Double Agents (`trees`)

La agencia de espionaje británica MI6 está planeando infiltrar una red de agentes dobles en la siniestra organización criminal SPECTRE. SPECTRE tiene N empleados, numerados de 0 a $N-1$, y su estructura organizativa es un árbol en esos N vértices.

MI6 seleccionará un conjunto no vacío S de empleados de SPECTRE que convertir en agentes dobles. Debido al riesgo de traiciones subsecuentes (posibles agentes triples), es esencial que la cadena de comunicación entre cualquier par de agentes dobles pase sólo a través de agentes 'inocentes'. Es decir, for cada par de empleados distintos a y b en S, el camino entre a y b en el árbol no debe contener ningún otro empleado en S.

Tu objetivo es contar el número de conjuntos posibles S de agentes dobles. Como este valor puede ser muy grande, debes retornar el valor módulo $10^9 + 7$.

Implementación

Deberás presentar un único fichero `.cpp`.

> ☞ Entre los archivos adjuntos encontrarás una plantilla `trees.cpp` con una implementación de ejemplo.

Tienes que implementar la siguiente función:

C++
```
int count_sets(int N, vector<int> U, vector<int> V);
```

- El entero N representa el número de empleados de SPECTRE.
- Los vectores U y V, indizados de 0 a $N-2$, contienen los valores $U_0, U_1, \ldots, U_{N-2}$ y $V_0, V_1, \ldots, V_{N-2}$, donde U_i y V_i son los extremos de la i-ésima arista en el árbol de la organización.
- La función debe retornar el número de conjuntos posibles módulo $10^9 + 7$.

El grader llamará a la función `count` e imprimirá su valor de retorno en el archivo de salida.

Sample Grader

El directorio de la tarea contiene una versión simplificada del grader del jurado, que puedes utilizar para probar tu solución localmente. El grader simplificado lee los datos de entrada de `stdin`, llama a la función que debes implementar, y finalmente escribe la salida a `stdout`.

La entrada consiste en N líneas:

- Línea 1: el entero N.
- Línea $2 + i$ $(0 \le i \le N-2)$: los enteros U_i y V_i.

La salida consiste en una sola línea, que contiene el valor retornado por la función `count`.

Restricciones

- $2 \le N \le 500\,000$.
- $0 \le U_i, V_i \le N-1$.

- Los nodos forman un árbol.

Puntuación

Tu programa será probado en un conjunto de casos de prueba agrupados por subtarea. Para obtener la puntuación asociada a una subtarea, debes resolver correctamente todos los casos de prueba que contiene.

- **Subtask 1** [**0 puntos**]: Casos de ejemplo.
- **Subtask 2** [**20 puntos**]: $N \leq 16$.
- **Subtask 3** [**15 puntos**]: El árbol es un camino. Un camino es un árbol en el que los nodos se pueden organizar en un orden $P_0, P_1, \ldots, P_{N-1}$ para el que existe una arista entre los nodos P_i y P_{i+1} para todo $0 \leq i \leq N-2$.
- **Subtask 4** [**15 puntos**]: $N \leq 500$, y el árbol es un *caterpillar*. Un *caterpillar* es un camino con algunas hojas adicionales conectadas. Es decir, los vértices del árbol se pueden escribir como $P_0, \ldots, P_k, Q_0, \ldots, Q_l$, donde $P_0, \ldots, P_k$ es un camino y cada Q_i es una hoja (es decir, tiene grado 1).
- **Subtask 5** [**15 puntos**]: $N \leq 5000$, y el árbol es un *caterpillar*.
- **Subtask 6** [**10 puntos**]: $N \leq 500\,000$, y el árbol es un *caterpillar*.
- **Subtask 7** [**25 puntos**]: Sin restricciones adicionales.

Ejemplos de entrada/salida

stdin	stdout
3 0 1 1 2	6
5 0 1 0 2 1 3 1 4	17

Explicación

En el **primer caso de ejemplo**, el árbol es un camino $(0-1-2)$. Hay 6 conjuntos posibles en este árbol: $\{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}$.

El conjunto $\{0,1,2\}$ no está permitido porque 1 está en el camino entre 0 y 2.

En el **segundo caso de ejemplo**, el árbol es un camino $(3-1-0-2)$ con una hoja adicional (4). Hay 17 conjuntos posibles.