Lab 11 Report

Ben Joye

I decided to solve this problem in a 3-step process.

* **1St Step:**
  + Search the file names for an exact match to the query.
* **2nd Step:**
  + If match wasn’t found in step 1, search the all of the text for an exact match to the query.
  + Implementation: I used std::search for both step 1 and 2. I was originally going to use the Boyer-Moore algorithm, but I decided to use std::search because it is more simple. It has a pretty long runtime O(S\*N) (S is the length of the text, and N is the length of the query), but I decided that this tradeoff was worth it because it doesn’t take noticeably long.
* **3rd Step:**
  + If match wasn’t found in step 2, find the closest non-exact match in the text using the Levenshtein Distance Algorithm.
  + Levenshtein Distance calculates and assigns a score to a substring based on how many deletions, insertions, or substitutions it takes to make them an exact match.
  + I looped through the text and calculated the levenshtein distance for all the strings, then chose the one with the lowest score as the closest match.
  + Complexity: O(n\*m) – product of both string lengths (this is called many times)

**How to Run:**

* When ran, you can enter a query to search for. After it is found, you can enter “y” or “n” to keep searching or terminate the program.
* Renamed int main(int argc, char\* argv[]) to int search()
  + Just call search() from main to run it.

**What I would do better next time:**

* Make it work on different string lengths
* Matches over spaces

***Sources:***

<https://en.cppreference.com/w/cpp/algorithm/search>

<http://www.levenshtein.net/>

<https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/>