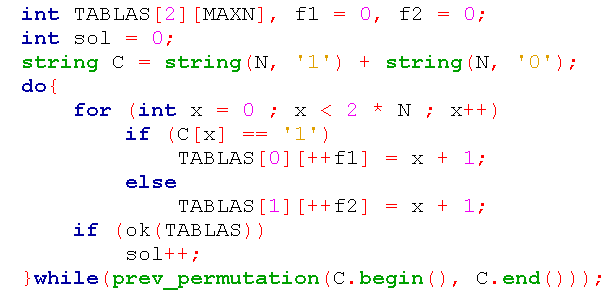
Abordemos el problema por Fuerza Bruta o Búsqueda Exhaustiva primeramente:

Para un N dado sea S = {1, 2, 3, …, 2N}. Notemos que si escogemos N números de S para la primera fila entonces los N restantes quedan para la segunda y ambas filas están en orden creciente, solo faltaría chequear que las columnas lo estén así como las restricciones de los conjuntos A y B. Por lo que hay configuraciones a chequear, las cuales podemos generar con las cadenas binarias de longitud 2N que tengan N unos (C[x] = 1 si x va para la primera fila, C[x] = 0 si x se ubica en la segunda) y comenzado con haciéndole prev\_permutation() las examinaremos todas hasta llegar a .



En la implementación anterior omitimos la función ok() que retorna true si la configuración en TABLAS es correcta según las restricciones del problema o false en caso contrario. Esta solución da en tiempo para N <= 12.

Observemos ahora que para cualquier columna x se cumpla que los dos elementos de la x-ésima columna estén en orden creciente entonces el x-ésimo 1 debe estar antes del x-ésimo 0. Ejemplo la cadena binaria “100110” que corresponde con configuración:

|  |  |  |
| --- | --- | --- |
| 1 | 4 | 5 |
| 2 | 3 | 6 |

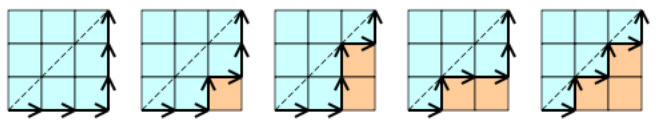
No es válida. Luego esto nos dice que solo debemos examinar las cadenas binarias de longitud 2N con N unos donde para todo prefijo de las mismas la cantidad de 1s sea mayor o igual a la cantidad de 0s. Si establecemos la correspondencia 1 con ‘(’ y 0 con ‘)’ entonces las configuraciones que estamos buscando son las parentizaciones balanceadas con algunos símbolos ‘(’s fijos [elementos en el conjunto A] y algunos ‘)’s fijos [elementos en el conjunto B]. La biyección se hace clara puesto que una cadena de paréntesis es balanceada sí y solo sí la cantidad de ‘(’s [1s] es igual a la cantidad de ‘)’s [1s] y para todo prefijo la cantidad de abiertos [1s] es mayor o igual a la cantidad de cerrados [0s]. Veamos un ejemplo:

N = 3; A = {3}; B = {}

|  |  |  |
| --- | --- | --- |
| Cadena binaria | Tablas | Parentización |
| 111000 | 1 2 3  4 5 6 | ( ( ( ) ) ) |
| 101100 | 1 3 4  2 5 6 | ( ) ( ( ) ) |
| 101010 | 1 3 5  2 4 6 | ( ) ( ) ( ) |

O sea, el problema se reduce a encontrar cuántas parentizaciones balanceadas tienen este formato: ? ? ( ? ? ?

Para resolver el problema de contar las parentizaciones balanceadas de N parejas de ()s observemos que son iguales a la cantidad de caminos en una matriz cuadrada de (N+1)x(N+1) de la celda 1, 1 a la N + 1, N + 1 donde solo está permitido a cada paso ir hacia abajo [ asociado a ( ] o ir a la derecha [asociado a ) ] y todo el tiempo estemos en el camino por debajo de la diagonal principal o en ella para asegurar la condición de que en todo momento la cantidad de cerrados no sobrepase la cantidad de abiertos.



Los elementos fijos de A y de B nos dicen que en algunas casillas solo podrá llegarse o bien de arriba o bien de la izquierda respectivamente. Dicho esto solo queda elaborar la definición por Programación Dinámica para el conteo:

P[x] = ‘(’ si x pertenece a A; P[x] = ‘)’ si x pertenece a B en otro caso P[x] = ‘?’.

dp(i, j) = cantidad de caminos de 1,1 a i,j que cumplen las condiciones anteriores.

dp(1, 1) = 1

Para i en [2..N+1]:

Para j en [1..i]:

cant = i – 1 + j – 1

dp(i, j) = dp(i, j – 1) si i == j o P[cant] == ‘)’

dp(i, j) = dp(i – 1, j) si P[cant] == ‘(’

dp(i, j) = dp(i – 1, j) + dp(i, j – 1) si P[cant] == ‘?’

Sol = dp(N + 1, N + 1)

La complejidad temporal y espacial esta solución es O(N2).

**Habilidades requeridas**: técnicas de conteo (biyección), programación dinámica

**Actividades propuestas**: 2260 - Dick Words (COJ). Investigar sobre los Números de Catalan.

PSN 2016 - Frank Arteaga Salgado, farteaga@ipvce.lt.rimed.cu