Που είναι η ρίζα?

`Το πρόβλημα είναι διαδραστικό`

Σας δίνεται ένα δέντρο το οποίο αποτελείται από n κορυφές. Αυτό το δέντρο είναι γράφος έτσι ώστε να υπάρχει ακριβώς ένα μονοπάτι μεταξύ κάθε ζεύγους κορυφών. **Είναι βέβαιο ότι τουλάχιστον μια κορυφή έχει τουλάχιστον 3 κορυφές συνδεδεμένες απευθείας (γείτονες)**. Μία από τις κορυφές είναι η ρίζα του δέντρου. Πρέπει να γράψετε ένα πρόγραμμα που να βρίσκει την ρίζα.

Μπορείτε να κάνετε ερωτήματα της μορφής:

- Για ένα σύνολο από κορυφές $a_1, a_2, \ldots, a_m$, ελέγχεται αν ο Χαμηλότερος Κοινός Πρόγονος (Lowest Common Ancestor / LCA) των κορυφών είναι μέσα στο σύνολο.

Πρέπει να βρείτε την ρίζα του δέντρου.

Μια κορυφή v είναι κοινός πρόγονος (common ancestor) ενός συνόλου από κορυφές S αν τα μονοπάτια από όλες τις κορυφές στο S προς την ρίζα περνάνε από το v. Ο Χαμηλότερος Κοινός Πρόγονος (LCA) ενός συνόλου S είναι ο κοινός πρόγονος του S ο οποίος είναι ο πιο μακρινός από την ρίζα.

Αλληλεπίδραση (Interaction)

Ξεκινήστε την αλληλεπίδραση διαβάζοντας έναν ακέραιο αριθμό n ($4 \leq n \leq 500$) - το πλήθος των κορυφών.

Μετά διαβάστε τις επόμενες $n-1$ γραμμές. Η i-οστή γραμμή θα περιέχει δύο ακέραιους αριθμούς a_i, b_i ($1 \leq a_i, b_i \leq n$), υποδεικνύοντας ότι υπάρχει μια ακμή μεταξύ των κορυφών a_i και b_i.

Είναι βέβαιο ότι αυτές οι $n-1$ ακμές σχηματίζουν ένα δέντρο και ότι υπάρχει πάντα τουλάχιστον μια κορυφή που έχει τουλάχιστον 3 κορυφές συνδεδεμένες απευθείας (γείτονες).

Για να κάνετε ένα ερώτημα, πρώτα τυπώστε "`?`", ακολουθούμενο από τον ακέραιο m, και τους m διαφορετικούς ακέραιους $a_1, a_2, \ldots, a_m$ ($1 \leq m \leq n$, $1 \leq a_i \leq n$, όλα τα a_i είναι διαφορετικά μεταξύ τους) - κορυφές για τις οποίες θέλετε να ελέγξετε αν ο Χαμηλότερος Κοινός Πρόγονος (LCA) τους είναι μια από αυτές τις κορυφές.

Ως απάντηση, ο βαθμολογητής θα τυπώνει "`YES`" αν το LCA τους είναι ένα από τα $a_1, a_2, \ldots, a_m$, αλλιώς θα τυπώνει "`NO`".

Μπορείτε να κάνετε μέχρι 1000 ερωτήματα αλλά θα πάρετε διαφορετικό αριθμό πόντων ανάλογα με το πόσα ερωτήματα θα κάνετε. Το τύπωμα της τελικής απάντησης δεν μετρά ως ερώτημα. Δείτε την ενότητα Βαθμολόγηση για λεπτομέρειες.

Όταν βρείτε την ρίζα του δέντρου, τυπώστε το σύμβολο "`!`, και έναν ακέραιο αριθμό v ($1 \leq v \leq n$) - την ρίζα. Στην συνέχεια τερματίστε το πρόγραμμα σας.

Αφού τυπώσετε ένα ερώτημα, μην ξεχάσετε να τυπώσετε το τέλος γραμμής (end of line) και να κάνετε flush την έξοδο. Για να γίνει αυτό:

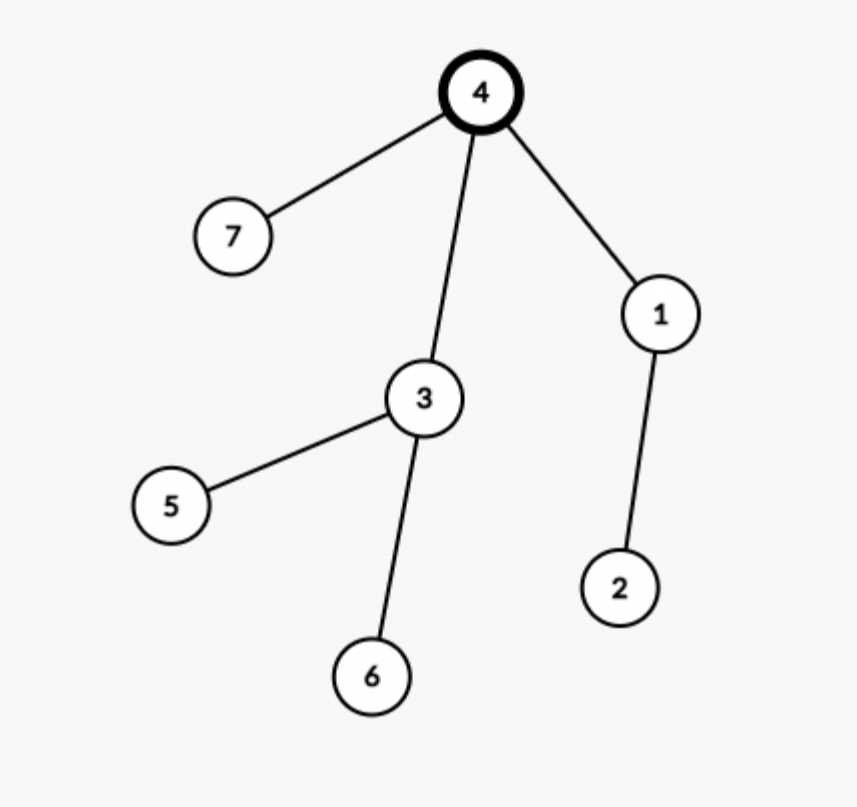
- `fflush(stdout)` ή `cout.flush()` στην C++;
- `stdout.flush()` στην Python;

Είναι βέβαιο ότι για κάθε περίπτωση ελέγχου (test case), το δέντρο και η ρίζα του είναι προκαθορισμένα από την αρχή της αλληλεπίδρασης. Με άλλα λόγια, **ο βαθμολογητής δέν είναι προσαρμοστικός (adaptive)**.

Παράδειγμα

```
Είσοδος:
7
4 1
1 2
4 3
3 5
3 6
4 7
Έξοδος:
? 2 5 6
Είσοδος:
NO
Έξοδος:
? 3 6 3 5
Είσοδος:
YES
Έξοδος:
? 2 1 7
Είσοδος:
NO
Έξοδος:
? 2 4 6
Είσοδος:
YES
Έξοδος:
! 4
```

Επεξήγηση



Η κρυμμένη ρίζα είναι η κορυφή 4.

Στο πρώτο ερώτημα, το LCA των κορυφών 5 και 6 είναι η κορυφή 3, η οποία δεν είναι μέσα στο σύνολο {5, 6} άρα η απάντηση είναι "`NO`".

Στο δεύτερο ερώτημα, το LCA των κορυφών 3, 5, και 6 είναι η κορυφή 3, άρα η απάντηση είναι "`YES`".

Στο τρίτο ερώτημα, το LCA των κορυφών 1 και 7 είναι η κορυφή 4, άρα η απάντηση είναι "`NO`".

Στο τέταρτο ερώτημα, το LCA των κορυφών 4 και 6 είναι η κορυφή 4, άρα η απάντηση είναι "`YES`".

Μετά από αυτό, μπορούμε να μαντέψουμε ότι η ρίζα είναι η κορυφή 4, η οποία είναι η σωστή απάντηση.

Βαθμολόγηση

Έστω k ο μέγιστος αριθμός ερωτημάτων που υποβάλατε για ένα υποπρόβλημα (subtask). Μπορείτε να ρωτήσετε το πολύ 1000 ερωτήματα, έτσι $k \leq 1000$.

1. (7 βαθμοί): $n \leq 9$
2. (10 βαθμοί): $n \leq 30$

3. (μέχρι 83 βαθμοί): $n \leq 500$

Στο τρίτο υποπρόβλημα, αν $k \leq 9$, θα πάρετε 83 πόντους. Διαφορετικά, θα πάρετε $\lfloor \max(10, 83 \cdot (1 - \frac{\ln(k-6)}{7})) \rfloor$ πόντους.

Επομένως, για να πάρετε όλους τους πόντους, η λύση σας πρέπει να χρησιμοποιεί το πολύ 9 ερωτήματα για κάθε περίπτωση ελέγχου (test case) του κάθε υποπροβλήματος (subtask).

Κώδικας C++ ο οποίος υπολογίζει τον αριθμό των πόντων:

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