

Table of Contents

[1. Topic 2](#_Toc42882293)

[2. Authors 2](#_Toc42882294)

[3. Summary 3](#_Toc42882295)

[Goal 3](#_Toc42882296)

[Main stages in accomplishment of the project: 3](#_Toc42882297)

[Planning 3](#_Toc42882298)

[Realization 3](#_Toc42882299)

[Presentation 3](#_Toc42882300)

[Level of difficulty main problems during realization: 3](#_Toc42882301)

[Explanation of some of the code: 3](#_Toc42882302)

[Diagram: 5](#_Toc42882303)

[4. Explanation of functions 6](#_Toc42882304)



# Authors

|  |  |
| --- | --- |
| Name | Role |
| Hussein Semir Abumelih | Scrum Trainer / Developer |
| Anton Slavchev Slavov | Developer Back-End |

…..............................................................................................................................................................

# Topic

To compare different algorithms for proximity between different words - advantages, disadvantages, speed.

To implement a chosen algorithm that searches for a word in a given list and arranges the results according to its proximity to the searched word.

…..............................................................................................................................................................

# Summary

* + Goal - To implement a chosen algorithm that searches for a word in a list and sorts the results by proximity
  + Main stages in accomplishment of the project:

Planning – In this stage, given the fact that we already had a set project, we only had to plan the execution and that is what we did. We made a schedule, distributed the work that had to be done, and started working.

Realization – In this second step we started writing the code of our project and creating the presentation and documentation which explain everything in further detail.

Presentation – The final step is presenting the finished product in front of the judges.

## Level of difficulty main problems during realization:

Along the way, we had a few misreadings with .NET because it was our first time using it. We also had a few accidents with the string metrics; however, we managed to overcome everything and complete our duties.

…..............................................................................................................................................................

Comparison of string metric algorithms:

|  |  |
| --- | --- |
| COLOR-LEVEL GRADES | |
| LOW |  |
| MEDIUM |  |
| HIGH |  |

|  |  |  |
| --- | --- | --- |
| Algorithm | Speed | Precision |
| Levenshtein distance |  |  |
| Damerau–Levenshtein distance |  |  |
| Jaro-Winkler distance |  |  |

**STRING METRICS**

…..............................................................................................................................................................

Hamming distance:

The Hamming distance between two equal-length strings of symbols is the number of positions at which the corresponding symbols are different.

## Examples:

* "**karolin**" and "**kathrin**" is 3.
* "**karolin**" and "**kerstin**" is 3.
* "**kathrin**" and "**kerstin**" is 4.
* **1011101** and **1001001** is 2.
* **2173896** and **2233796** is 3.

…..............................................................................................................................................................

# Levenshtein distance:

The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

## Examples:

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

* **k**itten → **s**itten (substitution of "s" for "k")
* sitt**e**n → sitt**i**n (substitution of "i" for "e")
* sittin → sittin**g** (insertion of "g" at the end).

# Damerau-Levenshtein distance:

The Damerau–Levenshtein distance between two words is the minimum number of operations (consisting of insertions, deletions or substitutions of a single character, or transposition of two adjacent characters) required to change one word into the other.

Examples:

* "**karolin**" and "**katloin**" is 2.
* "**karolin**" and "**kerstin**" is 3.
* "**kathrin**" and "**kerstin**" is 4.
* **1011101** and **1001001** is 2.
* **2173896** and **2713796** is 2.

…..............................................................................................................................................................

# Jaro–Winkler distance:

The Jaro–Winkler distance uses a prefix scale  which gives more favorable ratings to strings that match from the beginning for a set prefix length . The lower the Jaro–Winkler distance for two strings is, the more similar the strings are.

Formula:

