universal cup 3 stage 30 : Northern

Created: 4/25/2025 11:48
Updated: 4/25/2025 11:53
Exported: 4/25/2025 12:00

Dịch đề bài:

Bài A. Khảo cổ học

Một cổ vật bí ẩn được giấu tại một điểm nguyên trên lưới vuông $D = [1; N] \times [1; N]$, gồm N^2 điểm. Nhiệm vụ của bạn là tìm ra vị trí của cổ vật này bằng cách sử dụng một radar đặc biệt.

Radar hoạt động như sau:

- Mỗi lần sử dụng, bạn đặt radar tại một điểm nguyên (Rx, Ry).
- Nếu cổ vật ở đúng điểm đó, radar sẽ sáng xanh lá, hiển thị tọa độ cổ vật và kết thúc cuộc khai quật.
- Nếu không, radar sẽ sáng vàng và hiển thị tọa độ một điểm B sao cho góc $\angle ARB < 90°$, với:
  - A là vị trí cổ vật.
  - R là vị trí radar.
  - B được chọn ngẫu nhiên từ tất cả các điểm thỏa mãn điều kiện trên (có thể là chính điểm A, nhưng không bao giờ là R).

Bạn có tối đa **90 lần** sử dụng radar để tìm ra cổ vật.

Tương tác:

- Dòng đầu tiên của input chứa số nguyên N ($1 \le N \le 10^9$).
- Sau đó, bạn gửi các truy vấn bằng cách in ra hai số nguyên Rx, Ry ($1 \le Rx, Ry \le N$) – vị trí đặt radar.

- Sau mỗi truy vấn, bạn nhận lại hai số nguyên Bx, By – tọa độ điểm được radar hiển thị.
- Nếu Bx = Rx và By = Ry, bạn đã tìm thấy cổ vật và chương trình phải kết thúc ngay lập tức.
- Nếu sau 90 lần truy vấn vẫn chưa tìm được cổ vật, chương trình kết thúc với kết quả **Wrong Answer**.

Tóm tắt đề bài:

- Có một cổ vật ẩn tại một điểm (A) trên lưới vuông kích thước N × N.
- Bạn có radar, mỗi lần đặt radar tại điểm (R), radar báo:
  - **Xanh lá**: nếu R là A → cổ vật được tìm thấy.
  - **Vàng**: nếu R khác A → trả về một điểm B sao cho ∠ARB < 90°, B được chọn ngẫu nhiên (có thể chính là A).
- Có tối đa **90 lần** dùng radar để tìm cổ vật.
- Kết thúc ngay khi tìm được.

Nhiệm vụ: viết chương trình chọn các điểm đặt radar sao cho chắc chắn tìm ra cổ vật trong 90 lần.

Dịch đề bài:

Bài B. Những viên đá thủy tinh

Alice và Bob chơi một trò chơi trên một chuỗi chỉ gồm các ký tự **'L'** và **'R'** (với mã ASCII lần lượt là 76 và 82). Họ lần lượt thực hiện lượt đi, bắt đầu từ **Alice**. Ở mỗi lượt, người chơi sẽ **xóa một ký tự bất kỳ** trong chuỗi.

Luật thắng:

- Nếu sau khi xóa, chuỗi **không còn chứa đoạn con "LR"**, thì **Alice thắng**.
- Nếu sau khi xóa, chuỗi **không còn chứa đoạn con "RL"**, thì **Bob thắng**.
- Nếu **cả hai điều kiện trên đều đúng cùng lúc**, trò chơi **hòa**.

Luật này cũng áp dụng ngay từ ban đầu:

- Nếu chuỗi ban đầu chỉ chứa "RL" mà không chứa "LR", **Bob thắng ngay lập tức**.
- Nếu chuỗi ban đầu chỉ chứa "LR" mà không chứa "RL", **Alice thắng ngay lập tức**.
- Nếu **không có "LR" và không có "RL"**, **hòa ngay lập tức**.

Nhiệm vụ của bạn:

- Xác định kết quả cuối cùng nếu **cả hai chơi tối ưu**.
- Chỉ ra một **nước đi tối ưu** cho **Alice** (nếu có).

Input:

- Số nguyên **T** – số lượng test ($1 \le T \le 10^6$).
- Mỗi test là một chuỗi **s** gồm các ký tự 'L' và 'R' ($1 \le |s| \le 10^6$).
- Tổng độ dài tất cả các chuỗi không vượt quá 10^6.

Output:

- Với mỗi test, in ra kết quả: `"Alice"`, `"Bob"` hoặc `"Draw"` (không phân biệt hoa thường), theo sau là một số nguyên **a**:
  - Nếu trò chơi kết thúc ngay lập tức (**trước lượt đầu tiên**), in **a = 0**.
  - Ngược lại, in ra **a** là **vị trí của ký tự mà Alice nên xóa** ở lượt đầu tiên (**1 ≤ a ≤ |s|**) để đảm bảo kết quả không thay đổi khi cả hai chơi tối ưu.

Tóm tắt đề bài:

- Hai người chơi lần lượt xóa ký tự 'L' hoặc 'R' khỏi chuỗi.
- Alice thắng nếu không còn "LR", Bob thắng nếu không còn "RL", hòa nếu cả hai xảy ra đồng thời.
- Nếu chuỗi ban đầu đã đạt điều kiện thắng/hòa thì kết thúc ngay.
- Yêu cầu: Xác định kết quả cuối cùng khi cả hai chơi tối ưu, và chỉ ra nước đi tối ưu đầu tiên của **Alice** (nếu có).

Dịch đề bài:

Bài C. Tâm trí của Elegia

Có **n khối lập phương giống hệt nhau**, mỗi mặt của khối lập phương được sơn bằng một trong **6 màu khác nhau** (được đánh số từ **0** đến **5**). Tất cả các khối lập phương đều **giống hệt nhau kể cả khi xoay trong không gian ba chiều**. Cụ thể, mỗi khối có thể được xoay sao cho:

- Mặt trước có màu **0**.
- Mặt trên có màu **1**.
- Mặt phải có màu **2**.
- Mặt trái có màu **3**.
- Mặt dưới có màu **4**.
- Mặt sau có màu **5**.

Giả sử bạn **xếp các khối lập phương này thành một hàng từ trái sang phải**. Với cách xếp này, ta thu được **6 dãy màu**, mỗi dãy thể hiện màu của các mặt **trước / trên / phải / trái / dưới / sau** của các khối lập phương theo thứ tự từ trái sang phải.

Nhiệm vụ của bạn là **đếm số cách sắp xếp các khối lập phương** (bao gồm cả việc xoay từng khối), sao cho **tất cả 6 dãy màu thu được đều thuộc tập các dãy màu cho phép**.

- Do các khối lập phương **giống hệt nhau**, thứ tự đặt các khối nào trước hay sau **không quan trọng**, chỉ quan trọng việc **xoay các khối** thế nào.
- Mỗi khối có **24 cách xoay** (tổng cộng 24^n cách sắp xếp cho n khối lập phương).

Tập hợp các dãy màu cho phép được cho bởi một **xâu nhị phân gồm 6^n ký tự** ('0' hoặc '1'):

- Với một dãy màu **$(a_0, a_1, ..., a_{n-1})$** (mỗi **a_i** là một số từ 0 đến 5), dãy này được phép nếu và chỉ nếu ký tự thứ **$(a_0 \cdot 6^0 + a_1 \cdot 6^1 + ... + a_{n-1} \cdot 6^{n-1})$** trong xâu (đếm từ 0) là **'1'**.

Kết quả của bài toán có thể rất lớn, hãy in ra **số cách sắp xếp thỏa mãn**, **lấy dư với 998244353**.

Tóm tắt đề bài:

- Có **n khối lập phương giống hệt nhau**, mỗi mặt có **6 màu** (đánh số từ 0 đến 5).
- Xếp **n khối lập phương thành hàng**, mỗi khối có **24 cách xoay** → tổng **24^n cách sắp xếp**.
- Khi xếp xong, tạo ra **6 dãy màu** (mặt trước, trên, phải, trái, dưới, sau).
- Yêu cầu: Đếm số cách sắp xếp sao cho **cả 6 dãy màu đều nằm trong tập dãy màu cho phép** (cho bởi xâu nhị phân dài 6^n ký tự).
- In ra **số cách sắp xếp hợp lệ**, **modulo 998244353**.

Dịch đề bài:

Bài D. Chuẩn bị cho kỳ thi

Misha đang chuẩn bị cho kỳ thi triết học. Kỳ thi sẽ diễn ra sau **t giờ** nữa và gồm **n phần**. Ở **phần thứ i**, có **q_i câu hỏi**, mỗi câu hỏi cần **1 giờ** để học.

Trong kỳ thi:

- Với **mỗi phần**, **1 câu hỏi ngẫu nhiên** sẽ được chọn (độc lập và đồng đều từ các câu hỏi của phần đó).
- Để **vượt qua kỳ thi**, Misha phải trả lời **đúng tất cả n câu hỏi**.

Misha có thể sử dụng **toàn bộ t giờ** để học một số câu hỏi (có thể học bất kỳ câu nào trong các phần). Nhiệm vụ của bạn là tính **xác suất tối đa** để Misha **vượt qua kỳ thi**, khi **tối ưu hóa chiến lược học**.

Input:

- Dòng đầu: hai số nguyên **t** và **n** – số giờ còn lại và số phần của kỳ thi ($1 \le t \le 10^9$; $1 \le n \le 10^5$).
- Dòng tiếp: **n số nguyên $q_1, \ldots, q_n$** – số câu hỏi trong mỗi phần ($1 \le q_i \le 10^9$).

Output:

- Gọi **$p = k / \ell$** là xác suất cần tìm dưới dạng **phân số tối giản**.
- In ra **p dưới dạng hai số nguyên x và y** sao cho:
  - **$x \cdot \ell - y \cdot k$** chia hết cho **$M = 10^9 + 7$**.
  - **y không chia hết cho M**.
- Đảm bảo luôn tồn tại cặp số như vậy.

Lưu ý:

- Nếu quen với việc tính xác suất dạng **số dư modulo một số nguyên tố**, có thể in ra **x 1** (x là xác suất mod M), vẫn được chấp nhận.

Ví dụ:

Input	Output
1 2 2 2	0 4
3 2 2 2	2 4

Tóm tắt đề bài:

- Kỳ thi có **n phần**, mỗi phần có **q_i câu hỏi**.
- Mỗi phần, chọn **1 câu ngẫu nhiên** → cần học trước các câu để trả lời đúng.
- Misha có **t giờ học**, mỗi câu học mất **1 giờ**.
- Cần tính **xác suất tối đa để vượt qua kỳ thi** (trả lời đúng hết **n câu**), khi phân bổ **t giờ học tối ưu**.
- Xuất ra xác suất dưới dạng **phân số**, với quy ước đặc biệt về **modulo $10^9 + 7$**.

Dịch đề bài:

Bài E. Mê cung Fractal

Ta xây dựng một **mê cung gồm các ô vuông và các bức tường giữa các ô**. Bắt đầu với **một ô vuông duy nhất** được bao quanh bởi bốn bức tường. Sau đó, **mở rộng mê cung nhiều lần**.

Quá trình mở rộng như sau:

- Dán **4 bản sao của mê cung hiện tại** lại với nhau, tạo thành **hình vuông 2 × 2**.
- Ở **tâm mê cung mới**, giữa bốn ô vuông, có **4 bức tường dài** kéo từ tâm ra, phân chia các bản sao:
  - Gọi các bức tường là:
    - **r** (phải),
    - **u** (trên),
    - **l** (trái),
    - **d** (dưới).
- Trong **3 bức tường**, ta tạo **một lối đi**, mỗi lối đi rộng **1 ô**.
- **Bức tường còn lại được giữ nguyên** (không có lối đi).

Quy ước mô tả mỗi lần mở rộng:

- Cho bốn số nguyên **r, u, l, d**:
  - Một trong bốn số là **0** (bức tường không có lối đi).
  - Ba số còn lại chỉ vị trí lối đi, tính từ tâm ra ngoài (1 là ngay sát tâm, 2 là đoạn thứ hai,...).

Mê cung sau cùng **có đúng một đường đi đơn giản giữa bất kỳ hai ô nào** (đường đi đơn giản là đường đi qua mỗi ô nhiều nhất một lần).

Nhiệm vụ:

- Trả lời **q truy vấn**. Mỗi truy vấn yêu cầu tính **độ dài đường đi đơn giản giữa hai ô A và B** (tính bằng số bước đi qua các ô liền kề).

Input:

- Dòng đầu: số nguyên **n** ($1 \le n \le 30$) – số lần mở rộng mê cung.
- **n dòng tiếp theo**, mỗi dòng gồm bốn số nguyên **r, u, l, d** – mô tả chi tiết từng lần mở rộng.
- Dòng tiếp: số nguyên **q** ($1 \le q \le 1000$) – số lượng truy vấn.
- **q dòng tiếp theo**, mỗi dòng gồm bốn số nguyên: **rowA, colA, rowB, colB** – tọa độ hai ô (hàng và cột từ 1 đến 2^n).

Output:

- Với mỗi truy vấn, in ra **độ dài đường đi đơn giản** giữa hai ô.

Tóm tắt đề bài:

- Xây dựng mê cung **fractal** qua **n lần mở rộng**:
  - Mỗi lần dán **4 bản sao** mê cung hiện tại → tạo thành mê cung lớn hơn.
  - Tại **tâm mê cung**, có **4 bức tường**, ba tường có lối đi, một tường bịt kín.
- Sau **n lần mở rộng**, mê cung có kích thước $\mathbf{2^n \times 2^n}$.
- Cần trả lời **q truy vấn**: tính **độ dài đường đi đơn giản giữa hai ô A và B** trong mê cung.

Dịch đề bài:

Bài F. Kiểm tra số nguyên tố (tương tác)

Đây là **bài toán tương tác**.

Ban giám khảo đã chọn **T số nguyên** ($1 \le T \le 10$), mỗi số nằm trong khoảng **[1; 10^{18}]**. Nhiệm vụ của bạn là **đoán từng số một**.

Cách tương tác:

- Giả sử bạn đang cố đoán số **x**.
- Ở mỗi lượt, bạn được phép **hỏi một số nguyên y** ($1 \le y \le 10^{18}$), bằng cách in ra:

```
? y
```

- Phản hồi sẽ là:
  - `"Prime"` nếu **x + y là số nguyên tố**.
  - `"Composite"` nếu **x + y là hợp số**.
- Khi bạn **đoán được x**, in ra:

```
! z
```

  - Nếu **z đúng là x**, phản hồi là `"Correct"` (chuyển sang đoán số tiếp theo hoặc kết thúc nếu hết T số).
  - Nếu **z không đúng hoặc sai cú pháp**, phản hồi là `"Busted"` (chương trình phải kết thúc ngay lập tức).

Giới hạn:

- **Tổng số truy vấn tối đa: 8750** (cho tất cả các số).
- Nếu vượt quá số lượng này hoặc hỏi sai định dạng hay giá trị **y không hợp lệ** → nhận `"Busted"`.

Ví dụ:

```
1           <- số lượng T = 1
Composite   <- hỏi ? 6 → x + 6 không nguyên tố
Prime       <- hỏi ? 5 → x + 5 nguyên tố
Prime       <- hỏi ? 3 → x + 3 nguyên tố
Prime       <- hỏi ? 1 → x + 1 nguyên tố
Correct     <- đoán ! 2 → x = 2 → đúng
```

Tóm tắt đề bài:

- Có **T số nguyên** (1 ≤ T ≤ 10), mỗi số trong **[1; 10^{18}]** cần đoán.
- Bạn **hỏi một số y**, nhận phản hồi **x + y** là **nguyên tố** hay **hợp số**.
- Khi đoán được x, **báo đáp án**.
- **Tổng số truy vấn ≤ 8750** cho toàn bộ T số.
- Mục tiêu: đoán **chính xác tất cả T số**.

Dịch đề bài:

Bài G. Ếch nhảy

Bạn đã từng chơi trò **"Tap the Frog"** chưa? Đó là một bộ sưu tập các mini-game giúp chú ếch hoàn thành các nhiệm vụ hàng ngày. Hôm nay, chúng ta sẽ nói về mini-game thứ hai có tên **"Ếch nhảy"**.

Trò chơi được mô phỏng như sau:

- Có một **dải lưới dài** trên mặt hồ, mỗi ô có thể chứa **lá sen** hoặc chỉ là **mặt nước**.
- Ở **ô ngoài cùng bên trái**, có một **lá sen** và **con ếch** đang ngồi trên đó.

- Trò chơi có **k nút nhảy**:
  - “Nhảy sang phải 1 ô”, “Nhảy sang phải 2 ô”, ..., “Nhảy sang phải k ô”.
- Giả sử rằng **ếch chỉ có thể nhảy lên các ô có lá sen**. Nếu còn **r < k ô** phía trước, bạn **không thể nhấn nút nhảy i ô** nếu **i > r**.

Nhiệm vụ:

- Bạn muốn thiết kế các **màn chơi** cho trò chơi này. Bạn có sẵn một số **mẫu bản đồ** (template), mỗi mẫu là một **xâu ký tự** gồm:
  - `'0'` – ô có **lá sen**.
  - `'~'` – ô là **mặt nước**.
- Bạn có thể **tạo mẫu mới** bằng cách:
  - Chọn hai mẫu s_i và s_j (có thể giống nhau hoặc khác nhau).
  - **Nối hai mẫu** lại để tạo mẫu mới $s_i s_j$.
- Với mỗi **mẫu bản đồ**, bạn cần tính **số cách điều khiển ếch** từ **ô ngoài cùng bên trái** đến **ô ngoài cùng bên phải** mà **không bao giờ nhảy lên ô nước**.
- **Hai cách đi khác nhau** nếu:
  - Số lần nhấn nút khác nhau, hoặc
  - Dù số lần nhấn nút bằng nhau, nhưng **một nút nhấn tại vị trí nào đó khác nhau**.
- Nếu **ô đầu hoặc ô cuối không có lá sen**, **số cách đi là 0**.
- Vì kết quả có thể rất lớn, bạn cần in **số cách đi mod 998244353**.

Input:

- Dòng đầu: ba số nguyên **n**, **k**, **a**:
  - **n**: số mẫu bản đồ ban đầu ($1 \le n \le 10^5$).
  - **k**: số lượng nút nhảy ($1 \le k \le 20$).
  - **a**: số mẫu bản đồ được tạo thêm ($1 \le a \le 10^5$).
- **n dòng tiếp theo**: mỗi dòng là một **xâu bản đồ** ban đầu (**tổng độ dài các xâu ≤ 10^5**).
- **a dòng tiếp theo**: mỗi dòng gồm hai số nguyên l_i và r_i, nghĩa là:
  - **mẫu thứ n + i** được tạo bằng cách nối **mẫu l_i** và **mẫu r_i** ($1 \le l_i, r_i < n + i$).

Output:

- In ra **n + a số nguyên**, mỗi số ứng với **số cách nhảy ếch từ trái qua phải** trên từng mẫu bản đồ, **modulo 998244353**.

Tóm tắt đề bài:

- Có **n mẫu bản đồ ban đầu** (dải ô gồm `'0'` và `'~'`).
- Có **k nút nhảy** (nhảy 1 → k ô).
- Tạo thêm **a mẫu mới** bằng cách **nối hai mẫu cũ** lại với nhau.
- Với mỗi mẫu bản đồ, tính **số cách điều khiển ếch** từ ô đầu đến ô cuối mà chỉ nhảy lên lá sen (khác nhau về số nút nhấn hoặc trình tự nhấn nút).
- In ra **số cách nhảy mod 998244353** cho từng mẫu.

Dịch đề bài:

Bài H. Máy đánh bạc

Misha đang ngồi trước một **máy đánh bạc** và muốn giành **giải độc đắc**. Máy có **k ô hiển thị**, tạo thành một **xâu dài k**. Mỗi ô có thể hiển thị:

- Một **chữ số thập phân** (0-9), hoặc
- Một **dấu hỏi chấm** `'?'`.

Ban đầu, tất cả các ô đều hiển thị **'?'**.

Máy cũng **lưu trữ bí mật một xâu k chữ số** – **xâu bí mật**. Nhưng **không phải mọi xâu k chữ số đều có thể là xâu bí mật**:

- Misha tìm thấy **sổ tay hướng dẫn**, trong đó (chương 007) liệt kê tất cả **các xâu bí mật hợp lệ**.

Mục tiêu: để giành **giải độc đắc**, Misha phải:

- Làm cho **các ô hiển thị khớp chính xác với xâu bí mật**.
- Nhấn **nút đỏ lớn** sau đó.

Hoạt động của máy khi nhấn nút đỏ:

- Nếu **xâu hiển thị khớp xâu bí mật** → **Misha thắng**.
- Nếu **không khớp**:
  - Máy chuyển **cả hai xâu** (xâu hiển thị và xâu bí mật) thành **số nguyên** (có thể có **số 0 đứng đầu**).
  - Máy thông báo **xâu nào lớn hơn**.

- Nếu **còn dấu hỏi '?'** trong xâu hiển thị, máy **không nói gì**.

Hành động của Misha:

- Trước mỗi lần nhấn nút (và giữa các lần), Misha có thể **thay đổi các ký tự** trong các ô hiển thị (từ **chữ số** hoặc **'?'** thành ký tự khác).
- **Mỗi lần thay đổi một ký tự → tốn 1 rúp**.

Misha muốn **chi ít tiền nhất có thể** để **chắc chắn thắng**.

Nhiệm vụ:

- Với **T test case**, xác định **số tiền tối thiểu** (số lần thay đổi ký tự) cần thiết để **Misha chắc chắn thắng**.

Input:

- Dòng đầu: số nguyên **T** ($1 \le T \le 10^4$).
- Mỗi test:
  - Dòng đầu: số nguyên **k** ($1 \le k \le 5$) – số ô hiển thị.
  - Dòng tiếp: xâu nhị phân **s** dài $\mathbf{10^k}$:
    - **s[i] = 1**: xâu số **i** (theo hệ thập phân, độ dài k) **có thể là xâu bí mật**.
    - **s[i] = 0**: **không thể là xâu bí mật**.
  - Có ít nhất một vị trí **s[i] = 1**.

Output:

- Với mỗi test, in ra **số rúp ít nhất cần chi** để **Misha chắc chắn thắng**.

Ví dụ:

Input	Output
2	
1	
1110001010	3
1	
1111111111	4

Tóm tắt đề bài:

- Máy đánh bạc có **k ô hiển thị** (hiện chữ số hoặc `'?'`).
- Có **một xâu bí mật** dài **k chữ số** (nằm trong danh sách hợp lệ cho trước).
- **Misha** có thể:
  - **Thay đổi ký tự** (tốn tiền).
  - **Nhấn nút đỏ** để biết xâu hiển thị **so với xâu bí mật**.
- **Mục tiêu**: tìm ra **số tiền ít nhất** để **chắc chắn thắng** (hiển thị đúng xâu bí mật).

Dịch đề bài:

Bài I. Tích

Cho một **dãy số nguyên** $(a_0, a_1, ..., a_{mk-1})$ có độ dài **mk**. Ta định nghĩa **trọng số (weight)** của dãy này là:

$$\prod_{i=0}^{m-1} a_{ik}$$

(tức là **lấy tích của các phần tử** đứng ở các vị trí **ik** trong dãy, với $i = 0$ đến $m - 1$).

Nhiệm vụ:

- Với mỗi **n₀** từ **1** đến **n**, hãy tính **tổng trọng số** của tất cả các dãy số ((a_0, a_1, ..., a_{mk-1})** thỏa mãn:
  - $1 \leq a_0 \leq a_1 \leq ... \leq a_{mk-1} \leq n_0$
- Kết quả **lấy modulo 998244353**.

Input:

- Gồm ba số nguyên **n**, **m**, **k**:
  - $1 \leq n, k \leq 250000$
  - $1 \leq m \leq 10^{18}$

Output:

- In ra **n dòng**, mỗi dòng là **kết quả với n_0 = 1, 2, ..., n** (theo thứ tự tăng dần), **modulo 998244353**.

Ví dụ:

Input	Output
2 2 2	
1	1
	10

Tóm tắt đề bài:

- Cho dãy số có độ dài **mk**.
- Tính **tổng trọng số** của tất cả các dãy **không giảm** trong đoạn **[1, n_0]** (với **n_0 = 1 → n**).
- **Trọng số** của dãy là **tích các phần tử** ở vị trí **ik** ($0 \leq i < m$).
- **In kết quả mod 998244353** cho từng **n_0** từ **1 → n**.

Dịch đề bài:

Bài J. Sinh cây ngẫu nhiên

Giả sử bạn cần **sinh ra một cây có gán nhãn đỉnh** (labeled tree) ngẫu nhiên **đều** trên **n đỉnh**. Xét thuật toán sau:

- Khi đồ thị **chưa liên thông**, lặp lại:
  - Chọn **ngẫu nhiên hai đỉnh khác nhau**.
  - Nếu chúng **chưa cùng thành phần liên thông**, thêm một cạnh nối hai đỉnh đó.

Việc này **chỉ cần DSU (Disjoint Set Union)** đơn giản để kiểm tra liên thông.

Tuy nhiên, **thuật toán này không tạo ra cây ngẫu nhiên đều**! Mục tiêu của bạn là **phân biệt** các cây sinh ra từ thuật toán này với các cây được **chọn ngẫu nhiên đều** từ tập hợp tất cả các cây gán nhãn.

Cụ thể:

- Có **2k cây** trên **n đỉnh**:
  - **k cây** được chọn **ngẫu nhiên đều** từ tất cả cây gán nhãn.
  - **k cây** sinh ra từ **thuật toán DSU** ở trên.
- Thứ tự các cây **ngẫu nhiên**.
- **Nhiệm vụ**: với mỗi cây, **phán đoán** nó là:
  - `` `"DSU"` `` – nếu nghĩ cây sinh từ thuật toán DSU.
  - `` `"Uniform"` `` – nếu nghĩ cây được chọn ngẫu nhiên đều.

Bạn **được phép sai**, nhưng:

- Tỷ lệ đúng phải đạt ít nhất **80%** (tức là sai tối đa **20%**).

Input:

- Dòng đầu: hai số nguyên **n** và **k**:
  - **n = 4, k = 2** (mẫu).
  - **n = 10^4, k = 50** (các test chính).
- Mỗi cây gồm **n - 1 dòng**, mỗi dòng **u v** – cạnh nối hai đỉnh.
- Tổng có **2k cây** → **(2k)(n - 1)** dòng mô tả các cạnh.

Output:

- In **2k dòng**, mỗi dòng là `"DSU"` hoặc `"Uniform"` – dự đoán của bạn cho từng cây.

Ví dụ:

Input	Output
4 2	
3 2	DSU
1 4	Uniform
1 3	DSU
2 1	Uniform
...	

Tóm tắt đề bài:

- Có **2k cây gán nhãn** trên **n đỉnh**:
  - **k cây ngẫu nhiên đều**.
  - **k cây từ thuật toán DSU**.
- Cần **phân loại** mỗi cây là `"DSU"` hay `"Uniform"`.
- **Yêu cầu đúng ≥ 80%** (sai tối đa **20%**).

Dịch đề bài:

Bài K. Vortex

Đây là một **bài toán chạy hai lần (run-twice)**.

- Cả **hai lần chạy**, bạn sẽ được cung cấp một **cây** (tree) với **n đỉnh**, nhưng:
  - Cây được **shuffle (xáo trộn)** nhãn đỉnh một cách **ngẫu nhiên**.
  - **Cấu trúc cây** (dạng cây không có nhãn) **giống nhau** ở cả hai lần chạy.
  - Tuy nhiên, **nhãn đỉnh có thể khác nhau** giữa hai lần chạy.

Nhiệm vụ của bạn:

- **Sắp xếp lại thứ tự nhãn các đỉnh** của cây ở mỗi lần chạy (bằng cách xuất ra một **hoán vị các đỉnh**).
- Sau khi sắp xếp lại:
  - **Hai cây** (sau hoán vị) phải **giống hệt nhau** về mặt **cấu trúc và nhãn đỉnh**.

Cụ thể:

- Với mỗi lần chạy, bạn **in ra một hoán vị** các đỉnh:
  - $p_1, p_2, ..., p_n$ cho lần chạy 1.
  - $q_1, q_2, ..., q_n$ cho lần chạy 2.
- Đáp án được coi là **đúng** nếu:
  - Với **mọi cặp i, j**:
    - Có **cạnh giữa** p_i **và** p_j trong cây 1 **nếu và chỉ nếu** có **cạnh giữa** q_i **và** q_j trong cây 2.

Input:

- Dòng đầu: số nguyên **n** ($2 \le n \le 2 \cdot 10^5$) – số đỉnh của cây.
- **n - 1 dòng tiếp theo**: mỗi dòng hai số nguyên **u v** – cạnh nối hai đỉnh.

Output:

- Một dòng gồm **n số nguyên** – hoán vị của các đỉnh (1 → n).

Ví dụ:

Lần chạy 1:

```
7
2 1
```

```
7 1
3 7
4 7
6 5
5 2
```

Output:

```
1 2 3 4 5 6 7
```

Lần chạy 2:

```
7
1 5
6 7
3 6
2 3
2 1
1 4
```

Output:

```
2 3 4 5 6 7 1
```

Tóm tắt đề bài:

- Có hai cây **cùng cấu trúc** nhưng **khác nhãn đỉnh** (do shuffle).
- Cần **hoán vị lại nhãn đỉnh** sao cho:

    - **Cấu trúc và nhãn đỉnh** giống nhau giữa hai lần chạy.
- **Xuất ra hoán vị các đỉnh** cho mỗi lần chạy.

Dịch đề bài:

Bài L. Tổng ngẫu nhiên

Bạn có một số **x**, ban đầu **x = 0**, và một **số nguyên tố p**. Tiếp theo, với mỗi **i = 1, 2, ..., m**:

- Bạn **tăng x lên a_i** với **xác suất q_i**.

Cuối cùng, yêu cầu tính **xác suất để x chia hết cho p**.

Cách tính xác suất q_i:

- Với mỗi bước **i**, cho hai số nguyên:
    - **a_i** ($0 \le a_i < p$) – giá trị có thể cộng vào x.
    - **r_i** ($0 \le r_i \le 10^8$) – trong đó:
        - **$q_i = r_i / 10^8$**.

Output:

- Xác suất cần tìm là một **phân số**, hãy in ra kết quả **dưới dạng một số nguyên**, theo công thức:

$$\text{Kết quả} = \left(\frac{x}{y}\right) \mod 998244353 = x \cdot y^{-1} \mod 998244353$$

- Trong đó, **y^{-1}** là **nghịch đảo modulo của y** theo modulo **998244353**.

Input:

- Dòng đầu: hai số nguyên **p** và **m** ($2 < p < 30000$; $1 \le m \le 10^6$; p là số nguyên tố).
- **m dòng tiếp theo**: mỗi dòng gồm hai số nguyên:
  - $\mathbf{a_i}$ ($0 \le a_i < p$).
  - $\mathbf{r_i}$ ($0 \le r_i \le 10^8$).

Output:

- In ra **một số nguyên** – **xác suất x chia hết cho p** sau quá trình, theo **modulo 998244353**.

Ví dụ:

Input	Output
2 3	499122177
0 100000000	
1 100000000	
1 50000000	

Tóm tắt đề bài:

- Bắt đầu từ **x = 0**.
- Với mỗi bước **i = 1 → m**:
  - **Tăng x lên a_i** với **xác suất $q_i = r_i / 10^8$**.
- Tính **xác suất cuối cùng để x chia hết cho p**.
- In ra kết quả **dưới dạng $x \cdot y^{-1} \bmod 998244353$**.