Lab 11 Formal Report

Thomas Griffith

Algorithms and Data Structures

Dr. Hill

**Content:**

1. Overall Summary[page 2]
2. Exact Match one word and phrases(for only files) [page 2]
   1. description on how I solved this problem and why [page 2]
3. Misspelled one word [page 2]
   1. description on how I solved this problem and why [page 2]
   2. Pictures, drawing, or written notes that visually help understanding[page ?]
4. Misspelled phrase
   1. description on how I solved this problem and why [page ?]
   2. Pictures, drawing, or written notes that visually help understanding[page ?]
5. Sources(links) of ideas/code that I referenced that I used to help understand and help solve the problems at hand [page ?]
6. How to use my code [page ?]

**Overall Summary:**

**Overall Summary:**

Use exact match only on file names then if query has two or more words use phrase match to check match and misspelled phrase in file and their content or if query has one word search using misspelled word. Misspelled word and phrase functions both check everywhere and get the closest score. Exact only checks file name to be fast and lets misspell function check the file contents since file names are the most important than a random file content word.

**Exact Match one word and phrases(for only files):**

**Description:**

I solved this problem by only looking at the file names for both a phrase query or a word query. The reason behind this is that I made file names more important than any texts from within files so if there is a match, it is quicker than going through all the content of each file when a file is of top priority. This uses the idea of BFS since you start at the files and if is not there, you go deeper(in the files themselves/children). Look at all the file names and see if they are the same to query.

**Misspelled one word:**

**Description(short):**

To solve the misspelled word problem, I had an idea on how to tell what is close and later the closest using a scoring system. This score is first deduced by calculating every two or more letter combination of the query and a number that is the number of letters cubed. This then compares every query letter combination with the combinations of letters from word you are looking at and gets the largest similar combination to get the query combo associated number subtracted by the letters not in the combination from the word you are looking at. This is the score for every file name and content word. However with the content word you also subtract 4 so that if the file name is the same as the content word, the file name will be higher. I do this because files are much more important that an instance if a random file. Then at the end the highest scoring word will have been saved in a string int pair and is announced as the closest match

Cubing the letters were necessary because if I did not, and I subtract letters from a long word with a higher combo than a short word with a smaller combo, then it will choose the wrong one. Cubing prevents words that are not very related be the closest.

Some things that are a problem are one letter queries since they are so common that I do not check for them then also best and test are scored the same so different words are possible to be chosen as the best. I would improve my check and make the score from the query able to detect that difference in combos.

****

How I got the score and combo

****

Will get test.txt = 63, testing = 60, tess = 26, best = 26

**Misspelled phrase**

**Description:**

To solve this problem, I took my solution for a misspelled word and used bits from that to help get the closest match. I got a list of all the combos for each word in the query and the score for them. For Files, I got however many words there were in the name and got vectors for each with vectors of their combinations of them. Then I kept a list of the file words and recorded the query word location it had the highest match with and its score. Then I would go through a list of the query words and record that had the highest score and then add up all the numbers that each query word had. I did this because it allows me to be able to tell if all the words are out of order or missing some parts or even a word. Then for the content in the files, I looped through it by starting at the location of the number of words in the query and increase by one every time until you hit the end. Doing this, I can take the words that make up the same size as the query and check each phrase with one word switching every time. This is important because they don’t have to be in the same order or have words in-between and will still find the best phrase with the most combinations.





Can tell that the and end are the same so you add up the points

**Sources(links):**

Photos in here are mine

<http://www.cplusplus.com/reference/sstream/stringstream/>

I used this to help me get each individual word from one string. The whole contents of a file string\*

<https://stackoverflow.com/questions/313970/how-to-convert-stdstring-to-lower-case>

as a refresher on how to make all characters lower case in a string

**How to use my code:**

To use my code, have a directory named contents with all the files and their txt files within them that you want to check for a word. Once you have them all there, go on your terminal and make you way to the directory that views the directory contents and the .cpp files I provide then

Type make

It will prompt you for a query will loop through if you type yes and each search

Decisions on what I’m doing and why