Mục lục

Bài A. Bữa tiệc

File dữ liệu vào:	`party.inp`
File kết quả:	`party.out`
Hạn chế thời gian:	0.1 giây
Hạn chế bộ nhớ:	512 Mb

Hôm nay Linh mời khách tới tham dự bữa tiệc sinh nhật của mình. Tham gia bữa tiệc có tất cả n người bao gồm cả Linh. Mọi người trong bữa tiệc ngồi xung quanh một chiếc bàn lớn gồm đúng n chỗ ngồi.

Linh rất muốn nói chuyện với nhiều khách nhưng không muốn to tiếng làm ảnh hưởng đến cả bàn. Vì vậy Linh thỉnh thoảng lại nhờ người bên trái hoặc bên phải đổi chỗ ngồi cho mình. Dĩ nhiên tất cả các khách mời đều sẵn sàng chấp nhận yêu cầu của Linh.

Sau khi tiễn khách về, Linh chợt nhớ ra mình để quên điện thoại ở chỗ cuối cùng mà Linh ngồi. Linh không nhớ chỗ Linh ngồi cuối cùng ở đâu, nhưng lại nhớ chỗ Linh ngồi đầu tiên và nhớ rằng Linh đã đổi chỗ đúng k lần với những người bên cạnh.

Bây giờ Linh muốn tính số lượng chỗ ngồi mà cô ấy có thể đã ngồi trong suốt bữa tiệc.

Yêu cầu: Cho biết n và k, hãy giúp Linh xác định số chỗ ngồi mà cô ấy có thể đã ngồi trong suốt bữa tiệc.

Dữ liệu vào

Gồm một dòng chứa hai số nguyên n và k ($3 \le n \le 10^9, 0 \le k \le 10^9$).

Kết quả

Ghi ra một số nguyên duy nhất là số lượng chỗ ngồi mà Linh có thể đã ngồi trong suốt bữa tiệc.

Ví dụ

party.inp	party.out
5 2	3
3 3	3

Giải thích

Trong ví dụ đầu tiên, Linh có hai lần đổi chỗ với người bên cạnh. Do đó Linh có thể đã ngồi lại chỗ ban đầu, hoặc là ngồi một trong hai vị trí cách chỗ đầu tiên đúng một chỗ ngồi. Trong ví dụ thứ hai, Linh có thể đã ngồi ở tất cả các chỗ xung quanh bàn.

Bài B. Chơi bi-a

File dữ liệu vào:	`bia.inp`
File kết quả:	`bia.out`
Hạn chế thời gian:	0.1 giây
Hạn chế bộ nhớ:	512 MB

Trên một bàn bi-a kích thước chiều ngang là a inh và kích thước chiều dọc là b, một viên bi được bắn đi từ chính giữa bàn. Sau khoảng thời gian $s > 0$ giây, viên bi trở lại vị trí mà nó được bắn đi sau khi va chạm m lần theo chiều dọc và n lần theo chiều ngang của bàn. Tìm góc bắn A ban đầu của viên bi (so với chiều cạnh chiều ngang) với giá trị nằm trong khoảng 0° và 90° và vận tốc bắn ban đầu của quả viên bi.

Giả thiết là va chạm với cạnh bàn là va chạm đàn hồi (không mất năng lượng), và vận tốc của quả viên bi so với hai cạnh của bàn là không đổi. Và cũng giả thiết rằng quả viên bi có bán kính là 0 và bàn bi-a không có lỗ.

Dữ liệu vào

Bao gồm nhiều dòng, mỗi dòng chứa 5 số nguyên không âm cách nhau bởi dấu cách. Năm số đó là a, b, s, m và n. Tất cả các số đều là số nguyên dương không lớn hơn 10 000.

Dòng cuối cùng chứa 5 số 0.

Kết quả

Đối với mỗi dòng trong dữ liệu vào ngoại trừ dòng cuối, ghi ra một dòng gồm có hai số thực (được làm tròn chính xác đến hai chữ số sau dấu phẩy) cách với nhau bằng dấu cách. Số đầu tiên là độ lớn của góc bắn A ban đầu và số thứ hai là vận tốc của viên bi theo đơn vị inh trên giây, theo mô tả phía trên.

Ví dụ

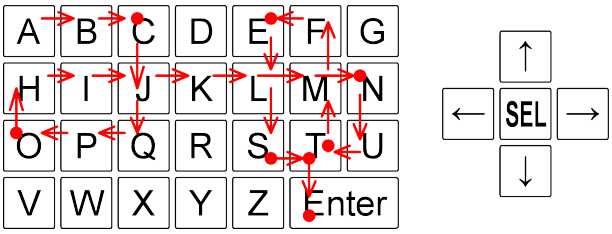
bia.inp	bia.out
100 100 1 1 1 200 100 5 3 4 201 132 48 1900 156 0 0 0 0 0	45.00 141.42 33.69 144.22 3.09 7967.81

Bài C. Gõ phím

File dữ liệu vào:	`stdin`
File kết quả:	`stdout`
Hạn chế thời gian:	0.5 giây
Hạn chế bộ nhớ:	512 MB

Đối với máy tính và đa số các thiết bị, các phím trên bàn phím được gõ trực tiếp để có được các ký tự mong muốn. Nhưng đối với các thiết bị nhỏ gọn như điện thoại, số lượng phím không đủ để thể hiện tất cả các ký tự, nên nhiều phương pháp đã được áp dụng. Một trong các giải pháp là dùng bàn phím ảo, có thể mô tả như sau:

Bàn phím ảo được hiển thị trên màn hình với đầy đủ các ký tự cần gõ, với một con trỏ phím di chuyển để chọn phím. Để di chuyển con trỏ, ta có 5 nút bấm, gồm 4 nút di chuyển theo hướng mũi tên, và nút select để xác nhận chọn ký tự. Ví dụ như hình vẽ:



Ta quy ước con trỏ sẽ bắt đầu từ phím góc trên trái của bàn phím ảo. Vậy để gõ chữ CONTEST ta phải thực hiện di chuyển và chọn ký tự như trên. Trong cách làm trên, cần 30 lượt bấm để gõ được chữ CONTEST (kết thúc bằng gõ phím enter). Chú ý là trên bàn phím ảo, mỗi ký tự có thể nằm trên nhiều ô vuông đơn vị liền nhau (vd như phím Enter trong hình).

Khi bấm một nút di chuyển, chưa chắc ô vuông liền kề đã là đích, mà là ô vuông của ký tự khác nó trên hướng đi. Nếu không thể chuyển sang phím khác ký tự theo hướng di chuyển thì con trỏ sẽ nằm nguyên mà không di chuyển. Ví dụ:

Yêu cầu: cho một đoạn văn bản là một chuỗi ký tự, hãy xác định số lần bấm ít nhất để gõ ra đoạn văn bản đó, bao gồm cả phím Enter ở cuối. Dữ liệu đảm bảo đoạn văn bản luôn có thể gõ được.

Dữ liệu vào

- Dòng đầu tiên gồm r và c là số dòng và số cột của bàn phím ($1 \le r, c \le 50$);
- R dòng tiếp theo là các ký tự của bàn phím, chỉ gồm các chữ cái hoa, chữ số, gạch ngang và * thể hiện cho ký tự enter. Mỗi ký tự có thể trải dài trên nhiều ô, nhưng phải đi liền nhau.
- Dòng cuối cùng là dòng chữ cần gõ, độ dài không quá 10000.

Kết quả

Ghi ra duy nhất một số là số lần bấm ít nhất để gõ ra đoạn văn bản cho trong input bao gồm cả phím Enter ở cuối.

Ví dụ

stdin	stdout
4 7 ABCDEFG HIJKLMN OPQRSTU VWXYZ** CONTEST	30
2 19 ABCDEFGHIJKLMNOPQZY X*****************Y AZAZ	19
6 4 AXYB BBBB KLMB OPQB DEFB GHI* AB	7

Bài D. Trò chơi đồ thị

File dữ liệu vào:	`graphgame.inp`
File kết quả:	`graphgame.out`
Hạn chế thời gian:	0.1 giây
Hạn chế bộ nhớ:	512 Mb

Khi học lý thuyết đồ thị, Ninh và Giang cùng nhau chơi một trò chơi trên đồ thị. Đầu tiên Ninh có một đơn đồ thị vô hướng G có N đỉnh được đánh số từ 1 đến N. Tiếp theo Ninh tiếp tục ghi ra giấy danh sách các đỉnh kề của từng đỉnh. Sau đó Giang thay đổi danh sách các đỉnh kề của 1 hoặc 2 đỉnh bằng cách thay đổi một số đỉnh kề với chúng. Cụ thể, nếu một đỉnh ban đầu có X đỉnh kề, Giang sẽ ghi ra đủ X đỉnh kề mà một số đỉnh trong đó khác với danh sách đỉnh kề ban đầu.

Yêu cầu: Cho biết số lượng đỉnh mà Giang đã thay đổi và danh sách các đỉnh kề của từng đỉnh sau khi đã thay đổi, hãy giúp Ninh tìm ra các đỉnh kề bị Giang thay đổi.

Dữ liệu vào

- Dòng thứ nhất chứa một số nguyên duy nhất $P \in \{1, 2\}$ là số lượng đỉnh mà Giang đã thay đổi danh sách các đỉnh kề của chúng.
- Dòng thứ hai ghi duy nhất một số nguyên dương N ($3 \leq N \leq 10^5$) là số lượng đỉnh của đồ thị G.
- Dòng thứ i trong số N dòng tiếp theo mô tả danh sách kề của đỉnh i đã bị Giang thay đổi:
  - đầu tiên là một số nguyên dương $K_i \leq N - 1$, là số lượng đỉnh kề của đỉnh i;
  - K_i số nguyên tiếp theo là các đỉnh kề của đỉnh i.
- Biết rằng $K_1 + K_2 + \ldots + K_N \leq 4 \times 10^5$.

Các số trên cùng một dòng cách nhau bởi dấu cách.

Kết quả

- Nếu $P = 1$, ghi ra duy nhất một đỉnh mà Giang đã thay đổi.
- Nếu $P = 2$, ghi ra trên cùng dòng 2 đỉnh mà Giang đã thay đổi theo thứ tự tăng dần.

Dữ liệu đảm bảo luôn có lời giải và có duy nhất một lời giải.

Ví dụ

graphgame.inp	graphgame.out
2 7 4 7 3 2 4 2 6 1 4 7 6 4 1 4 1 3 5 6 2 4 1 3 7 5 1 1 3	1 6

Bài E. Đổi chỗ

File dữ liệu vào:	`switch.inp`
File kết quả:	`switch.out`
Hạn chế thời gian:	0.1 giây
Hạn chế bộ nhớ:	256 MB

Người ta định ra một quan hệ hai ngôi đối xứng trên tập hợp các chữ số $\{0, 1, \ldots, 9\}$ gọi là quan hệ có thể đổi chỗ được.

Một số nguyên bất kỳ được biến đổi theo cách sau: đổi chỗ chữ số thứ i và chữ số thứ $i + 1$ nếu hai chữ số này thuộc quan hệ đang xét. Phép biến đổi như thế được ký hiệu là i (các chữ số trong một số nguyên được xếp thứ tự 1,2,... theo chiều từ trái sang phải).

Hãy tìm dãy biến đổi ngắn nhất trên một số nguyên cho trước sao cho thu được một số nguyên lớn nhất.

Dữ liệu vào

Dòng đầu ghi số cặp chữ số có thể đổi chỗ được.

Các dòng tiếp theo, mỗi dòng ghi một cặp chữ số có thể đổi chỗ được, các chữ số ghi cách nhau ít nhất một dấu trắng.

Dòng cuối cùng ghi số nguyên cần biến đổi, các chữ số ghi sát nhau. Số nguyên này có số chữ số không vượt quá 1000 chữ số, có thể có những chữ số 0 ở đầu.

Kết quả

Dòng đầu ghi số phép biến đổi (có thể bằng 0).

Dòng thứ hai ghi số nguyên lớn nhất thu được.

Ví dụ

switch.inp	switch.out
7 1 7 4 3 6 4 5 6 5 4 9 2 0 8 3546	3 4365

Bài F. Thêm cạnh

File dữ liệu vào:	`addedge.inp`
File kết quả:	`addedge.out`
Hạn chế thời gian:	0.1 giây
Hạn chế bộ nhớ:	256 MB

Bình đang quan sát một đồ thị vô hướng. Bình băn khoăn liệu trong đồ thị này có tồn tại hai đỉnh nào mà sau khi nối hai đỉnh đó lại với nhau thì đồ thị sẽ có thêm duy nhất một chu trình đơn mới. Nhắc lại chu trình đơn là một chuỗi các đỉnh bắt đầu và kết thúc tại cùng một đỉnh mà không được phép lặp lại đỉnh ngoài đỉnh đầu và đỉnh cuối.

Yêu cầu: hãy giúp Bình tính số lượng cặp đỉnh thoả mãn điều kiện trên.

Dữ liệu vào

Dòng đầu tiên chứa hai số nguyên dương n, m ($n, m \leq 10^5$) tương ứng là số đỉnh và số cạnh của đồ thị.

m dòng tiếp theo mỗi dòng chứa hai số nguyên dương u, v ($u, v \leq n$) là hai đỉnh của một cạnh.

Kết quả

Ghi ra duy nhất một số là số cặp đỉnh tìm được.

Ví dụ

addedge.inp	addedge.out
5 4 1 2 2 3 3 4 4 5	6
5 5 1 2 2 3 1 3 3 4 4 5	1

Bài G. Thành phần liên thông mạnh

File dữ liệu vào:	`stdin`
File kết quả:	`stdout`
Hạn chế thời gian:	1 giây
Hạn chế bộ nhớ:	256 MB

Bài toán yêu cầu tính số thành phần liên thông mạnh của đồ thị.

Dữ liệu vào

- Dòng đầu chứa ba số nguyên dương $n \leq 10^4$, $m \leq 2 \times 10^4$, $k \leq 10^4$;
- m dòng sau mỗi dòng chứa hai số nguyên dương u và v tương ứng với một cung (u, v);
- k dòng tiếp theo mỗi dòng chứa hai số nguyên dương i và j tương ứng với một thao tác thêm cung (i, j) vào đồ thị. Mỗi dòng là một yêu cầu truy vấn: sau thao tác thêm cung này yêu cầu chương trình của bạn trả lời số thành phần liên thông mạnh của đồ thị.

Kết quả

- Dòng đầu chứa duy nhất 1 số nguyên là số thành phần liên thông mạnh tìm được của đồ thị.
- k dòng tiếp theo mỗi dòng ghi ra số thành phần liên thông mạnh tìm được của đồ thị tương ứng với yêu cầu truy vấn trong file dữ liệu vào.

Ví dụ

stdin	stdout
7 5 4 1 2 2 3 3 1 5 1 4 5 3 5 6 7 7 4 4 6	5 4 4 4 2

Bài H. Danh sách móc nối

File dữ liệu vào: `linklist.inp`
File kết quả: `linklist.out`
Hạn chế thời gian: 0.1 giây
Hạn chế bộ nhớ: 256 MB

0	1	2	3	4	5	6	7	8	9
2	5	4	4	-1	1	-1	3	0	8

Hàng dưới bảng trên chứa thông tin của mảng A kèm các chỉ số mảng ở hàng trên. Mảng A chứa các con trỏ của một danh sách móc nối tuyến tính. Trong bài toán này, một con trỏ đơn giản chỉ là một giá trị nguyên. Con trỏ tại nốt đầu tiên được lưu tại A[0], có nghĩa là giá trị A[0] chỉ ra vị trí của nốt thứ hai. Con trỏ của nốt thứ hai được lưu tại A[A[0]], rồi con trỏ của nốt thứ ba được lưu tại A[A[A[0]]], ... Con trỏ có giá trị -1 biểu thị điểm cuối của danh sách móc nối.

Trong ví dụ trên, giá trị của A[0] là 2 nghĩa là con trỏ thứ hai được lưu tại A[2]. Giá trị của A[2] là 4 nghĩa là con trỏ thứ 3 được lưu tại A[4]. Giá trị của A[4] là -1 nghĩa là không còn nốt nối sau đó nữa. Như vậy chuỗi con trỏ trong danh sách móc nối trên sẽ gồm 3 nốt như sau: A[0]=2 → A[2]=4 → A[4]=-1.

Cho trước các giá trị của mảng con trỏ A và biết rằng có một vị trí đã bị thay đổi giá trị mới nhưng không được biết thông tin về vị trí và giá trị mới đó. Lưu ý là giá trị mới có thể bằng giá trị cũ, nghĩa là bản chất mảng không bị thay đổi. Đây là một ứng dụng trong hình sự để truy vết tội phạm. Vì vậy để khôi phục lại mảng ban đầu thì cần sửa giá trị tại đúng một vị trí trong mảng mới sao cho mảng được sửa biểu diễn một danh sách móc nối có nhiều nốt nhất. Với ví dụ trên:

- Đổi giá trị A[4] thành 6 sẽ tạo ra danh sách móc nối với 4 nốt.
- Đổi giá trị A[0] thành 7 sẽ tạo ra danh sách móc nối với 4 nốt.
- Đổi giá trị A[0] thành 9 sẽ tạo ra danh sách móc nối sai.
- Đổi giá trị A[2] thành 7 sẽ tạo ra danh sách móc nối với 5 nốt.

Trong số tất cả các cách đổi có thể, kể cả cách không đổi gì, thì danh sách móc nối mới tạo ra với 5 nốt là dài nhất. Do vậy nhiều khả năng giá trị ban đầu của A[2] là 7.

Yêu cầu: tìm cách thay đổi giá trị một nốt để có được mảng biểu diễn danh sách móc nối dài nhất bắt đầu từ vị trí 0.

Dữ liệu vào

Dòng đầu tiên chứa duy nhất một số nguyên N là kích thước của mảng A.

Dòng thứ hai ghi ra N số nguyên là các giá trị trong mảng, bắt đầu từ A[0]. Mỗi số nguyên ≥ -1 và $< N$.

Kết quả

Ghi ra duy nhất một số là số nốt của danh sách móc nối có nhiều nốt nhất tìm được.

Ví dụ

linklist.inp	linklist.out
10 2 5 4 4 -1 1 -1 3 0 8	5

Hạn chế

- Subtask 1 (20%): $1 < N \leq 20$
- Subtask 2 (20%): $20 < N \leq 3000$ và số lượng nốt tối đa của danh sách móc nối tìm được không quá 100.
- Subtask 3 (20%): $20 < N \leq 3000$
- Subtask 4 (40%): $3000 < N \leq 20000$