Белорусский государственный технологический университет

**Курс «Математическое программирование»**

**Отчёт по лабораторной работе №4**

**Динамическое программирование**

**Вариант 15**

Выполнил: Сураго Д.А.

ФИТ 2 курс 5 группа

**Лабораторная работа 4**

**ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

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

**Ход работы**

**Задание 1.** На языке С++ сгенерировать случайным образом строку букв латинского алфавита  длиной  символов и длиной .

**Решение:**

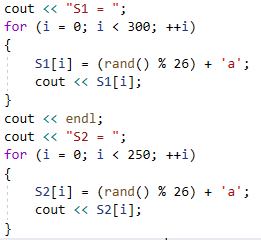
****

Рис.1-1 – Код генерации строк на языке С++

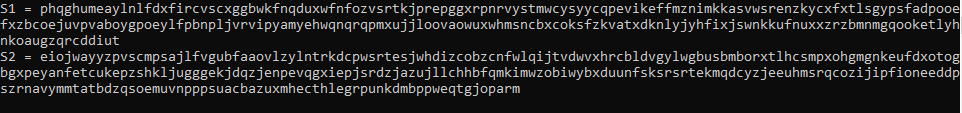


Рис. 1-2 – Результат работы

**Задание 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).

**Решение:**

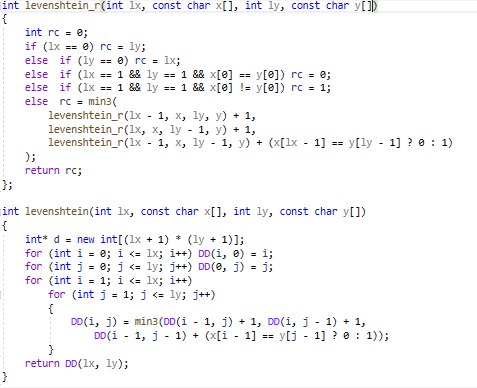


Рис. 2-1 – Код реализации

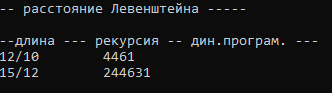


Рис. 2-2 – Результат работы рекурсивного алгоритма

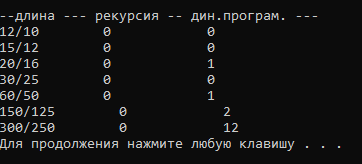


Рис. 2-3 – Результат динамического алгоритма

**Задание 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).

**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).

**Решение:**

|  |  |
| --- | --- |
| Сын | Фасон |

1. L(“сын”, “фасон”) = min

1 этап

2. L(“сы”, “фасон”) = min

3. L(“сын”, “фасо”) = min

4. L(“сы”, “фасо”) = min

2 этап

5. L(“с”, “фасон”) = min

L(“”,” фасон”) = 5

L(“”,”фасо”)=4

6. L(“с”, “фасо”) = min

L(“”,”фасо”) = 4

L(“”,”фас”)=3

7. L(“сын”, “фас”) = min

8. L(“сы”, “фас”) = min

9. L(“с”, “фас”) = min

L(“”,”фас”) = 3

L(“”,”фа”) = 2

3 этап

10. L(“сын”, “фа”) = min

11. L(“сы”, “фа”) = min

12. L(“с”, “фа”) = min

L(“”,”фа”) = 2

L(“”,”ф”) = 1

4 этап

13. L(“сын”, “ф”) = min

L(“сын”,””) = 3

L(“сы”,””) = 2

14. L(“сы”, “ф”) = min

L(“сы”,””) = 2

L(“c”,””) = 1

5 этап

16 L(“с”, ”ф”) = 1 |

17 L(“сы”, ”ф”) = min (2, 3, 2) = 2 |

18 L(“сын”, ”ф”) = min (3, 4, 3) = 3 |

19 L(“с”, “фа”) = min (3, 2, 2) = 2 |

20 L(“сы”, ”фа”) = min (3, 3, 2) = 2 |

21 L(“сын”, ”фа”) = min (3, 4, 3) = 3 |

22 L(“с”, ”фас”) = min (4, 3, 2) = 2 |

23 L(“сы”, “фас”) = min (3, 3, 3) = 3 |

24 L(“сын”, “фас”) = min (3, 4, 2) = 2 |

25 L(“с”, ”фасо”) = min(5, 3, 4) = 3 |

26 L(“с”, “фасон”) = min (6, 4, 5) = 4 |

27 L(“сы”, “фасо”) = min (4, 3, 3) = 3 |

28 L(“сын”, “фасо”) = min (4, 3, 3) = 3 |

29 L(“сы”, “фасон”) = min (5, 4, 4) = 4 |

30 L (“сын”, “фасон”) = min (5, 4, 3) = 3 |

Дистанция = 3

**Задание 5. (нет решения, есть excel)**

**Нечетные варианты**. 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 . **Отобразить ход решения в отчете**(по примеру из лекции) + код и копии экрана.

|  |  |
| --- | --- |
| MIOPLKJ | GUIOLW |

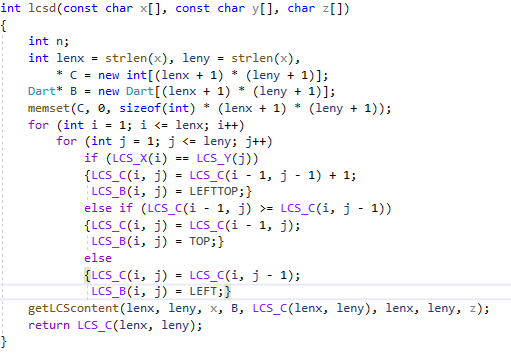


Рис. 5-1 – Динамический метод

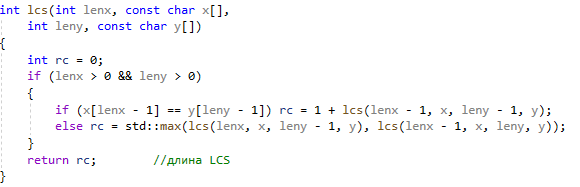


Рис. 5-1 – Рекурсивный метод

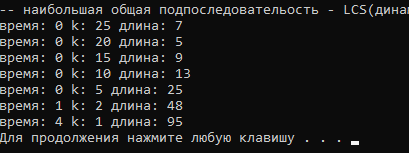


Рис. 5-1 – Результаты динамического алгоритма

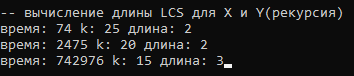


Рис. 5-1 – Результаты рекурсивного алгоритма

в

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C |  | G | U | I | O | L | W |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| I | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| O | 0 | 0 | 0 | 1 | 2 | 2 | 2 |
| P | 0 | 0 | 0 | 1 | 2 | 2 | 2 |
| L | 0 | 0 | 0 | 1 | 2 | 3 | 3 |
| K | 0 | 0 | 0 | 1 | 2 | 3 | 3 |
| J | 0 | 0 | 0 | 1 | 2 | 3 | 3 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| B |  | G | U | I | O | L | W |
|  |  |  |  |  |  |  |  |
| M |  |  |  |  |  |  |  |
| I |  |  |  |  |  |  |  |
| O |  |  |  |  |  |  |  |
| P |  |  |  |  |  |  |  |
| L |  |  |  |  |  |  |  |
| K |  |  |  |  |  |  |  |
| J |  |  |  |  |  |  |  |

Наибольшая общая подпоследовательность: IOL

Порядок заполнения таблицы:

– первая строка и первый столбец таблицы заполняются нулями;

– каждый элемент последовательно заполняется по формуле для c[i, j]:

если символы для позиции i, j совпадают, то в неё записывается значение c[i−1, j −1]+1, иначе вычисляется максимум от соседей слева и сверху.

Элемент в правом нижнем углу показывает длину наибольшей общей

подпоследовательности.

Вторая матрица заполняется следующим образом: все ячейки кроме технических строк заполняются стрелками, направленными вверх.

Если если символы для позиции i, j совпадают, то стрелка меняется на лево-вверх.

Если числовое значение от соседа слева больше, чем от соседа сверху, то стрелка меняется на лево.