|  |  |
| --- | --- |
|  |  |

# Môn thi: Tin học

Thời gian làm bài:***180*** *phút, không kể thời gian giao đề*

Ngày thi thứ 2

#### Tổng quan bài thi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên bài** | **File nguồn** | **File Input** | **File Output** | **Thời gian** |
| Cho táo | apples.\* | apples.in | apples.out | 1 giây |
| Đếm số lượng đường đi | colpaths.\* | colpaths.in | colpaths.out | 2 giây |
| Trò chơi trên lá bài | cardgame.\* | cardgame.in | cardgame.our | 1.5 giây |

**Dấu \* là Pas hoặc Cpp tương ứng với ngôn ngữ lập trình Pascal hoặc C++**

**Hãy lập trình giải các bài toán sau:**

**Câu I.** (*6 điểm*) **Cho táo**

Nam cùng với N - 1 người bạn còn lại của mình, mỗi người hiện đang có một số lượng táo. Vì là bạn thân của nhau, từng người một sẽ lần lượt cho nhau táo.

Để cho thuận tiện, N người sẽ được đánh số từ 1 tới N, trong đó người thứ nhất là Nam.

Đầu tiên, Nam sẽ cho mỗi một trong N - 1 người bạn còn lại của mình một lượng táo đúng bằng số lượng táo mà người đó đang có. Nói cách khác, nếu một người bạn của Nam đang có X quả táo, Nam sẽ cho người bạn này thêm X quả táo nữa.

Tiếp theo, người thứ hai cũng sẽ cho mỗi một trong N - 1 người bạn còn lại của mình một lượng táo đúng bằng số lượng táo mà người bạn đó đang có.

Sau đó, người thứ ba, bốn, …, cho tới người thứ N đều lần lượt làm như thế.

Sau khi người thứ N phân phát táo xong, mỗi người đều đếm lại số lượng táo mà mình đang có. Ngạc nhiên thay, mỗi người đều sở hữu một lượng táo như nhau!

**Yêu cầu:** Biết rằng ban đầu Nam có A quả táo. Bạn hãy xác định số lượng táo mà N - 1 người còn lại có ban đầu.

**Input:** đọc từ file **apples.in**

Dòng đầu tiên chứa số nguyên T (1 <= T <= 1000) là số lượng test.

T dòng sau, mỗi dòng gồm 2 số nguyên dương N, A (1 <= N <= 50, 1 <= A <= 10^16) mô tả một test.

**Output:** ghi ra file **apples.out**

Với mỗi test, in ra trên một dòng N số nguyên là số lượng táo ban đầu của N người. Nếu có nhiều đáp án, in ra một đáp án bất kỳ. Ngược lại nếu không có đáp án, in ra -1.

**Subtask:**

*Subtask 1 (15%):* T <= 50, N = 2, A <= 10^6.

*Subtask 2 (15%):* T <= 50, N = 3, A <= 10^9

*Subtask 3 (40%):* T <= 50, N <= 50, A <= 10^5..

*Subtask 4 (30%):* Không có ràng buộc gì thêm.

**Ví dụ**

|  |  |
| --- | --- |
| **apples.in** | **apples.out** |
| 2 3 13 2 10 | 13 7 4 10 6 |

**Câu II.** (*7 điểm*) **Đếm số lượng đường đi**

Cho một đồ thị đơn, vô hướng gồm có N đỉnh và M cạnh. Đỉnh được đánh số từ 1 tới N. Mỗi đỉnh đã được tô một trong K màu, các màu được đánh số từ 1 tới K.

Một đường đi trên đồ thị gồm một chuỗi các đỉnh trong đó 2 đỉnh liên tiếp trên chuỗi đó có cạnh nối với nhau. Một đường đi được gọi là đường đi đơn khi không có đỉnh nào xuất hiện hơn một lần trên đường đi đó. Đường đi là một chuỗi đỉnh có thứ tự, do đó các đường đi 5->6->7, 5->7->6 và 7->6->5 được xem là khác nhau.

**Yêu cầu:** Đếm số lượng đường đi đơn khác nhau trong đó không có 2 đỉnh nào cùng màu xuất hiện trên đường đi.

**Input:** đọc từ file **colpaths.in**

Dòng đầu tiên chứa 3 số nguyên dương N, M, K.

Dòng tiếp theo chứa N số nguyên dương không quá K, số thứ i (1 <= i <= N) là màu của đỉnh i.

M dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v (1 <= u, v <= N, u khác v) mô tả có một cạnh nối giữa 2 đỉnh u và v.

**Output:** ghi ra file **colpaths.out**

In ra trên một dòng số lượng đường đi đơn khác nhau trong đó không có 2 đỉnh nào cùng màu xuất hiện trên đường đi. Dữ liệu đảm bảo đáp án không vượt quá 10^18.

**Subtask:**

*Subtask 1 (25%):* 1 <= N, M <= 100, 1 <= K <= 4.

*Subtask 2 (25%):* 1 <= N, M <= 300000, 1 <= K <= 3.

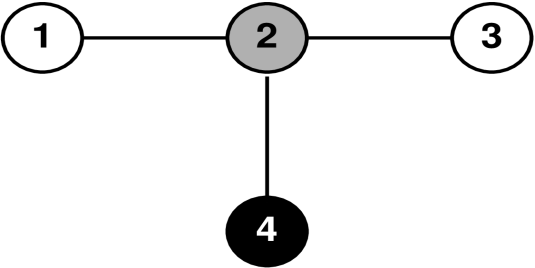
*Subtask 3 (25%):* 1 <= N, M <= 300000, 1 <= K <= 4.

*Subtask 4 (25%):* 1 <= N, M <= 100000, 1 <= K <= 5.

**Ví dụ**

|  |  |
| --- | --- |
| **colpaths.in** | **colpaths.out** |
| 4 3 3 1 2 1 3 1 2 2 3 4 2 | 10 |

*Hình minh họa ví dụ trên:*

****

|  |  |
| --- | --- |
| **colpaths.in** | **colpaths.out** |
| 9 11 4 1 2 3 4 1 2 1 2 2 1 2 1 3 2 3 2 4 3 6 6 2 6 5 4 3 4 5 7 8 9 8 | 70 |

**Câu III** (*7 điểm*). **Trò chơi trên lá bài**

Có N lá bài. Mỗi lá bài có một mặt màu xanh và mặt kia màu đỏ. Đồng thời, ở mỗi mặt có một số nguyên dương ghi trên đó.

Có một trò chơi với luật chơi như sau:

* Người chơi bắt đầu với 0 điểm. Ban đầu, N lá bài được đặt vào trong một chiếc hộp.
* Trong khi trong hộp còn có ít nhất 2 lá bài, người chơi phải thực hiện thao tác sau:
  + Lấy ra khỏi hộp 2 lá bài bất kỳ.
  + Chọn số nguyên R được viết lên mặt đỏ của một lá bài được lấy ra và chọn số nguyên B được viết trên mặt màu xanh của lá bài còn lại.
  + Điểm của người chơi sẽ tăng lên một lượng bằng R XOR B.
  + Sau đó, đặt một lá bài bất kỳ vào lại trong hộp và vứt lá bài còn lại đi.
* Trò chơi kết thúc khi trong hộp chỉ còn đúng 1 lá bài.

**Yêu cầu:** Hãy xác định số điểm ít nhất mà người chơi có thể nhận được khi trò chơi kết thúc.

**Input: Đ**ọc từ file **cardgame.in**

Dòng đầu chứa số nguyên T (1 <= T <= 100) là số lượng test.

T nhóm dòng sau, mỗi dòng mô tả một test.

Mỗi nhóm dòng có định dạng:

* Dòng đầu tiên chứa số nguyên dương N (2 <= N <= 100) là số lượng lá bài.
* Dòng thứ hai chứa N số nguyên dương không quá 1 tỷ, số thứ i (1 <= i <= N) là số được ghi trên mặt đỏ của lá bài thứ i.
* Dòng thứ ba chứa N số nguyên dương không quá 1 tỷ, số thứ i (1 <= i <= N) là số được ghi trên mặt xanh của lá bài thứ i.

**Output:** ghi ra file **cardgame.out**

Với mỗi test theo đúng thứ tự được cho trong input, in ra trên một dòng số điểm ít nhất mà người chơi có thể đạt được khi trò chơi kết thúc.

**Subtask:**

*Subtask 1 (25%):* N <= 5

*Subtask 2 (25%):* N <= 12

*Subtask 3 (25%):* N <= 16

*Subtask 4 (25%):* Không có ràng buộc gì thêm.

**Ví dụ**

|  |  |
| --- | --- |
| **cardgame.in** | **cardgame.out** |
| 2 2 1 2 3 3 3 1 101 501 3 2 3 | 1  5 |

---------------------------**HẾT**---------------------------