Bruno Marquez A00834415

Para esta situación problema usamos 3 funciones principales para cada parte:

Parte 1:

‘contains: Esta función utiliza el método find para verificar si un patrón está contenido en un texto. Retorna un booleano que indica su presencia y la posición.

La complejidad es **O(N*M)**, donde NN es la longitud del texto y M la del patrón.

Parte 2:

‘findlongestpalindrome’: Itera sobre cada carácter, tratándolo como el centro de un palíndromo de longitud impar y se expande hacia afuera mientras los caracteres coincidan

La complejidad es **$O(N^2)$** para un texto de longitud NN

Parte 3:

‘longestcommonsubstring’: Se utiliza programación dinámica. Una tabla 2D dp registra la longitud del substring común que termina en los índices i (texto1) y j (texto2). Si los caracteres coinciden, dp[i][j] = dp[i-1][j-1] + 1; de lo contrario, es 0. El valor máximo en la tabla representa el substring común más largo.

La complejidad es **O(M*N)**, donde M y N son las longitudes de los dos textos.

Si bien había varios algoritmos para resolver el problema, a consideración personal estos fueron los mas sencillos de realizar.