Problema 1

Orden

Link

https://codeforces.com/problemset/problem/1671/E

Definicion

Dado un árbol binario con $2^n - 1$ vértices donde cada vértice es una hoja o tiene exactamente dos hijos (un hijo izquierdo y un hijo derecho), y el valor de cada vertice es 'A' o 'B', la tarea es encontrar el número de cadenas de preorden distintas que se pueden formar a partir del árbol cuando puedes intercambiar los hijos de cualquier vértice cualquier número de veces.

- Cadena de Preorden: Para cada nodo, la cadena de preorden se define concatenando el carácter en ese nodo con las cadenas de preorden de sus hijos izquierdo y derecho.
- Operación de Intercambio: Puedes intercambiar los hijos izquierdo y derecho de cualquier vértice que no sea hoja, lo que cambiará el orden de concatenación en la cadena de preorden.

Solucion Divide and Conquer

Solucion

Propongamos la siguiente solucion usando divide and conquer:

- Si el arbol es una hoja, entonces solo hay una cadena de preorden posible, el que incluye a si misma.
- Si el arbol no es una hoja, entonces podemos dividir el arbol en dos subarboles, el subarbol izquierdo y el subarbol derecho. Llamemos `L` al numero de cadenas de preorden posibles para el subarbol izquierdo y `R` al numero de cadenas de preorden posibles para el subarbol derecho. Entonces, el numero de cadenas de preorden posibles para el arbol completo puede caer en 1 de dos casos:
  - Si el conjunto de preordenes del subarbol derecho es igual al conjunto de preordenes del subarbol izquierdo, entonces el numero de cadenas de preorden posibles para el arbol completo es igual a $L * R$, ya que usando un enfoque combinatorio, se cogeria de raiz al nodo actual, y por cada uno de los preordenes de la izquierda escogemos uno de la derecha para concadenarlo, intercambiar los dos hijos del nodo actual seria inutil pq darian los mismos preordenes.
  - Si el conjunto de preordenes del subarbol derecho no es igual al conjunto de preordenes del subarbol izquierdo, entonces el numero de cadenas de preorden posibles para el arbol completo es igual a $2 * L * R$, seria el mismo enfoque que si fueran iguales, pero ahora como no lo son, puedo intercambiar los hijos, dando el doble de posibilidades.
- Luego, nos podemos dar cuenta (despues hago la demostracion), que esto es muy complejo de calcular, pero podemos simplificarlo de gran manera. No nos hace falta comprobar que todo el conjunto de preordenes son iguales, solo necesito comprobar cual es el menor de ellos lexicograficamente, o sea, si el menor lexicografico de la izquierda comparado con el menor lexicografico de la derecha, eso debe ser exactamente igual que comprobarlos todos sin la complejidad añadida.

```python
def solve(n: int, s: [str]) -> int:
    """Divide and conquer solution"""
    s = " " + s

    def solve_rec(x: int) -> (int, str):
```

```
        l = 2 * x
        r = 2 * x + 1

        if l >= len(s):
            return 1, s[x]

        left_pre, left_small = solve_rec(l)
        right_pre, right_small = solve_rec(r)

        t = min(s[x] + left_small + right_small, s[x] + right_small + left_small)

        if left_small == right_small:
            return (left_pre * right_pre) % MOD, t
        else:
            return 2 * left_pre * right_pre % MOD, t

    return solve_rec(1)[0]
```

Comlejidad por Teorema de Master:

$T(n) = 2T(\frac{n}{2}) + O(1)$

$a = 2, b = 2, f(n) = O(1)$

⁂ Entra en el caso 1 del teorema de Master

$O(n^{\log_b a}) = O(n^{\log_2 2}) = O(n)$

Demostracion

Necesito 2 demostraciones para la solucion:

1. Demostrar que puedo simplificar el problema comprobando solo el preorden mas pequeño de cada subarbol
2. Demostrar que en efecto, el enfoque de divide and conquer funciona

Demostracion 1

Para demostrar que dicha simplifacion funciona entonces tenemos que demostrar lo siguiente:

`Sean A y B dos arboles binarios perfectos con la misma cantidad de nodos, y sean {A} y {B} los preordenes de A y B respectivamente, siguiendo las mismas reglas de cambio del problema, entonces si el preorden mas pequeño de A es igual al preorden mas pequeño de B, entonces {A} = {B}`

`Hipotesis:` Supongamos que el preorden mas pequeño de A es igual al preorden mas pequeño de B, llamemoslos `a` y `b` respectivamente.

`Demostracion:` Como ambos son arboles binarios perfectos con la misma cantidad de nodos, entonces ambos tienen topologicamente hablando la misma estructura. Luego, como todos los preordenes de ambos conjuntos tienen la misma longitud, y por reglas de cambio, y como $a = b$, entonces puedo hacer un cambio en A que me de B. O sea existe una forma de cambiar nodos en A, tal que me daria B. Luego, haciendo ese cambio, me quedaria un un arbol que tiene el mismo preorden que B, y como tienen la misma topologia, un arbol igual que B. Luego partiendo con ese `A'` y `B`, voy a empezar a buscar preordenes cambiando nodos, pero lo voy a hacer a la par, o sea si ya busque un preorden en `A'` y `B`, y voy a cambiar un nodo en `A'` con otro, voy a hacer lo mismo en `B` para seguir la busqueda. De esta forma, cuando la busqueda termine, voy a tener los mismos preordenes en ambos conjuntos, o sea $\{A'\} = \{B\}$. Pero como $\{A'\} = \{A\}$, ya que `A'` es A cambiando algunos nodos con sus hermanos, entonces $\{A\} = \{B\}$, y por tanto sus preordenes son

iguales. En terminos del problema, esto significa que para todo elemento de {A} hay un elemento en {B} igual a él y viceversa.

`PD: Me di cuenta tarde que puedo hacer una demostracion mas simple y mas fuerte, esta demostracion cumple que hayan cualquiera 2 preordenes en ambos conjuntos que sean iguales, no solo el mas pequeño. Pero me voy a quedar con el más pequeño ya que es algo que algoritmicamente puedo mantener en cada iteracion, o sea que en cada llamado si siempre devuelvo el preorden mas pequeño, entonces puedo mantenerlo en cada llamado subsecuente.`

Por tanto, puedo simplificar el problema comprobando solo el preorden mas pequeño de cada subarbol.

Demostracion 2

Usemos induccion matematica para la demostracion en n, donde n es el numero de niveles del arbol, ya que es un arbol binario perfecto y el numero de nodos es $2^{n+1} - 1$.

Caso Base n=1

Para n=1, el arbol es una hoja, y el numero de preordenes posibles es 1, ya que solo hay una cadena de preorden posible, el que incluye a si misma.

Caso Inductivo

`Hipotesis de induccion:` Supongamos que la solucion es correcta para n=k.

`Paso inductivo:` Queremos demostrar que la solucion es correcta para n=k+1.

Para n=k+1, el arbol no es una hoja, llamemos a este árbol `x`, luego tiene hijo izquierdo e hijo derecho, cada uno conforma un subarbol de n=k. Como la solucion es correcta para n=k, entonces ya tengo `L` y `R`, seanse el numero de preordenes posibles para el subarbol izquierdo y el subarbol derecho respectivamente. Cada uno de los preordenes de `x`, es el resultado de concatenar el valor de `x` con algun miembro de `{L}` + algun miembro de `{R}`. Luego, combinatoriamente hablando se tiene que el resultado deberia ser $2 * L * R$, o sea hacer una permutacion de cada miembro de `L` con cada miembro de `R`, y como puedo intercambiar los hijos de `x`, entonces tengo el doble de posibilidades ya que podria hacer la concatenacion de algun miembro de `{R}` primero y despues concatenar algun miembro de `{L}`.

Pero pasa algo, puede haber casos repetidos. En el hipotetico caso de que el miembro de `{L}` y de `{R}` que escoja, llamemoslos `l` y `r` respectivamente, sean ambos iguales, entonces el preorden `xlr` seria el mismo que el preorden `xrl`, por lo que no puedo contarlos dos veces. Es aqui donde la Demostracion 1 es importante. Sea `l` el preorden mas pequeno de `{L}` y `r` el preorden mas pequeno de `{R}`, entonces si `l=r`, entonces para cada elemento de `{L}` hay un elemento en `{R}` igual a el y viceversa, por lo que no puedo contarlos dos veces ya que `xlr=xrl`. O sea, que el numero de preordenes posibles para `x` es $L * R$.

Por tanto, la solucion es correcta para n=k+1, y por tanto es correcta para todo n.