Ricardo Alfredo Calvo Pérez

A01028889

During this project we used 3 main algorithms to complete our different tasks.

Malicious Code Detection

First we used and Naive algorithm that reads character by character of our string searching for any coincidences and comparing subsequences for matches. This algorithm has a complexity of $O(N \times M)$ where N is the number of characters in the first string and M is the number of characters in the second string.This approach is straightforward and works well for short strings, which is typically in this context given the length of transmission and code files.

Palindrome Detection

On our Palindrome detection algorithm is based on the Manacher's Algorithm designed to find the longest palindromic substring in linear time, $O(N)$where N is the length of the given string This linear complexity is achieved by using previously calculated palindrome lengths to "*skip*" redundant checks.

Longest Common Substring Detection

This problem was solved with an algorithm based on dynamic programming using a tale `dp[i][j]` represents the length of the common substring ending at positions `i`and `j` in the two transmission strings. The complexity on this algorithm is $O(N \times N)$ where N and M are the length of each string. The dynamic programming approach is chosen because it ensures accuracy by exhaustively comparing all character pairs.

Overall, the algorithms chosen were used because of the balance between their efficiency and relevance to the respective problems. Each approach allowed us to analyze the transmission data such as the malicious code, palindromic patterns, and common substrings.