Double Agents (trees)

Der britische Geheimdienst MI6 plant, ein Netzwerk von Doppelagenten in die sinistre kriminelle Organisation SPECTRE einzuschleusen. SPECTRE hat N Mitarbeiter, nummeriert von 0 bis $N-1$, und ihr Organigramm ist ein Baum auf diesen N Knoten.

MI6 wird eine nicht-leere Menge S von SPECTRE-Mitarbeitern auswählen, um sie in Doppelagenten zu verwandeln. Aufgrund des Risikos weiterer Verräter (d.h. möglicher Dreifachagenten) ist es unerlässlich, dass die Kommunikationskette zwischen zwei beliebigen Doppelagenten nur durch ünschuldigeMitarbeiter verläuft. Das heisst, für zwei unterschiedliche Mitarbeiter a und b in S darf der einfache Pfad zwischen a und b im Baum keine anderen Mitarbeiter in S enthalten.

Deine Aufgabe ist es, die Anzahl der möglichen Mengen S von Doppelagenten zu zählen. Da dieser Wert ziemlich gross sein kann, gib den Wert modulo $10^9 + 7$ aus.

Implementierung

Du musst eine einzelne `.cpp`-Quelldatei einreichen.

> ☞ Unter den Anhängen dieser Aufgabe findest du eine Vorlage `trees.cpp` mit einer Beispielimplementierung.

Du musst die folgende Funktion implementieren:

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

- Die Ganzzahl N repräsentiert die Anzahl der Mitarbeiter von SPECTRE.
- Die Arrays U und V, indiziert von 0 bis $N-2$, enthalten die Werte $U_0, U_1, \ldots, U_{N-2}$ und $V_0, V_1, \ldots, V_{N-2}$, wobei U_i und V_i die Endpunkte der i-ten Kante im Organisationsbaum sind.
- Die Funktion soll die Anzahl der möglichen Mengen modulo $10^9 + 7$ zurückgeben.

Der Grader wird die Funktion `count` aufrufen und deren Rückgabewert in die Ausgabedatei schreiben.

Beispielgrader

Das Verzeichnis der Aufgabe enthält eine vereinfachte Version des Jury-Graders, die du verwenden kannst, um deine Lösung lokal zu testen. Der vereinfachte Grader liest die Eingabedaten aus `stdin`, ruft die von dir zu implementierenden Funktionen auf und schreibt schliesslich die Ausgabe in `stdout`.

Die Eingabe besteht aus N Zeilen, die enthalten:

- Zeile 1: die Ganzzahl N.
- Zeile $2+i$ ($0 \le i \le N-2$): die Ganzzahl U_i und V_i.

Die Ausgabe besteht aus einer einzigen Zeile, die den von der Funktion `count` zurückgegebenen Wert enthält.

Einschränkungen

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

- Die Knoten bilden einen gültigen Baum.

Punktevergabe

Dein Programm wird an einem Satz von Testfällen getestet, die in Unteraufgaben gruppiert sind. Um die Punktzahl zu erhalten, die mit einer Unteraufgabe verbunden ist, musst du alle darin enthaltenen Testfälle korrekt lösen.

- **Teilaufgabe 1** [**0 Punkte**]: Beispiel-Testfälle.
- **Teilaufgabe 2** [**20 Punkte**]: $N \leq 16$.
- **Teilaufgabe 3** [**15 Punkte**]: Der Baum ist ein Pfad. Ein Pfad ist ein Baum, in dem die Knoten in einer Reihenfolge $P_0, P_1, \ldots, P_{N-1}$ angeordnet werden können, in der eine Kante zwischen den Knoten P_i und P_{i+1} für alle $0 \leq i \leq N-2$ existiert.
- **Teilaufgabe 4** [**15 Punkte**]: $N \leq 500$, und der Baum ist eine Raupe. Eine Raupe ist ein Pfad mit einigen zusätzlichen Blattknoten, die angehängt sind. Das heisst, die Knoten des Baumes können als $P_0, \ldots, P_k, Q_0, \ldots, Q_l$ geschrieben werden, wobei $P_0, \ldots, P_k$ ein Pfad ist und jedes Q_i ein Blatt (d.h. hat Grad 1) ist.
- **Teilaufgabe 5** [**15 Punkte**]: $N \leq 5000$, und der Baum ist eine Raupe.
- **Teilaufgabe 6** [**10 Punkte**]: $N \leq 500\,000$, und der Baum ist eine Raupe.
- **Teilaufgabe 7** [**25 Punkte**]: Keine zusätzlichen Einschränkungen.

Beispiele

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

Erklärung

Im **ersten Beispiel** ist der Baum ein Pfad $(0-1-2)$. Es gibt 6 mögliche Mengen in diesem Baum: $\{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}$.

Die Menge $\{0,1,2\}$ ist nicht erlaubt, da 1 auf dem einfachen Pfad zwischen 0 und 2 liegt.

Im **zweiten Beispiel** ist der Baum ein Pfad $(3-1-0-2)$ mit einem zusätzlichen Blatt (4). Es gibt 17 mögliche Mengen in diesem Baum.