

*Государственное образовательное учреждение высшего профессионального образования*

*«Московский Государственный Технический Университет*

*имени Н.Э. Баумана»*

**ОТЧЁТ**

По лабораторной работе №1

По курсу «Тестирование и отладка ПО»

Тема **«**Расстояние Левенштейна**»**

|  |  |  |
| --- | --- | --- |
| Исполнитель: | Студенты: | Макаров Д.В.  Фадеев П.В.  Желудков А.В. |
|  | Группа: | ИУ7-72 |
| Принял: | Преподаватель: | Рогозин О.В. |
|  |  |  |
|  | Москва, 2015 |  |

**Цель работы**

Написать программу, вычисляющую расстояние Левенштейна между двумя строками, четырьмя способами:

* алгоритм Вагнера-Фишера матричная реализация;
* алгоритм Вагнера-Фишера рекурсивная реализация;
* алгоритм Вагнера-Фишера стековая реализация;
* модифицированный алгоритм Вагнера-Фишера(расстояние Дамерау-Левенштейна).

Затем провести модульное тестирование части данной программы, которая отвечает за реализацию вычислительных алгоритмов (тестирование UI проводиться не будет).

**Модули программы**

Программа состоит из 5 нетривиальных модулей, 4 из которых будут тестироваться:

1. int damerauLevensteinDistance(string source, string comp)

Эта функция реализует модифицированный алгоритм Вагнера-Фишера.

1. int levenshteinDistStack(string source, string comp)

Эта функция реализует алгоритм Вагнера-Фишера с помощью стека.

1. int recursiveLevenshteinPart(string source, string comp, int i, int j)

int levenshteinDist(string source, string comp)

Эти две функции реализуют алгоритм Вагнера-Фишера с помощью рекурсии. Но тестироваться будет только одна из них, т.к. другая является тривиальной вспомогательной функцией.

1. int levenshteinDistMatrix(string source, string comp)

Эта функция реализует алгоритм Вагнера-Фишера с помощью матрицы.

1. void btnDist\_Click(object sender, EventArgs e)

Это обработчик нажатия на кнопку, запускающую расчеты. Тестироваться он не будет.

**Классы эквивалентности**

В связи с тем, что все тестируемые модули выполняют одну и ту же задачу, для их тестирования составлен один набор классов эквивалентности, который подойдет ко всем модулям.

Ниже представлена таблица классов эквивалентности и тестов, которые им соответствуют:

|  |  |  |
| --- | --- | --- |
| Тестируемые данные | Ожидаемый результат (расстояние) | Класс эквивалентности и название теста в программе |
| Пустая строка  Пустая строка | 0 | BothStringsAreEmpty()  Поиск расстояние между двумя пустыми строками. |
| Пустая строка  abcd | 4 | FirstStringIsEmpty()  Поиск расстояния между пустой исходной строкой и непустой строкой. |
| abcd  Пустая строка | 4 | SecondStringIsEmpty()  Поиск расстояния между непустой исходной строкой и пустой строкой. |
| abcd  abcd | 0 | EqualStrings()  Поиск расстояния между одинаковыми непустыми строками. |
| abcde  abc | 2 | CharactersMissing\_AtTheEnd()  Поиск расстояния между строками, при условии, что в исходной строке больше символов, при этом все остальные символы совпадают, начиная с первого. |
| abc  abcde | 2 | CharactesSuperfluous\_AtTheEnd()  Поиск расстояния между строками, при условии, что в исходной строке меньше символов, при этом все остальные символы совпадают, начиная с первого. |
| abcde  cde | 2 | CharactersMissing\_AtTheBeginning()  Поиск расстояния между строками, при условии, что в исходной строке, больше символов, при этом все остальные символы совпадают, начиная не с первого, но до последнего. |
| cde  abcde | 2 | CharactesSuperfluous\_AtTheBeginning()  Поиск расстояния между строками, при условии, что в исходной строке, меньше символов, при этом все остальные символы совпадают, начиная не с первого, но до последнего. |
| abcdef  abef | 2 | CharactersMissing\_InTheMiddle()  Поиск расстояния между строками, при условии, что в исходной строке, больше символов, при этом во второй строке отсутствуют некоторые символы из середины исходной строки. |
| abef  abcdef | 2 | CharactesSuperfluous\_InTheMiddle()  Поиск расстояния между строками, при условии, что в исходной строке, меньше символов, при этом во второй строке несколько лишних символов, по сравнению с исходной строкой. |
| abcd  efgh | 4 | AllCharactersDoNotMatch()  Сравнение двух строк одинаковой длины в которых все символы разные. |
| abcd  efg | 4 | DifferentStringLength\_AllCharactersDoNotMatch()  Сравнение двух строк разной длины в которых все символы разные. |
| abcdef  abedcf | 2 | WrongCharactersOrder()  Сравнение двух строк одинаковой длины, причем, у одной из них, два несоседних символа поменяли местами, по сравнению с первой. |
| abcdef  abdcef | 2(Для алгоритма Вагнера-Фишера)  1(Для **модифицированного** алгоритма Вагнера-Фишера) | WrongCharactersOrder\_between\_AdjacentCharacters()  Сравнение двух строк одинаковой длины, причем, у одной из них, два соседних символа поменяли местами, по сравнению с первой.  *Примичание: другой результат этого теста для модифицированного алгоритма Вагнера-Фишера обусловлен тем, что в этом алгоритме считается не расстояние Левенштейна, а расстояние Демерау-Левенштейна, для которого операция перестановки соседних символов имеет расстояние 1* |

Программа, прошедшая тесты, перечисленные выше, является полностью протестированной в связи с тем, что рассмотрены все нетривиальные классы эквивалентности, а значит, что правильно выполняются и остальные тесты, которые принадлежат данным классам эквивалентности.