Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

Электронный конспект по дисциплине

**Лабораторная работа № 4**

**Динамическое программирование**

Выполнила:

Студентка 2 курса 3 группы

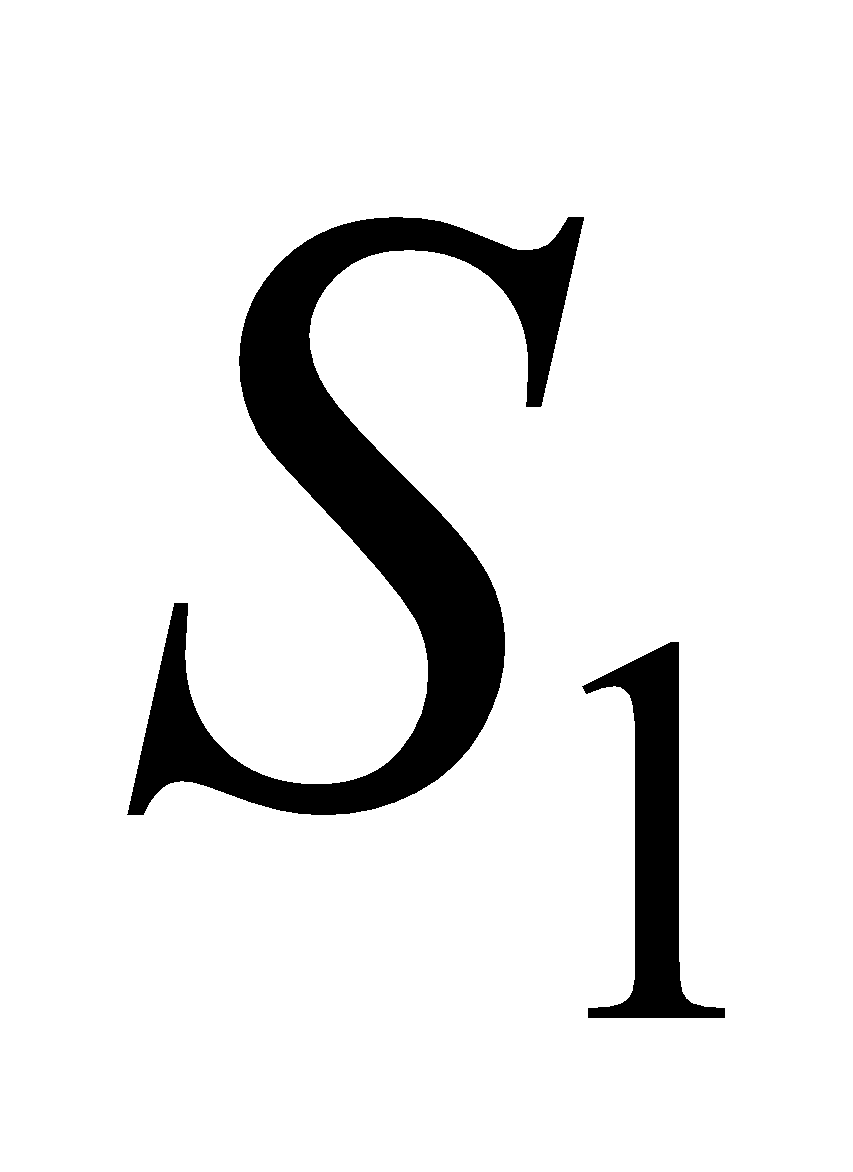
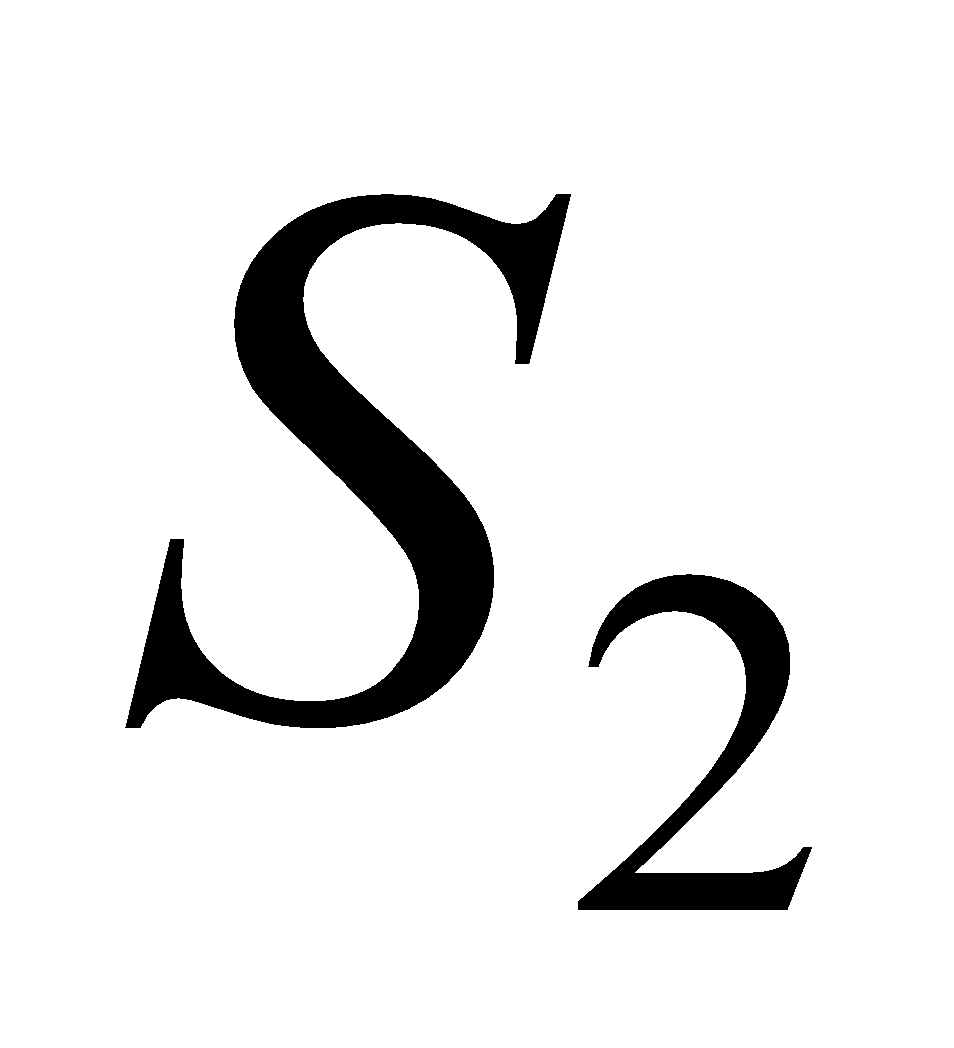
Кохнюк Александра Сергеевна

**Вариант 5**

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

Динамическое программирование используется при оптимальном планировании управляемых процессов и наиболее эффективно в случае многошаговых или многоэтапных процессов принятия решений.

К преимуществам метода динамического программирования по сравнению с “классическими” методами оптимизации относятся более высокая скорость расчетов и широкая область применимости. В частности, для него некритично требование линейности и дифференцируемости функций ***fj***(***xj***) и функция выигрыша может быть задана не в аналитическом, а в табличном виде. Другими словами, метод применим и при решении задач нелинейного и дискретного программирования.

**Задание 1.** На языке С++ сгенерировать случайным образом строку букв латинского алфавита  длиной 300 символов и длиной 250. Ниже на *рисунке 1* представлен код генерации строк и на *рисунке 2* результат.

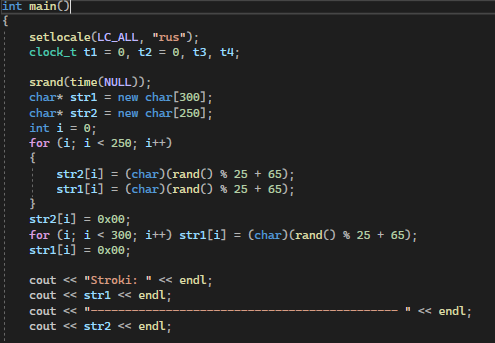


Рис. *1* – Код программы

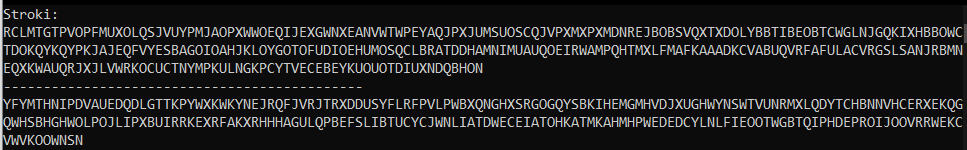


Рис. *2* – Результат выполнения программы

**Задание 2.** Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки . Ниже на *рисунках 3-5* представлен код двух способов и на результат.

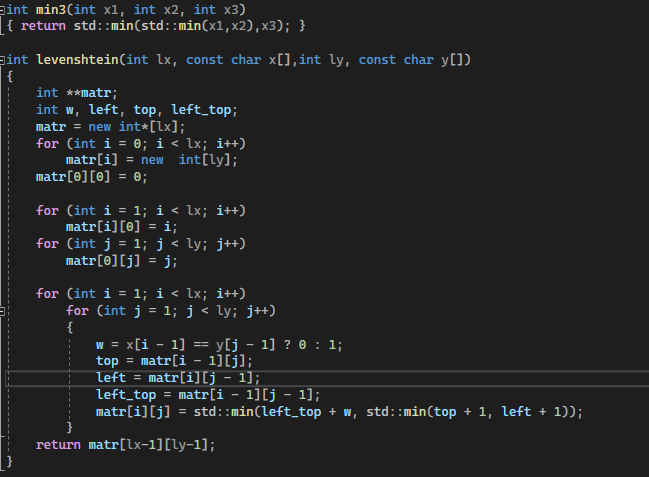
+

Рис. *3* – Код реализации программы через динамическое программирование

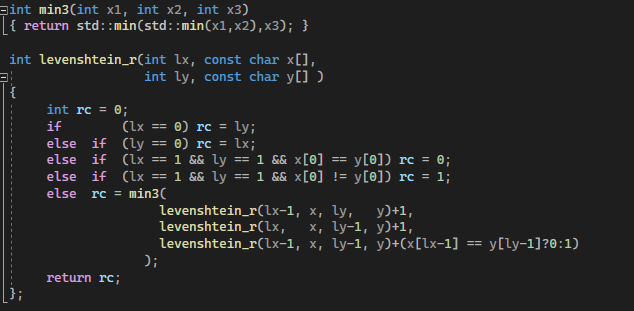


Рис. *4* – Код реализации программы рекурсивным методом

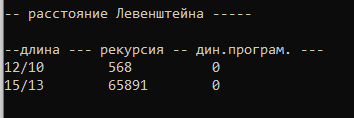


Рис. *5* – Результат выполнения программы

**Задание 3.** Выполнить сравнительный анализ времени затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от .

На графике, представленном на *рисунке 6* можно заметить, что выполнение с помощью динамического алгоритма, вычисления производятся в разы быстрее, чем с помощью рекурсивного алгоритма.

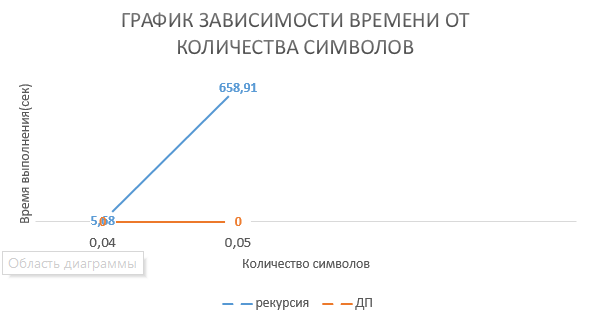
****

Рис. *6* – Анализ вычислений

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

***Рекурсивный алгоритм*** – это алгоритм, решающий задачу путем сведения ее к решению одной или нескольких таких же задач, но в сокращенном их варианте.

Первое определение рекурсивной функции относится к теории вычислимости и является синонимом понятия вычислимой функции, т. е. функции, для вычисления значения которой можно указать алгоритм.

Второе определение, которое и будет использоваться здесь, происходит из области теории программирования.

***Рекурсивная функция*** – это функция, которая вызывает саму себя.

Рекурсивный алгоритм может быть записан в виде рекурсивной функции.

Классическими примерами рекурсивных функций являются функции для вычисления факториала, чисел Фибоначчи и наибольшего общего делителя с помощью алгоритма Эвклида.

Рекурсивную функцию всегда можно преобразовать в цикл, и, наоборот любой цикл можно представить в виде рекурсивной функции.

Рекурсивная запись алгоритма, как правило, не дает выигрыша в скорости его работы. Скорее наоборот, так как вызов любой функции связан с сохранением и восстановлением контекста вызывающей функции, что является затратной по времени операцией.

|  |  |  |
| --- | --- | --- |
| 5 | Сон | Донор |

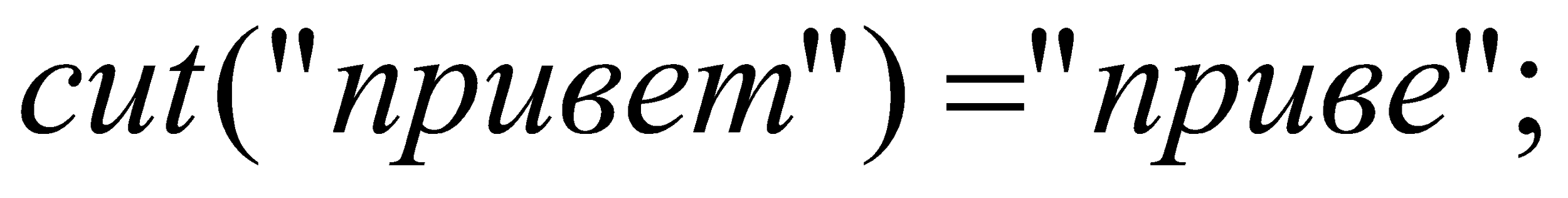
***Дистанция Левенштейна (расстояние Левенштейна, редакционное расстояние, дистанция редактирования)*** определяется между двумя строками и равна минимальному количеству операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

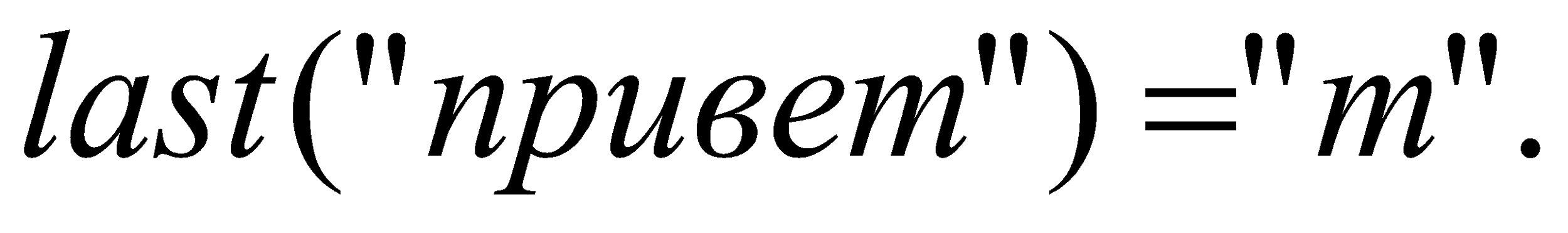
Расстояние Левенштейна активно применяется для исправления ошибок в поисковых системах, в текстовых редакторах, а также в биоинформатике.

**Вычисление дистанции Левенштейна**



 – количество символов в заданной строке. Например, 

 – заданная строка без последнего символа. Например, 

 – последний символ заданной строки. Например, 



**Дистанция Левенштейна = 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 в соответствии с вариантом. Построить графики зависимости времени вычисления от . Ниже на *рисунке 7* представлен код программы.

|  |  |  |
| --- | --- | --- |
| 5 | HRWCYUJ | WLPCAU |

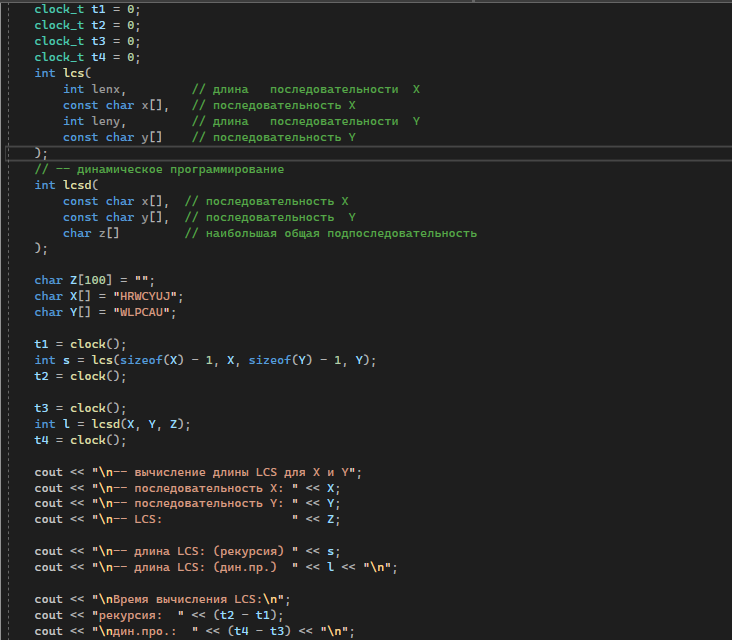


Рис. *7* – Код программы файла main.cpp

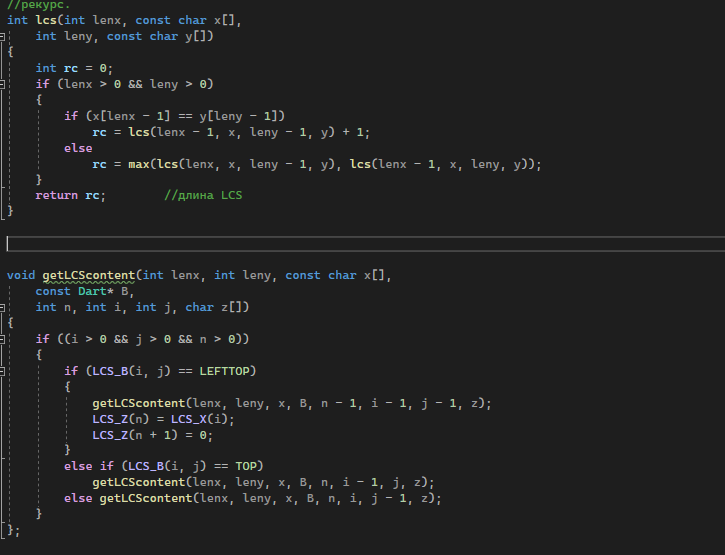


Рис. *8* – Код программы файла 5.cpp

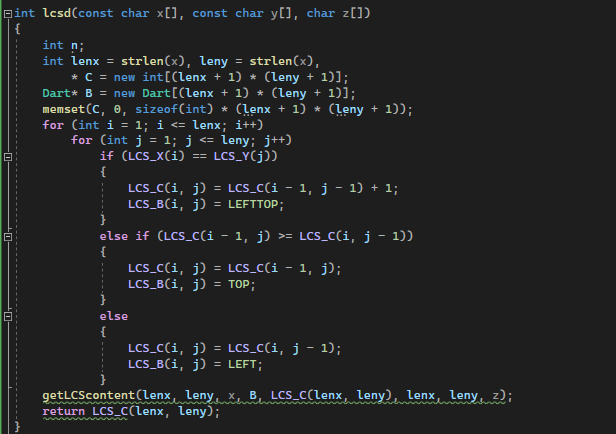


Рис. *9* – Код программы файла 5.cpp

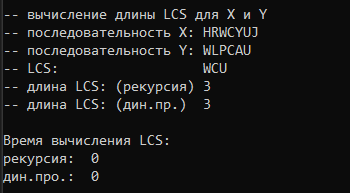


Рис. *10* – Результат выполнения программы

На графике, представленном на *рисунке 11* видно, что вычисление с помощью динамического алгоритма идет быстрее, чем с помощью рекурсивного алгоритма.

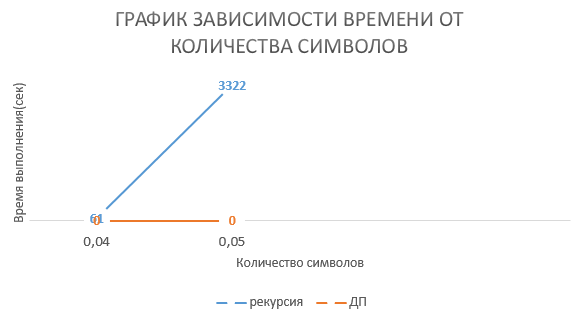


Рис. *11* – Анализ вычислений

**Вывод**: Мы освоили общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.