Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Математическое программирование»

**Лабораторная работа №4**

**Тема «Динамическое программирование»**

Выполнил:

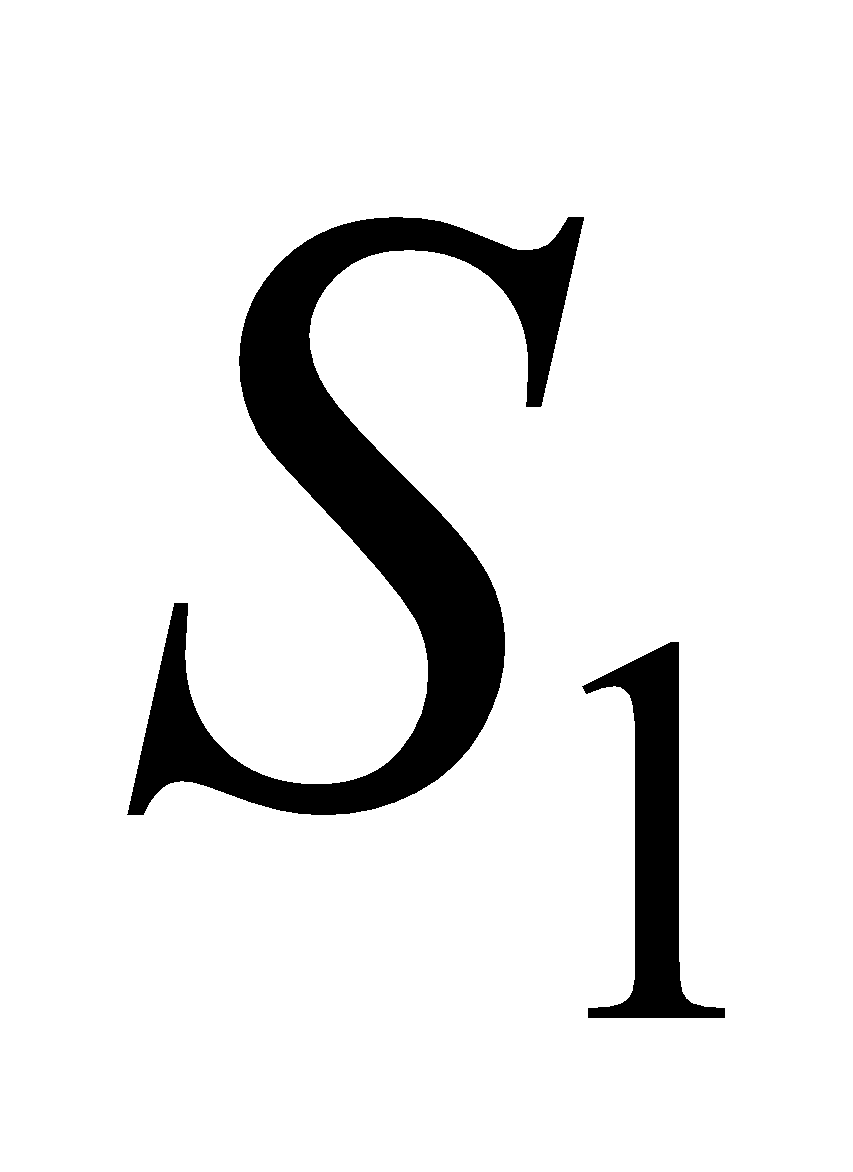
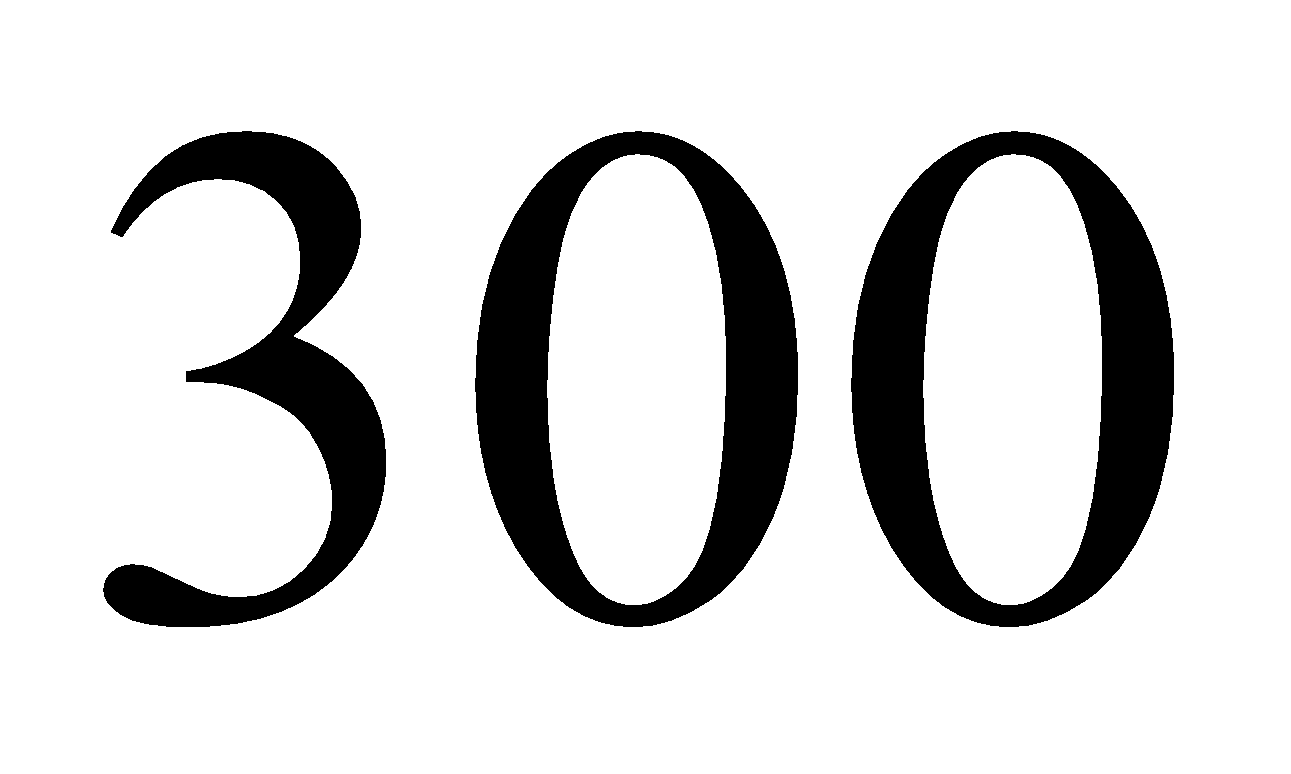
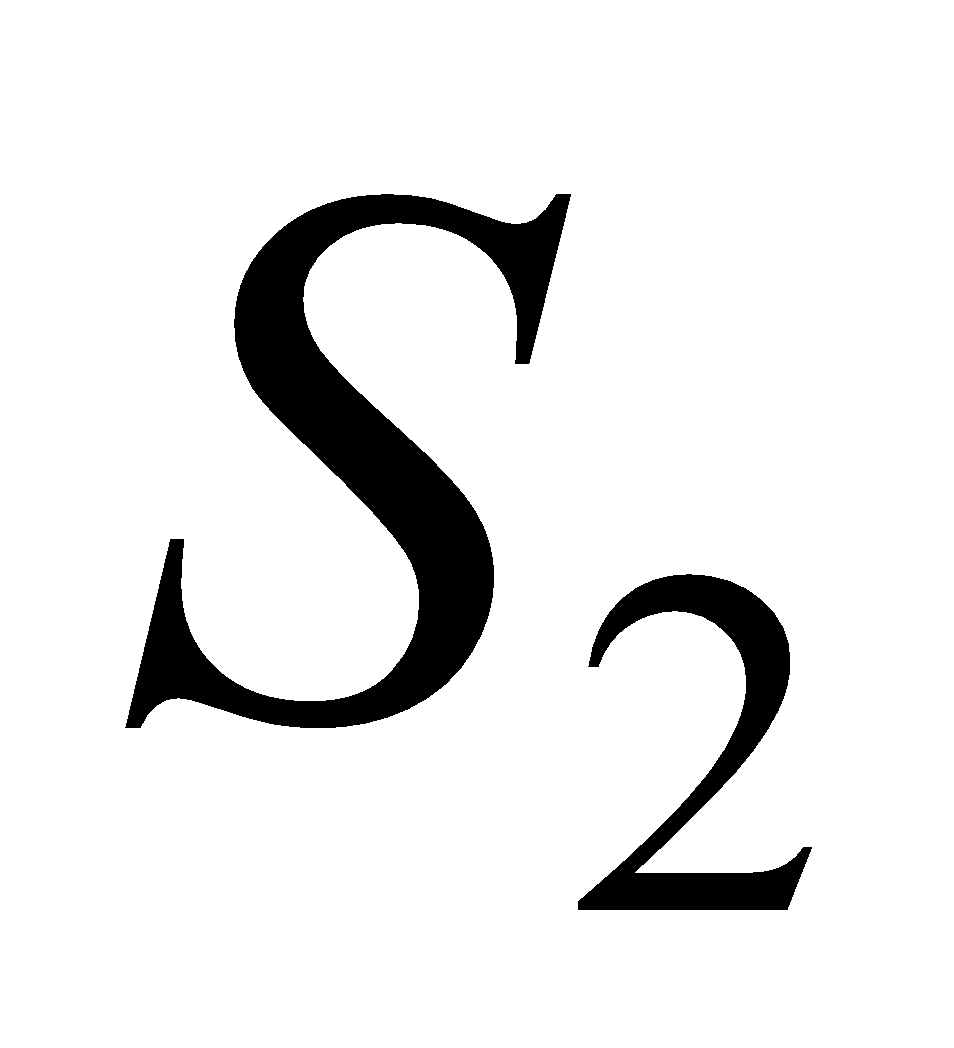
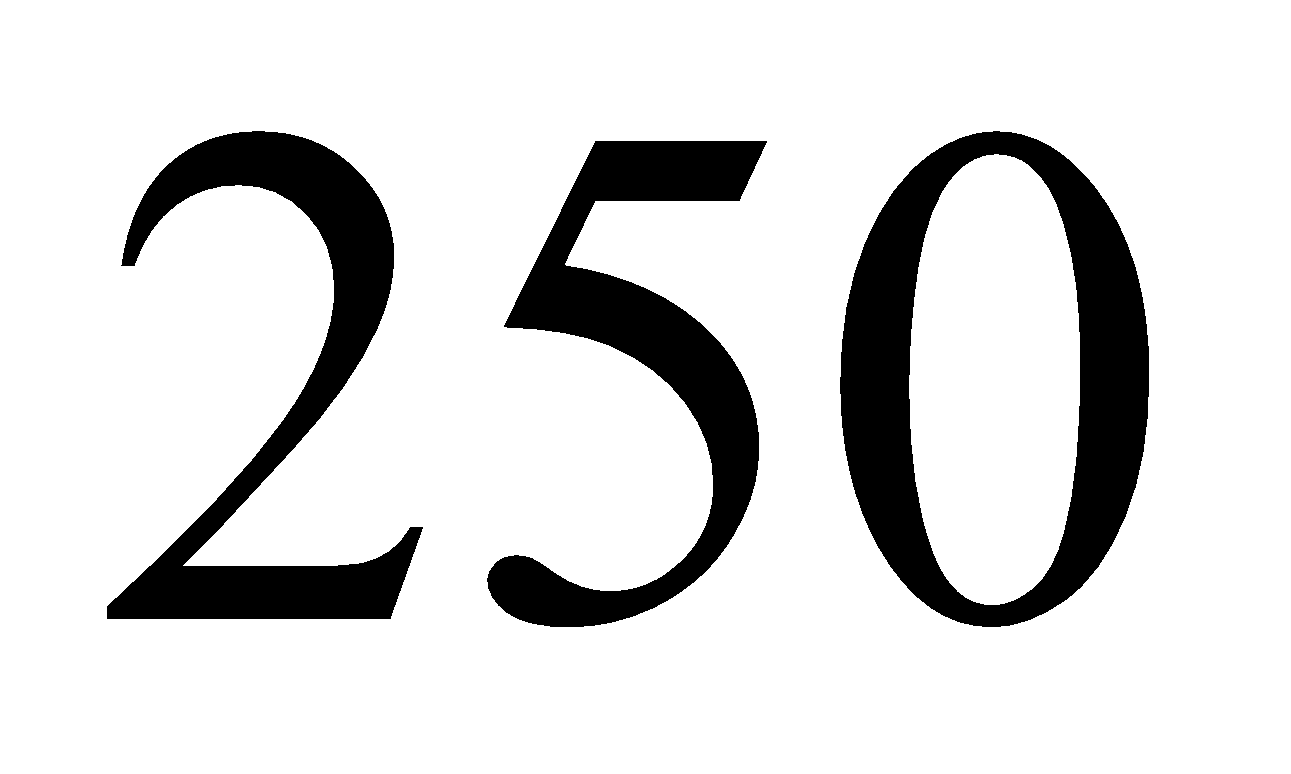
Студент 2 курса 7 группы ФИТ

Ильин Н. С.   
 Проверил:   
 Доц. Буснюк Н. Н.

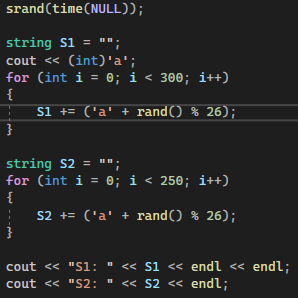
Минск 2023

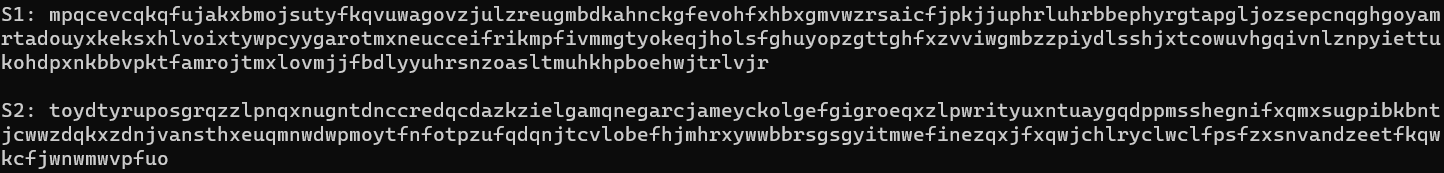
**Цель работы:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

**Задание №1**

**Условие:** на языке С++ сгенерировать случайным образом строку букв латинского алфавита  длиной  символов и длиной .

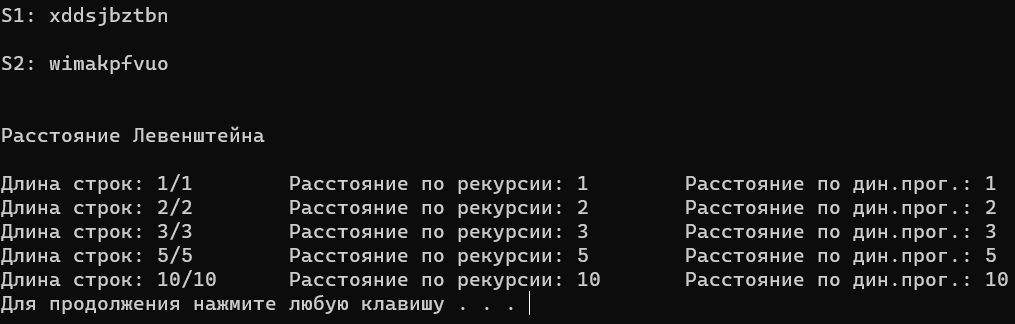
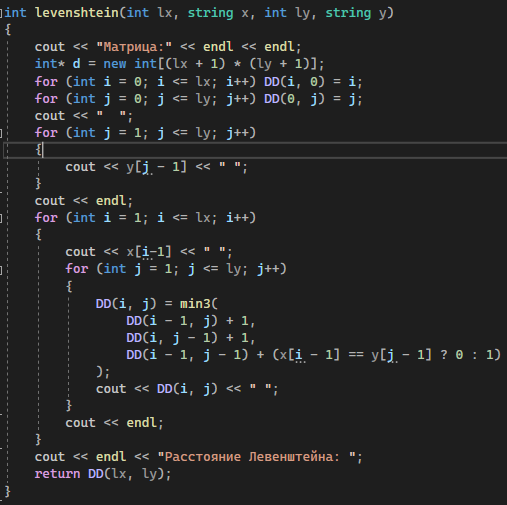
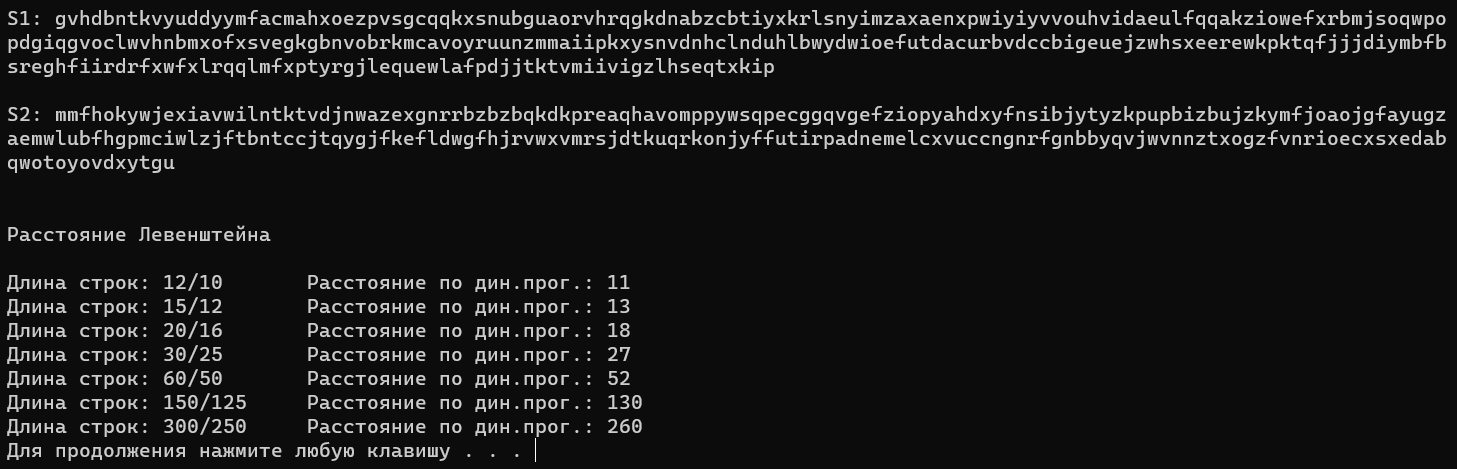
**Выполнение:** код на C++, генерирующий случайным образом букву латинского алфавита, путём добавления к первой букве ‘a’ числового значения, полученного путём остатка от деления числа, сгенерированного функцией rand() на 26 (в латинском алфавите 26 букв), и автоматического неявного преобразования его в уже другую букву латинского алфавита в соответствии с таблицей кодировки ASCII.





**Задание №2**

**Условие:** 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).

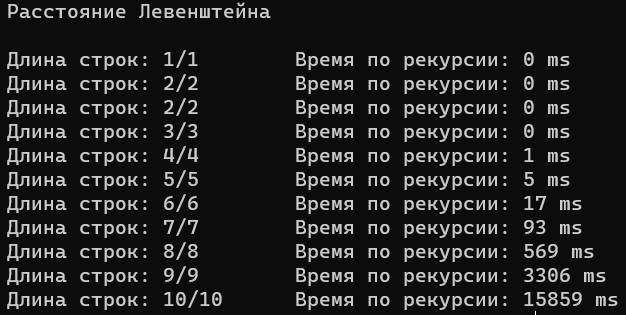
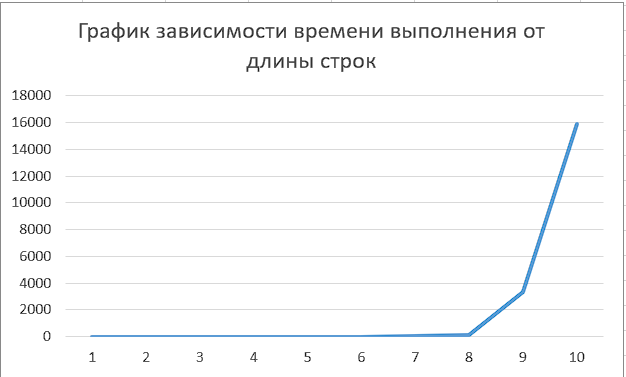
**Выполнение:** взял длину строк гораздо меньшую по сравнению с заданием №1, так как рекурсивный метод даже строчку длиной в 15 символов высчитывал очень долго (с разрешения лектора). Для метода динамического программирования использовал длины строк из задания №1, так как метод очень быстр.  
  
  
  


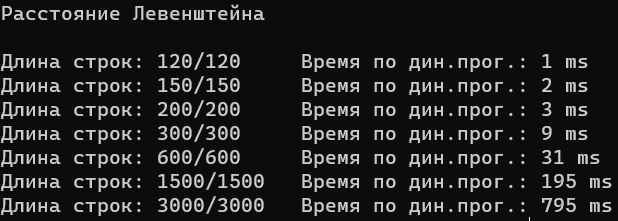
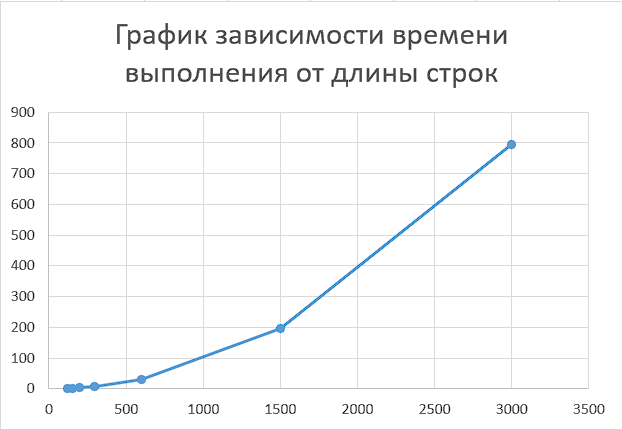
**Задание №3**

**Условие:** выполнить сравнительный анализ времени затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от длины строки (копии экрана и график вставить в отчет).

**Выполнение:** программный подсчет времени вычисления и график зависимости времени выполнения рекурсивного алгоритма и метода динамического программирования от длины строк:

Рекурсивный алгоритм:

Метод динамического программирования:  
  


**Задание №4**

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

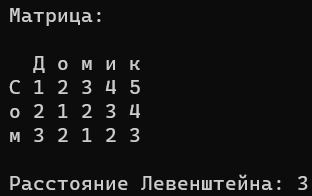
Слово 1: Сом; Слово 2: Домик

**Выполнение:** реализовал рекурсивный алгоритм вычисления дистанции Левенштейна «от руки».

1. L(“сом”, “домик”) = min  
  
2. L("со", "домик") = min  
3. L(“сом”, “доми”) = min  
  
4. L(“со”, “доми”) = min

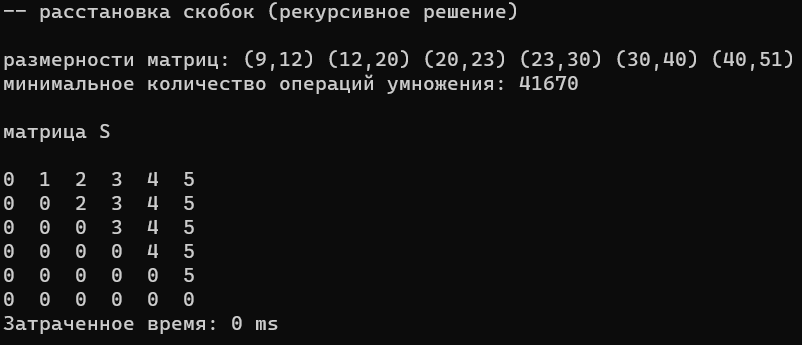
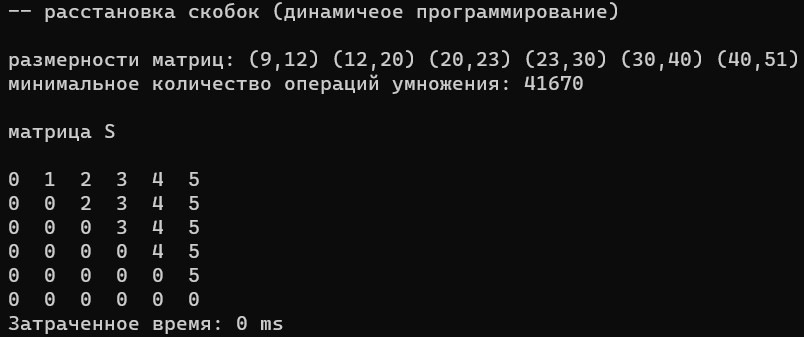
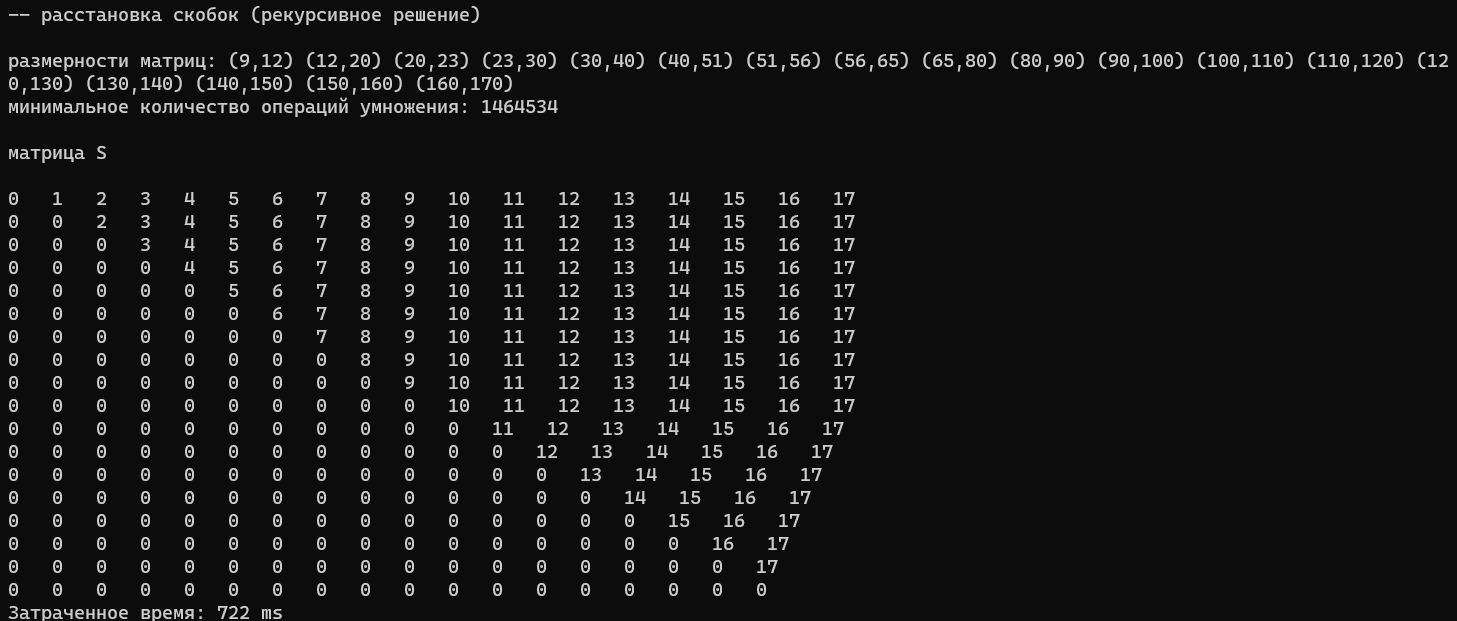
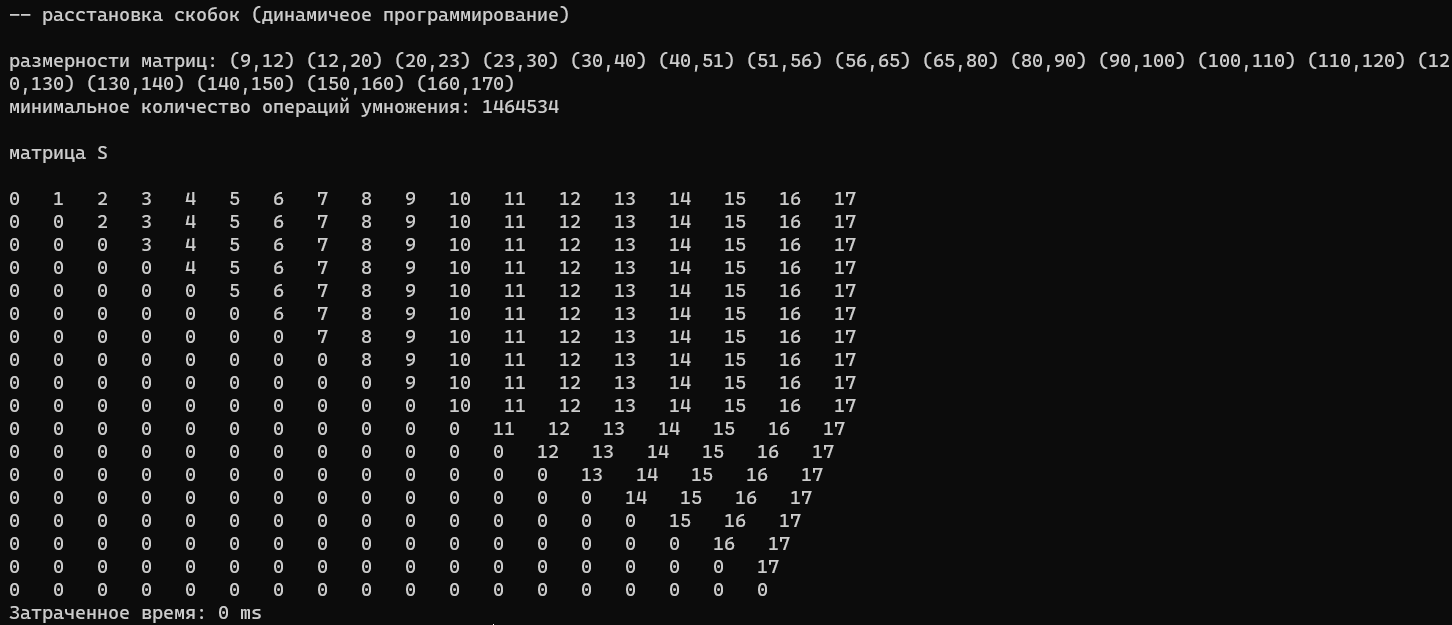
5. L(“с”, “домик”) = min  
  
L(“”, “домик”) = 5, L(“”, “доми”) = 4  
  
6. L(“с”, “доми”) = min  
  
L(“”, “доми”) = 4, L(“”, “дом”) = 3  
  
7. L(“сом”, “дом”) = min  
  
8. L(“со”, “дом”) = min  
  
9. L(“сом”, “до”) = min  
  
10. L(“сом”, “д”) = min  
  
L(“сом”, “”) = 3, L(“со”, “”) = 2  
  
11. L(“с”, “дом”) = min  
  
L(“”, “дом”) = 3, L(“”, “до”) = 2  
  
12. L(“со”, “до”) = min  
  
L(“с”, “д”) = 1  
  
13. L(“с”, “до”) = min  
  
L(“”, “до”) = 2, L(“с”, “д”) = 1, L(“”, “д”) = 1  
  
14. L(“со”, “д”) = min  
  
L(“с”, “д”) = 1, L(“со”, “”) = 2, L(“с”, “”) = 1  
  
15. L(“со”, “д”) = min(2, 3, 2) = 2  
  
16. L(“с”, “до”) = min(3, 2, 2) = 2  
  
17. L(“со”, “до”) = min(3, 3, 1) = 1  
  
18. L(“с”, “дом”) = min(4, 3, 3) = 3  
  
19. L(“сом”, “д”) = min(3, 4, 3) = 3  
  
20. L(“сом”, “до”) = min(2, 4, 3) = 2  
  
21. L(“со”, “дом”) = min(4, 2, 3) = 2  
  
22. L(“сом”, “дом”) = min(3, 3, 1) = 1  
  
23. L(“с”, “доми”) = min(5, 4, 4) = 4  
  
24. L(“с”, “домик”) = min(6, 5, 5) = 5  
  
25. L(“со”, “доми”) = min(5, 3, 4) = 3  
  
26. L(“сом”, “доми”) = min(4, 2, 3) = 2  
  
27. L(“со”, “домик”) = min(6, 4, 5) = 4  
  
28. L(“сом”, “домик”) = min(5, 3, 4) = 3

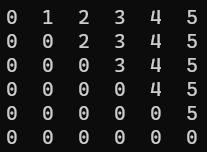
Ответ: 3.

Проверка:  
******

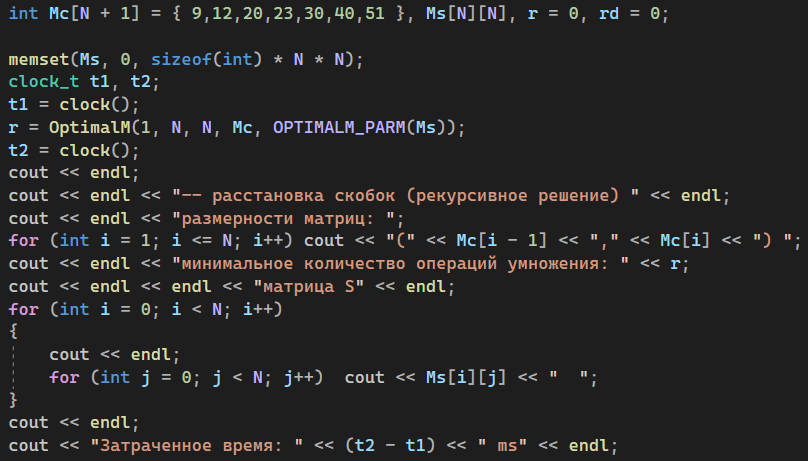
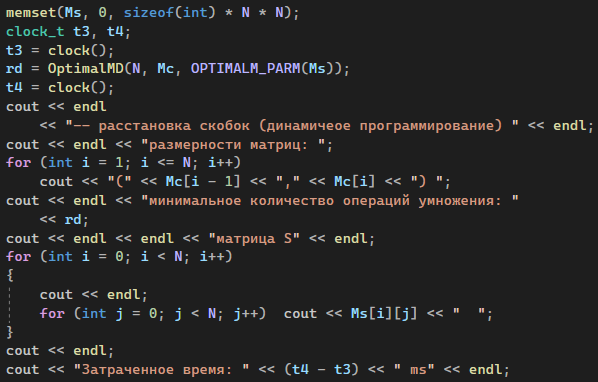
**Задание №5**

**Условие:** выполнить сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления скобок по итоговой матрице + код + копии экрана.  
Условие: 9\*12, 12\*20, 20\*23, 23\*30, 30\*40, 40\*51

**Выполнение:** выполнил сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения(рекурсивного и дин.прогр.)  
  
  
При малом количестве матриц, расчет их оптимальной расстановки выполняется довольно быстро, как для метода динамического программирования, так и для рекурсивного. Однако при большом количестве матриц, рекурсивный метод, естественно, затрачивает гораздо большее время для расчета, в то время как метод динамического программирования высчитывает всё примерно с такой же скоростью:  
  




**!!!ЭЛЕМЕНТЫ МАТРИЦЫ НАЧИНАЮТСЯ С 1!!!**

Исходные матрицы: (9,12); (12,20); (20,23); (23,30); (30,40); (40,51)  
Принцип расставления скобок по итоговой матрице:   
Скобки расставляются по принципу «сначала внешние – затем внутренние». Элемент матрицы, который имеет индекс (1, 6) – это число 5, оно означает, что точка разрыва скобок находится между 1 и 6 матрицей после 5 матрицы, что позволяет расставить скобки следующим образом:  
  
(A1 x A2 x A3 x A4 x A5) x A6  
  
Точка разрыва между 1 и 5 матрицей находится в элементе матрицы под индексом (1, 5) – это число 4, что означает, что точка разрыва скобок находится между 1 и 5 матрицей после 4 матрицы, что позволяет расставить скобки следующим образом:  
  
((A1 x A2 x A3 x A4) x A5) x A6  
  
Совершая те же действия, получается следующая последовательность перемножений:  
  
((((A1 x A2) x A3) x A4) x A5) x A6  
  
Проверка:  
  
9\*12\*20 + 9\*20\*23 + 9\*23\*30 + 9\*30\*40 + 9\*40\*51 = 41670. Ответ сошёлся.  
  
Код:  
  


**Вывод:** освоил общие принципы решения задач методом динамического программирования, сравнил полученные решения задач с рекурсивным методом. Рекурсивный метод при малом количестве данных высчитывает результат довольно быстро, однако при большом количестве резко возрастает время ожидания. Однако рекурсивный метод более прост в понимании и реализации. Метод динамического программирования заключается в разбиении задач на подзадачи и работает всегда очень быстро и гораздо быстрее рекурсивного метода. Однако этот метод более сложен в понимании и реализации.