HƯỚNG DẪN GIẢI ĐỀ THI VÒNG LOẠI CODE TOUR UNI 2020 HCMUT

A. DISTINGUISHED NUMBERS

Solution:

C++	https://ideone.com/1PQaI4
Java	https://ideone.com/ktucSU
Python	https://ideone.com/oYDjM1

Tóm tắt đề:

Cho một dãy số gồm N số nguyên dương. Hãy liệt kê các số phân biệt trong dãy số, mỗi số in 1 lần, theo thứ tự xuất hiện của số đó trong dãy.

Input

- Dòng đầu tiên chứa số nguyên dương N (1 ≤ N ≤ 100.000) là số lượng các số trong dãy số.
- Dòng tiếp theo chứa N số nguyên dương x (1 ≤ x ≤ 1.000), mỗi số cách nhau 1 khoảng trắng.

Output

Liệt kê các số phân biệt trong dãy số, mỗi số in 1 lần, theo thứ tự xuất hiện của số đó.

9 5 1000 5 2 3 1 2 5 8	5 1000 2 3 1 8

Giải thích:

Theo đề bài, ta chỉ in các số một lần duy nhất theo thứ tự xuất hiện của chúng. Như vậy ta có:

5 – In ra vì chưa có 5 ở trước

1000 – In ra vì trước đó không tồn tại 1000

5 – Không in vì đã có 5 phía trước

2 – In vì chưa có 2

3 – In vì chưa có 3

1 – In vì chưa có 1

2 – Không in vì đã có 2

5 – Không tin vì đã có 5

8 – In vì chưa có 8

Hướng dẫn giải:

Giả sử ta đã đọc toàn bộ N số vào một mảng, để giải quyết bài toán này ta chỉ cần xét lần lượt từng phần tử của mảng và kiểm tra xem trước đó nó đã xuất hiện hay chưa. Như vậy với cách làm này, với mỗi phần tử của mảng, ta cần phải dùng thêm một vòng lặp nữa để duyệt lại những phần tử trước đó nhằm mục đích kiểm tra xem phần tử đó đã xuất hiện hay chưa. Độ phức tạp của cách làm này là $O(N^2)$ và với N có giá trị tối đa là 10^5, cách làm này không thể giải quyết được những test có giá trị N lớn trong thời gian quy định là 1 giây.

Để ý rằng giá trị của các phần tử x_i chỉ nằm trong khoảng từ 1 đến 1000, do đó ta có thể dùng thêm một mảng nữa để thực hiện kỹ thuật đặt cờ nhằm đánh dấu lại những giá trị nào đã xuất hiện trước phần tử mình đang xét. Nhờ vào mảng đánh dấu này, ta không cần vòng lặp kiểm tra bên trong nữa mà đơn thuần chỉ cần một vòng lặp để duyệt qua các phần tử trong mảng mà thôi.

Độ phức tạp: O(N) với N là số lượng phần tử của dãy số.

B. LOTTERY

Solution:

C++	https://ideone.com/DIWyay
Java	https://ideone.com/sUdK2i
Python	https://ideone.com/3l6Go0

Tóm tắt đề:

Cho thông tin về tấm một vé số đã được mua và kết quả giải đặc biệt của N ngày. Hỏi tấm vé đó trúng giải đặc biệt, khuyến khích hay không trúng gì cả.

Input

- Dòng đầu tiên, gồm 3 con số nguyên dương và một chuỗi gồm 6 chữ số, cách nhau 1 khoảng trắng, đại diện cho ngày, tháng, năm và giá trị của vé số cần dò.
- Dòng thứ hai, gồm một số nguyên dương N (1 ≤ N ≤ 100.000) là số lượng các kết quả trúng số độc đắc.
- N dòng tiếp theo, mỗi dòng gồm 3 con số nguyên dương và một chuỗi 6 chữ số, cách nhau 1 khoảng trắng, đại diện cho ngày, tháng, năm và giá trị của vé số giải đặc biệt.

Output

Cho biết vé số đó trúng giải đặc biệt (JACKPOT), giải khuyến khích (CONSOLATION), hay không trúng giải (GOODLUCK).

5 2 2020 552202 4 3 2 2020 012345 5 2 2020 552202 7 2 2020 227702 10 2 2020 101002	JACKPOT
5 2 2020 551202 4 3 2 2020 012345 5 2 2020 552202 7 2 2020 227702 10 2 2020 101002	CONSOLATION

5 2 2020 551202 4 3 2 2020 012345 5 2 2020 552021 7 2 2020 227702 10 2 2020 101002	GOODLUCK
5 2 2020 225502 4 3 2 2020 012345 5 2 2020 552202 7 2 2020 227702 10 2 2020 101002	GOODLUCK

Giải thích:

- Ở ví dụ đầu tiên, ngày 5/2/2020 xổ ra giải đặc biệt là 552202 nên vé trúng giải đặc biệt;
- Ở ví dụ thứ hai, ngày 5/2/2020 xổ ra giải đặc biệt là 55**2**202 nhưng vé được mua có số là 55**1**202, vì vé chỉ khác biệt 1 số nên vé này chỉ trúng giải khuyến khích;
- Ở ví dụ thứ ba, vé được mua có số là 55**1202** nhưng giải đặc biệt trong ngày đó là 55**2021**, vì số lượng số khác biệt là 4 nên vé này không trúng giải nào cả.

Hướng dẫn giải:

Đây là một bài toán đơn giản, đầu tiên ta chỉ cần so sánh xem ngày của vé mình mua có trùng với ngày xổ hay không. Vì N ngày khác nhau (mỗi ngày chỉ có một vé đặc biệt) nên ta có thể làm như sau:

- Nếu trùng ngày và dãy số khớp hoàn toàn thì ta in ra JACKPOT và thoát chương trình;
- Nếu trùng ngày và dãy số sai đúng một vị trí thì ta in ra CONSOLATION và thoát chương trình;
- Nếu trùng ngày nhưng dãy số sai từ 2 vị trí trở lên, ta in ra GOODLUCK và thoát chương trình.

Độ phức tạp: O(N) với N là số lượng ngày xổ.

C. SHOOTING GAME

Solution:

C++	https://ideone.com/u87DxC
Java	https://ideone.com/GwpisC
Python	https://ideone.com/6Axpio

Tóm tắt đề:

John đi chơi trò chơi bắn súng, ban đầu Ban Tổ Chức sẽ phát cho mỗi đội 1 số áo chống đạn. Tuy nhiên, mỗi người sẽ chỉ mặc vừa một số áo chống đạn trong những chiếc áo đã được phát. John là nhóm trưởng vì thế cậu cần tính toán làm sao để số người mặc được áo chống đạn là nhiều nhất có thể.

Input:

Có T ($T \leq 10$) là số lượng trận chiến. Mỗi trận bao gồm các thông tin:

- Số N là số đại diện cho số lượng áo chống đạn ($1 \leq N \leq 200$).
- Số M là số lượng người chơi đang có ($1 \leq M \leq 200$). M dòng tiếp theo mỗi dòng đại diện cho thông tin của một người có thể mặc được áo chống đạn nào.
    - Số đầu tiên là số lượng áo chống đạn có thể mặc được.
    - Các số tiếp theo cách nhau bởi dấu khoảng trắng lần lượt là số thứ tự các áo mà người đó có thể mặc được.

Output:

In ra số áo tối đa bạn có thể mặc vừa cho người chơi. Theo định dạng Test Case #: <Kết quả> . Với # là đại diện cho số trận chiến bắt đầu từ số 1.

Thông tin các trò chơi cách nhau bởi dấu xuống dòng.

Ví dụ:

3 3 3 1 1 2 1 2 2 2 3 5 6 1 2	Test Case #1: 3 Test Case #2: 5 Test Case #3: 1

2 2 3 2 2 1 3 1 2 5 1 2 4 1 4 2 3 2 2 0 2 1 2	

Giải thích ví dụ:

Ở ví dụ thứ nhất, ta có thể phân phối áo chống đạn như sau:

- Người chơi 1 mặc áo chống đạn 1
- Người chơi 2 mặc áo chống đạn 2
- Người chơi 3 mặc áo chống đạn 3

Ở ví dụ thứ hai, ta có thể phân phối áo chống đạn như sau:

- Người chơi 1 mặc áo chống đạn 3
- Người chơi 2 mặc áo chống đạn 1
- Người chơi 3 mặc áo chống đạn 5
- Người chơi 5 mặc áo chống đạn 4
- Người chơi 6 mặc áo chống đạn 2

Ở ví dụ thứ ba, người chơi 1 không mặc vừa áo chống đạn nào. Ta có thể cho người chơi 2 có thể mặc áo chống đạn 1.

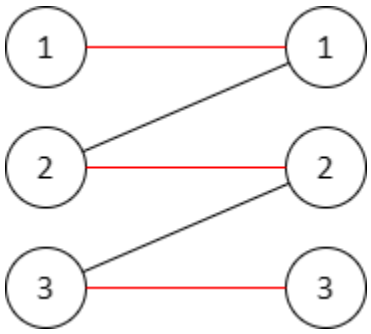
Hướng dẫn giải:

Ta sẽ xây dựng một đồ thị hai phía, trong đó:

- M đỉnh ở phía bên trái đại diện cho M người chơi (đỉnh i tương ứng với người chơi i)
- N đỉnh ở phía bên phải đại diện cho N áo chống đạn (đỉnh j tương ứng với áo chống đạn j).
- Nếu người i mặc vừa áo chống đạn j thì ta sẽ thêm cạnh giữa đỉnh i của phía bên trái với đỉnh j của phía bên phải

Khi đó, đáp án sẽ là kích thước của một tập cạnh lớn nhất sao cho các đỉnh chỉ kề với tối đa một cạnh trong tập cạnh được chọn (mỗi người chơi chỉ mặc tối đa một áo chống đạn, cũng như mỗi áo chống đạn chỉ được mặc bởi tối đa một người chơi).

Hình vẽ dưới đây minh họa đồ thị hai phía ứng với test ví dụ thứ nhất (các cạnh tô màu đỏ là tập cạnh lớn nhất cần tìm):



Bài toán nói trên chính là bài toán tìm bộ ghép cực đại trong đồ thị hai phía. Đây là một bài toán kinh điển, có thể được giải bởi thuật toán đường mở với độ phức tạp $O(VE)$ hoặc thuật toán Hopcroft-Karp với độ phức tạp $O\left(E\sqrt{V}\right)$, với V là số đỉnh, E là số cạnh của đồ thị hai phía.

Các bạn có thể tìm hiểu thêm về các thuật toán trên qua các bài viết sau:

- Thuật toán đường mở: https://www.geeksforgeeks.org/maximum-bipartite-matching/
- Thuật toán Hopcroft-Karp: https://www.geeksforgeeks.org/hopcroft-karp-algorithm-for-maximum-matching-set-1-introduction/

Độ phức tạp: $O(TNM(N+M))$ (nếu dùng thuật toán đường mở) với T là số bộ dữ liệu vào, N là số áo chống đạn, M là số người chơi.

D. THE GAME

Solution:

C++	https://ideone.com/JD2ArM
Java	https://ideone.com/9GEnw7
Python	https://ideone.com/69lQN9

Tóm tắt đề:

Cho một số quân lính được ghép cặp với nhau để chia thành nhiều phe. Hỏi có thể chọn được duy nhất một phe có số quân lính nhiều nhất hay không?

Input

- Dòng đầu tiên chứa 2 số nguyên m và n lần lượt là số lượng lính trong trò chơi và số lượng ghép nối các quân lính lại với nhau. *n* (1≤ n ≤ 50.000) m (1≤ m ≤ 50.000).
- Các dòng tiếp theo, mỗi dòng gồm 2 số nguyên dương a, b, thể hiện là quân lính a ghép với quân lính b.

Output

Nếu có thể tìm ra được 1 nhóm nhiều quân lính thì in ra YES và số lượng quân lính trong nhóm có nhiều quân nhất, ngược lại in ra NO.

10 7 1 2 4 5 1 3 7 8 1 9 4 6 2 3	YES 4

Giải thích:

Ở ví dụ này chúng ta có 10 người lính (đánh số thứ tự từ 1 tới 10) được chia thành 4 phe:

- Phe 1 có 4 người: 1, 2, 3, 9.

- Phe 2 có 3 người: 4, 5, 6.
- Phe 3 có 2 người: 7, 8.
- Phe 4 có 1 người: 10.

Như vậy phe nhiều người nhất có 4 người, và chỉ có đúng 1 phe có 4 người, nên kết quả là YES.

Hướng dẫn giải:

Chúng ta đưa bài toán về một đồ thị vô hướng có **M** đỉnh mà mỗi đỉnh đại diện cho 1 người lính, và đỉnh đại diện cho 2 người lính có cạnh nối khi 2 người lính đó cùng phe.

Như vậy nhiệm vụ bây giờ là tìm Thành phần liên thông có nhiều đỉnh nhất của đồ thị trên, và kiểm tra số lượng đỉnh trong thành phần liên thông đó có duy nhất hay không.

Để tìm các thành phần liên thông của đồ thị, các cách phổ biến là các thuật toán duyệt đồ thị như Depth-first Search, Breadth-first Search hay cấu trúc Disjoint-set Union.

Độ phức tạp: O(M + N) với M là số lượng người lính và N là số lượng các quan hệ cùng phe giữa các người lính.

E. THE POSTER

Solution:

C++	https://ideone.com/eygWRR
Java	https://ideone.com/1HFDlw
Python	https://ideone.com/tX9VCb

Tóm tắt đề:

Có N tấm áp phích. Mỗi tấm áp phích sẽ dán từ trái sang hoặc từ phải sang. Áp phích đầu tiên được dán bắt đầu tại vị trí số 0 của hệ trục tọa độ Ox. Các tấm tiếp theo sẽ được dáp tiếp lên từ vị trí kết thúc của tấm trước.

Hỏi phần bị chồng lên nhau (đoạn có ít nhất 2 áp phích cùng được dán qua) có độ dài là bao nhiêu.

Input

- Dòng đầu tiên chứa số **N (1 ≤ N ≤ 100.000)** số lượng tấm áp phích cần dán.
- **N** dòng tiếp theo lần lượt là thông tin của tấm áp phích.
  - Đầu tiên chữ P hoặc T nghĩa là dán áp phích sang phải hoặc sang trái
  - Tiếp theo là số nguyên dương thể hiện độ dài của áp phích. **($\leq 10^9$).**

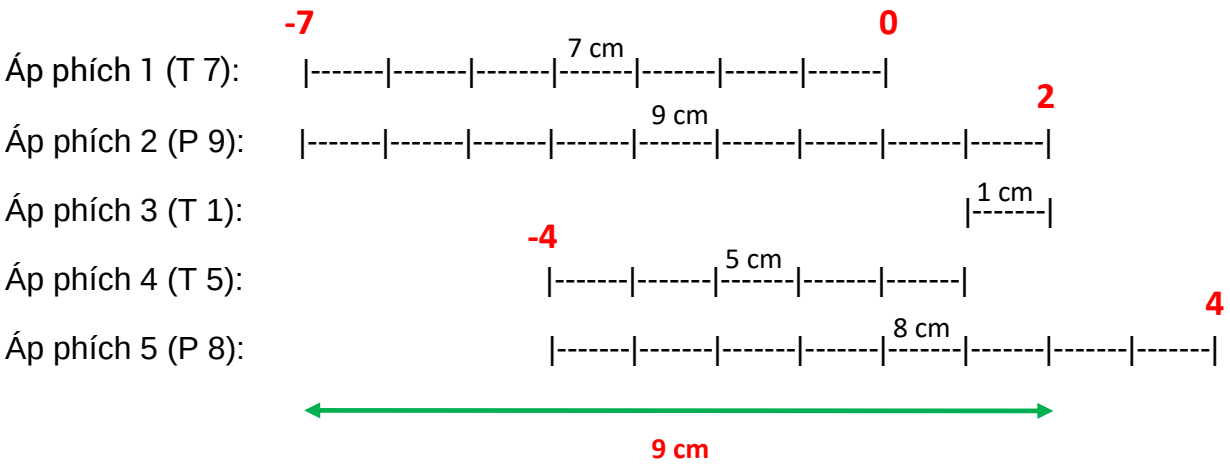
Output

In ra độ dài phần bị chồng lên nhau của các áp phích.

Ví dụ

5 T 7 P 9 T 1 T 5 P 8	9

Giải thích:



Hướng dẫn giải:

Chúng ta xét từng đoạn thẳng mà 2 đầu của nó là đầu của một trong các các áp phích. Nếu có nhiều hơn 1 áp phích bao phủ đoạn này thì ta cộng độ dài đoạn này vào kết quả.

Như vậy đầu tiên cần xây dựng tất cả các toạ độ là đầu mút của các áp phích (gọi là mảng x[]) rồi sắp xếp các toạ độ này lại theo thứ tự tăng dần, sau đó xét từng cặp toạ độ liên tiếp x[i], x[i + 1].

Để đếm được có bao nhiêu áp phích bao phủ đoạn này, chúng ta cần duy trì số lượng các áp phích bắt đầu từ x[i] trở về trước và kết thúc sau x[i]:

- Nếu x[i] là đầu mút trái của một áp phích thì tăng currentOverlaps lên 1.
- Nếu x[i] là đầu mút phải của một áp phích thì giảm currentOverlaps đi 1.
- Sau đó so sánh currentOverlaps với 1 để quyết định xem có cộng khoảng cách (x[i + 1] - x[i]) vào kết quả hay không.

Độ phức tạp: O(NlogN) với N là số lượng áp phích.