A strong method for cracking monoalphabetic substitution ciphers is Frequency Analysis. We **count the appearances of each symbol in the ciphertext to determine the frequency of each character**.

How do you break a Monoalphabetic substitution cipher?

To break a monoalphabetic substitution using a known plaintext attack, we can take advantage of the fact that **any pair of letters in the original plaintext message is replaced by a pair of letters with the same pattern**.

**Approach:**The problem can be solved based on the following observations:

1. **Frequency analysis** is one of the known ciphertext attacks. It is based on the study of the frequency of letters or groups of letters in a ciphertext. In all languages, different letters are used with different frequencies.
2. The frequency array attack is based on the observation that in an English text, not all letters occur with the same **frequency**.
3. In the given problem, the string, **T = “ETAOINSHRDLCUMWFGYPBVKJXQZ”** is used for deciphering.
4. Therefore, the idea is to find the difference between **ith** maximum occurring letter in the given string and the string **T**and then shift all the letters of the given string with that difference. The string obtained will be one of the possible decrypted strings.

Follow the steps below to solve the problem:

* Initialize a string say **T** as **“ETAOINSHRDLCUMWFGYPBVKJXQZ”.**
* [Find the frequency of each character of the string](https://www.geeksforgeeks.org/frequency-of-each-character-in-a-string-using-unordered_map-in-c/) **S**, and store it in a variable, say **freq[].**
* [Iterate over the range](https://www.geeksforgeeks.org/range-based-loop-c/)**[0, 5]** using the variable **i** and perform the following steps:
  + Find the **ith** most occurring element in the string **S** and store it in a variable, say **ch**.
  + Find the difference between the **ch** and **ith** character of the string **T** and store it in a variable, say **x**.
  + [Iterate over the characters of string](https://www.geeksforgeeks.org/iterate-over-characters-of-a-string-in-c/) **S,**and shift all characters by **x**and then push the obtained string into an array **plaintext[].**
* Finally, after the above steps, print the strings obtained in the array **plaintext[].**

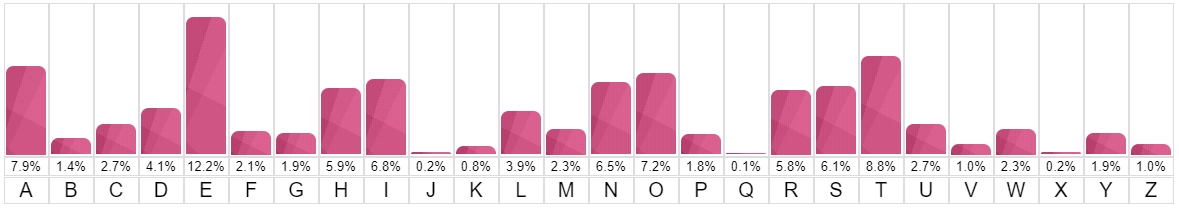
**Frequency Analysis**

Posted on [November 9, 2019](https://www.101computing.net/frequency-analysis/) Posted in [Computer Science](https://www.101computing.net/category/computer-science/), [Computing Concepts](https://www.101computing.net/category/computing-concepts/), [Cryptography](https://www.101computing.net/category/cryptography/)

In cryptography, frequency analysis is the study of the **frequency of letters** or groups of letters in a ciphertext. The method is used as an aid to breaking **substitution ciphers** (e.g. [mono-alphabetic substitution cipher](https://www.101computing.net/mono-alphabetic-substitution-cipher/), [Caesar shift cipher](https://www.101computing.net/caesar-cipher/), [Vatsyayana cipher](https://www.101computing.net/vatsyayana-cipher/" \t "_blank)).

Frequency analysis consists of **counting the occurrence of each letter** in a text. Frequency analysis is based on the fact that, in any given piece of text, certain letters and combinations of letters occur with varying frequencies. For instance, given a section of English language, letters **E, T, A and O** are the most common, while letters Z, Q and X are not as frequently used.

The following chart shows the frequency of each letter of the alphabet for the English language:



We can assume that most samples of text written in English would have a similar distribution of letters. However this is only true if the sample of text is long enough. A very short text may lead to a significantly different distribution.

When trying to decrypt a cipher text based on a substitution cipher, we can use a frequency analysis to help identify the most recurring letters in a cipher text and hence make **hypothesis** of what these letters have been encoded as (e.g. E, T, A, O, etc). This will help us decrypt some of the letters in the text. We can then recognise **patterns/words** in the partly decoded text to identify more substitutions.

You can perform a frequency analysis on the following text to try to decrypt this text step by step:

Below is the implementation of the above approach:

// C++ program for the above approach

#include <bits/stdc++.h>

using namespace std;

// Function to decrypt a monoalphabetic

// substitution cipher using the letter

// frequency attack

void printString(string S, int N)

{

// Stores final 5 possible deciphered

// plaintext

string plaintext[5];

// Store the frequency of each letter in

// cipher text

int freq[26] = { 0 };

// Stores the frequency of each letter

// in cipher text in descending order

int freqSorted[26];

// Store which alphabet is used already

int Used[26] = { 0 };

// Traverse the string S

for (int i = 0; i < N; i++) {

if (S[i] != ' ') {

freq[S[i] - 'A']++;

}

}

// Copy the frequency array

for (int i = 0; i < 26; i++) {

freqSorted[i] = freq[i];

}

// Stores the string formed from concatenating

// the english letters in the decreasing frequency

// in the english language

string T = "ETAOINSHRDLCUMWFGYPBVKJXQZ";

// Sort the array in descending order

sort(freqSorted, freqSorted + 26, greater<int>());

// Iterate over the range [0, 5]

for (int i = 0; i < 5; i++) {

int ch = -1;

// Iterate over the range [0, 26]

for (int j = 0; j < 26; j++) {

if (freqSorted[i] == freq[j] && Used[j] == 0) {

Used[j] = 1;

ch = j;

break;

}

}

if (ch == -1)

break;

// Store the numerical equivalent of letter at

// ith index of array letter\_frequency

int x = T[i] - 'A';

// Calculate the probable shift used

// in monoalphabetic cipher

x = x - ch;

// Temporary string to generate one

// plaintext at a time

string curr = "";

// Generate the probable ith plaintext

// string using the shift calculated above

for (int k = 0; k < N; k++) {

// Insert whitespaces as it is

if (S[k] == ' ') {

curr += ' ';

continue;

}

// Shift the kth letter of the

// cipher by x

int y = S[k] - 'A';

y += x;

if (y < 0)

y += 26;

if (y > 25)

y -= 26;

// Add the kth calculated/shifted

// letter to temporary string

curr += 'A' + y;

}

plaintext[i] = curr;

}

// Print the generated 5 possible plaintexts

for (int i = 0; i < 5; i++) {

cout << plaintext[i] << endl;

}

}

// Driver Code

int main()

{

// Given string

string S = "B TJNQMF NFTTBHF";

int N = S.length();

// Function Call

printString(S, N);

return 0;

}