NWEN 243 Frequency Analysis Outline of Program

David Barnett ID: 300313764

Functions

`freq_map(char *map, int *freq_table)`

Takes a `map` which is the the translation table to be used to attempt to decode the text. The frequencies of the characters is passed as a parameter `freq_table`. `freq_table` is the in format of the index represents the character `index + 'A'`. `map` is in the format such that given a character to decode that it can find the index through `c - 'A'` and that value in `map` is the 'decoded' character. The algorithm used is iterating through the `CHFREQ` array and finds the character from `freq_table` with the highest frequency, maps the pairing to `map` then sets the frequency for that character to `-1`.

Using this algorithm is constant time.

`main`

The algorithm used in the `main` function is that it iterates through all the alphabets to test, then calculates the frequency of each character in all alphabets. From the frequency tables builds the translation tables, after applies the translation tables on the correct characters that corresponds with its the alphabet.

The main reason why I choose to make multiple frequency tables over splitting the text into multiple sub-texts due to using less memory, constant 26 * keys vs text length * keys.

Testing

I tested the code with running `cat alice.txt | ./crack 1` and `cat alice.txt | ./crack 2` then running `diff` with the example output given. I also tested the code by running it with `valgrind` to check for memory leaks. During development I also used to print out the translation & frequency tables.