Primer Proyecto de DAA: Procrastinación++

Javier A. Oramas López
Eduardo García Maleta

April 3, 2023

1 Descripción del problema

El problema pide, dada una cadena de Caracteres S, y una cadena T Contar todas las cadenas A que tienen a T como prefijo y que se forman a partir de el siguiente procedimiento:

Comenzamos con $A = ''$; Se toma el primer caracter de S, este caractrer se puede colocar al inicio o al final de la cadena A. Se repite este proceso hasta que no queden Caracteres en S.

1.1 Enfoque Backtracking

La solución más ingenua que se puede generar consiste en simular el proceso descrito anteriormente, Se toma cada caracter de S y se inserta primero delante de la cadena A, se llama recursivamente al método con la cadena nueva. Una vez termina el proceso, se verifica si la cadena que se formó contiene a T como prefijo, si es así se cuenta una solución. Una vez se retorna, se elimina el caracter insertado al inicio de la cadena y se coloca al final, análogamente se recorre todo el camino hasta el final.

La correctitud de este algoritmo se infiere de la definición del problema, dado que está simulando lo descrito el enunciado.

La complejidad de estee algorirtmo es $O(2^|S|)$ por cada caracter de S se revisatodo el árbol derivado de colocarlo al principio y luego al final.

2 Enfoque ??

Para pasar a explicar este enfoque es necesario realizar primero unas observaciones.

Existen solo dos formas de generar la cadena:

1. Que aparezca T en S de forma consecutiva
2. Formar T al principio de la nueva cadena

El segundo caso solo es posible si se colocan los caracteres que forman T al principio de la cadena y los que no, al final.

Con esto claro, vamos a pasar a procesar la cadena, lo rimero que nos va a interesar es tomar S al revés

Por cada prefijo de T se recorre S y se guarda cuantas formas hay de generar el prefijo X_i de T para cada posición de S. Esto es: Si se tienen las cadenas $S =$ "$abcddacbbccd$" y $T =$ "$abcd$" Para X_1 ('a') se guarda en un array la cantidad de formas de generar ese prefijo para cada posición de S;

Esto quedaría: $dp[1] = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2]$

Con esto se puede entonces generar $dp[2]$ buscando entonces la cantidad de formas de construir "ab" Como ya se tiene la cantidad de cadenas que se pueden generar para el prefijo de tamaño anterior de la siguiente forma:

Para cada posición de S, si el caracter no es el que estoy buscando (el nuevo caracter del prefijo) se suma la cuenta actual con 0, dado que no se formaron nuevas cadenas en esa posición

En caso de que si sea el caracter deseado, se suma la cuenta actual más la cantidad de formas de generar el prefijo anterior en esta posición.

En este caso por cada caracter de T se recorre toda la cadena S, dejando una complejidad de $O(|T||S|)$

S = "abcddacbbccd" T = "abcd"
'a' → 1 1 1 1 1 2 2 2 2 2 2 2
'ab' → 0 1 1 1 1 1 1 3 5 5 5 5
'abc' → 0 0 1 1 1 1 2 2 2 7 12 12
'abcd' → 0 0 0 1 2 2 2 2 2 2 2 14

3 Results

4 Conclusion