1. Enunciado del problema

Dado un árbol con N nodos, calcular para todo $K \leq N$ el mínimo número de aristas a ser cortadas tal que quede una componente conexa con exactamente K nodos.

2. Solución

Resolvamos el problema utilizando programación dinámica. Rootiemos el árbol T, llamemos T_u al subárbol del nodo u. Formulemos nuestra DP de la siguiente forma:

Sea $dp[u][i]$ la mínima cantidad de aristas que hay que eliminar del árbol para obtener una componente conexa en T_u que contenga al nodo u y sea de tamaño i. Es trivial que teniendo este arreglo rellenado podemos calcular la solución para cada K en $O(N)$.

Veamos cómo calcular esta DP. Denotemos T_u^j como el árbol T_u pero conteniendo solo sus j primeros hijos y sus subárboles.

Llamemos $dp_j[u][i]$ el valor de $dp[u][i]$ en T_u^j, lo cual es básicamente habiendo procesado los j primeros hijos de u. La idea es calcular $dp_h[u][i]$ donde h es la cantidad de nodos hijos de u, es decir, $h = |\{v_1, v_2, v_3, \ldots, v_h\}|$ donde v_i es el hijo i-ésimo de u

.

Empezamos con $j = 0$, en ese caso u es tomado como una hoja en T_u^j, así que $dp_j[u][1] = 0$. Luego computamos los valores de $dp_j[u][i]$ siguiendo la siguiente fórmula:

$$dp_j[u][i] = \min\left(dp_{j-1}[u][i] + 1, \min_{\substack{a+b=i \\ a,b\geq 1}}(dp_{j-1}[u][a] + dp[v_j][b])\right)$$

Lo cual es básicamente tomar el mínimo entre:

- Cortar la arista hacia el j-ésimo hijo de u.
- El mínimo para todo a, b de tener a nodos en T_u^{j-1} y tener b nodos en el subárbol del j-ésimo hijo de u.

3. Análisis de complejidad

3.1. Cota máxima del algoritmo

Podríamos establecer una cota máxima de $O(N^3)$ para nuestro algoritmo, ya que para cada nodo u, debemos llenar todos los valores de i en $dp[u][i]$, y para hacer esto tenemos que recorrer todas las posibles particiones a, b tal que $a + b = i$. Sin embargo, podemos encontrar una mejor cota.

3.2. Enunciado

Para un nodo fijo u, el número de operaciones necesarias para computar todos los valores de $dp[v][:]$ donde v es un nodo que pertenece a T_u es $O(|T_u|^2)$.

3.3. Demostración

Demostremos este resultado por inducción sobre la profundidad del árbol T_u.

- Caso base: Si T_u es una hoja, el lema se cumple trivialmente, ya que solo necesitamos $O(1)$ operaciones.

- Paso inductivo: Supongamos que el nodo u tiene h hijos, denotados como $v_1, v_2, \ldots, v_h$, y definamos $a_j = |T_{v_j}|$.

  Al fusionar el j-ésimo hijo, es decir, al calcular $dp_j[u][:]$, recorremos todos los valores de $dp_{j-1}[u][a]$ y $dp[v_j][b]$ para todo a y b. Sin embargo, el valor de a puede ser escogido en $|T_u^{j-1}|$ formas y el de b en $|T_{v_j}|$ formas, lo que nos da un número total de operaciones:

  $$O(|T_u^{j-1}| \cdot |T_{v_j}|) = O\left((1 + a_1 + a_2 + \cdots + a_{j-1}) \cdot a_j\right)$$

  Por la hipótesis de inducción, calcular todos los valores de dp para los nodos en el subárbol de u tomará:

  $$O\left(\sum_{j=1}^{h} a_j^2\right)$$

  Sumando el número de operaciones necesarias para calcular los valores de $dp[v][:]$ para todos los vértices v pertenecientes a T_u, obtenemos:

  $$O\left(\sum_{i=1}^{h}\sum_{j=1}^{h} a_i \cdot a_j + \sum_{j=1}^{h} a_j + \sum_{j=1}^{h} a_j^2\right)$$

  Como el primer término es una doble sumatoria sobre los tamaños de los subárboles, podemos agrupar los términos para obtener:

  $$O\left(\left(1 + \sum_{j=1}^{h} a_j\right)^2\right) = O(|T_u|^2)$$

3.4. Demostración combinatoria alternativa

También podemos obtener una demostración combinatoria del enunciado. Al fusionar un subárbol se realizan:

$$O(|T_u^{j-1}| \cdot |T_{v_j}|)$$

operaciones. Esto se puede interpretar como el número de pares de nodos que tienen a u como su ancestro común más cercano (LCA). Como cualquier par de nodos tiene un único LCA, cada par se cuenta una sola vez, lo que nos lleva a una cota total de:

$$O(N^2)$$

4. Demostración de Correctitud

Sea un árbol T con N nodos y sea T_u el subárbol enraizado en el nodo u. Para cada nodo u y para cada entero i en $1, \ldots, |T_u|$, definimos $dp[u][i]$ como el mínimo número de aristas a cortar en T_u para obtener una componente conexa de tamaño i que contiene a u. Además, al procesar el nodo u consideramos sus hijos en un orden fijo y definimos T_u^j como el árbol formado por u y los subárboles de sus primeros j hijos. Sea

$dp_j[u][i] =$ costo mínimo en T_u^j para obtener una componente conexa de tamaño i que contiene a u.

El caso base es $j = 0$, donde el único nodo es u, de modo que

$$dp_0[u][1] = 0,$$

y para $i \neq 1$ se establece $dp_0[u][i] = +\infty$ (o un valor que indique imposibilidad).

Procedemos a demostrar la correctitud mediante doble inducción: primero sobre el número de hijos procesados (índice j) y luego sobre la estructura del árbol.

1. Inducción sobre la fusión de hijos en u

Caso base: Con $j = 0$, el subárbol parcial T_u^0 consiste únicamente en u; por lo tanto,

$$dp_0[u][1] = 0,$$

lo cual es correcto.

Paso inductivo: Supongamos que para cierto $j - 1$ la tabla $dp_{j-1}[u][\cdot]$ calcula correctamente el costo mínimo para cada tamaño i en el subárbol T_u^{j-1}. Sea v_j el j-ésimo hijo de u. Al incorporar el subárbol T_{v_j}, se tienen dos opciones:

1. **No incluir T_{v_j}:** Se corta la arista (u, v_j) y se conserva el estado anterior, con un costo adicional de 1:
$$dp_j[u][i] \leq dp_{j-1}[u][i] + 1.$$

2. **Fusionar T_{v_j}:** Se toma una componente en T_u^{j-1} de tamaño a (con costo $dp_{j-1}[u][a]$) y una componente en T_{v_j} de tamaño b (con costo $dp[v_j][b]$); al fusionarlas se obtiene una componente de tamaño $a+b$ con costo:

$$dp_j[u][a+b] \leq dp_{j-1}[u][a] + dp[v_j][b].$$

Por lo tanto, para cada i se tiene:

$$dp_j[u][i] = \text{mín}\left\{ dp_{j-1}[u][i] + 1,\ \min_{\substack{a+b=i \\ a,b\geq 1}} \Big(dp_{j-1}[u][a] + dp[v_j][b] \Big) \right\}.$$

Dado que, por hipótesis inductiva, $dp_{j-1}[u][\cdot]$ y $dp[v_j][\cdot]$ son correctos, la fusión produce correctamente $dp_j[u][\cdot]$ en T_u^j.

Al procesar todos los hijos de u (es decir, $j = h$, donde h es el número total de hijos de u), se obtiene:

$$dp_h[u][\cdot] = dp[u][\cdot],$$

lo que demuestra la corrección del estado $dp[u][\cdot]$ en T_u.

2. Inducción en la estructura del árbol

Asumiendo que para cada hijo v de u la tabla $dp[v][\cdot]$ es correcta, el proceso de fusión en u descrito anteriormente garantiza que $dp[u][\cdot]$ se calcula correctamente. Por inducción sobre la estructura del árbol, la DP es correcta para todo el árbol T.

5. Pseudocódigo en $O(N^2)$

A continuación se presenta el pseudocódigo que implementa la solución mediante DFS y fusión de estados. Se aprovecha que, al fusionar el estado del nodo u (de tamaño size[u]) con el del hijo v (de tamaño size[v]), se realizan size[u] $\times$ size[v] operaciones, y gracias a la propiedad combinatoria (cada par de nodos se fusiona una única vez, correspondiente a su LCA) la complejidad total es $O(N^2)$.

```
// Constante que representa un valor "infinito"
const INF = infinito;

// dp[u][i]: Costo minimo para obtener una componente conexa de
   tama o i en T_u que contiene a u.
// size[u]: Tamano actual del arreglo dp[u] (inicialmente 1, pues
   solo se cuenta el nodo u).

procedure DFS(u, parent)
    // Inicializar dp[u] con el caso base: solo el nodo u.
```

```
    dp[u] := array[1 ... (tamano maximo posible)] of INF;
    dp[u][1] := 0;
    size[u] := 1;

    // Procesar cada hijo v de u (evitando regresar al padre)
    for each v in vecinos(u) do
        if v == parent then continue;
        DFS(v, u);

        // Preparar un nuevo arreglo para fusionar dp[u] y dp[v]
        newSize := size[u] + size[v];
        newDP := array[1 ... newSize] of INF;

        // Opcion 1: No fusionar v (cortar la arista (u,v))
        for i from 1 to size[u] do
            newDP[i] := min(newDP[i], dp[u][i] + 1);
        end for;

        // Opcion 2: Fusionar el subarbol de v sin cortar (usar dp[v])
        for i from 1 to size[u] do
            for j from 1 to size[v] do
                newDP[i + j] := min(newDP[i + j], dp[u][i] + dp[v][j]);
            end for;
        end for;

        // Actualizar dp[u] y su tamano
        size[u] := newSize;
        dp[u] := newDP;
    end for;
end procedure;

// Funcion principal: se asume que el arbol tiene N nodos.
procedure Main()
    // Se asume que el arbol esta almacenado en una lista de adyacencia: G.
    // Se elige un nodo arbitrario como raiz, por ejemplo, 1.
    DFS(1, NIL);

    // Actualizar la respuesta global para cada tamano K.
    // La solucion final es el minimo dp[u][K] entre todos los nodos u.
    answer := array[1 ... N] of INF;
    for each nodo u de 1 a N do
```

```
        for k from 1 to size[u] do
            answer[k] := min(answer[k], dp[u][k]);
        end for;
    end for;

    // Imprimir o retornar la respuesta para cada K = 1, 2,    , N.
end procedure;
```

Listing 1: Pseudocódigo en $O(N^2)$