|  |  |
| --- | --- |
| **Summary** | |
| Program | frog.\* |
| Input | frog.inp |
| Output | frog.out |
| Time/test | 1 giây |

# ẾCH SĂN MỒI

Có *m* bậc thang đánh số từ 1 đến *m* từ trên xuống dưới. Mỗi bậc thang được chia đều thành *n* ô. Ô thứ *j* của bậc thang *i* được gọi là ô (*i,j*) và trên đó có lượng thức ăn *aij*.

Một con ếch muốn đi săn mồi trên những bậc thang. Ếch được xuất phát từ một ô tùy ý trên bậc thang 1 và nhảy dần xuống bậc thang *m*. Khi nhảy tới ô nào thì ếch sẽ ăn hết thức ăn trong ô đó. Tuy nhiên có một hạn chế là từ ô (*x*, *y*) chú ếch chỉ được phép nhảy sang ô (*x'*, *y'*) nếu:



**Yêu cầu:** Tìm một cách đi kiếm ăn cho chú ếch sao cho tổng lượng thức ăn kiếm được là lớn nhất.

**Input:**

* Dòng 1 chứa ba số nguyên dương 
* *m* dòng tiếp theo, dòng thứ *i* chứa *n* số nguyên dương, số thứ *j* là 

**Output:**

* Dòng 1 ghi tổng lượng thức ăn kiếm được
* *m* dòng tiếp theo, dòng thứ *i* ghi một số nguyên là số hiệu ô đi qua trên bậc thang *i*.

**Example:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 3 5 2  4 3 2 1 1  4 3 5 4 9  1 2 3 7 5 | 18  3  5  4 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | gnomes.\* |
| Input | gnomes.inp |
| Output | gnomes.out |
| Time/test | 1 giây |

# CÁC CHÚ LÙN

“Ôi, không phải là các chú lùn mà là một cực hình!” – Bạch tuyết than thở khi ru các chú lùn ngủ. Trong trại có cả thảy ***n*** chú lùn (1 <***n***< 105). Ở thế giới của các chú lùn đêm khá dài.Mỗi chú lùn chỉ ngủ một lần trong đêm. Bạch tuyêt chỉ được ngủ khi mọi chú lùn đều ngủ. Các chú lùn rất khác nhau về thói quen. Để ru chú lùn thứ ***i*** ngủ Bạch tuyết cần ***ai***đơn vị thời gian (gọi là phút) và chú lùn sẽ ngủ trong ***bi*** phút ( 1 <***ai***, ***bi***<109, ***i*** = 1÷ ***n***, 1 <***n*** ≤ 105).

Ví dụ, với ***n*** = 2 và ***a1***= 1, ***b1*** = 10, ***a2***= 10, ***b2*** = 20, nếu Bạch tuyết ru chú lùn thứ nhất ngủ rồi chuyển sang chú lùn thứ 2 thì khi chú lùn thứ 2 ngủ, chú lùn thứ nhất đã thức dậy. Nếu Bạch tuyết ru chú lùn thứ 2 ngủ trước rồi sau đó ru chú thứ nhất – Bạch tuyết sẽ có 9 phút nghỉ ngơi.

***Yêu cầu***: Hãy xác định trình tự ru các chú lùn để Bạch tuyết có ít nhất một phút nghỉ ngơi. Nếu không tồn tại một trình tự ru nào thì đưa ra số -1.

***Dữ liệu***:

* Dòng đầu tiên chứa số nguyên ***n***,
* Dòng thứ ***i*** trong ***n*** dòng sau chứa 2 số nguyên ***ai*** và ***bi***.

***Kết quả***:Một dòng chứa n số nguyên xác định trình tự ru hoặc số -1 nếu không có trình tự ru.

***Ví dụ***:

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 2  1 10  10 20 | 2 1 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | symmetry.\* |
| Input | symmetry.inp |
| Output | symmetry.out |
| Time/test | 1 giây |

# ĐỐI XỨNG

Dãy số đối xứng là dãy số đọc từ trái sang phải giống như đọc từ phải sang trái. Các dãy số sau là dãy đối xứng:

1 2 3 4 5 4 3 2 1

1 2 1 2 2 1 2 1

***Yêu cầu***: Cho dãy số ***n*** số nguyên dương (1 ≤ ***n*** ≤ 200 000), mỗi số có giá trị không quá 1 000. Hãy xác định số lượng số ít nhất phải bổ sung vào cuối dãy đã cho để nhậndãy số đối xứng và chỉ ra dãy số phải bổ sung.

***Dữ liệu***:

* Dòng đầu tiên chứa số nguyên ***n***,
* Dòng thứ 2 chứa ***n*** số nguyên xác định dãy số cho trước.

***Kết quả***: Dòng thứ nhất chứa số ***m*** – số phần tử cần bổ sung,

***Ví dụ***:

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 5  1 2 3 4 5 | 4 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | capital.\* |
| Input | capital.inp |
| Output | capital.out |
| Time/test | 1 giây |

# THÀNH PHỐ TRUNG TÂM

Ở đất nước ByteLand có *n* thành phố (đánh số thứ tự từ 1 đến *n*) được nối với nhau bằng *m* con đường hai chiều). Do yêu cầu về sự thịnh vượng của đất nước, cần thiết phải di chuyển thủ đô hiện thời về một thành phố nào đó trong số các thành phố nói trên. Quốc hội ByteLand muốn chọn một thành phố trung tâm nhất để di chuyển thủ đô đến đó.

Thành phố được gọi là trung tâm nếu như khi cấm các hoạt động giao thông qua thành phố này và trên các con đường nối với nó thì các thành phố còn lại tạo thành nhiều cụm thành phố nhất. Một cụm thành phố là một nhóm các thành phố mà giữa hai thành phố bất kỳ có thể đi đến được với nhau thông qua các thành phố trong cụm. Tất nhiêm, nếu thêm một thành phố khác vào cụm thì tính tính chất trên bị phá vỡ.

Viết chương trình tìm thành phố trung tâm cho Quốc Hội

**Input:**

* Dòng đầu tiên ghi hai số nguyên dương *n*, *m* (2≤*n*,*m*≤300000)
* *m* dòng tiếp theo, mỗi dòng ghi hai số *ai, bi* thể hiện có một đường đi hai chiều nối thành phố *ai* với thành phố *bi*.

**Output:** Ghi hai số Cmax và C thể hiện số cụm lớn nhất tạo ra khi cấm đường giao thông qua thành phố trung tâm và số hiệu của thành phố trung tâm này. Nếu có nhiều thành phố trung tâm thì chỉ in ra thành phố có chỉ số nhỏ nhất

**Example:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 5 5  1 2  3 2  4 2  5 2  1 3 | 3 2 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | newfunc.\* |
| Input | newfunc.inp |
| Output | newfunc.out |
| Time/test | 1 giây |

# HÀM MỚI

An có một dãy *n* số nguyên không âm .

Cô định nghĩa hàm *f(l,r)* (*l, r* là các số nguyên ) trên dãy *a* như là toán từ bít OR của tất cả các số nguyên có chỉ số từ *l* đến *r* :



Ở đây phép toán bit là phép **or** trong pascal và **|** trong C++

An lấy một mảnh giấy và ghi lại tất cả các giá trị của *f(l,r*) với mọi *l*, *r* (*l*, *r* nguyên nguyên ). Bây giờ cô ta muốn biết có bao nhiêu giá trị khác nhau đã được ghi lên tờ giấy?.

Viết chương trình giúp An tính toán xem có bao nhiêu giá trị *f(l,r)* khác nhau với dãy *a* đã cho.

**Input:**

* Dòng đầu tiên ghi số nguyên dương *n* ≤ 105là số phần tử của dãy *a*
* Dòng thứ hai chứa *n* số nguyên -các phần tử của dãy *a*

**Output:** In ra một số nguyên duy nhất là số các giá trị khác nhau của *f(l,r*) trên dãy *a*.

**Example:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 3  1 2 0 | 4 |  |
| 10  1 2 3 4 5 6 1 2 9 10 | 11 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | lottery.\* |
| Input | lottery.inp |
| Output | lottery.out |
| Time/test | 1 giây |

# XỔ SỐ

Các rô bốt trong khu mỏ có số xê ri liên tục trong khoảng [***a***, ***b***] (1 ≤ ***a*** ≤ ***b*** ≤ 109, ***b*** – ***a*** ≤ 105). Vì điều kiện thời tiết xấu chỉ có một lô hàng mỡ bôi trơn được chở tới. Lô hàng này không đủ cho tất cả các rô bốt. Người ta quyết định tổ chức một cuộc “xổ số” nhỏ phân phối số mỡ này: Dùng máy tính đưa ra ***n*** số nguyên dương ***p1***, ***p2***, . . ., ***pn*** và một số nguyên ***k*** (0 ≤ ***k*** ≤ ***n***). Những rô bốt nào có số xê ri chia hết cho đúng ***k*** số trong dãy số nguyên trên sẽ được thay mỡ mới.

**Yêu cầu:** Hãy cho biết số lượng rô bốt được thay mỡ.

**Input:** Vào từ file văn bản LOTTERY.INP:

* Dòng đầu tiên chứa 2 số nguyên ***n*** và ***k*** (1 ≤ ***n*** ≤ 105),
* Dòng thứ 2 chứa ***n*** số nguyên ***p1***, ***p2***, . . ., ***pn*** (1 ≤ ***pi*** ≤ 109, ∀***i***),
* Dòng thứ 3 chứa 2 số nguyên ***a*** và ***b***.

**Output:** Đưa ra file văn bản LOTTERY.OUT một số nguyên – kết quả tìm được.

**Example:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 2 1  2 3  1 10 | 6 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | candies.\* |
| Input | candies.inp |
| Output | candies.out |
| Time/test | 1 giây |

# HỘP KẸO

Các bạn gọi điện thoại cho Steve hẹn đến nhà chia vui với kết quả cao mà Steve đã đạt được trong kỳ thi Tin học vừa kết thúc. Steve đi mua ***n*** hộp kẹo để đón bạn, mỗi hộp một loại kẹo và hộp thứ ***i*** có ***ai*** viên.

Có tất cả ***m*** người tới. Các bạn tới không cùng một lúc mà là lần lượt từng người một. Steve hiểu rất rõ các bạn của mình. Người thứ ***j*** có độ tế nhị ***bj***. Điều này có nghĩa là bạn đó sẽ chỉ ăn kẹo ở các hộp có số lượng còn lại không ít hơn ***bj*** chiếc và sẽ ăn ở những hộp này, mỗi hộp một viên. Nếu một bạn nào đó có độ tế nhị 1 thì bạn đó sẽ ăn ở mỗi hộp một viên kẹo.

Chiều tối, khi các bạn đã về hết, Steve vừa dọn dẹp vừa nhẫm tính xem mỗi bạn đã ăn bao nhiêu viên kẹo.

**Dữ liệu:** Vào từ file văn bản CANDIES.INP:

* Dòng đầu tiên chứa số nguyên ***n*** (1  ***n***  105),
* Dòng thứ 2 chứa ***n*** số nguyên ***a1***, ***a2***, . . ., ***an*** (1  ***ai***  109, ***i*** = 1  ***n***),
* Dòng thứ 3 chứa số nguyên ***m*** (1  ***m***  105),
* Dòng thứ 4 chứa ***m*** số nguyên ***b1***, ***b2***, . . ., ***bm*** (1  ***bj***  109, ***j*** = 1  ***m***).

**Kết quả:** Đưa ra file văn bản CANDIES.OUT ***m*** số nguyên, mỗi số trên một dòng. Số thứ ***j*** là số viên kẹo bạn thứ ***j*** đã ăn.

**Ví dụ:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 3  3 1 1  2  1 2 | 3  1 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | rooms.\* |
| Input | rooms.inp |
| Output | rooms.out |
| Time/test | 1 giây |

# XẾP PHÒNG

Có *n* cuộc hội thảo (đánh số từ 1 đến *n*) đăng ký sử dụng phòng của khu nhà do bạn quản lý. Cuộc hội thảo thứ *i* cần bắt đầu ngay sau thời điểm *si* và kết thúc tại thời điểm *fi*. Có thể hiểu thời gian cuộc hội thảo thứ *i* diễn ra trong khoảng (*si*, *fi*] trên trục thời gian. Hãy bố trí các phòng phục vụ toàn bộ các cuộc hội thảo thỏa mãn các yêu cầu sau:

* Tại một thời điểm, mỗi phòng chỉ được dùng cho một cuộc hội thảo. Hay nói cách khác, hai cuộc hội thảo chỉ có thể bố trí trong cùng một phòng nếu khoảng thời gian làm việc của chúng là không giao nhau.
* Số phòng cần huy động để phục vụ cho toàn bộ *n* cuộc hội thảo là ít nhất có thể

**Input:**

* Dòng 1: chứa số nguyên dương *n* ≤ 105.
* *n* dòng tiếp theo, mỗi dòng chứa hai số tự nhiên 

**Output:**

* Dòng 1: ghi số lượng phòng cần huy động (*k*)
* *k* dòng tiếp theo, dòng thứ *j* ghi chỉ số các cuộc hội thảo sẽ được tổ chức tại phòng thứ *j* trong *k* phòng đã huy động

**Example:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 4  0 2  2 4  1 3  3 5 | 2  1 2  3 4 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | guards.\* |
| Input | guards.inp |
| Output | guards.out |
| Time/test | 1 giây |

# VỆ SỸ

Đội bảo vệ của một tổng thổng mới đắc cử có *n* vệ sỹ đánh số từ 1 đến *n*. Vệ sỹ thứ *i* có thể bảo vệ tổng thống từ thời điểm *si* đến hết thời điểm *fi*: [*si*, *fi*]. Hãy huy động một số ít nhất các vệ sỹ sao cho tại bất kỳ thời điểm nào của lễ nhậm chức diễn ra từ thời điểm A đến hết thời điểm B ([A,B]) tổng thống luôn có ít nhất *k* vệ sỹ bảo vệ mình.

**Input:**

* Dòng 1 chứa 4 số nguyên 
* *n* dòng tiếp theo, dòng thứ *i* chứa hai số nguyên 

**Output:**

* Dòng 1 ghi ra số vệ sỹ cần huy động (*m*), trong trường hợp không thể đáp ứng được yêu cầu đề ra ghi -1
* Trong trường hợp có phương án thực hiện, dòng 2 ghi *m* chỉ số vệ sỹ cần huy động

**Example:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 5 2 1 9  1 5  4 9  1 3  1 7  5 9 | 4  4 1 2 5 |  |

|  |  |
| --- | --- |
| **Summary** | |
| Program | addedge.\* |
| Input | addedge.inp |
| Output | addedge.out |
| Time/test | 1 giây |

# ĐỒ THỊ

Người ta khởi tạo một đồ thị có hướng gồm 109 đỉnh, các đỉnh được đánh số từ 1 đến 109. Ban đầu đồ thị không có cung nào. Người ta lần lượt thêm các cung vào đồ thị bởi *m* lệnh dạng *Add(u,v)*: thêm một cung nối từ đỉnh *u* đến đỉnh *v* trên đồ thị.

Cho trước hai đỉnh *s* và *t*. Hãy cho biết số thứ tự của lệnh *Add* đầu tiên mà sau thời điểm thực hiện lệnh *Add* đó, ta có thể đi từ *s* đến *t* theo các cung của đồ thị

**Input:**

* Dòng 1 chứa ba số nguyên dương *m*, *s*, *t* ()
* *m* dòng tiếp theo, mỗi dòng ghi hai số nguyên *u*, *v* tương ứng là một lệnh *Add(u,v)*

**Output:** Một số duy nhất là số thứ tự lệnh *Add* tìm được, trong trường hợp không thể đi từ *s* đến *t* cho dù thực hiện tất cả các lệnh *Add* thì ghi số 0.

**Example:**

|  |  |  |
| --- | --- | --- |
| Input | Output |  |
| 51 5  1 2  3 5  3 1  2 3  2 4 | 4 |  |