Reflection Integrated Activity 1

For integrated activity number 1, 3 different functionalities were requested, hence, there were 3 different algorithms implemented for each functionality.

The first functionality consists of searching subsequences from mcode to the transmission files. For this part, a KMP algorithm was used. This algorithm utilizes a Longest Prefix Suffix array, helping the algorithm to jump positions that are not a possible match with the LPS. Because of this, the complexity is reduced from $O(m \times n)$ (naive implementation) to $O(m + n)$ (GeeksforGeeks, 2025d). It is important to mention that the KMP algorithm is especially efficient for bigger character words because it will result in the shifting of numerous positions as opposed to having a smaller character word.

For the second functionality the code must look for the longest palindrome inside mcode1 and mcode2 and output the position where it was found. In here, Manacher's Algorithm was utilized. Some methods for palindrome searching have a complexity of $O(n^2)$ or $O(n \times \log n)$, nevertheless, Manacher's Algorithm maintains a $O(n)$ by preprocessing strings to make all of them and odd-length string and using each character as a possible center for the palindrome (GeeksforGeeks, 2025d).

Lastly, an LCS problem solving algorithm was used for the third functionality. According to CITAR AL PROFE, naive implementation of this type of algorithm could result in a $O(n \times m^2)$complexity, being n the length of string number 1 and m the length of string number 2; in order to optimize this, we implemented a matrix to store the information of previous character matches between suffixes using the following formula

$$\text{If } M[i] == N[j] \Rightarrow dp[i][j][i-1][j-1] + 1$$

After this implementation, the complexity lowers to $O(m \times n)$ on both time and space.

Test Cases for part 2

For the second part, the test cases created had the functionality of testing the output of the cases where a text file has no palindromes, expecting a 1 1 result, and when it has a palindrome, expecting the final position minus the initial position being the length of the palindrome, and each position being

Test Case 1 No palindromes:

Files:

- Transmission1.txt: wertyuiorefcqgaryklcvweyugfdiopñzamqdtrejhgfiuytam
- Transimission2.txt: zxcasdqwevbndfgertmnlkjhyuiopñwertyuiopñgdazmcdjx

Results

Expected results Transmission1: 1 1

Results: 1 1

Expected results Transmission2: 1 1

Results: 1 1

Test Case 2 Palindromes:

Files:

- Transmission1.txt: calkiuxfdqsxbazyunmakqncawqpwprlfkowvlegknxbhhwmajjhlqqpimqymoczqnoilmjybd nyrojjgkibkpkolhbuknpnlr**jikagdcqffchdjtzmkmztjdhcffqcdgakij**axa
- Transmission2.txt: gixwjdlpgogmocqolikvcibybrvfaywtypttsyldwrxndjhgfkfkqscevmjhpxphjmvecsqkfkfghjd nxrwdlysttpythnwp**btysedkrapuecptfvlruqmtuhrlizzuccuzzilrhutmqurlvftpceuparkde sytb**z

Results

Expected results Transmission1: 99 133

Results: 99 133

Expected results Transmission2: 97 160

Results: 97 160

References

GeeksforGeeks. (2025d, August 27). *KMP algorithm for pattern searching*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/kmp-algorithm-for-pattern-searching/

GeeksforGeeks. (2025d, July 30). *Manacher's Algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/manachers-algorithm-linear-time-longest-palindromic-substring-part-1/