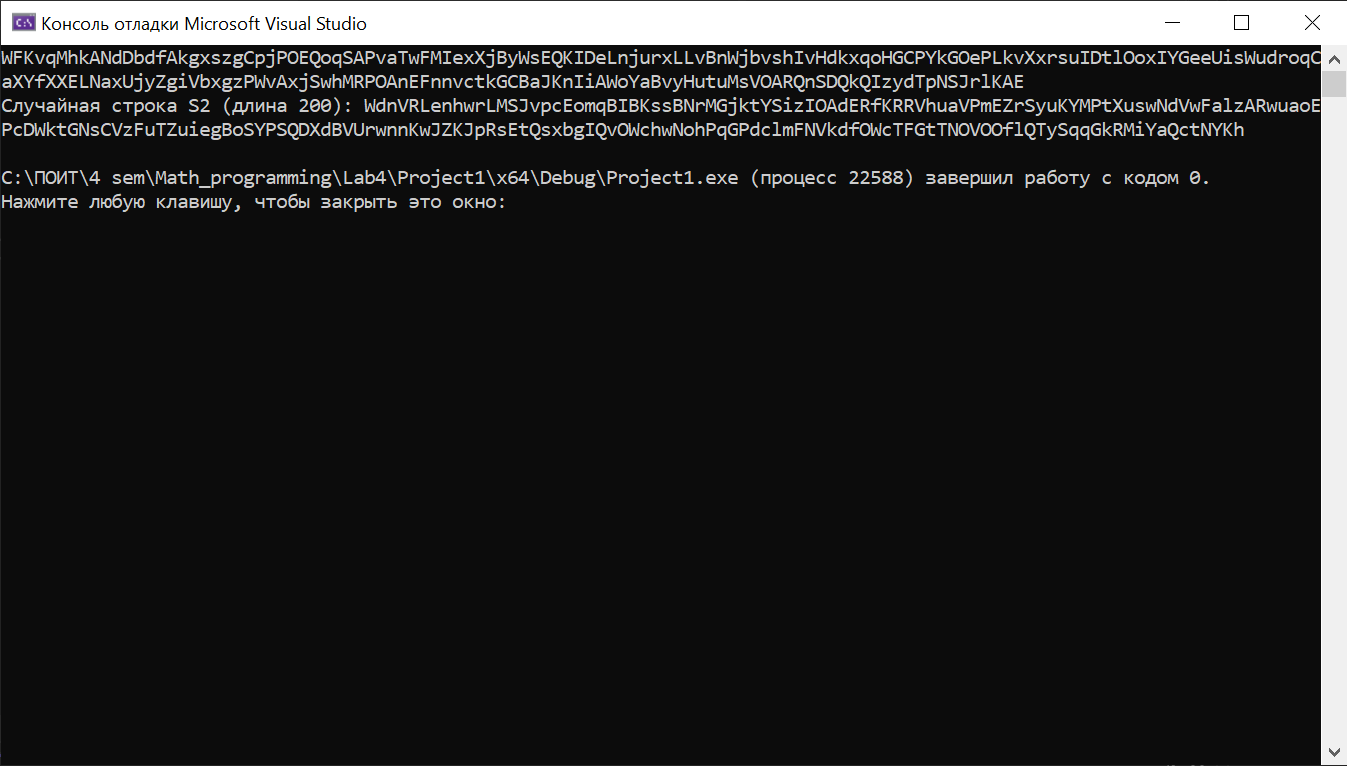
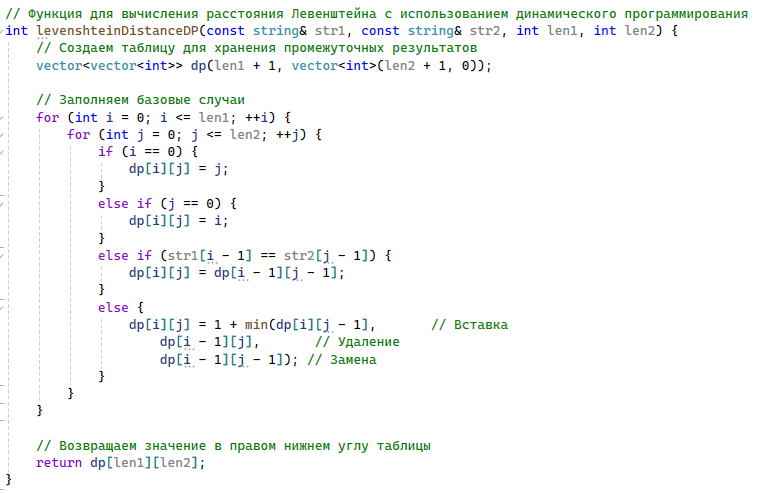


Рисунок 1. Результат генерации строк

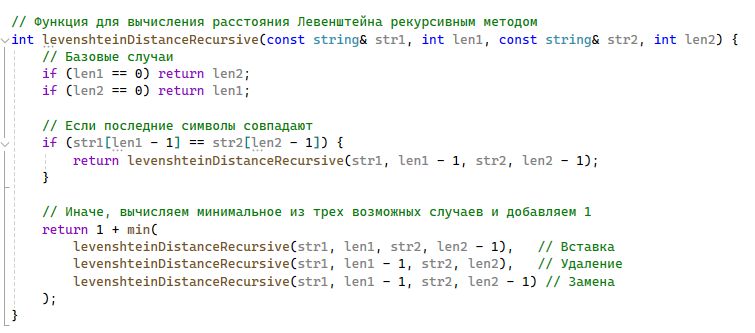
**2 Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки . (копии экрана и код вставить в отчет).**

Реализация через динамическое программирование:



Листинг 2.1. Реализация через динамическое программирование

Реализация через рекурсию:



Листинг 2.2. Реализация через рекурсию

Результат:

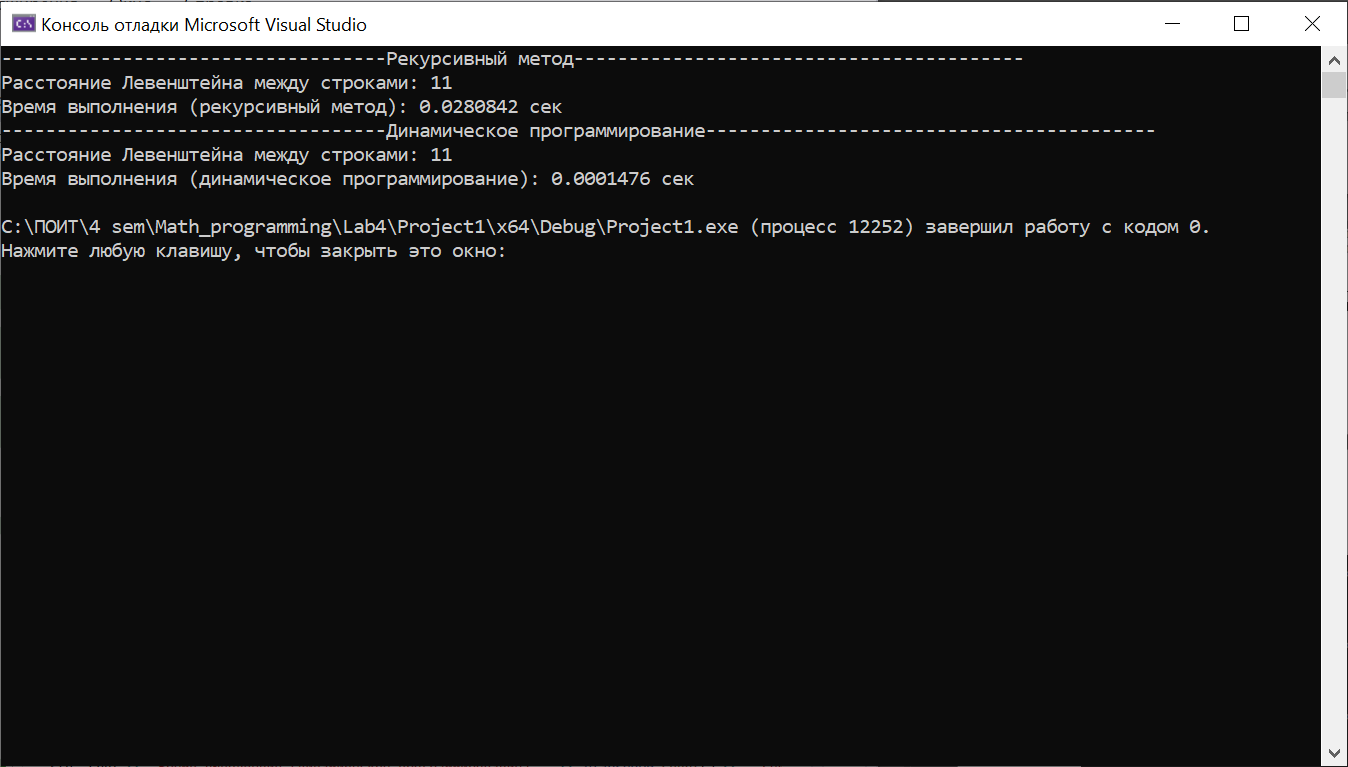


Рисунок 2. Результаты динамического программирования и рекурсии

**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).**

Метод динамического программирования значительно эффективнее, чем рекурсивный метод так, как выполняется значительно быстрее.

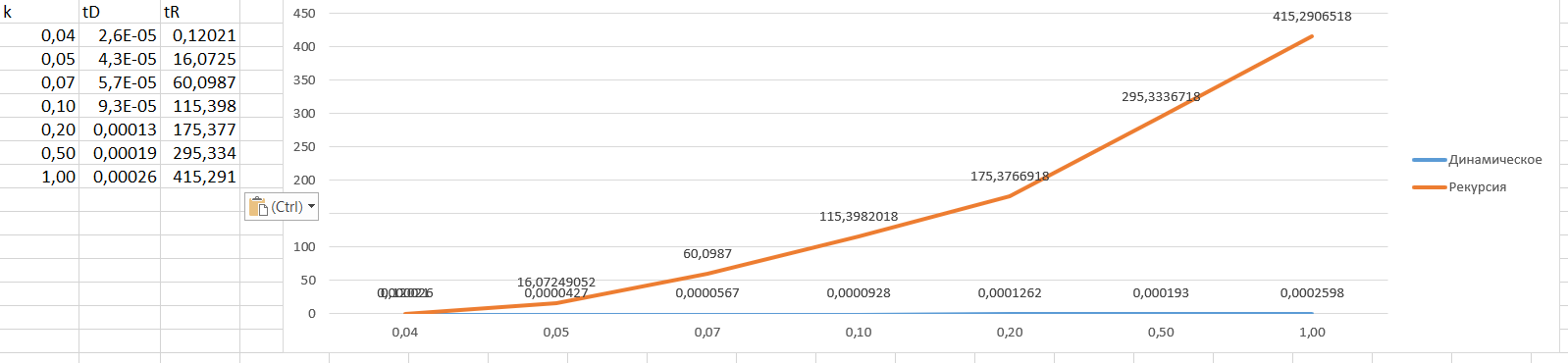


Рисунок 3. График времени выполнения рекурсивным способом

**4 Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом).**

****

Найти расстояние Левенштейна между словами «Сан» и «Сонар».

1. L(«Сан», «Сонар») = min

2. L(«Са», «Сонар») = min

3. L(«Сан», «Сона») = min

4. L(«Са», «Сона») = min

5. L(«С», «Сонар») = min

L(«», «Сонар») = 5,

L(«», «Сона») = 4

6. L(«С», «Сона») = min

L(«», «Сона») = 4,

L(«», «Сон») = 3

7. L(«Сан», «Сон») = min

8. L(«Са», «Сон») = min

9. L(«С», «Сон») = min

L(«», «Сон») = 3,

L(«», «Со») = 2,

10. L(«Сан», «Со») = min

11. L(«Са», «Со») = min

12. L(«С», «Со») = min

L(«», «Со») = 2,

L(«», «С») = 1,

13. L(«Сан», «С») = min

L(«Сан», «») = 3,

L(«Са», «») = 2

14. L(«Са», «С») = min

L(«Са», «») = 2,

15. L(«С», «С») = 0

16. L(«Са», «С») = min (1, 3, 2) = 1

17. L(«Сан», «С») = min (2, 4, 3) = 2

18. L(«С», «Со») = min (3, 1, 2) = 1

19. L(«Са», «Со») = min (2, 2, 1) = 1

20. L(«Сан», «Со») = min (2, 3, 2) = 2

21. L(«С», «Сон») = min (4, 2, 3) = 2

22. L(«Са», «Сон») = min (3, 2, 2) = 2

23. L(«Сан», «Сон») = min (3, 3, 1) = 1

24. L(«С», «Сона») = min (5, 3, 4) = 3

25. L(«С», «Сонар») = min (6, 4, 5) = 4

26. L(«Са», «Сона») = min (4, 3, 3) = 3

27. L(«Сан», «Сона») = min (4, 3, 2) = 2

28. L(«Са», «Сонар») = min (5, 4, 4) = 4

29. L(«Сан», «Сонар») = min (5, 3, 4) = 3

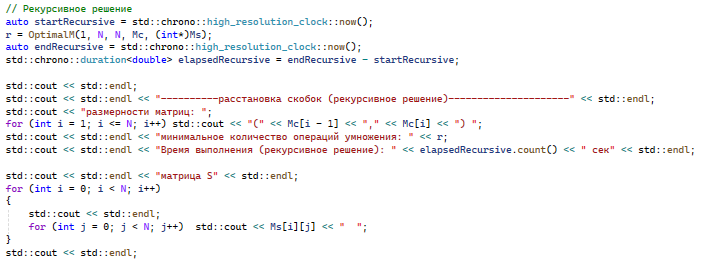
Дистанция Левенштейна для слов «Сон» и «Сонар»: 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| С | А | Н | - | - |
| С | О | Н | А | Р |

**5 Выполнить сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование).**

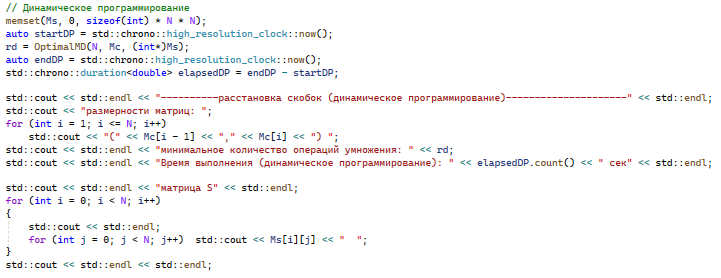


Реализация через рекурсию:



Листинг 5.1. Реализация через рекурсию

Реализация через динамическое программирование:



Листинг 5.2. Реализация через динамическое программирование

Результат:

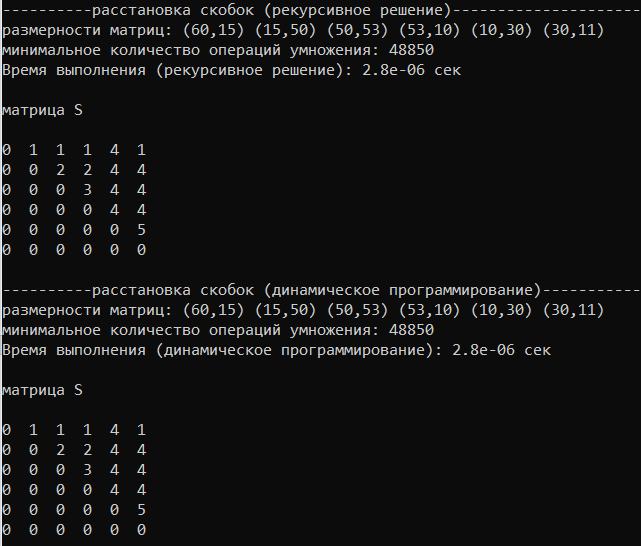


Рисунок 4. Результаты динамического программирования и рекурсии

Принцип расстановки скобок по итоговой матрице:

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

А1= 60\*15,

А2= 15\*50,

А3= 50\*53,

А4 =53\*10,

А5 =10\*30,

А6 =30\*11.

Матрица S:

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

Далее берем элемент (2,6) и получаем, что он равен 4. Следовательно получаем:

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, равное 48850.