Де Корінь?

`Це інтерактивна задача`

Вам дано дерево з n вершин. Дерево - це граф, в якому між кожною парою вершин існує рівно один простий шлях. **Також гарантується, що існує принаймні одна така вершина, що безпосередньо з'єднана ребром з принаймні 3 іншими вершинами**. Одна з вершин - це корінь, і ваше завдання - визначити його. Для цього вам дозволено робити запити наступної форми:

- Для даної множини вершин $a_1, a_2, \ldots, a_m$ перевірити, чи їх найменший спільний предок належить до цієї множини.

Вершина v є спільним предком множини вершин S, якщо шляхи від усіх вершин множини S до кореня проходять через v. Найменший спільний предок (LCA) множини вершин S є спільний предком S, який є найвіддаленішим від кореня.

Взаємодія з інтерактором

На початку зчитайте єдине ціле число n ($4 \le n \le 500$) - кількість вершин.

Потім зчитайте наступні $n-1$ рядки. i-й рядок міститиме два цілі числа a_i, b_i ($1 \le a_i, b_i \le n$), що вказують на наявність ребра між вершинами a_i та b_i у дереві.

Гарантується, що ці $n-1$ ребер утворюють дерево і принаймні одна вершина безпосередньо з'єднана ребром з принаймні 3 вершинами.

Щоб зробити запит, спочатку виведіть "`?`", потім ціле число m, а потім m різних цілих чисел $a_1, a_2, \ldots, a_m$ ($1 \le m \le n$, $1 \le a_i \le n$, усі a_i є різними) - вершини, для яких ви хочете перевірити, чи є серед них їх LCA.

У відповідь інтерактор виведе "`YES`", якщо їх LCA є одним із $a_1, a_2, \ldots, a_m$, і "`NO`" в іншому випадку.

Ви можете зробити не більше 1000 запитів, але ви отримаєте різну кількість балів залежно від кількості запитів, які ви зробите. Виведення відповіді не вважається запитом. Дивиться подробиці у розділі оцінювання.

Коли ви визначили корінь, виведіть символ "`!`", а потім одне ціле число v ($1 \le v \le n$) - корінь. Потім завершіть свою програму.

Після виведення запиту не забудьте вивести кінець рядка та скинути вивід. Для цього використовуйте:

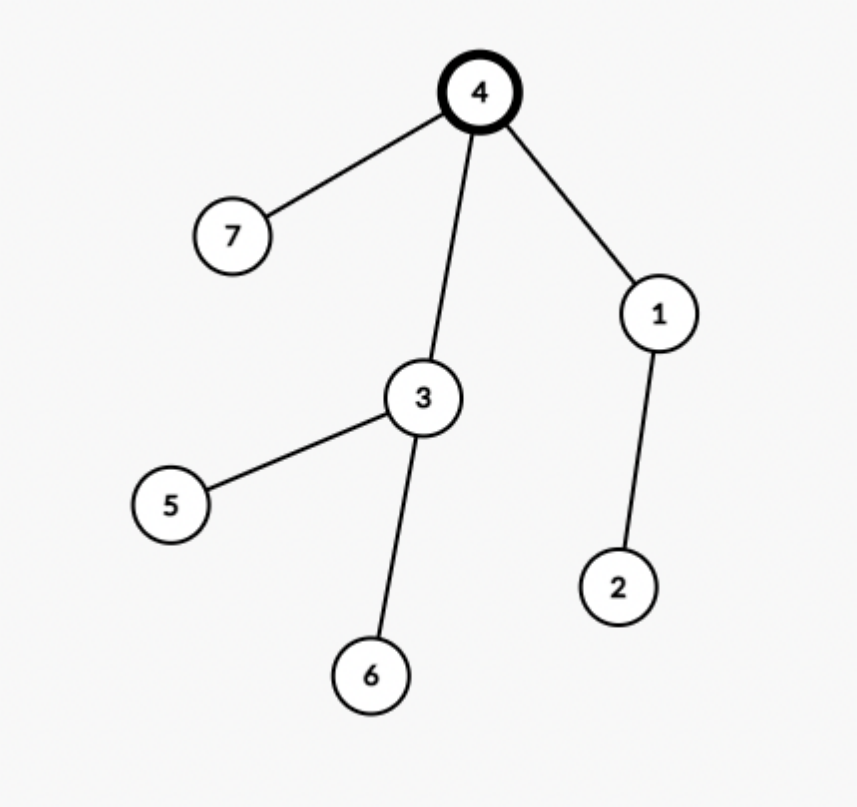
- `fflush(stdout)` або `cout.flush()` в C++;
- `stdout.flush()` в Python;

Гарантується, що для кожного тестового випадку дерево та його корінь фіксуються перед початком взаємодії. Іншими словами, **інтерактор не є адаптивним**.

Приклад

```
Ввід:
7
4 1
1 2
4 3
3 5
3 6
4 7
Вивід:
? 2 5 6
Ввід:
NO
Вивід:
? 3 6 3 5
Ввід:
YES
Вивід:
? 2 1 7
Ввід:
NO
Вивід:
? 2 4 6
Ввід:
YES
Вивід:
! 4
```

Примітка



Загаданий корінь — це вершина 4.

У першому запиті LCA вершин 5 і 6 — це вершина 3, яка не входить до множини вершин 5 і 6, тому відповідь: "NO".

У другому запиті LCA вершин 3, 5 і 6 є вершиною 3, тому відповідь "YES".

У третьому запиті LCA вершин 1 і 7 є вершиною 4, тому відповідь: "NO".

У четвертому запиті LCA вершин 4 і 6 є вершиною 4, тому відповідь "YES".

Після цього ми можемо припустити, що вершина 4 є коренем, що є правильною відповіддю.

Оцінювання

1. (7 балів): $n \leq 9$
2. (10 балів): $n \leq 30$
3. (до 83 балів): $n \leq 500$

У першій та другій підзадачах ви можете зробити не більше 1000 запитів.

У третій підзадачі, нехай k буде максимальною кількістю запитів, які ви зробили в рамках цієї підзадачі. Якщо $k \leq 9$, то ви отримаєте 83 бали. Інакше ви отримаєте $\lfloor \max(10, 83 \cdot (1 - \frac{\ln(k-6)}{7})) \rfloor$ балів.

Код C++, який обчислює кількість балів:

```
((k <= 9) ? 83: max(10, int(83 * (1 - log(k - 6.0) / 7))))
```