Mục lục

Bài 1. ATM

Vinh làm việc cho một công ty chuyên lập trình điều khiển các cây ATM. Nhiệm vụ của Vinh là phải lập trình điều khiển cây ATM sao cho mỗi lần nhận lệnh rút một lượng tiền trị giá W đồng thì cây ATM sẽ đưa ra một số N ít nhất tờ tiền sao cho tổng giá trị bằng W. Biết rằng trong cây ATM có các loại tiền mệnh giá $1000, 2000, 3000, 5000, 1000\cdot10^1, 2000\cdot10^1, 3000\cdot10^1, 5000\cdot10^1, \ldots, 1000\cdot10^c, 2000\cdot10^c, 3000\cdot10^c, 5000\cdot10^c$ với c là một số nguyên dương, và số lượng tờ tiền mỗi loại là không hạn chế.

Dữ liệu vào

Dòng đầu tiên chứa một số nguyên dương T là số lượng test ($T \leq 1000$).

Mỗi dòng trong số T dòng tiếp theo là dữ liệu cho từng test bao gồm hai số nguyên dương W và c với $W \leq 10^{18}$ và $c \leq 15$.

Kết quả

Kết quả tương ứng với mỗi test được viết trên một dòng duy nhất bao gồm 2 số nguyên dương N và S, trong đó S là số lượng các cách rút ra số N ít nhất các tờ tiền. Ghi ra số 0 nếu như không có cách rút.

Ví dụ

test	answer
2 1000 1 7000 1	1 1 2 1

Bài 2. Tìm đồng xu giả 1

File dữ liệu vào: `Gọi hàm`
File kết quả: `Trả về`

Có n đồng xu được đánh số thứ tự từ 1 đến n, trong số đó có một đồng xu là giả, đồng xu giả có thể nặng hơn hoặc nhẹ hơn các đồng xu khác. Những đồng xu thật có cùng một trọng lượng. Có n đồng xu thật khác được đánh số từ $n+1$ đến $2*n$.

Yêu cầu: xác định đồng xu giả.

Chương trình phải sử dụng một thư viện riêng fakecoinlib1.h (cho C++). Trong chương trình của bạn cần khai báo các thư viện này ở đầu chương trình:

- #include "fakecoinlib1.h"

Thư viện cung cấp các hàm sau:

- Các hàm khởi tạo trò chơi
  - int get_T();
    Chương trình phải gọi hàm này để khởi tạo trò chơi và chỉ được gọi duy nhất 1 lần. Hàm này trả về một giá trị T là số lượng bộ test.
  - int get_n();
    Chương trình phải gọi hàm này T lần sau hàm get_T(), lần thứ i hàm này trả về số n tương ứng là số lượng đồng xu tương ứng với bộ test thứ i.
- Hàm thực hiện truy vấn:
  int compare(int *A, int l);
  Mảng A phải chứa $2*l$ số, các số từ $A[1]$ đến $A[l]$ chứa số thứ tự các đồng xu đặt lên đĩa cân bên trái và các số từ $A[l+1]$ đến $A[2*l]$ chứa số thứ tự các đồng xu đặt lên đĩa cân bên phải. Hàm sẽ trả về kết quả là một số res. Nếu $res=-1$ có nghĩa các đồng xu bên trái nhẹ hơn, $res=0$ tương đương với hai bên đĩa cân thăng bằng. Nếu $res=1$ thì các đồng xu bên trái nặng hơn. Một đồng xu chỉ xuất hiện nhiều nhất một lần trong một lần cân
- Hàm trả lời câu hỏi:
  void answer(int res, int r);
  Nếu $r=1$ có nghĩa đồng xu res là đồng xu giả nặng hơn các đồng xu khác. Nếu $r=-1$ có nghĩa đồng xu res là đồng xu giả và nhẹ hơn các đông xu khác.

Ví dụ

Gọi hàm	Trả về
get_T()	1
get_n()	9
compare(A,3) với A=[0,1,2,3,4,5,6]	-1
compare(A,1) với A=[0,4,5]	0
compare(A,1) với A=[0,1,6]	0
compare(A,1) với A=[0,3,2]	1
answer(2,-1)	Chương trình kết thúc

Hạn chế

- $n \leq 40$;
- Mỗi bộ test bạn chỉ được thực hiện không quá 4 lần cân.

Bài 3. MBAGS

Cho n thỏi vàng. Các thỏi vàng được đánh số từ 0 đến $n-1$, thỏi thứ i có khối lượng a_i. Cần may m cái túi để đựng, các túi có sức chứa giống nhau và bằng M. Tìm M nhỏ nhất có thể để chứa được hết số vàng trên. Cụ thể hơn, cần chia n thỏi vàng thành m phần sao cho tổng của phần có tổng lớn nhất là nhỏ nhất có thể.

Đây là bài toán chỉ cần nộp output, kết quả càng tốt điểm của bạn càng cao. Cụ thể cách tính điểm cho 1 test như sau:

- Gọi M_p là sức chứa bạn tìm được, M_j là sức chứa ban giám khảo tìm được.
- Đặt $x = \frac{M_p - M_j}{M_j}$.
- Nếu $x > 0.1$ bạn được 0 điểm cho test này.
- Nếu $x < 0$ bạn được 10 điểm cho test này.
- Nếu $0 \le x \le 0.1$ bạn được $-2.5 \log_{10}(x + 0.0001)$ điểm cho test này.

Điểm của bài thi là tổng điểm của từng test, mỗi test đều tự động lấy điểm cao nhất của tất cả các lần nộp. Bạn có thể nộp output từng test hoặc nén nhiều output thành file submission.zip rồi nộp (cần đặt tên các file là output_0.txt, output_1.txt, ..., output_9.txt). Các input được cho sẽ đính kèm trên mục đề bài của hệ thống.

Dữ liệu vào

- Dòng đầu chứa: n m
- Dòng tiếp theo chứa: a_0 a_1 ... a_{n-1}

Kết quả

Gồm m dòng, mỗi dòng ghi danh sách các thỏi vàng cho vào một túi.

- Số đầu tiên trên dòng là số lượng thỏi: k
- Tiếp theo là k chỉ số của k thỏi vàng: i_0 i_1 ... i_{k-1}

Ví dụ

test	answer
3 3 1 2 3	2 0 1 0 1 2

Hạn chế

- $1 \le n \le 500$. $1 \le a_i \le 4 \times 10^6$
- 30% test với $n \le 20$
- 30% test với $a_i \le 10^4$

Bài 4. Trò chơi Trăng Sao

File dữ liệu vào: `Gọi hàm`
File kết quả: `Trả về`

Thành thấy một đám trẻ con đang chơi một trò chơi rất thú vị. Thành lại gần bắt chuyện:

- Chào các cháu, các cháu đang chơi trò gì thế?
- Chúng cháu có một trọng tài, còn những người còn lại được chia vào hai đội: đội Sao và đội Trăng. Có n cháu trong đội Sao và m cháu trong đội Trăng. Chú hãy đoán xem ai trong số chúng cháu là trọng tài rồi chúng cháu sẽ nói cho chú biết chi tiết trò chơi?"
- Đồng ý, thế luật chơi thế nào?
- Chú có thể hỏi ai đó nằm trong đội nào nhưng không được phép hỏi trực tiếp ai đó có phải là trọng tài hay không. Biết rằng người trong đội Sao sẽ luôn nói thật, còn người trong đội Trăng luôn nói dối, còn trọng tài thì trả lời tùy theo thứ tự câu hỏi: luôn nói dối ở những câu hỏi lẻ: nghĩa là những câu hỏi thứ nhất, thứ 3, thứ 5, ..., và luôn nói thật ở những câu hỏi chẵn: nghĩa là những câu hỏi thứ 2, thứ 4, thứ 6, ...
- Thế có hạn chế gì không?
- Chú không được hỏi một người về chính bản thân người đó. Ví dụ chú không được hỏi câu :" Cháu ở đội Sao phải không?". Thêm nữa chú chỉ được phép hỏi một cháu tối đa là hai câu hỏi và hai câu hỏi này phải hỏi về hai cháu khác nhau. Ví dụ chú hỏi Dũng liệu Tuấn có thuộc đội Sao không, thì sau đó chú không được hỏi Dũng là liệu Tuấn có thuộc đội Trăng không.
- Nhất trí. Vậy chú sẽ tìm ra tất cả các cháu, mỗi cháu thuộc đội nào, không chỉ tìm ra trọng tài thôi đâu.

Chương trình phải sử dụng một thư viện riêng `kidgamelib.h` (cho C++). Trong chương trình của bạn cần khai báo các thư viện này ở đầu chương trình:

```
#include ‘‘kidgamelib.h’’
```

Thư viện cung cấp các hàm sau:

- Các hàm khởi tạo trò chơi
  - `int get_T();`
    Chương trình phải gọi hàm này để khởi tạo trò chơi và chỉ được gọi duy nhất 1 lần. Hàm này trả về một giá trị T là số lượng bộ test.
  - `void get_nm(&`n`,&`m`);`
    Chương trình phải gọi hàm này T lần sau hàm `get_T()`, lần thứ i hàm này trả về hai số n và m tương ứng là số cháu trong đội Sao và Trăng tương ứng với bộ test thứ i. Các cháu được đánh số từ 1 đến $n + m + 1$.
- Hàm thực hiện truy vấn:
  `int ask(int` a`, int` b`, int` d`);`
  Hỏi cháu a liệu cháu b có nằm trong đội d không? Với $1 \le a, b \le n + m + 1$, và $a \ne b$. $d = 0$ chỉ đội Sao và $d = 1$ chỉ đội Trăng. Hàm trả về 1 nếu câu trả lời là 'Đúng', còn trả về 0 nếu câu trả lời là 'Sai'. Chỉ được phép hỏi một cháu về cùng một cháu khác tối đa 1 lần.
- Hàm trả lời câu hỏi:
  `void answer(int *`C`);`
  Các giá trị từ phần tử C[1] đến phần tử thứ C[n+m+1] chỉ vai trò từng cháu trong trò chơi: 0 nghĩa là thuộc đội Sao, 1 nghĩa là thuộc đội Trăng và 2 nghĩa là trọng tài. Phần tử C[0] không quan trọng, nhận giá trị tùy ý.

Ví dụ

Gọi hàm	Trả về
`get_T()`	`1`
`get_nm(n,m)`	`1 1`
`ask(1,2,0)`	`0`
`ask(2,1,0)`	`1`
`ask(3,1,1)`	`0`
`answer(C) với C=[-1,2,1,0]`	`Chương trình kết thúc`

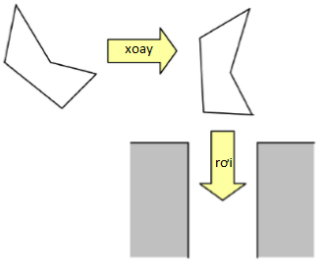
Hạn chế

$2 \leq n + m \leq 500$

Bài 5. Máng rác

ACM là một công ty xây dựng máng rác. Máng rác là một ống hình trụ được đặt trong các tòa nhà cao tầng. Rác được bỏ vào máng sẽ rơi xuống dưới và được thu gom tại tầng hầm. Máng phải được thiết kế vừa với các loại rác được thả vào. Do kích thước của máng càng lớn thì càng tốn chi phí, nên các công ty luôn muốn xây dựng máng rác nhỏ nhất có thể.

Để đơn giản hóa vấn đề, chúng ta sẽ chỉ xem xét việc thiết kế máng rác dưới dạng 2D. Một máng rác dạng thẳng đứng và chiều rộng cố định. Các đồ vật được thả vào máng có là các đa giác. Trước khi thả, các đa giác có thể được xoay để vừa với máng, và chúng sẽ không bị xoay trong quá trình rơi.



Nhiệm vụ: tính toán độ rộng bé nhất của máng rác để một đa giác cho trước lọt qua.

Dữ liệu vào

Đầu vào bao gồm nhiều test (số lượng test <400). Mỗi test bắt đầu bằng một số nguyên n ($3 \leq n \leq 100$) chỉ số đỉnh của đa giác. Tiếp theo là n dòng chứa các cặp x_i và y_i ($0 \leq x_i, y_i \leq 10^4$) cho biết tọa độ các đỉnh của đa giác. Tất cả các điểm này đôi một khác nhau và đảm bảo các cạnh đa giác không cắt nhau (trừ tại các đỉnh của đa giác).

Test cuối cùng theo sau bởi một số 0.

Kết quả

Kết quả mỗi test tương ứng ghi trên một dòng chỉ số test (Case i:) tiếp theo là một giá trị là độ rộng nhỏ nhất của máng rác với độ chính xác đến hai chữ số thập phân sau dấu phẩy.

Ví dụ

test	answer
3 0 0 3 0 0 4 4 0 10 10 0 20 10 10 20 0	Case 1: 2.40 Case 2: 14.14

Bài 6. BUSFREE

File dữ liệu vào: `input_xy.txt`
File kết quả: `output_xy.txt`

Một bản đồ xe buýt được cho trong một hình chữ nhật kích thước $p \times q$ trên hệ tọa độ Oxy, tọa độ điểm dưới trái là (0, 0), điểm trên phải là (p, q). Có n tuyến xe buýt, mỗi tuyến gồm hai chặng tương ứng với hai đường gấp khúc có chung hai điểm đầu cuối, nằm trọn vẹn trong bản đồ. Có c ($c \leq p \times q$) ô vuông quan trọng nằm trong bản đồ, mỗi ô vuông có cạnh dài 1 đơn vị, các cạnh song song với trục tọa độ và các đỉnh có tọa độ nguyên. Mỗi tuyến xe chỉ có đúng 1 xe buýt duy nhất chạy.

Có m ($m \leq n$) sensor cần được đặt lên các xe buýt để quan sát các điểm quan trọng. Mỗi khi được bật lên, sensor sẽ quan sát được tất cả các ô vuông quan trọng có điểm chung với hình tròn bán kính r tính từ vị trí bật sensor. Vì hạn chế năng lượng, mỗi sensor sẽ chỉ được phép bật không quá c lần trên mỗi chặng và sẽ được tắt ngay sau khi quan sát xong (thời gian quan sát rất nhỏ, có thể coi như bằng 0).

Yêu cầu: Chọn ra m tuyến xe để đặt sensor và các vị trí bật sensor trên mỗi tuyến nhằm tối đa hóa số lượng ô vuông quan trọng có thể quan sát được.

Đây là bài toán chỉ cần nộp các file kết quả đầu ra. Thí sinh được cho 10 file đầu vào tương ứng với 10 test, đối với mỗi file đầu vào thí sinh cần nộp một file kết quả đầu ra tương ứng. Với mỗi file kết quả đầu ra, điểm của thí sinh phụ thuộc vào độ tốt của đáp án so với đáp án của ban tổ chức (xem cách tính điểm trong phần Chấm điểm).

Dữ liệu vào

Thí sinh được cung cấp 10 file dữ liệu đầu vào với tên tương ứng là: input_00.txt, input_02.txt,..., input_09.txt. Mỗi file dữ liệu đầu vào có khuôn dạng như sau:

- Dòng đầu ghi hai số nguyên dương p, q và số thực không âm r.
- Dòng thứ hai ghi hai số nguyên dương n, m.
- Các dòng tiếp theo ghi thông tin của từng tuyến xe buýt. Mỗi tuyến xe buýt gồm có:
  - Dòng đầu ghi tổng số điểm gấp khúc trên toàn tuyến.
  - Dòng tiếp theo ghi số điểm gấp khúc của chặng thứ nhất và theo sau đó là tọa độ các điểm gấp khúc của chặng thứ nhất, mỗi tọa độ nằm trên một dòng.
  - Dòng tiếp theo đó là số điểm gấp khúc của chặng thứ hai và theo sau là tọa độ các điểm gấp khúc của chặng thứ hai, mỗi tọa độ nằm trên một dòng.
- Dòng tiếp theo ghi số nguyên không âm c.
- c dòng tiếp theo ghi tọa độ góc trái dưới của các ô vuông quan trọng.

Kết quả

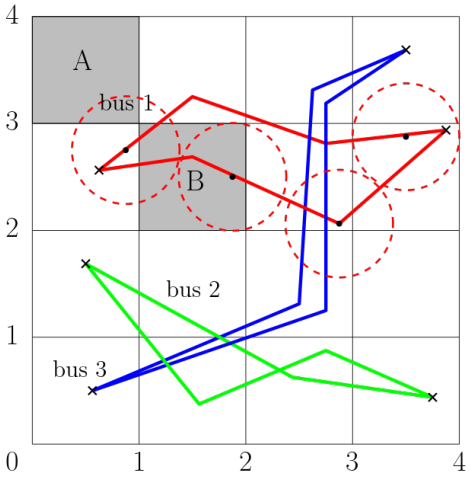
Đối với mỗi file dữ liệu đầu vào, thí sinh cần nộp một file kết quả đầu ra có tên tương ứng là: output_00.txt, output_02.txt, ..., output_09.txt. Mỗi file kết quả đầu ra có khuôn dạng:

- Dòng đầu in ra số ô vuông quan trọng có thể quan sát được.
- Các dòng tiếp theo in ra các tuyến xe buýt được lắp sensor và các vị trí mà sensor được bật trên tuyến đó. Mỗi tuyến gồm có:
  - Dòng đầu tiên in ra số hiệu của tuyến xe.
  - c dòng tiếp theo, mỗi dòng gồm tất cả các ô vuông quan trọng mà một vị trí bật sensor trên chặng thứ nhất có thể quan sát được. Mỗi dòng sẽ bao gồm một số nguyên không âm d là số ô vuông quan trọng quan sát được, theo sau là tọa độ của d ô vuông quan trọng đó.
  - c dòng sau đó, mỗi dòng gồm tất cả các ô vuông quan trọng mà một vị trí bật sensor trên chặng thứ hai có thể quan sát được. Mỗi dòng sẽ bao gồm một số nguyên không âm d là số ô vuông quan trọng quan sát được, theo sau là tọa độ của d ô vuông quan trọng đó.

Ví dụ

input_xy.txt	output_xy.txt
4 4 0.5 3 1 8 4 0.625 2.5625 1.5 3.25 2.75 2.8125 3.875 2.9375 4 3.875 2.9375 2.875 2.0625 1.5 2.6875 0.625 2.5625 7 3 0.5 1.6875 2.3475 0.625 3.75 0.4375 4 3.75 0.4375 2.75 0.875 1.6875 0.375 0.5 1.6875 8 4 0.5625 0.5 2.5 1.3125 2.625 3.3125 3.5 3.6875 4 3.5 3.6875 2.75 3.1875 2.75 1.25 0.5625 0.5 2 0 3 1 2	2 1 2 0 3 1 2 0 0 1 1 2

Giải thích



Có 3 tuyến xe và 2 ô quan trọng là A và B. Đáp án mẫu chọn tuyến xe 1 để lắp đặt sensor. Với các vị trí bật

sensor như trên hình, cả hai ô quan trọng đều được quan sát. Trên chặng thứ nhất, vị trí bật sensor thứ nhất sẽ quan sát được cả A và B, vị trí bật thứ hai không theo dõi được ô nào. Trên chặng thứ hai, vị trí bật sensor thứ nhất không theo dõi được ô nào, trong khi vị trí bật thứ hai sẽ theo dõi được ô vuông B.

Hạn chế

- $1 \leq p, q \leq 50$
- $n = 60$ trong tất cả các test

Chấm điểm

Với mỗi file dữ liệu đầu vào, phần trăm điểm mà thí sinh nhận được tính bằng công thức: $100 \times \min(1, \frac{S_{ts}^2}{S_{btc}^2})(\%)$. Trong đó S_{btc} và S_{ts} tương ứng là số ô vuông quan trọng quan sát được trong đáp án của ban tổ chức và đáp án trong file kết quả đầu ra của thí sinh.

Thí sinh có thể nộp từng file kết quả đầu ra tương ứng với file dữ liệu đầu vào hoặc nén các file kết quả đầu ra thành một file có tên submission.zip để nộp. Điểm số của mỗi test là điểm cao nhất đạt được trong các lần nộp file kết quả đầu ra của test đó. Điểm số của bài là tổng điểm của từng test.

Lưu ý

Một số công thức hữu ích:

- Phương trình đường thẳng đi qua hai điểm (x_1, y_1) và (x_2, y_2) là:

$$(y_2 - y_1)x - (x_2 - x_1)y - (y_2 - y_1)x_1 + (x_2 - x_1)y_1 = 0$$

- Công thức tìm giao điểm của đường tròn tâm (0, 0) bán kính r và đường thẳng $ax + by + c = 0$:
  - Nếu $c^2 > r^2(a^2 + b^2)$ thì hai đường không cắt nhau.
  - Nếu $c^2 = r^2(a^2 + b^2)$ thì hai đường có giao điểm duy nhất $(\frac{-ac}{a^2+b^2}, \frac{-bc}{a^2+b^2})$.
  - Nếu $c^2 < r^2(a^2 + b^2)$ thì hai đường có hai giao điểm.
  + Đặt $d = r^2 - \frac{c^2}{a^2+b^2}$
  + Đặt $mul = \sqrt{\frac{d}{a^2+b^2}}$
  + Hai giao điểm là: $(\frac{-ac}{a^2+b^2} + b * mul, \frac{-bc}{a^2+b^2} - a * mul)$, $(\frac{-ac}{a^2+b^2} - b * mul, \frac{-bc}{a^2+b^2} + a * mul)$

Bài 7. ROUTE

Trong kho có n điểm $1, 2, \ldots, n$ chứa hàng. Biết rằng điểm i có lượng hàng là $a_i (i = 1, \ldots, n)$. Nhân viên kho vận đứng tại cửa kho (điểm 0) và cần lấy 1 lượng hàng là Q. Biết $d(i, j)$ là khoảng cách giữa điểm i và j $(i, j = 0, 1, \ldots, n)$. Hãy tìm lộ trình cho nhân viên kho xuất phát từ cửa kho (điểm 0), đi qua 1 số điểm chứa hàng để lấy đủ lượng hàng Q và quay trở ra cửa kho với tổng độ dài quãng đường là nhỏ nhất. Lưu ý là đến mỗi điểm lấy hàng i thì không nhất thiết phải lấy hết toàn bộ lượng hàng a_i.

Đây là bài toán NP-khó có dạng Output-Only, nghĩa là bạn sẽ được cung cấp toàn bộ input chuẩn và bạn chỉ cần nộp lên chấm output bạn tìm được. Kết quả output càng tốt thì điểm của bạn càng cao với cách tính điểm cho 1 test như sau:

- Gọi GK là tổng quãng đường cần đi theo kết quả của Ban giám khảo (giá trị này thí sinh không được biết, chỉ dùng khi chấm), TS là tổng quãng đường cần đi theo kết quả của thí sinh.
- Đặt $x = \frac{TS-GK}{GK}$.
- Nếu $x < 0$ bạn được 10 điểm cho test này.
- Nếu $x \geq 0$ bạn được $\frac{1}{1+10*x}$ điểm cho test này.

Điểm của bài thi là tổng điểm của từng test, mỗi test đều tự động lấy điểm cao nhất của tất cả các lần nộp. Bạn có thể nộp output từng test hoặc nén nhiều output thành file submission.zip rồi nộp (cần đặt tên các file là output_0.txt, output_1.txt, ..., output_9.txt). Các input được cho sẽ đính kèm trên mục đề bài của hệ thống. Bạn cũng được cho sẵn một file output ví dụ (output_0.txt) là một output hợp lệ cho input_0.txt.

Dữ liệu vào

Dữ liệu đầu vào có cấu trúc như sau:

- Dòng 1 ghi giá trị nguyên dương n, Q $(1 \leq n \leq 50)$
- Dòng 2 chứa n số nguyên không âm $a_1, a_2, \ldots, a_n$ $(a_i \leq 10^4, \forall i = 1, \ldots n)$
- Dòng $i + 3 (i = 0, \ldots, n)$ ghi các phần tử ở hàng thứ i của ma trận d $(1 \leq d(i, j) \leq 100)$

Kết quả

Nếu không có hành trình nào lấy đủ lượng hàng Q thì in ra -1, ngược lại:

- Dòng đầu tiên ghi giá trị tổng quãng đường của lộ trình tìm được.
- Dòng tiếp theo chứa k là số lượng điểm lấy hàng
- Dòng tiếp theo ghi k số là vị trí các điểm lấy hàng theo thứ tự sẽ lấy

Ví dụ

test	answer
5 10 3 6 2 4 1 0 3 6 4 6 2 3 0 7 8 2 1 6 7 0 1 4 9 4 8 1 0 3 4 6 2 4 3 0 1 2 1 9 4 1 0	12 4 1 5 4 3

Giải thích

Hành trình ngắn nhất lấy đủ lượng hàng là 0 - 1 - 5 - 4 - 3 - 0 với độ dài là $3 + 1 + 1 + 3 + 4 = 12$ và tổng lượng hàng của các điểm đi qua là $3 + 1 + 4 + 2 = 10$

Hạn chế

- Có 30% số test với $n \leq 10$
- Có 30% số test với $n \leq 20$

Bài 8. Tàu điện ngầm

Thành phố mà Hải đang sống có hệ thống tàu điện ngầm bao gồm N trạm và M tuyến đường ngầm mỗi tuyến nối trực tiếp hai trạm khác nhau. Hai trạm được gọi là kề nhau nếu như có tuyến đường nối trực tiếp hai trạm. Không có 2 tuyến nào nối cùng 1 cặp trạm. Trạm i có số lượng khách trung bình hàng ngày là p_i.

Hải mới được giao nhiệm vụ xây dựng và phát triển các chi nhánh công ty quảng cáo của mình tại các trạm tàu điện ngầm. Sau khi khảo sát, Hải nhận thấy rằng điểm đặc biệt của hệ thống tàu điện ngầm là có không quá 15 trạm mà mỗi trạm kề với ít nhất 3 trạm khác.

Do không đủ kinh phí để xây dựng các chi nhánh trên toàn tuyến tàu điện ngầm nên Hải quyết định chỉ xây dựng trên một tập con các trạm sao cho không có hai trạm nào trong tập con được chọn là kề nhau, thêm nữa tổng số lượng khách trung bình hàng ngày của các trạm trong tập đó phải là lớn nhất.

Yêu cầu: Hãy giúp Hải tìm ra tập các trạm đó.

Dữ liệu vào

Dòng đầu tiên ghi hai số nguyên dương N và M ($N \leq 100000$, $M \leq 150000$).

Dòng thứ hai ghi N số nguyên dương, số thứ i là số lượng khách trung bình hàng ngày p_i ($p_i \leq 10000$) của trạm i.

Mỗi dòng trong số M dòng tiếp theo chứa hai số nguyên dương x và y ($x, y \leq N$) là hai trạm đầu mút của một tuyến đường hầm nối chúng.

Kết quả

Ghi ra duy nhất một số là tổng số hành khách trung bình hàng ngày của các trạm trong tập các trạm tìm được.

Ví dụ

test	answer
8 10 1 3 2 5 4 1 2 1 1 2 2 3 3 4 4 5 5 3 3 6 2 6 2 7 7 8 8 3	9

Giải thích

Có 8 trạm tàu điện ngầm và 10 đường nối trực tiếp. Tập con các trạm $\{2, 4, 8\}$ cho tổng số lượng khách trung bình lớn nhất và bằng 3+5+1=9. Tập này thoả mãn điều kiện là không có hai trạm nào có đường nối trực tiếp.

Bài 9. Lệnh tiến công — ATTACK

"*Ngày* `DD` *tới, vào giờ* `HH:MM`*, toàn quân tổng tiến công*!"

Đó là quân lệnh từ sở chỉ huy cần được truyền tới tất cả các đơn vị chiến đấu. Tuấn làm việc trong bộ phận phụ trách thiết kế các module đảm bảo thông tin được truyền đi an toàn và chính xác. Thông tin được truyền đi sẽ gồm ba giai đoạn:

1. Mã hóa thông tin (encode): từ sở chỉ huy, thông tin (`DD, HH, MM`) sẽ được mã hóa thành một dãy bit;
2. Truyền tin (transmission): sử dụng những phương tiện thô sơ để truyền dãy bit tới các máy nhận ở các đơn vị;
3. Giải mã thông tin (decode): từ dãy bit nhận được, giải mã lại thông tin (`DD, HH, MM`).

Tuấn phụ trách thiết kế module 1 và 3 nhưng gặp rất nhiều khó khăn. Lý do chính là do kĩ thuật truyền tin thô sơ và phụ thuộc vào yếu tố kĩ năng của con người, nên không phải thông tin lúc nào cũng được truyền đi một cách hoàn toàn chính xác. Thông tin truyền đi có thể bị sai sót tối đa 1 bit. Vì vậy, yêu cầu của hệ thống mã hóa này phải có khả năng phát hiện lỗi, và tự sửa chữa sai sót. Nói cách khác, cho dù thông tin truyền đi có thể sai 1 bit nhưng vẫn phải tự khôi phục được.

Yêu cầu: Hãy giúp Tuấn thực hiện được công việc của mình.

Giao tiếp:

Bạn cần viết hai hàm sau:

1. `string encode (string message)`

   Hàm nhận vào một xâu độ dài 8 có dạng `"DD HH:MM"` trong đó `DD` là một số nguyên trong đoạn [1,31], `HH` là một số nguyên trong đoạn [0,23] và `MM` là một số nguyên trong đoạn [0,59]. Hàm cần trả về một xâu không quá 50 kí tự chỉ gồm các kí tự 0 hoặc 1 là xâu được mã hóa.

2. `string decode (string encryptedMessage)`

   Hàm nhận vào một xâu có độ dài không quá 50 kí tự, chỉ gồm các kí tự 0 hoặc 1. Hàm cần trả ra một xâu có độ dài 8 có định dạng giống xâu đầu vào của hàm encode là xâu đã được giải mã.

Làm bài:

Trên hệ thống, các bạn có thể download file `attack_public_cpp.zip` chứa các file hỗ trợ làm bài. Trong đó:

- `attack.cpp` là file các bạn cần làm việc và viết thuật toán mã hóa.
- `stub.cpp` là file hỗ trợ các chức năng nhập xuất dữ liệu.
- `compile_cpp.sh` là script hỗ trợ việc biên dịch chương trình.

Bạn có thể làm các việc sau:

- Viết thuật toán mã hóa của bạn vào file `attack.cpp`. Các bạn có thể viết thêm các hàm phụ trợ khác.
- Biên dịch và thực thi bằng script được cung cấp như sau:
  - Vào thư mục chứa file `attack.cpp`, click chuột phải và chọn Open in Terminal.
  - Biên dịch file mã nguồn ra file thực thi bằng cách chạy lệnh `./compile_cpp.sh`.

  - Thực thi bằng cách chạy lệnh `./attack`.

- File thực thi `attack` nhận dữ liệu vào từ dòng nhập chuẩn (stdin) ở một trong hai dạng:

  1. `ENCODE DD HH MM`
     - Ví dụ: ENCODE 1 12 5 với ý nghĩa gọi hàm encode("01 12:05")
  2. `DECODE str`
     - Ví dụ DECODE 00011101 với ý nghĩa gọi hàm decode("00011101")
  3. File thực thi sẽ in ra một xâu là kết quả của hàm tương ứng mà chương trình bạn chạy.

Ngoài ra, trong quá trình làm bài, các bạn có thể tự thay đổi các file `stub.cpp` để phục vụ theo cách nhập xuất dự liệu mà bạn muốn.

Nộp bài:

Các bạn chỉ nộp file `attack.cpp` mà không nộp file nào khác.

Chấm bài:

Bài của các bạn sẽ được chấm như sau. Ban giám khảo chuẩn bị một file hệ thống chấm (được gọi là manager).

- Chương trình manager sẽ gọi hàm encode của bạn với một bộ dữ liệu (`DD, HH, MM`) nào đó và thu được xâu `encryptedMessage`.
- Chương trình manager sẽ lặp lại các bước sau nhiều lần:
  - Sửa một bit hoặc không sửa bit nào tạo thành xâu `receivedMessage`.
  - Chương trình manager sẽ gọi hàm decode của bạn với đầu vào là xâu `receivedMessage` và so sánh kết quả thu được với bộ dữ liệu ban đầu.

Lưu ý rằng, những lần hàm encode và decode của bạn được gọi sẽ được chạy ở các tiến trình độc lập. Vì vậy, nếu hàm encode có sử dụng và lưu trữ dữ liệu vào các biến toàn cục, các dữ liệu này sẽ không tồn tại khi hàm decode được thực thi.

Chấm điểm:

Với mỗi bộ dữ liệu, bạn sẽ không được điểm nếu:

- Tương tác sai quy cách (dữ liệu trả ra không đúng định dạng)
- Chạy sinh lỗi
- Chạy quá thời gian
- Xâu sau khi giải mã khác với xâu ban đầu

Gọi độ dài xâu mã hóa của bạn là L, bạn sẽ nhận được điểm dựa trên độ tốt của L như sau:

Độ dài xâu mã hóa của bạn	Điểm của test
$1 \leq L \leq 21$	100%
$L = 22$	90%
$L = 23$	80%
$24 \leq L \leq 25$	60%
$26 \leq L \leq 30$	40%
$31 \leq L \leq 35$	30%
$36 \leq L \leq 40$	20%
$41 \leq L \leq 50$	10%

Bài 10. Alice ở xứ sở 01

File dữ liệu vào: `Gọi hàm`
File kết quả: `Trả về`

Mọi lần Alice lạc vào xứ sở thần tiên, nhưng lần này Alice lại lạc vào xứ sở của những con số 0 và 1. Do không hứng thú với xứ sở này nên Alice muốn nhanh chóng quay lại thế giới thực của mình. Alice đã tìm đến cửa ra thông minh nhưng để mở được cánh cửa phải giải mã được câu hỏi sau:

"Cho biết các số đứng ở một vài vị trí trên một dãy nhị phân độ dài n ($1 \leq n \leq 10^5$) (các vị trí trên dãy được đánh số từ 0 đến $n-1$), hỏi dãy có chứa hai số 0 và 1 (theo đúng thứ tự 0 rồi đến 1) ở hai vị trí liên tiếp nhau hay không?".

Để có đủ dữ kiện trả lời câu hỏi này, cánh cửa thông minh cho phép Alice đặt ra các truy vấn, mỗi truy vấn có dạng sau:

"Số đứng ở vị trí thứ i của dãy là số nào?".

Câu trả lời cho mỗi truy vấn như vậy chỉ là 0 hoặc 1. Cánh cửa thông minh chỉ mở khi Alice trả lời đúng cho câu hỏi đặt ra với số lần thực hiện truy vấn là ít nhất có thể.

Yêu cầu: Hãy giúp Alice viết chương trình tìm cách thoát khỏi xứ sở 01.

Chương trình phải sử dụng một thư viện riêng `alicelib.h` (cho C/C++).

Thư viện cung cấp các hàm sau:

- Hàm khởi tạo trò chơi
- Đối với C/CPP: `int get_n(int *&a)`

  Chương trình phải gọi hàm này để khởi tạo trò chơi. Hàm get_n trả về một giá trị n là độ dài của dãy nhị phân và mảng a độ dài n với các phần tử là 0 hoặc 1 hoặc -1, trong đó các vị trí đứng số 0 (hoặc số 1) là các vị trí mà số đứng ở đó được biết trước là 0 (hoặc 1), còn các vị trí đứng số -1 là các vị trí mà số đứng tại đó là chưa được biết.
- Hàm thực hiện truy vấn: `int get_01(int i)`. Hàm này trả về số đứng ở vị trí thứ i (là câu trả lời cho truy vấn về vị trí thứ i).
- Hàm trả lời câu hỏi: void guess(int res). Để kết thúc chương trình cần gọi hàm này với res bằng 1 nếu câu trả lời cho câu hỏi đặt ra là khẳng định và bằng 0 nếu câu trả lời là phủ định. Sau khi gọi hàm này chương trình sẽ tự động kết thúc. Số lượng câu hỏi của chương trình của bạn sẽ bằng tổng số lần gọi hàm `get_01`.

Lưu ý: Mỗi hàm `get_n` và `guess` chỉ được gọi một lần duy nhất.

Nhiệm vụ của bạn là phải gọi hàm `guess` để đoán kết quả khi đã xác định được đáp án.

- Đối với C/C++ thì cần include file thư viện này theo mẫu:

  `#include ''alicelib.h''`

Bạn có thể xem các file được cung cấp trên hệ thống để hiểu rõ hơn về cách tương tác với hệ thống.

Ví dụ

Gọi hàm	Trả về
get_n(*a)	5; a= [-1, 1, 0, 0, -1]; độ dài của dãy là 5, các vị trí 1, 2, 3 là các vị trí được biết trước
get_01(4)	0
get_01(0)	0
guess(1)	Kết thúc. Chương trình đã trả lời đúng với số lần thực hiện truy vấn là ít nhất. Cánh cửa thông minh sẽ tự động mở ra.

Bài 11. Thành phần liên thông trên cây

Cho cây (đơn đồ thị vô hướng không chứa chu trình) $T = (V, E)$, trong đó V là tập đỉnh và E là tập cạnh. Ta định nghĩa thao tác xoá tập con S của tập đỉnh V khỏi cây T là việc loại bỏ tất cả các đỉnh trong S cùng với tất cả các cạnh kề với chúng khỏi cây. Giả sử S là một tập con các đỉnh của cây T, gọi $k(S)$ là số lượng thành phần liên thông của đồ thị thu được sau khi xoá tập đỉnh S khỏi cây T. Ký hiệu: $k_{\max} = \max\{k(S) : S \in V\}$.

Yêu cầu:

- Tính số $k_{\max}$;
- Tính M là số lượng tập con khác nhau của tập đỉnh V sao cho đồ thị thu được sau khi loại bỏ các đỉnh trong nó khỏi cây T có đúng $k_{\max}$ thành phần liên thông, nghĩa là số cách xoá bớt khỏi đồ thị một hoặc một số đỉnh để thu được đồ thị với số thành phần liên thông đúng bằng $k_{\max}$.

Dữ liệu vào

Dòng đầu tiên chứa số nguyên N là số lượng đỉnh của cây T, trong đó các đỉnh được đánh số từ 1 đến N ($1 \le N \le 10^5$);

Mỗi dòng trong số $N - 1$ dòng tiếp theo chứa hai số nguyên x và y được ghi cách nhau bởi dấu cách cho biết có cạnh nối giữa hai đỉnh x và y trên cây T.

Kết quả

Ghi ra trên một dòng duy nhất hai số nguyên $k_{\max}$ và M, với $k_{\max}$ là số lượng thành phần liên thông lớn nhất tìm được và M là số cách xoá bớt khỏi đồ thị một hoặc một số đỉnh để thu được đồ thị với số thành phần liên thông đúng bằng $k_{\max}$. Vì M có thể là số rất lớn nên chỉ cần đưa ra phần dư trong phép chia của M cho $10^9 + 7$.

Ví dụ

test	answer
6 1 2 1 3 1 4 4 5 4 6	4 1
4 1 2 2 3 3 4	2 5

Giải thích

Ví dụ 1: Số lượng thành phần liên thông lớn nhất tìm được sau xoá là 4. Có duy nhất một cách xoá để được 4 thành phần liên thông đó là xoá hai đỉnh 1 và 4.

Ví dụ 2: Số lượng thành phần liên thông lớn nhất tìm được sau xoá là 2. Có 5 cách xoá để đạt được 2 thành phần liên thông, đó là xoá các tập: {2}, {3},{2, 3}, {2, 4}, {1, 3}

Bài 12. Vị trí hạnh phúc — HAPPOS

Dây chuyền sản xuất nhà máy VAIP có n vị trí sản xuất được đánh số $\{1, 2, \ldots, n\}$ và bố trí dưới dạng một đồ thị có cấu trúc như sau:

- mỗi một vị trí sản xuất là một đỉnh của đồ thị;
- hai vị trí kề nhau tương ứng với cạnh của đồ thị;
- đồ thị là vô hướng liên thông và không có chu trình.

Có n công nhân được đánh số $\{1, 2, \ldots, n\}$ được được đào tạo để phụ trách các vị trí trong dây chuyền. Do sở trường của mỗi người nên người i sẽ cảm thấy hạnh phúc nếu được xếp phụ trách ở vị trí i ($\forall i, 1 \leq i \leq n$). Ban đầu các công nhân được bố trí đúng vị trí hạnh phúc của mình, nghĩa là người i được bố trí ở vị trí i. Để có thể thích ứng với nhiều vị trí khác nhau mà không làm ảnh hưởng sản xuất, sau mỗi ngày, người quản lý chọn ra đúng hai công nhân kề nhau, nghĩa là ở vị trí hai đỉnh kề nhau trên đồ thị, và đảo vị trí của họ. Sau một số ngày đảo chỗ, đến hôm nay người quản lý nhận ra là quên mất không ghi lại nhật ký vị trí các lần đảo để có thể làm ngược lại quá trình những ngày vừa qua nhằm đưa tất cả công nhân về vị trí hạnh phúc của họ.

Yêu cầu: Cho biết vị trí các công nhân hiện đang phụ trách ngày hôm nay, bạn hãy giúp người quản lý tìm ra thứ tự đảo vị trí hai công nhân mỗi ngày để đưa tất cả công nhân về vị trí hạnh phúc của họ sao cho số ngày phải sử dụng là ít nhất có thể.

Đây là bài toán chỉ cần nộp các file kết quả đầu ra. Thí sinh được cho 10 file đầu vào tương ứng với 10 test, đối với mỗi file đầu vào thí sinh cần nộp một file kết quả đầu ra mô tả kế hoạch hoán đổi vị trí theo từng ngày. Với mỗi file kết quả đầu ra mô tả đúng đắn quá trình hoán đổi, điểm của thí sinh phụ thuộc vào số ngày sử dụng để đưa toàn bộ công nhân về vị trí hạnh phúc (xem cách tính điểm trong phần **Chấm điểm**).

Dữ liệu vào

Thí sinh được cung cấp 10 file dữ liệu đầu vào với tên tương ứng là: input_0.txt, input_1.txt,..., input_9.txt. Mỗi file dữ liệu đầu vào có khuôn dạng như sau:

- Dòng đầu chứa một số nguyên dương n là số đỉnh của đồ thị;
- Mỗi dòng trong số $n - 1$ dòng tiếp theo chứa hai số nguyên dương u và v là hai đỉnh đầu mút một cạnh của đồ thị;
- Dòng tiếp theo chứa n số nguyên dương, số thứ i là số hiệu của công nhân hiện đang đứng tại vị trí i.

Các số trên một dòng cách nhau bởi dấu cách.

Kết quả

Đối với mỗi file dữ liệu đầu vào, thí sinh cần nộp một file kết quả đầu ra mô tả kế hoạch hoán đổi vị trí theo từng ngày, các file kết quả đầu ra có tên tương ứng là: output_0.txt, output_1.txt,..., output_9.txt. Mỗi file kết quả đầu ra có khuôn dạng:

- Dòng đầu tiên chứa một số nguyên t là số ngày bạn cần để đưa tất cả công nhân về vị trí hạnh phúc;
- Dòng thứ i trong số t dòng tiếp theo ghi hai số nguyên dương u và v là hai vị trí sản xuất mà công nhân ở đó phải đảo chỗ ngày thứ i.

Thí sinh có thể nộp từng file kết quả đầu ra tương ứng với file dữ liệu đầu vào hoặc nén các file kết quả đầu ra thành một file có tên submission.zip để nộp. Điểm số của mỗi test là điểm cao nhất đạt được trong các lần nộp file kết quả đầu ra của test đó. Điểm số của bài là tổng điểm của từng test.

Lưu ý: Kích thước tối đa cho phép của mỗi file nộp lên server là 10Mb.

Hạn chế

- Subtask 1 (30 điểm): $n \leq 10$;
- Subtask 2 (70 điểm): $10 < n \leq 1000$.

Chấm điểm:

Với mỗi file dữ liệu đầu vào, gọi GK là số ngày cần để đưa toàn bộ công nhân về vị trí hạnh phúc của họ theo phương án của Ban giám khảo (giá trị này thí sinh không được biết, chỉ dùng khi chấm), TS là số ngày cần để đưa toàn bộ công nhân về vị trí hạnh phúc của họ theo phương án của thí sinh trong file đầu ra tương ứng với file đầu vào. Đặt $P = (TS - GK)/GK$, khi đó, thí sinh sẽ nhận được:

- 0 điểm nếu $P > 1$;
- 10 điểm nếu $P \leq 0$;
- $-\log_{10}(P \times 0.9999 + 0.0001) \times 2.5$ điểm nếu $0 < P \leq 1$;

trên tổng số 10 điểm của test đó.

Lưu ý: Thí sinh sẽ không được điểm nếu file output nộp lên không hợp lệ.

Ví dụ

test	answer
3 1 2 2 3 2 1 3	1 1 2