Assignment1: Deciphering Substitution Cipher

Description of Assignment

In this assignment, I was tasked with deciphering a substitution cipher without being provided with a cipher key. The objective was to recover the original plaintext message from the encoded text, leveraging statistical analysis and linguistic knowledge.
To achieve this, I used several techniques rooted in the structure of the English language, namely dictionary and frequency-based analysis. Below is a detailed overview of the approach I used:

1. Understanding the Cipher

A substitution cipher works by replacing each letter of the plaintext with another letter of the alphabet. The challenge in this assignment was to decipher the message without the cipher key. As such, I needed to find patterns in the cipher text that could be linked to known patterns in the English language.

2. Language Model Support

Since English is a highly structured language, I utilized linguistic resources to guide the decryption process. Specifically, I employed two key files:

- **dictionary.txt:** This file contained a list of well-known English words. It was crucial for verifying potential solutions and matching meaningful words in the deciphered text.
- **quadgrams.txt:** This file provided a list of frequencies for quadgrams—groups of four consecutive letters—that are common in English text. By analyzing the frequency distribution of these quadgrams in the ciphertext and comparing them with the typical frequency distribution found in English, I could make educated guesses about letter mappings.

3. Approach to Decryption

- **Pattern Matching:** The algorithm for deciphering the substitution cipher begins by creating a pattern for each word based on the positions of its repeated letters. For example, the word "Egg" would have the pattern 1.2.2 (where "E" is unique and "G" repeats), and "See" would have the same pattern. Next, the ciphertext words are matched to these patterns, and for a ciphered word like a##, we generate possible substitutions for each letter based on matching words. For instance, a might map to "E" or "S", and # could map to "G" or "E". To refine the possibilities, we take the intersection of the possible substitutions for the same letters across multiple words. If, after the intersection, a letter has only one possible substitution (e.g., # must map to "G"), we finalize that substitution and remove it from the lists of other letters. This process is repeated iteratively for all unsolved letters, gradually narrowing down the list of possible mappings until a nearly complete decryption key is formed. The algorithm relies on these pattern matches and intersections to incrementally solve the cipher and resolve conflicts, ultimately leading to the nearly correct decryption of the ciphertext.
- **Fitness Function:** Using frequency analysis I tried to score a sentence using a quadgram scoring process. I used quadgram.txt file to get the frequencies of the quadgrams, scoring function is defined such as we have a sentence "I AM GONNA MAKE HIM AN OFFER HE CAN'T REFUSE" , I will remove all the white spaces and the punctuations and join all the words, now, I have a rough idea let's suppose a word "REFUSE" then probability of this in a sentence can be seen as P("REFUSE")=P("REFU")*P("EFUS")*P("FUSE"), this is an unstable function so we can take log both sides. Now, the function is submission of probability of all the quadgrams possible, this can be calculated using quadgram.txt file.
  Using this fitness function, I can tell how close a key to be correct, so I used gradient descent like approach to get to the best key possible through randomly switching two words in the nearly correct key and scoring the new sentence, it the score is lower then sentence is more correct therefore the key is better. Termination condition is simple if the key doesn't change for 1500 random switches then terminate the process and return the key, the key is possible the correct key, exceptions can be there as this is a heuristic algorithm and can have a succes rate which can be very high. As large as the substituted sentence the better the heuristic will perform.