Actividad Integradora
Diego Velázquez Moreno
A01632240
Viernes 17 de septiembre

Actividad Integradora

Dentro de la actividad integradora se expone que en el momento que se transmite información de un dispositivo a otro existe una transmisión sucesiva de bits. Sabemos que actualmente la comunicación entre dispositivos es indispensable para la vida moderna que llevamos y existen casos en donde personas con malas intenciones interceptan estas transmisiones de información para modificarlas y tener ventaja de los dispositivos de otras personas. Nuestro reto es encontrar la forma en que podamos combatir estos casos con algoritmos apropiados.

Primeramente, debíamos analizar un archivo de transmisión con un archivo *mcode* para reconocer si existe algún tipo de patrón que podría poner en riesgo la transmisión. En este caso se utilizó el algoritmo KMP, el cual es muy útil al momento de buscar patrones y no tener que pasar por todos los elementos. Cuando se utiliza el KMP se pueden saltar los espacios que por obviedad no podrían coincidir con el patrón que se está buscando. El algoritmo que se utilizó tiene una complejidad de O(m) por lo que le da una gran importancia sobre otras formas de solucionar el problema presentado.

Después de investigar si existiese algún patrón que pudiera poner en riesgo el archivo de transmisión, buscamos si dentro del archivo de transmisión había la posibilidad de que tuviera algún código espejado. Con el problema presentado la primera solución que se nos vino a la mente fue en implementar el algoritmo de Manacher. Éste es indispensable para buscar palíndromos debido a que tiene una complejidad de O(n), aunque en el peor de los casos podría ser de O(n^2). De igual manera, sigue siendo muy útil contra otros algoritmos que conocemos.

Para finalizar buscamos entre dos archivos de transmisión el patrón más largo que se presente. En este caso tuvimos bastantes dificultades para solucionarlo, debido a que no conocíamos algún algoritmo en específico que fuera muy útil para el problema. Sin embargo, logramos que la complejidad fuera de O(m*n). Consideramos que el algoritmo sigue siendo aceptable debido a que podríamos haber obtenido una complejidad más alta. En este algoritmo íbamos guardando todos los *substrings* más largos para finalmente comparar el *substring* mayor.