Вариант 2

Тема

Разработка класса для алгоритма Вагнера-Фишера вычисления расстояния между строками. Работа с динамической памятью. Оптимизация по памяти.

1. В математике **метрика** – это функция, определяющая расстояние между элементами множества. Чтобы функция d(x, y) являлась метрикой, она должна удовлетворять:

* Неотрицательность: d(x, y) >= 0.
* Тождественность: d(x, y) = 0 тогда и только тогда, когда x = y. Симметричность: d(x, y) = d(y, x).
* Неравенство треугольника: d(x, z) <= d(x, y) + d(y, z).

**Результаты вычислений и подтверждение свойств:**

Я вычислили расстояние Левенштейна между строками. В коде main.cpp была проведена проверка свойства неотрицательности.

1. Алгоритм Вагнера-Фишера вычисляет минимальное количество операций, необходимых для преобразования одной строки в другую. \

**Инициализация:** первая строка и первый столбец таблицы заполняются числами от 0 до длины соответствующего префикса. Это количество операций, необходимых для преобразования пустой строки в префикс другой строки.

**Заполнение таблицы:** для каждой ячейки (i, j) вычисляется минимальная стоимость преобразования префикса первой строки длиной i в префикс второй строки длиной j.

**Удаление:** Стоимость преобразования префикса длиной i-1 в префикс длиной j + 1 операция удаления. Значение берется из ячейки (i-1, j).

**Вставка:** Стоимость преобразования префикса длиной i в префикс длиной j-1 + 1 операция вставки. Значение берется из ячейки (i, j-1).

**Замена/Совпадение:** Стоимость преобразования префикса длиной i-1 в префикс длиной j-1 + стоимость замены символа (0, если символы совпадают, 1, если символы разные). Значение берется из ячейки (i-1, j-1).

**Результат:** Значение в последней ячейке таблицы (нижний правый угол) представляет собой расстояние Левенштейна между двумя строками.

1. В классе WF использовался двумерный массив, чтобы хранить все

промежуточные расстояния между префиксами строк. В оптимизированной версии WFOpt я заменили двумерный массив на два одномерных массива (p1 и p2), представляющих текущую и предыдущую строки. Это снизило потребление памяти. Вместо хранения MxN элементов, я храню только 2N элементов.

1. Функция DistanceOpt реализует оптимизированный алгоритм Вагнера

Фишера. Она использует только два одномерных массива для хранения текущей и предыдущей строк матрицы расстояний, что значительно снижает потребление памяти. Алгоритм остается тем же, что и в классе WF, но вместо доступа к элементам двумерного массива, он работает с элементами одномерных массивов.

Корректность DistanceOpt подтверждается сравнением ее результата с результатом WF в main.cpp. Если результаты совпадают, это указывает на то, что оптимизированный алгоритм работает правильно.

1. Если длины строк s и t сильно отличаются (например, M >> N), можно

улучшить DistanceOpt следующим образом:

**Всегда использовать меньшую строку для столбцов (N):** Алгоритм DistanceOpt требует выделения массивов размером N. Поскольку мы используем только *две* строки, а не всю матрицу, нет никакой причины, по которой N должно соответствовать более длинной строке. Поэтому, если M>N, поменяйте строки s и t местами, и всегда используйте меньшую строку для определения N. Это позволит уменьшить объем выделяемой памяти, если одна из строк значительно длиннее другой.

Эта оптимизация позволяет еще больше снизить потребление памяти в случаях, когда длины строк значительно отличаются.