Computer Science Project 5

# Overcoming obstacles

I relatively faced fewer problems with the standardizeRules function than with the determineQuality function. I initially had trouble taking each word from the document and putting it into an array of C strings. I later fixed the bug, by noticing that the program was accessing an index that is out of bounds. I also took some time to determine the logic behind checking whether a match rule has been satisfied. But soon, I was able to use certain Boolean variables to ensure that the match quality only increments once even if a match rule is satisfied multiple times in the document.

# Program Design and Pseudocodes

## standardizeRules function

This function repeatedly checks if every match rule satisfies a set of conditions. These conditions include a positive distance, only lowercase alphabetic characters and no repetition of match rules. If any of these conditions are not satisfied, the program either fixes the match rule (in the case of upper case letters), or deletes the entire match rule (in the case of non-positive distance and repetition of match rules). The pseudocode for this function is included below.

*repeatedly:*

*check if distance is non-positive*

*remove match rule if condition is true*

*decrease number of rules by 1*

*repeatedly:*

*check if character in word 1 is non-alphabetic*

*remove match rule if condition is true*

*decrease number of rules by 1*

*repeatedly:*

*check if character in word 2 is non-alphabetic*

*remove match rule if condition is true*

*decrease number of rules by 1*

*convert all letters in both word 1 and word 2 to lowercase*

*repeatedly:*

*check if word 1 in current match rule is repeated in any other match rule as word 1*

*check if word 2 in current match rule is repeated in match rule where the word 1s are similar*

*remove match rule with smaller distance if both conditions are true*

*decrease number of rules by 1*

*repeatedly:*

*check if word 1 in current match rule is repeated in any other match rule as word 2*

*check if word 2 in current match rule is repeated in the above match rule as word 1*

*remove match rule with smaller distance if both conditions are true*

*decrease number of rules by 1*

*return number of rules*

## determineQuality function

This function replicates the document as a new document, and makes changes to this new document to facilitate the program to easily identify if a match rule is satisfied by the document. The non-alphabetic and space characters are removed in this new document. The upper case letters are also converted to lower case letters. Then, the words in this new document are added to an array of C strings, which holds each word in each index. Finally, the words in this array are checked against the words in the match rules. Only if word1 and word2 is found within a specific distance, does the program increase match quality by one. The pseudocode for this function is included below.

*copy document to a new document*

*delete all non-alphabetic and non-space characters in the new document*

*convert all alphabetic characters in the new document to lower case*

*create a new array of C strings (2 dimensional array of characters)*

*repeatedly:*

*transfer current word in document to the current position in the array of C strings*

*move to next word in document and next index in the array of C strings*

*repeatedly:*

*check if word 1 matches any word in the array of C strings*

*check if word 2 matches any other word in the document that is a specific distance away from the word 1 match*

*increase match quality by 1*

*return match quality*

# Test Data

## standardizeRules function

From test case 3 onwards, the value of **nRules** and **distance** is always positive, and cases 3 – 6 inclusive assume word1 and word2 are not repeated in any other match rule.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Test Case | Sample Data | Changes to Match Rule |
| 1 | The value of **nRules** is non-positive |  | None (function returns 0) |
| 2 | The value of **distance** is non-positive | Index 0 | Match rule is deleted |
| 3 | **word1** and **word2** both only consist of lower case letters | Index 1 | None |
| 4 | **word1** and/or **word2** consist of at least one uppercase letter | Index 3 | Letters are converted to lower case |
| 5 | **word1** and/or **word2** are comprised of at-least one non-alphabetic character | Index 4 | Match rule is deleted |
| 6 | Both **word1** and **word2** are empty C strings or comprise of only space characters | Index 5, 6 | Match rule is deleted |
| 7 | **word1** and **word2** are repeated in other match rule(s) | Index 2, 7, 8 | Only the match rule with least distance is retained |

Sample Test Data

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Word 1 | Word 2 | Distance |
| 0 | fiction | land | -1 |
| 1 | deranged | robot | 3 |
| 2 | alchemy | scientist | 2 |
| 3 | Breakfast | BANANA | 5 |
| 4 | C++ | Synthesis! | 2 |
| 5 | " " | Everyday | 1 |
| 6 | Space | "" | 3 |
| 7 | scientist | alchemy | 4 |
| 8 | alchemy | scientist | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Word 1 | Word 2 | Distance |
| 0 | deranged | robot | 3 |
| 1 | breakfast | banana | 5 |
| 2 | scientist | alchemy | 4 |

Return Value = 3

After executing function

## determineQuality function

In test cases 2 - 6, the value of **nRules** is positive and the match rules are assumed to be standardized. The following match rules will be used in all following test cases.

const int MAX\_WORD\_LENGTH = 20;

char word1[4][MAX\_WORD\_LENGTH] = {deranged, breakfast, scientist, garden};

char word2[4][MAX\_WORD\_LENGTH] = {robot, banana, alchemy, garden};

int distance[4] = {3, 1, 2, 1};

int nRules = 4; // only for test cases 2 – 6

1. The value of **nRules** is non-positive

nRules = -4;

document = {“deranged robot is falling”};

*Return Value =* 0

1. The **document** C string has no characters

document = {“”};

*Return Value =* 0

1. The **document** C string has only alphabetic characters and space characters

document = {“While eating breakfast banana the alchemy scientist sent a robot to the garden”};

*Return Value =* 2

1. The **document** C string has non-alphabetic characters

document = {“The robot2655 is deranged! The alchemy scientist no.5 exclaimed”};

*Return Value =* 2

1. A single match rules is satisfied multiple times in the **document**

document = {“Robot2 is deranged, the robot is deranged, deranged it is, said the scientist”};

*Return Value =* 1

1. The words in a match rule are present in the **document** but not within the specified distance

document = {“The scientist had made bananas for breakfast”};

*Return Value =* 0

1. Both word1 and word2 are the same, but the word only appears once in the document

document = {“The garden is beautiful said the robot”};

*Return Value =* 0