

CHUYỆN ĐỀ TRÒ CHƠI

Nguyễn Thanh Tùng

4 – BÀI TOÁN TRÒ CHƠI

Trong công việc thường ngày để đạt được mục tiêu này hay mục tiêu khác, con người thường thường phải đưa ra sự lựa chọn tối ưu. Bài toán tối ưu gấp cả trong lý thuyết lẩn trong ứng dụng ở tất cả các ngành khoa học, từ y tế đến vật lý, từ toán học đến sinh học, từ quân sự đến kinh tế. Vấn đề lựa chọn giải pháp tối ưu được hình thức hóa và việc nghiên cứu mô hình toán học này dẫn đến sự ra đời của một số lĩnh vực nghiên cứu lý thuyết, trong đó tổng quát hơn cả là lý thuyết Nghiên cứu hỗ trợ quyết định.

Trong số các điều kiện tác động lên chiến lược lựa chọn quyết định **điều kiện xung đột đóng** một vai trò đặc biệt quan trọng. Nó quan trọng vì 2 lý do:

- Xung đột là động lực phát triển của cuộc sống và xã hội,
- Đặc thù của xung đột là yếu tố chính tác động lên quyết định phải lựa chọn.

Trong điều kiện có xung đột, quyết định phải dựa trên cơ sở:

- Lợi ích của mình và lợi ích của các đối tác mà thông thường các lợi ích này *mâu thuẫn lẫn nhau*,
- Khi ra quyết định, ta phải lưu ý đến *quyết định của đối tác*, điều mà thông thường ta *không biết trước*.

Chuyên ngành nghiên cứu về việc đưa ra quyết định trong trường hợp có xung đột là Lý thuyết trò chơi.

Là một lý thuyết nó có các khái niệm cơ bản, các mô hình toán học, tiên đề, bối cảnh, định lý, hệ quả, . . . Ở đây do hạn chế về thời gian chúng ta sẽ giản lược đi nhiều thứ, đi thẳng ngay vào một vài vấn đề phục vụ cho việc đào tạo bài dưỡng học sinh năng khiếu. Lưu ý là xung đột trong trò chơi thường được gọi là đối kháng.

Các vấn đề mà chúng ta sẽ xét bao gồm:

- Lý thuyết trò chơi với *một số mô hình thường gặp*,
- *Kỹ thuật lập trình* cho các bài toán trò chơi.

Ngoài ra, trong phần kỹ thuật lập trình ta sẽ xem xét một mô hình đơn giản hơn (nhưng không kém phần quan trọng với công tác đào tạo, giảng dạy của chúng ta) đó là các trò chơi không đối kháng.

4.1 – Lý thuyết trò chơi

Mỗi trò chơi bao giờ cũng liên quan tới 2 khía cạnh: không gian trạng thái và không gian điều khiển. Hàm mục tiêu thường được xác định trên tập trạng thái. Tập trạng thái của trò chơi có thể là hữu hạn hoặc vô hạn nhưng tập điều khiển luôn luôn luôn hữu hạn và thông thường là không lớn. Xét trò chơi đối kháng giữa 2 người.

Ví dụ, xét bài toán trò chơi lật xúc xắc.

Bài toán: Lật xúc xắc

Cho một con xúc xắc truyền thống, trên mỗi mặt của xúc xắc có một số chấm trong phạm vi từ 1 đến 6 xác định số điểm của mặt, không có 2 mặt nào có cùng số điểm và tổng điểm của 2 mặt đối luôn luôn bằng 7. Con xúc xắc được tung lên bàn và tổng S của trò chơi ban đầu nhận giá trị bằng số điểm mặt trên của xúc xắc. Hai người lần lượt đi. Khi đến lượt mình đi, người chơi lật một lần con xúc xắc qua cạnh của nó và cộng số điểm ở mặt trên mới vào S . Ai đến lượt mình đi làm tổng S lớn hơn S_{max} sẽ thua.

Cuộc chơi đang diễn ra sôi nổi thì người đến lượt đi có điện thoại và lúng túng xin lỗi phải đi làm một việc gấp. Vì lý do tế nhị, không ai hỏi đó là việc gì, nhưng bạn được chỉ định thay thế. Với tổng S và mặt trên v hiện có, hãy xác định cuối cùng bạn có thể thắng được hay không và nếu có thì chỉ ra các cách lật để thắng.

Dữ liệu: Vào từ file văn bản DICE.INP gồm một dòng chứa 3 số nguyên v , S và S_{max} ($1 \leq v \leq 6$, $1 \leq S \leq S_{max} \leq 10^5$). Các số ghi cách nhau một dấu cách.

Kết quả: Đưa ra file văn bản DICE.OUT trên một dòng số nguyên m – số các đi thắng và nếu $m > 0$ thì sau đó là m số nguyên – các mặt cần đưa thành mặt trên, đưa ra theo thứ tự tăng dần. Các số ghi cách nhau một dấu cách. $m = 0$ ứng với trường hợp không có cách thắng.

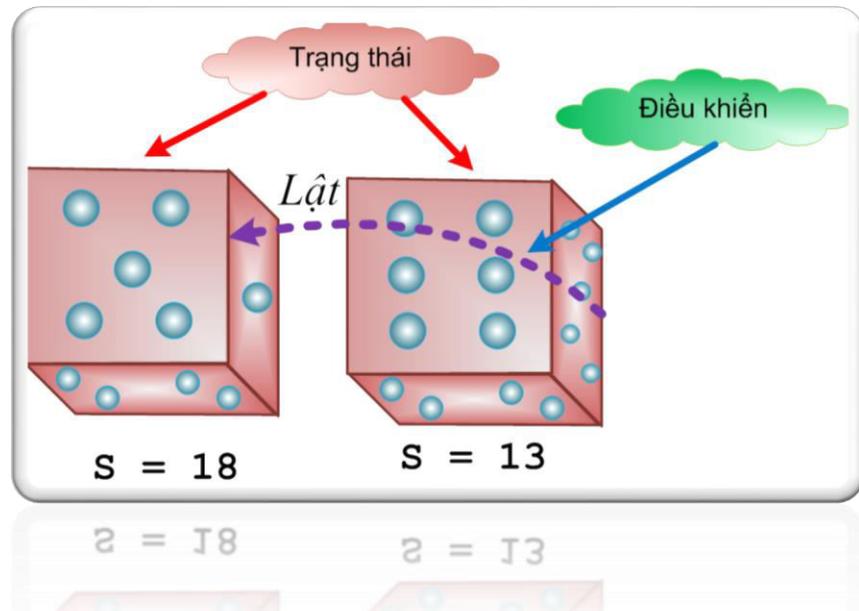
Ví dụ:

DICE.INP
6 13 20

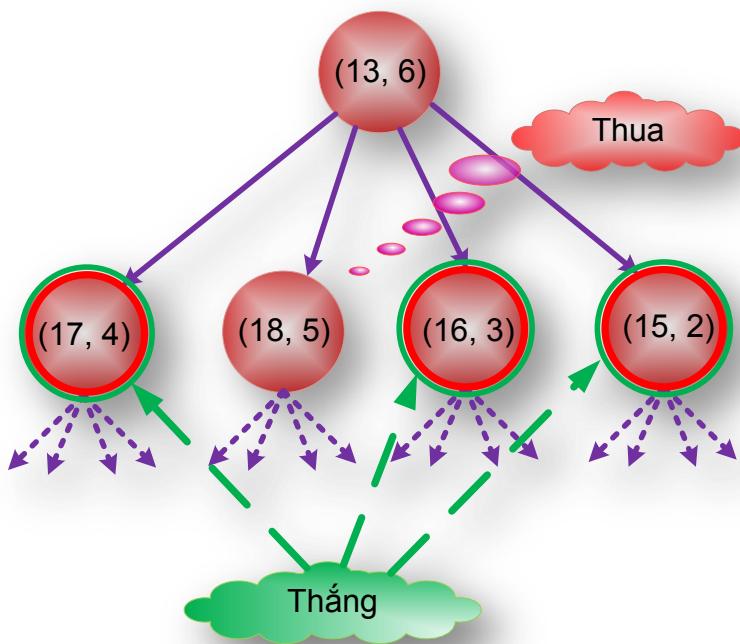
DICE.OUT
3 2 3 4

Mô hình:

Ở bài này, trạng thái của trò chơi là tổng S và mặt trên của con xúc xắc, còn điều khiển là cách lật. Có 4 cách lật: đưa mặt trước lên trên, đưa mặt phải lên trên, đưa mặt sau lên trên hoặc đưa mặt sau lên trên.



Ta có thể biểu diễn các nước đi của trò chơi dưới dạng một đồ thị có hướng, trong đó các nút là trạng thái. Điều khiển được thể hiện dưới dạng cung có hướng, nối các trạng thái với nhau.



4.1.1 – Trò chơi trên đồ thị

Trên nguyên tắc, mọi trò chơi đều có thể biểu diễn dưới dạng đồ thị có hướng. Tuy vậy, không phải lúc nào đồ thị có hướng này cũng dễ xây dựng, lưu trữ và thuận tiện trong việc hỗ trợ tìm chiến lược điều khiển.

Những trò chơi mà mô hình đồ thị cho phép chúng ta dễ dàng tìm ra chiến lược điều khiển để thắng (nếu có) được xếp vào lớp trò chơi trên đồ thị.

Trò chơi trên đồ thị có nội dung như sau:

Cho đồ thị có hướng $G = (V, E)$, trong đó V – tập đỉnh của đồ thị, mỗi đỉnh tương ứng với một trạng thái của trò chơi, E – tập cạnh, mỗi cạnh tương ứng với một nước đi.

Giả thiết cả 2 người chơi đều biết cách đi tối ưu, tức là nếu có khả năng thắng được thì họ sẽ không bỏ qua.

Ký hiệu W – tập con các đỉnh từ đó có cách đi để thắng, L – tập con các đỉnh từ đó không có cách đi để thắng và D – tập con các đỉnh từ đó trò chơi sẽ kết thúc với kết quả hòa. Khi đó $V = W \cup L \cup D$.

Mọi chiến lược điều khiển trò chơi là một phép ánh xạ $f: V \rightarrow E$ xác định nước đi cần thực hiện từ đỉnh đang đứng.

Số cung xuất phát từ một đỉnh được gọi là bậc của đỉnh đó. Các nút tương ứng với trạng thái kết thúc trò chơi có bậc bằng 0.

Giải thuật phân loại đỉnh đồ thị

a – Trường hợp không có chu trình

Bố đề 1. Nếu G là đồ thị không chứa chu trình và mọi đỉnh bậc 0 đều thuộc tập $W \cup L$, khi đó mỗi đỉnh của G đều có thể phân loại thành đỉnh thắng hoặc thua (không có trường hợp hòa).

Chứng minh: Sắp xếp các đỉnh của đồ thị theo thứ tự từ điển tăng dần khoảng cách tới đỉnh gốc. Gọi W_0 là tập các đỉnh bậc 0 thuộc W , L_0 là tập các đỉnh bậc 0 thuộc L . Như vậy, nếu u là một đỉnh bậc 0, nó phải thỏa mãn một trong 2 điều kiện:

- $u \in W_0 \Rightarrow u \in W$,

- $u \in L_0 \Rightarrow u \in L$.

Do phải có ít nhất một cung dẫn tới đỉnh u nên:

- Nếu $u \in W \Leftrightarrow \exists v: uv \in E$ và $v \in L$,
- Nếu $u \in L \Leftrightarrow \forall v$ nếu $uv \in E$ thì $v \in W$.

Như vậy giải thuật có độ phức tạp là $O(E+V)$.

b – Trường hợp đồ thị có chu trình

Xét trường hợp đồ thị G chứa chu trình. Một nước đi của trò chơi bao gồm 2 lần đi: lần đi của một người và lần đi tiếp theo của đối phương.

Gọi W_k là tập các nút mà xuất phát từ đó người chơi sẽ thắng được sau khi thực hiện không quá k lần đi của mình, L_k là tập các nút mà xuất phát từ đó người chơi sẽ chắc chắn thua sau không quá k lần đi của mình.

Rõ ràng là $W_{k-1} \subset W_k$, $L_{k-1} \subset L_k$.

Nếu như một người nào đó sẽ thắng sau một số hữu hạn lần đi thì tồn tại các đẳng thức:

- $W = \bigcup_{k=0}^{\infty} W_k$,
- $L = \bigcup_{k=0}^{\infty} L_k$,
- và tương ứng có $D = E \setminus (W \cup L)$.

Công thức truy hồi xác định W_i và L_i là như sau:

Xuất phát từ $i = 0$ (từ W_0 và L_0):

- $W_{i+1} = \{u \mid \exists v: uv \in E \text{ và } v \in L_i\}$,
- $L_{i+1} = \{u \mid \forall v: uv \in E \Rightarrow v \in W_i\}$.

c – Giải thuật xây dựng W và L độ phức tạp $O(E)$

Chuẩn bị mảng **count** với **count[v]** bằng số bậc của đỉnh v . Dựa các phần tử của W_0 và L_0 vào hàng đợi, sau đó bắt đầu lấy thông tin từ hàng đợi ra xử lý. Có thể xảy ra các trường hợp sau:

c1. $u \in L$, khi đó:

```
for vu ∈ E do
    if ( v ∉ W) then
        begin W+= v;
        push(v)
    end;
```

c2. $u \in W$, khi đó:

```
for vu ∈ E do
    begin count[v]--;
    if (count[v] ==0) then
        begin L += v;
        push(v)
    end
end;
```

4.1.2 – Tổng trực tiếp. Hàm Sprague – Grundy

Có những trò chơi G có thể phân rã thành k trò chơi độc lập dạng đồ thị G_1, G_2, \dots, G_k . Mỗi trò chơi độc lập này được gọi là trò chơi con. Với mỗi trò chơi con cho biết trạng thái thua. Ở trạng thái này và chỉ ở trạng thái này mới không tồn tại nước đi tiếp theo hợp lệ.

Người chơi sẽ chọn một trong số các trò chơi con và thực hiện nước đi của mình. Người chơi bị thua khi không còn trò chơi con nào có thể lựa chọn để thực hiện được nước đi. Trong trường hợp này G được gọi là trò chơi có tổng trực tiếp.

Trò chơi G trong trường hợp này có thể biểu diễn như trò chơi trên một đồ thị duy nhất:

- $G = \langle V_1 \times V_2 \times \dots \times V_k, E \rangle$,
- $\langle (v_1, v_2, \dots, v_k), (u_1, u_2, \dots, u_k) \rangle \in E \Leftrightarrow \exists i: u_i v_i \in E_i$.

Định nghĩa 1. Hàm Sprague – Grundy SG là ánh xạ đinh sang tập nguyên không âm và thỏa mãn điều kiện:

$$SG(u) = \min\{m \geq 0: m \neq SG(v), uv \in E\}.$$

SG(u) được gọi là số Sprague – Grundy.

Như vậy, SG(u) là số nguyên không âm nhỏ nhất chưa tìm thấy trong tập giá trị số Sprague – Grundy đối với u.

Số Sprague – Grundy có các tính chất sau:

- $SG(u) = 0$ nếu bậc của u bằng 0,
- $SG(u) = 0$ nếu $u \in L$,
- Nếu $SG(u) = 0$ và $uv \in E$, thì $SG(v) \neq 0$,
- Nếu $SG(u) \neq 0$ thì $\exists v: uv \in E$ và $SG(v) = 0$.

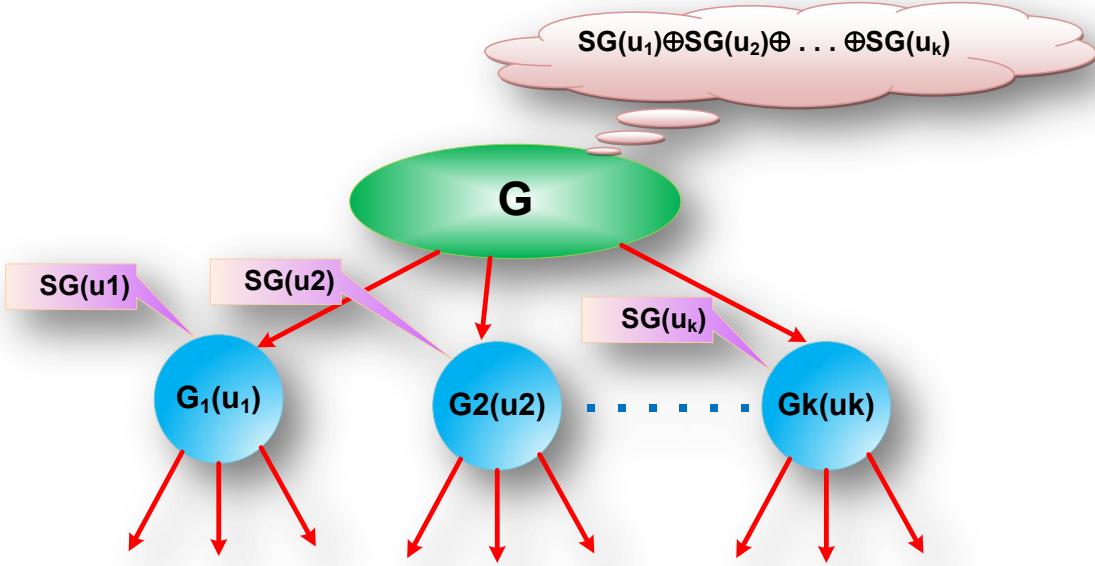
Định lý Grundy

$G = G_1 \times G_2 \times \dots \times G_k$ và SG_i là hàm Sprague – Grundy của G_i , $i = 1 \div k$, khi đó có

$$SG(\langle u_1, u_2, \dots, u_k \rangle) = SG(u_1) \oplus SG(u_2) \oplus \dots \oplus SG(u_k)$$

Trong đó \oplus là phép tính cộng từng bit không nhón:

(Phép XOR trong PASCAL, \wedge trong C).



Chứng minh

Ta sẽ chứng minh định lý này bằng phương pháp quy nạp. Với $k = 2$ ta có G_1, G_2 và các hàm SG_1, SG_2 . Ký hiệu $SG(u_1, u_2) = SG_1(u_1) \oplus SG_2(u_2)$. Ta sẽ chứng minh đây chính là hàm Sprague – Grundy cho G .

Nếu tồn tại cạnh $(u_1, u_2) \rightarrow (v_1, v_2)$ thì $SG_1(u_1, u_2) \neq SG_2(v_1, v_2)$. Giả thiết điều này không đúng, tức là có $SG(u_1, u_2) = SG(v_1, v_2)$. Khi đó $SG_1(u_1) \oplus SG_2(u_2) = SG_1(v_1) \oplus SG_2(v_2)$, suy ra $SG_1(u_1) = SG_1(v_1)$ – mâu thuẫn với định nghĩa của hàm SG_1 !

Giả thiết $x < SG_1(u_1) \oplus SG_2(u_2)$. Ta sẽ chứng minh rằng tồn tại cạnh nối nút (u_1, u_2) tới nút (v_1, v_2) thỏa mãn điều kiện $SG_1(v_1) \oplus SG_2(v_2) = x$.

Gọi b là vị trí đầu tiên của bít trong x mà bít x_b của x nhỏ hơn bít thứ b của $SG_1(u_1) \oplus SG_2(u_2)$. Khi đó tại vị trí thứ b các số $SG_1(u_1)$ và $SG_2(u_2)$ có giá trị bít khác nhau, còn $x_b = 0$.

Không mất tính chất tổng quát, ta có thể coi bít thứ b của $SG_1(u_1)$ bằng 1, còn bít thứ b của $SG_2(u_2)$ bằng 0. Rõ ràng là $SG_2(u_2) \oplus x < SG_1(u_1)$. Ngược lại trong đồ thị G_1 có cạnh nối $u_1 \rightarrow v_1$ và $SG(v_1) = SG_1(u_1) \oplus x$. Còn trong G ta có cạnh $(u_1, u_2) \rightarrow (v_1, v_2)$ và $SG(< v_1, v_2 >) = SG_1(v_1) \oplus SG_2(v_2) = x$.

Điều này nói lên rằng SG là hàm Sprague – Grundy của G. Đó là điều phải chứng minh.

Bằng cách quy nạp, ta có thể dễ dàng chứng minh trong trường hợp tổng quát.

Ứng dụng

a – Trò chơi NIM.

Có n đống sỏi, đống sỏi thứ i có a_i viên ($a_i > 0, i = 1 \div n$). Có 2 người chơi. Mỗi người, khi đến lượt mình phải bốc một số lượng sỏi tùy ý, lớn hơn 0 từ một đống tùy chọn. Ai đến lượt mình không còn cách bốc thì người đó thua.

Phân tích

Với mỗi đống có số lượng \mathbf{x} $SG(\mathbf{x}) = \mathbf{x}$. Như vậy hàm Sprague – Grundy có dạng:

$$SG(<\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n>) = \mathbf{a}_1 \oplus \mathbf{a}_2 \oplus \dots \oplus \mathbf{a}_n.$$

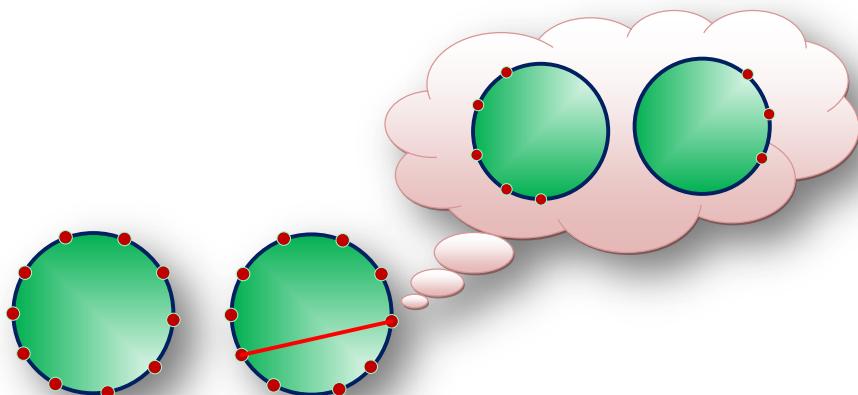
Như vậy người chơi sẽ thua khi đến lượt mình đi có $\mathbf{a}_1 \oplus \mathbf{a}_2 \oplus \dots \oplus \mathbf{a}_n = 0$.

b – Trò chơi Omaks

Cho n điểm khác nhau trên đường tròn. Có 2 người chơi. Mỗi người khi đến lượt mình tạo một dây cung mới bằng cách nối 2 trong số các điểm đã cho sao cho dây cung mới không có điểm chung với bất kỳ dây cung nào đã có trước đó. Ai đến lượt mình đi không tạo được dây cung mới là thua.

Phân tích

Sau khi kẻ một dây cung ta có thể hình dung như bây giờ có 2 bài toán trò chơi tương tự ban đầu với một đường tròn có m_1 điểm và đường tròn kia có m_2 điểm, $m_1, m_2 > 0$ và $m_1 + m_2 + 2 = k$.



Khi đó có :

$$SG(\mathbf{k}) = \min\{ \mathbf{i}, \mathbf{i} \neq SG(\mathbf{m1}) \oplus SG(\mathbf{m2}), \mathbf{m1+m2+2=k} \}$$

c – Trò chơi Grundy

Xét trò chơi giữa 2 người. Ban đầu có đống gồm n đồng tiền xu. Mỗi người đến lượt mình đi, chọn một đống và chia nó thành 2 đống với số xu ở mỗi đống là khác nhau (trong số 2 đống này). Ai đến lượt mình không còn cách chia là thua.

Phân tích

Người ta đã khảo sát hàm Sprague – Grundy với n đạt tới 2^{35} và trong khoảng nói trên có 42 giá trị 0. Trên cơ sở các giá trị đã khảo sát, người ta đã đưa ra giả thuyết là hàm có tính chất tuần hoàn. Tuy vậy, điều này vẫn chưa được chứng minh và bài toán trò chơi này vẫn thuộc loại chưa giải được trong trường hợp tổng quát.



4.1.3 – Trò chơi trên ma trận

Cho ma trận $\mathbf{A} = ((\mathbf{a}_{ij}))$, $i = 1 \div m$, $j = 1 \div n$. Các trò chơi xác định trên ma trận \mathbf{A} là trò chơi trên ma trận.

Ví dụ 1: Với ma trận \mathbf{A} nói trên, với tổng s ban đầu bằng 0 và chưa có hàng hay cột nào bị đánh dấu xóa hai người thực hiện k lần đoán số ($k < \min\{m, n\}$). Mỗi lần hai người đồng thời chọn, người thứ nhất chọn một hàng i trong số các hàng chưa bị đánh dấu xóa, người thứ 2 chọn cột j trong số các cột chưa bị đánh dấu xóa. Phần tử nằm ở giao của hàng i cột j này, tức là phần tử a_{ij} sẽ là điểm số của người thứ nhất và được cộng vào s , sau đó hàng i và cột j bị đánh dấu xóa. Hãy xác định tổng s lớn nhất có thể đạt được nếu cả hai cùng có sự lựa chọn tối ưu.

Ở bài toán này, người thứ nhất cố gắng cực đại hóa giá trị sẽ cộng vào s , còn người thứ 2 – tìm cách cực tiểu hóa giá trị này.

Phần lớn các trò chơi trên ma trận liên quan tới bài toán Minimax. Những bài toán này có nhiều ứng dụng trong kinh tế. Việc giải những bài toán thực tế đã là động lực ra đời và phát triển những lý thuyết như quy hoạch tuyến tính, quy hoạch cầu phương, lý thuyết Maximin, v. v. . .

Trong phạm vi chương trình trung học phổ thông chúng ta gặp những bài toán đơn giản hơn và trong nhiều trường hợp – được phát biểu không ở dưới dạng một trò chơi tinh minh.

Ví dụ 2:

QUÂN TƯỢNG

Mirko vẽ một bàn cờ kích thước $2n \times 2n$ ô và nghĩ ra trò chơi như sau:

Trên mỗi ô Mirko ghi một số nguyên gọi là giá trị của ô. Tại 2 ô ở giữa hàng trên cùng (ở cột n và $n+1$) anh đặt 2 quân tượng, mỗi quân ở một ô, ghi nhận những ô mà các quân tượng này kiểm soát, tức là những ô nằm trên đường chéo hoặc đường chéo của ô có quân tượng, gọi đó là những ô nhìn thấy được. Những ô nhìn thấy được không bao gồm ô có quân đứng!

Tổng giá trị các ô nhìn thấy được là điểm số ban đầu của trò chơi.

Mirko thực hiện k nước đi, ở mỗi nước Mirko di chuyển một quân tượng nào đó sang ô mới theo cách đi của quân tượng trên bàn cờ. Ở vị trí mới, một số ô trước kia chưa nhìn thấy bao giờ bây giờ trở thành nhìn thấy. Giá trị của những ô này sẽ được cộng vào điểm số của trò chơi.

Hãy xác định điểm số lớn nhất có thể đạt được.

Dữ liệu: Vào từ file văn bản BISHOP.INP:

- Dòng đầu tiên chứa 2 số nguyên n và k ($1 \leq n \leq 10, 0 \leq k \leq 100$),
- Mỗi dòng trong $2n$ dòng tiếp theo chứa $2n$ số nguyên xác định giá trị các ô trong một dòng, các số nguyên có giá trị tuyệt đối không vượt quá 10^6 .

Kết quả: Đưa ra file văn bản BISHOP.OUT một số nguyên – điểm số lớn nhất có thể đạt được.

Ví dụ:

BISHOP.INP						
2	1					
0	-9	-9	0			
0	1	1	0			
1	0	0	1			
0	0	6	0			

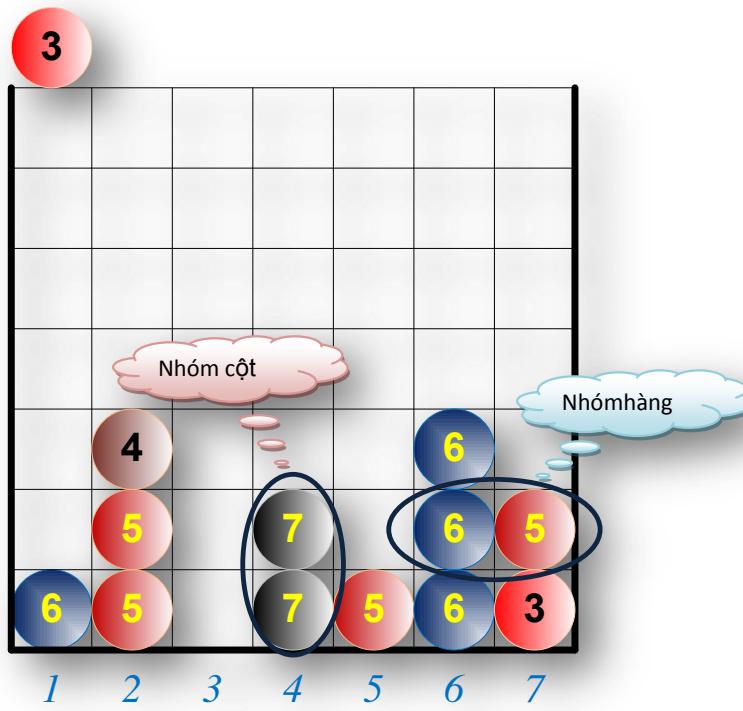
BISHOP.OUT						
1						

Ví dụ 3:

XÓA BI

Xóa bi là trò chơi một người. Cho lưới ô vuông kích thước 7×7 , các cột được đánh số từ 1 đến 7 từ trái sang phải. Ban đầu lưới ô vuông rỗng. Ở mỗi lượt đi sẽ có một viên bi rơi xuống từ trên cột i . Viên bi được đánh một trong các số từ 1 đến 7. Ở đây chúng ta chỉ xem xét việc xử lý bảng khi viên bi rơi xuống cột i . Viên bi sẽ rơi xuống đáy cột i nếu cột này rỗng hoặc dừng lại khi chạm viên bi trên cùng trong cột. Mỗi viên bi chiếm đúng 1 ô.

Ví dụ, nếu ta thả viên bi 3 ở cột 1 nó sẽ dừng lại trên viên bi 6, nếu thả nó ở cột 3, nó sẽ dừng lại ở đáy, còn nếu thả ở cột 7 – dừng trên viên bi 5. Các viên bi trong một cột tạo thành *nhóm cột*. Dãy các viên bi kề nhau trong một hàng hai đầu của dãy là ô trống hoặc biên của lưới tạo thành một *nhóm hàng*. Số lượng bi trong nhóm gọi kích thước của nhóm. Nếu có



một hoặc một vài viên bi trong nhóm có số đúng bằng kích thước của nhóm thì những viên bi đó sẽ bị bốc hơi, biến mất khỏi bảng, những ô chứa các viên bi này sẽ trở thành ô trống và các viên bi còn lại (nếu có) trong cột sẽ rơi xuống dưới, lấp vào các ô trống.

Ví dụ, nếu viên bi 3 được thả vào cột 4, nó sẽ tạo thành một nhóm cột kích thước 3 và nó sẽ bị bốc hơi. Nếu thả nó vào cột 7, một nhóm cột kích thước 3 cũng sẽ hình thành, các viên bi ở dòng 1 và 3 sẽ bị bốc hơi và viên bi 4 sẽ rơi xuống đáy. Việc các viên bi bốc hơi làm thay đổi cấu hình bi trong bảng, hình thành những nhóm hàng, nhóm cột mới và lại có thể làm một số viên bi nào đó khác bốc hơi!

Cho bảng rỗng ban đầu. Xét n viên bi lần lượt rơi xuống, viên bi thứ i có số là v_i . Hãy chọn cột cho các viên bi rơi xuống để sao cho sau khi các viên bi đã rơi hết trong bảng còn lại ít viên bi nhất. Nếu ở một thời điểm nào đó, mọi cột đã đầy mà vẫn có bi rơi xuống thì đưa ra thông báo “**Game Over!**”.

Dữ liệu: Vào từ file văn bản DROP.INP:

- Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 1000$),
- Dòng thứ 2 chứa xâu độ dài n chỉ chứa các ký tự số trong phạm vi từ 1 đến 7. Ký tự thứ i là giá trị vi.

Kết quả: Đưa ra file văn bản DROP.OUT một số nguyên – số ít nhất các viên bi còn lại trong bảng hoặc đưa ra dòng thông báo “**Game Over!**”.

Ví dụ:

DROP.INP
6
236416

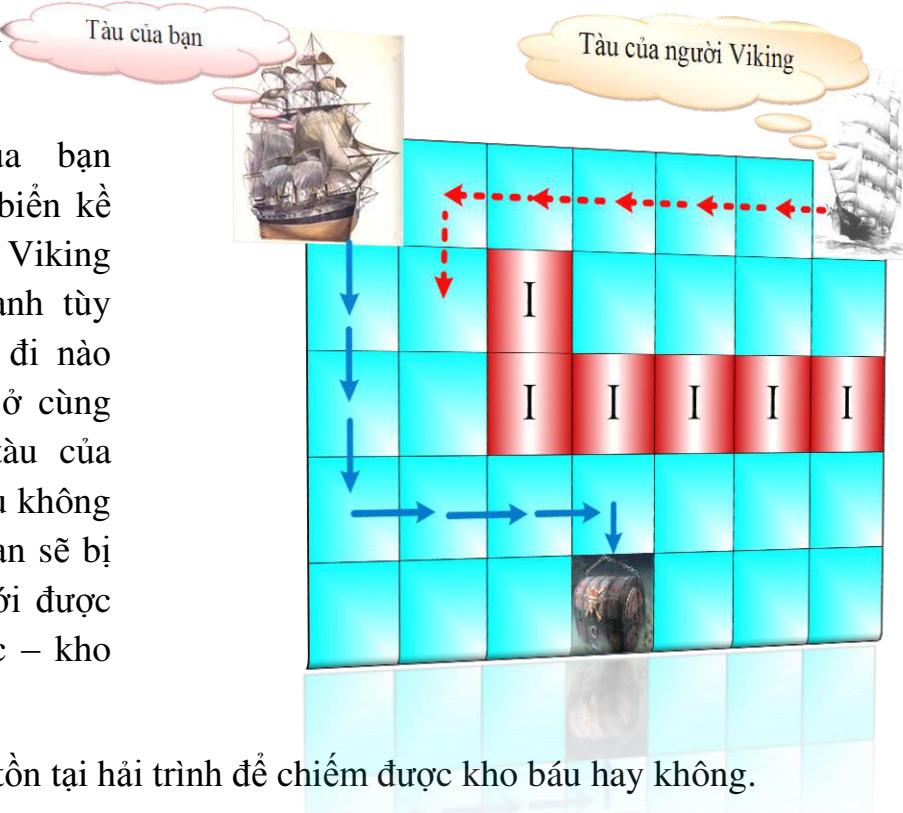
DROP.OUT
2

Ví dụ 4: NGƯỜI VIKING

Bạn đang có trong tay hải đồ vùng biển hình chữ nhật kích thước $n \times m$ ô, một số ô là đảo, còn lại là biển. Trên hải đồ có đánh dấu một ô mà dưới mặt nước ở đó có một kho báu vô giá. Tàu của bạn đang ở một ô trên vùng biển này, ngoài ra còn có một tàu của cướp biển

Viking cũng đang trên vùng biển này.

Mỗi bước, tàu của bạn chuyển động sang ô biển kè cạnh và tàu của người Viking đi sang một ô kè cạnh tùy chọn. Sau một bước đi nào đó, nếu tàu của bạn ở cùng hàng hoặc cột với tàu của Viking và ở giữa 2 tàu không có ô đảo nào chắn, bạn sẽ bị bắn chìm. Nếu bạn tới được ô chứa kho báu trước – kho báu sẽ là của bạn!



Hãy xác định xem có tồn tại hải trình để chiếm được kho báu hay không.

Dữ liệu: Vào từ file văn bản VIKINGS.INP:

- Dòng đầu tiên chứa 2 số nguyên n và m ($1 \leq n, m \leq 700$),
- Mỗi dòng trong n dòng sau chứa xâu độ dài m từ tập ký tự $\{., I, Y, V, T\}$, ký tự “.” ứng với ô biển, “I” – ô đảo, “Y” – vị trí ban đầu của tàu bạn, “V” – vị trí tàu của Viking, “T” – nơi có kho báu. Các ký tự “Y”, “V” và “T” chỉ xuất hiện đúng một lần.

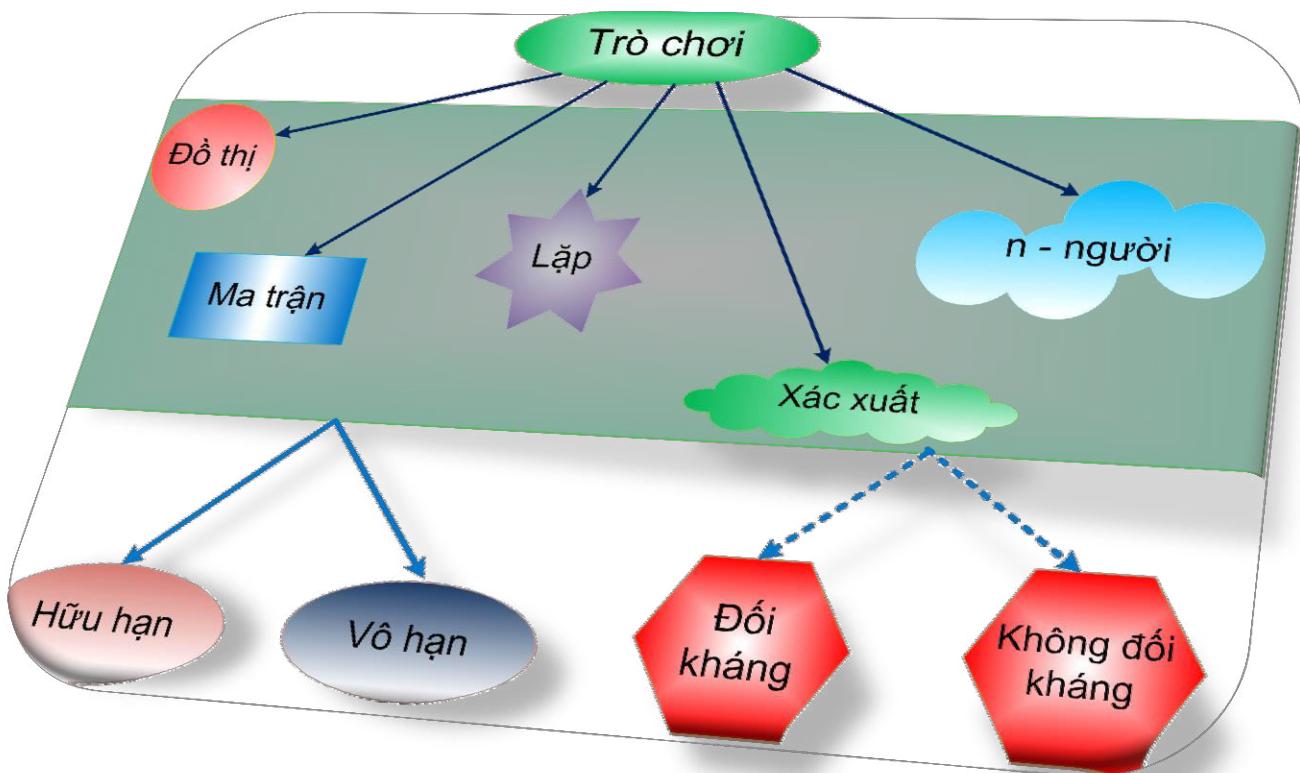
Kết quả: Đưa ra file văn bản VIKINGS.OUT thông báo “YES” hoặc “NO”.

Ví dụ:

VIKINGS.INP
5 7
Y.....V
..I....
..IIIII
.....
...T...

VIKINGS.OUT
YES

4.1.4 – Các loại trò chơi khác

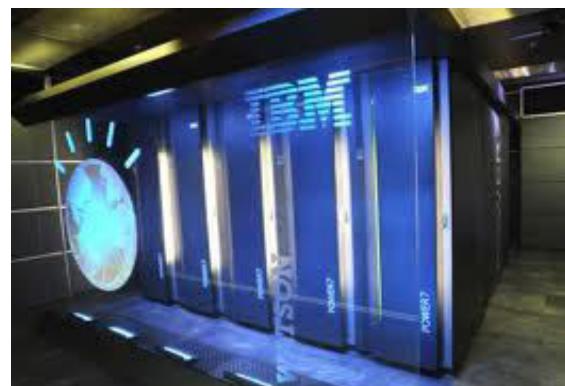


4.2 – KỸ THUẬT LẬP TRÌNH

Việc lập trình giải các bài toán trò chơi là một vấn đề phức tạp. Mỗi chương trình giải bài toán trò chơi trên thực tế là một hệ thống trí tuệ nhân tạo thu nhỏ. Công ty IBM đã từng lắp ráp siêu máy tính Deep Blue chỉ để thử nghiệm và chứng minh khả năng xây dựng trí tuệ nhân tạo. Chương trình chơi cờ vua (cờ Quốc tế) đã được xây dựng, cài đặt trên Deep Blue và đại kiện tướng, vô địch cờ thế giới Garry Kasparov đã được mời tới đấu cờ với máy tính.



Garry Kasparov đấu cờ với máy tính
Deep Blue



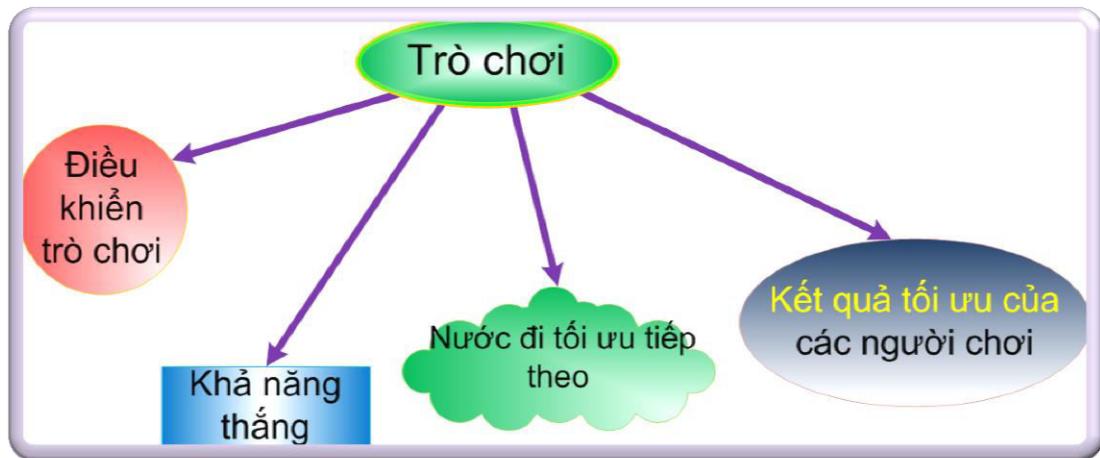
Máy tính Deep Blue

Các bài toán trò chơi có đặc trưng chung:

- Các phép xử lý trạng thái ít và đơn giản,
- Lô gic để dẫn xuất, lựa chọn phép xử lý – phức tạp.

Sự phức tạp này tạo ra đặc thù cho việc tiếp cận và lập trình giải quyết các bài toán trò chơi. Về nguyên tắc với các bài toán trò chơi trên đồ thị ta có thể tính giá trị hàm Sprague – Grundy (gọi ngắn gọn là **số Grundy**) và dựa vào nó để điều khiển trò chơi. Tuy vậy, “Tránh vỏ dưa, gấp vỏ dừa”! Việc tính số Grundy trong nhiều trường hợp là hết sức phức tạp, thậm chí có thể là không tính được. Khi đó người ta phải tìm cách vòng tránh.

Các bài toán lập trình thường gặp với bài toán trò chơi



Dưới đây chúng ta sẽ xem xét một số kỹ thuật giải bài toán trò chơi.

4.2.1 – Tính trực tiếp hàm Sprague – Grundy

Bản chất của việc điều khiển trò chơi dựa trực tiếp vào hàm Sprague – Grundy là **xác định** hoặc **đánh giá** số lượng nước đi còn lại. Trong trò chơi 2 người, nếu các người chơi đều biết cách đi đúng thì người đi ở bước tiếp theo thắng được khi và chỉ khi để lại một số chẵn nước đi. Nếu đổi phương còn nước đi, ta cũng sẽ còn nước để đi. Nếu hết nước đi – đổi phương hết trước.

Trò chơi NIM: Với trò chơi NIM đã phát biểu ở phần lý thuyết, hãy xác định ở lần đi đầu tiên người thứ nhất có mấy cách bốc sỏi để thắng. Hai cách bốc gọi là khác nhau nếu nó được thực hiện ở những đống khác nhau.

Có n đống sỏi, đống i thứ i có a_i viên ($a_i > 0$, $i = 1 \dots n$). Có 2 người chơi. Mỗi người, khi đến lượt mình phải bốc một số lượng sỏi tùy ý, lớn hơn 0 từ một đống tùy chọn. Ai đến lượt mình không còn cách bốc thì người đó thua.

Dữ liệu: Vào từ file văn bản NIM.INP:

- Dòng đầu tiên chứa số nguyên n ($2 \leq n \leq 1000$),
- Dòng thứ i trong n dòng sau chứa số nguyên a_i ($1 \leq a_i \leq 10^9$).

Kết quả: Đưa ra file văn bản NIM.OUT:

- Dòng đầu tiên đưa ra số nguyên k – số cách bốc khác nhau có thể thực hiện. $k = 0$ nếu không có cách bốc để thắng.
- Nếu $k > 0$ thì mỗi dòng trong k dòng sau đưa ra 2 số nguyên j và b_j theo thứ tự tăng dần của j , xác định cần bốc b_j viên sỏi từ đống thứ j .

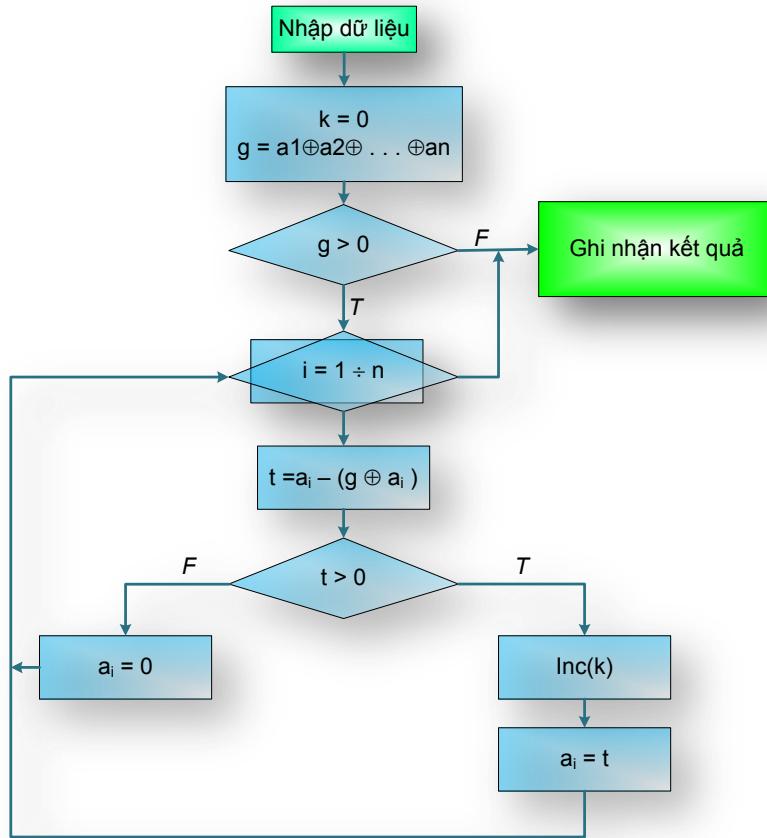
Ví dụ:

NIM.INP	NIM.OUT
4	3
12	1 9
8	2 1
5	4 13
14	

$$\begin{aligned}
 a_1 &= 12_{10} = 1100_2 \rightarrow g-a_1 = 3 \rightarrow \text{số sỏi cần bốc: } 9 \\
 a_2 &= 8_{10} = 1000_2 \rightarrow g-a_2 = 7 \rightarrow \text{số sỏi cần bốc: } 1 \\
 a_3 &= 5_{10} = 0101_2 \rightarrow g-a_3 = 10 \rightarrow \text{không bốc được từ đống này} \\
 a_4 &= 14_{10} = 1110_2 \rightarrow g-a_4 = 1 \rightarrow \text{số sỏi cần bốc: } 13 \\
 &\underline{\qquad\qquad\qquad} \\
 g &= 1111
 \end{aligned}$$

Lưu ý: Giá trị $g - a_i$ có thể nhận được bằng phép tính lô gic $g \oplus a_i$.

Sơ đồ giải thuật:



Chương trình PASCAL

```

Program NIM;
Const tfi='NIM.INP';
      tfo='NIM.OUT';

Var a:array[1..1000] of longint;
    i,n,g,t,k:longint;
    fi,fo:text;

BEGIN
  assign(fi,tfi);
  reset(fi);
  readln(fi,n);
  for i := 1 to n do readln(fi,a[i]);
  close(fi);
  g:=0;
  
```

```

for i:=1 to n do g:=g xor a[i];
k:=0;
for i:=1 to n do
begin
  t:=a[i]-(a[i] xor g);
  if t>0 then
    begin
      inc(k);
      a[i]:=t
    end
    else a[i]:=0
  end;
assign(fo,tfo); rewrite(fo);
writeln(fo,k);
if k > 0 then
  for i:=1 to n do if a[i]>0 then writeln(fo,i,' ',a[i]);
close(fo)
END.

```

Chương trình C++

```

#include <iostream>
#include <conio.h>
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{long n,g,t,k;
ifstream fi ("NIM.INP");
ofstream fo ("NIM.OUT");
fi >> n;
{long a[n+1];
for (long i = 1; i <=n;i++) fi>>a[i];
k=0;
g=0;
for (long i=1; i <=n;i++) g^=a[i];
if (g>0)
{ for (long i=1;i<=n;i++)
  { t = a[i] - (a[i]^g);
    if (t>0)
      {k++;a[i]=t;
       } else a[i]=0;
     }
}
fo<<k<<endl;
if (k>0)

```

```

        for (long i= 1; i<=n;i++) if (a[i]>0) fo<<i << "
"<<a[i]<<endl;
fi.close(); fo.close();
}
}

```

4.2.2 – Kỹ thuật bảng phương án (Decide Table)

Giả thiết hành động cần thực hiện ở bước tiếp theo phụ thuộc vào kết quả kiểm tra n điều kiện C_1, C_2, \dots, C_n . Điều kiện C_i ($i = 1 \div n$) có thể là điều kiện lô gic với kết quả kiểm tra là Đúng (**True**) hoặc Sai (**False**) hoặc có thể là điều kiện với kết quả kiểm tra nằm trong tập có nhiều hơn 2 giá trị. Ví dụ, việc so sánh 2 số p và q có thể cho kết quả <, = hoặc >.

Gọi v_i là kết quả kiểm tra điều kiện C_i . Việc kiểm tra tất cả các điều kiện từ C_1 đến C_n sẽ cho ta bộ giá trị $\mathcal{V} = (v_1, v_2, \dots, v_n)$. Bộ giá trị này được gọi là *vec tơ điều kiện*. Với mỗi \mathcal{V} cần phải thực hiện một hành động A nào đó. Véc tơ (\mathcal{V}, A) được gọi là *quy tắc hành động*.

Bảng phương án (Decide Table) là *bảng liệt kê các quy tắc hành động*.

Lý thuyết bảng phương án nghiên cứu các vấn đề:

- Phân loại bảng phương án,
- Cách xây dựng và biểu diễn bảng phương án,
- Tính chất bảng phương án,
- Sử dụng bảng phương án như công cụ phân tích và thiết kế giải thuật,
- Sử dụng bảng phương án như công cụ tự động hóa lập trình.

Bảng phương án đóng một vai trò hết sức quan trọng trong lập trình lô gic, trong các hệ thống trí tuệ nhân tạo, hệ hỗ trợ quyết định v. v. . .

Bằng bảng phương án ta có thể dễ dàng kiểm định giải thuật trong các sách báo, tài liệu giới thiệu trò chơi, chỉnh lý các sai sót liên quan tới việc xử lý các trường hợp tinh tế của trò chơi. Các sai sót này không phải là quá hiếm!

Ở đây chúng ta không đi sâu vào lý thuyết bảng phương án mà chỉ xem xét cách triển khai nó để giải quyết một số bài toán trong lĩnh vực trò chơi.

Ví dụ: Cho bàn cờ kích thước $m \times n$, các dòng ô được đánh số từ 1 đến m từ dưới lên trên, các cột ô được đánh số từ 1 đến n từ trái qua phải. Ở vị trí (p, q) có một quân mã ($1 \leq p \leq m, 1 \leq q \leq n$).

Hai người chơi lần lượt đi quân mã theo quy tắc sau:

- Không được đi ra ngoài bàn cờ,
- Không được vượt lên trên đường thẳng chưa đường chéo chính của ô con mã đang đứng.

Ai đến lượt mình không thể đi được là thua.

Hãy lập trình trò chơi này giữa người và máy trong chế độ đối thoại, đảm bảo khả năng thắng của máy là lớn nhất. Máy là người đi trước. Thông tin về nước đi của người được nhập từ bàn phím. Để đơn giản, giả thiết thông tin nhập vào là đúng, không cần kiểm tra.

Dữ liệu: Vào từ file INPUT chuẩn của hệ thống gồm một dòng chứa 4 số nguyên m, n, p và q ($2 \leq m, n \leq 1000$).

Kết quả: Đưa ra file OUTPUT chuẩn của hệ thống các nước đi của máy, mỗi nước đi là 2 số nguyên u và v xác định ô con mã tới.

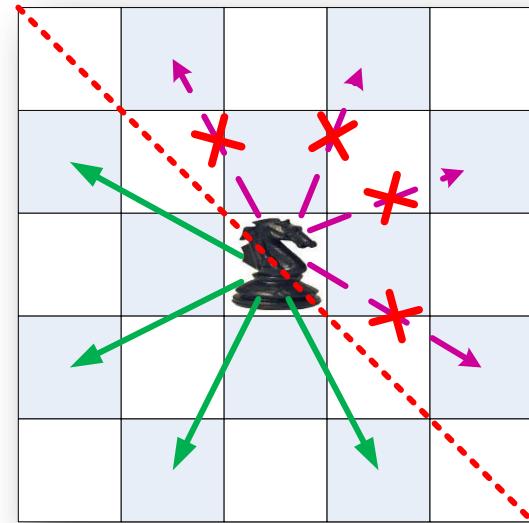
Bài toán có mô hình tương đương:

Có 2 đống sỏi với số lượng tương ứng là m và n viên. Hai người chơi bốc sỏi. Khi đến lượt mình người chơi được bốc theo quy tắc sau:

- Bốc 2 viên ở một đống nào đó và 1 viên ở đống kia,
- Bốc 2 viên ở một đống nào đó và thả một viên vào đống kia (kể cả khi nó trống).

Ai đến lượt mình không có cách bốc là thua.

Giải thuật:



Để giải bài này ta cần xây dựng bản đồ đánh dấu tất cả các ô trên bàn cờ, phân chung thành 2 loại: ô từ đó có cách đi thắng (đánh dấu +) và ô từ đó không có cách đi thắng nếu đối phương không phạm sai lầm, những ô này được đánh dấu -.

Bản đồ được xây dựng như sau:

- Đánh dấu 4 ô tạo thành hình vuông ở góc dưới trái.
- Bằng cách loang (BFS) từ 4 ô ban đầu này có thể đánh dấu được tất cả các ô còn lại. Một ô chưa được đánh dấu sẽ bị đánh dấu - nếu tồn tại một cách đi tới ô đã được đánh dấu +. Ngược lại, một ô chưa được đánh dấu sẽ được đánh dấu + nếu tồn tại một cách đi tới ô đã được đánh dấu -.

									$m = 8, n = 9$
8	+	+	+	+	+	+	+	-	
7	+	+	+	+	+	+	+	-	
6	-	-	+	+	-	-	+	-	
5	-	-	+	+	-	-	+	-	
4	+	+	+	+	+	+	+	-	
3	+	+	+	+	+	+	+	-	
2	-	-	+	+	-	-	+	-	
1	-	-	+	+	-	-	+	-	
	1	2	3	4	5	6	7	8	9

Bản đồ này gọi là bảng phương án, dùng để điều khiển trò chơi.

Nhận xét:

Với điều kiện đơn giản của bài toán, trong trường hợp này ta có thể dẫn xuất công thức giải tích (phụ thuộc vào m và n) để xác định một (i, j) thuộc loại nào, khi đó bảng phương án chủ yếu phục vụ cho việc phân tích, thiết kế giải thuật và tồn tại trong chương trình dưới dạng ẩn.

Việc xây dựng tường minh bảng phương án cho phép:

- Dễ dàng tìm ra nước đi cần thiết tiếp theo, đặc biệt khi người chơi thứ 2 phạm sai lầm,
- Cho phép điều khiển trò chơi theo một trong số các chiến lược:
 - Chiến thắng trong thời gian nhanh nhất có thể,
 - Kéo dài thời gian đến mức tối đa có thể,
 - Làm cho trò chơi diễn ra đa dạng: nếu chơi lại và người thứ 2 lặp lại nước đi cũ, chương trình cho nước đi khác, đảm bảo xác xuất các nước đi trùng nhau ở 2 ván là gần như bằng 0.

Trong bảng phương án này (cũng như trong tuyệt đại bộ phận các bảng phương án cho trò chơi) trong bảng chỉ chứa các thông tin cho phép người chơi tìm được sự lựa chọn thù hợp với chiến lược điều khiển mình theo đuổi.

Bảng phương án cho bài toán Lật xúc xắc

Để điều khiển trò chơi này cần có bảng phương án B kích thước $6 \times S_{\max}$.

$$i = 1 \div 6, j = 1 \div S_{\max}.$$

Bảng phương án cho bài toán này được xây dựng trên cơ sở phân tích diễn biến trò chơi từ cuối về đầu, tức là với j thay đổi từ S_{\max} về 1.

Bài toán: Lật xúc xắc

Cho một con xúc xắc truyền thống, trên mỗi mặt của xúc xắc có một số chẵn trong phạm vi từ 1 đến 6 xác định số điểm của mặt, không có 2 mặt nào có cùng số điểm và tổng điểm của 2 mặt đối luôn luôn bằng 7. Con xúc xắc được tung lên bàn và tổng S của trò chơi ban đầu nhận giá trị bằng số điểm mặt trên của xúc xắc. Hai người lắc lượt đi. Khi đến lượt mình đi, người chơi lật một lần con xúc xắc qua cạnh của nó và cộng số điểm ở mặt trên mới vào S . Ai đến lượt mình đi làm tổng S lớn hơn S_{\max} sẽ thua.

Cuộc chơi đang diễn ra sôi nổi thi người đến lượt đi có điện thoại và lúng túng xin lỗi phải đi làm một việc gấp. Vì lý do tế nhị, không ai hỏi đó là việc gì, nhưng bạn được chỉ định thay thế. Với tổng S và mặt trên v hiện có, hãy xác định cuối cùng bạn có thể thắng được hay không và nếu có thì chỉ ra các cách lật để thắng.

Đứa liệu: Vào từ file văn bản DICE.INP gồm một dòng chứa 3 số nguyên v , S và S_{\max} ($1 \leq v \leq 6, 1 \leq S \leq S_{\max} \leq 10^5$). Các số ghi cách nhau một dấu cách.

Kết quả: Đưa ra file văn bản DICE.OUT trên một dòng số nguyên m – số các đi thắng và nếu $m > 0$ thì sau đó là m số nguyên – các mặt cần đưa thành mặt trên, đưa ra theo thứ tự tăng dần. Các số ghi cách nhau một dấu cách. $m = 0$ ứng với trường hợp không có cách thắng.

Ví dụ:

DICE.INP
6 13 20

DICE.OUT
3 2 3 4

Ở bài toán đang xét ta chỉ cần một phần của bảng phương án. Trên

$$B_{ij} = \begin{cases} 1 - \text{để thắng cần đưa xúc xắc về trạng thái mặt}\\ \text{trên là } i \text{ khi } S = j, \\ 0 - \text{sẽ thua nếu đưa xúc xắc về trạng thái mặt}\\ \text{trên là } i \text{ khi } S = j. \end{cases}$$

phương diện giải thuật, việc xây dựng một phần hay toàn bộ bảng phương án là như nhau.

Bảng phương án được xây dựng theo nguyên tắc truy hồi. Giải thiết B_{ij} đã được xác

$$t = \sum_{\substack{p=1, \\ p \neq i, p \neq 7-i}}^6 B_{p,k+i}$$

$$B_{ik} = \begin{cases} 1 & \text{nếu } t = 0, \\ 0 & \text{trong trường hợp ngược lại.} \end{cases}$$

định với $i = 1 \div 6, j = S_{\max} \div k+1$. B_{ik} sẽ được xác định như sau:

Để áp dụng được công thức truy hồi trên ta cần có 6 cột cuối cùng của bảng. Các cột này có thể dễ dàng xây dựng dựa trên nhận xét: khi $j = S_{\max}$ người đi ở bước tiếp theo, dù lật mặt nào của xúc xắc lênh trên cũng thua vì làm cho tổng $S > S_{\max}$, như vậy, $B_{i,S_{\max}} = 0, i = 1 \div 6$.

Các cột này cũng có thể được tính theo sơ đồ chung nếu khởi tạo $B_{ij} = 1, i = 1 \div 6, j = S_{\max}+1 \div S_{\max}+6$.

Nhận xét:

- Để xác định giá trị một cột ta chỉ cần dựa trên giá trị của 6 cột trước đó,
- Nếu xây dựng bảng phương án với 10^5 cột, ta có thể dùng nó để điều khiển trò chơi với S_{\max} bất kỳ thỏa mãn $1 \leq S_{\max} \leq 10^5$ mà không cần xây dựng lại bảng phương án (bằng cách sử dụng S_{\max} cột cuối cùng của bảng),
- Với bài toán đang xét, ta chỉ cần B với các cột từ S_{\max} đến S.

Chương trình PASCAL

```
Program DICE;
Const tfo='DICE.INP';
      tfo='DICE.OUT';

Var b:array[1..6,-5..1000] of byte;
```

```

i,j,v,s,smax,n,m:longint;
fi,fo:text;

Procedure xd_b(p,q:longint);
var t,i,j:longint;
Begin
  t:=0;
  for i:=1 to 6 do t:= t + b[i,q-p];
  t:=t-b[p,q-p]-b[7-p,q-p];
  if t >0 then b[p,q]:=0 else b[p,q]:=1
End;

BEGIN
  assign(fi,tfi);
  reset(fi);
  readln(fi,v,s,smax);
  close(fi);
  assign(fo, tfo); rewrite(fo);
  for j := -5 to 0 do
    for i := 1 to 6 do b[i,j]:=1;
n:=smax-s+1;
  for j:=1 to n do
    for i := 1 to 6 do xd_b(i,j);
  { Dua ra bang phuong an }

  for i:=1 to 6 do
    begin
      for j:=1 to n do write(fo,b[i,j],' ');
      writeln(fo);
    end;
    writeln(fo);
  { Het dua ra bang phuong an}
m:=0; b[v,n]:=0; b[7-v,n]:=0;
  for i:= 1 to 6 do m:= m+b[i,n];
  write(fo,m);
  if m > 0 then
    for i:= 1 to 6 do if b[i,n]>0 then write(fo,' ',i);
  close(fo)
END.

```

Chương trình C++

```

#include <conio.h>
#include <iostream>
#include <stdlib.h>
using namespace std;
long v,n,s,smax,m;
int b[6][10007];

```

```

ifstream fi ("DICE.INP");
ofstream fo ("DICE.OUT");
int xd_b(long i,long j)
{long t,r,j1;
t=0;j1=j-i-1;
for (int k=0; k<6; k++) t+=b[k][j1];
t-=(b[i][j1]+b[5-i][j1]);
if (t>0) b[i][j]=0; else b[i][j]=1;
}

int main()
{
    fi>>v>>s>>smax;
    n=smax-s+7;
    for (int j=0; j<6; j++)
        for (int i=0; i<6; i++) b[i][j]=1;
    for (long j=6; j<n; j++)
        for (int i=0; i<6; i++) xd_b(i,j);
/*
    Dua ra bang phuong an de tham khao
    for (int i=0;i<6;i++)
        {for (long j=6; j<n; j++) fo<<b[i][j]<<' ';
        fo<<endl;
        }
        fo<<endl;
    Het dua ra bang phuong an
*/
    n--; m=-b[v-1][n];
    for(int i=0;i<6;i++) m+=b[i][n];
    fo<<m; b[v-1][n]=0;
    if (m>0) for(int i=0;i<6;i++) if (b[i][n] ==1) fo<<" "<<i+1;
    fi.close(); fo.close();
}

```

Lưu ý:

- Có sự khác biệt giữa giải thuật hiện thực hóa trong chương trình với giải thuật trình bày theo mô hình toán học. Ở phần phân tích, giải thuật được trình bày phù hợp tối đa với lý thuyết, để làm rõ nguồn gốc và bản chất của giải thuật. Khi lập trình, giải thuật cần được cụ thể hóa để thuận tiện cho việc lập trình và hiệu chỉnh. Sơ đồ giải thuật cần được chuẩn bị lại trước khi bắt tay vào lập trình.
- Trong các chương trình trên đều đoạn các câu lệnh (đã được biến thành thông tin chủ thích) đưa ra bảng phương án. Đây là thông tin trung gian cần thiết phục vụ hiệu chỉnh trong trường hợp phải lập trình điều khiển trò chơi.

Dĩ nhiên, sau khi hoàn thiện chương trình, đoạn câu lệnh này cần phải loại bỏ.

Kỹ thuật bảng phương án cho phép ta vòng tránh việc phải tính tường minh giá trị hàm Sprague – Grundy khi trò chơi không bị phân rã thành các trò chơi con độc lập.

Với những bài toán bị phân rã thành các trò chơi độc lập (không nhất thiết phải theo cùng một quy tắc chơi như bài toán ban đầu) bảng giá trị số Grundy là một thành phần của bảng phương án với 3 chức năng:

- Cho biết người đi ở nước tiếp theo có thể thắng được hay không,
- Cung cấp các thông tin để tìm ra chiến lược điều khiển (tìm nước đi),
- Làm cơ sở để tính số Grundy cho các trò chơi với bộ tham số có giá trị lớn hơn.

Ở các loại trò chơi này việc lưu trữ điều khiển là không tối ưu vì miền xác định của tập trạng thái lớn. Với mỗi trạng thái của trò chơi điều khiển được tìm bằng con đường giải thuật, dựa vào bảng số Grundy đã có. Đó là kỹ thuật áp dụng để giải các bài toán như Trò chơi Omaks, Trò chơi Grundy.

Lưu ý rằng người ta với trò chơi thứ 2 vẫn chưa tính được số Grundy trong trường hợp tổng quát!

Trong trường hợp này bảng phương án được xây dựng theo *nguyên tắc loang* với các giá trị n lần lượt bằng 0, 1, 2, . . . (loang từ cuối về điểm xuất phát của trò chơi).

Xét trò chơi giữa 2 người. Ban đầu có đồng gồm n đồng tiền xu. Mỗi người đến lượt mình đi, chọn một đồng và chia nó thành 2 đồng với số xu ở mỗi đồng là khác nhau (trong số 2 đồng này). Ai đến lượt mình không còn cách chia là thua.

Phân tích

Người ta đã khảo sát hàm Sprague – Grundy với n đạt tới 2^{35} . Trong khoảng nói trên giá trị của hàm được khảo sát đã vượt quá 10^{12} ! Trên cơ sở các giá trị đã khảo sát, người ta đã đưa ra giả thuyết là hàm có tính chất tuần hoàn. Tuy vậy, điều này vẫn chưa được chứng minh và bài toán trò chơi này vẫn thuộc loại chưa giải được trong trường hợp tổng quát.



Đây là một ví dụ về cách phân tích một số bài toán trò chơi. Ví dụ, ta có một ván cờ với 8x8 ô. Mục tiêu là đưa quân cờ từ ô A1 đến ô H8. Các quy tắc di chuyển là: quân cờ chỉ di chuyển theo đường thẳng hoặc theo đường cong, không được đi qua ô đã qua. Quân cờ không thể di chuyển qua ô H8. Sau đó, ta sẽ phân tích ván cờ.

Wikipedia.org

Now player 2 has to split the 4-heap into 3 + 1, and player 1 subsequently splits the 3-heap into 2 + 1:

player 2: $4+2+1+1 \rightarrow 3+1+2+1+1$
player 1: $3+1+2+1+1 \rightarrow 2+1+1+2+1+1$
player 2 has no moves left and loses

Mathematical theory

The game can be analysed using the Sprague–Grundy theory. This requires the heap sizes in the game to be mapped onto equivalent nim heap sizes. This Line Encyclopedia of Integer Sequences as A002188.

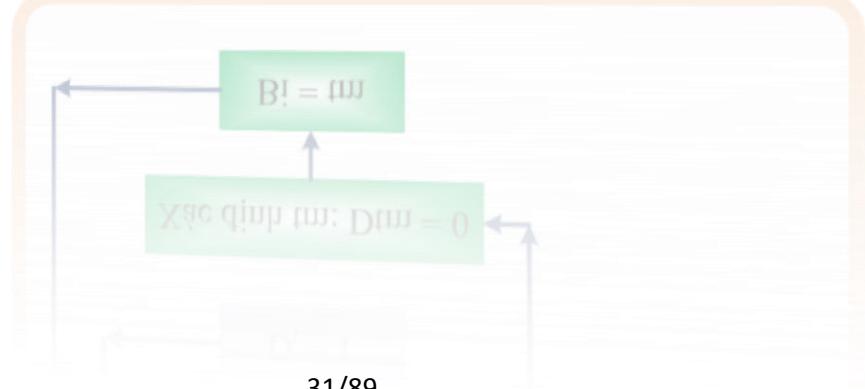
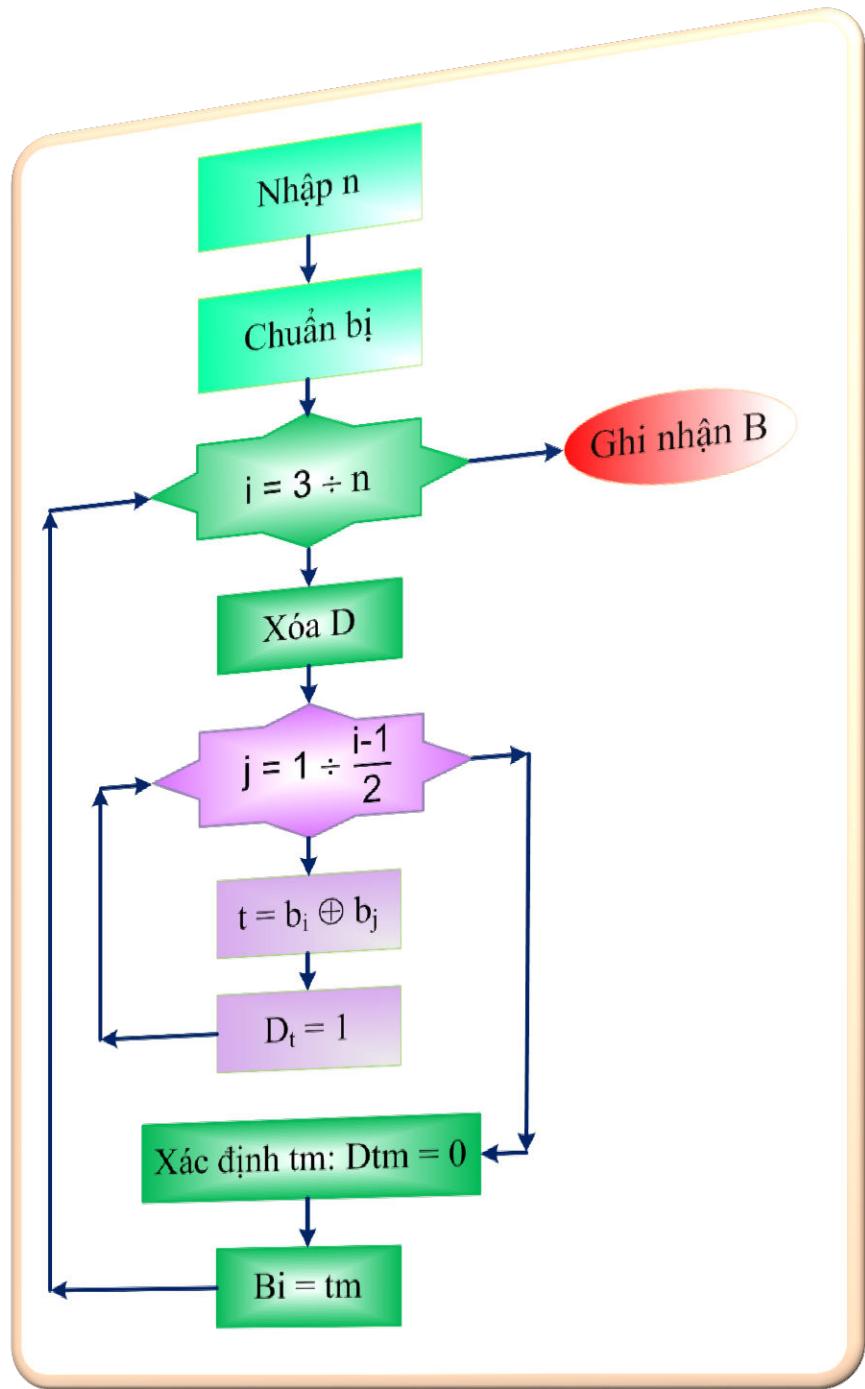
Heap size	: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...
Equivalent Nim heap	: 0 0 0 1 0 2 1 0 2 1 0 2 1 3 2 1 3 2 4 3 0 ...

Using this mapping, the strategy for playing the game Nim can also be used for Grundy's game. Whether the sequence of nim-values of Grundy's game ever becomes periodic is an unsolved problem. Elwyn Berlekamp, John Horton Conway, and Richard Guy have conjectured^[1] that the sequence does become periodic eventually, but did not prove it. Achim Flammenkamp, the question has not been resolved.

Bảng phương án

Giải thuật cho Trò chơi Grundy nếu trong Bách khoa toàn thư mở Wikipedia.org.

Xây dựng bảng phương án cho trò chơi Grundy



Chương trình PASCAL

```
Var
    a,b,c : array[1..1000] of integer;
    n,dem : integer;

Procedure Cal(p : integer);
    Var
        i,thu,g,g1,g2 : integer;
        kiemtra : array[0..1000] of integer;
    Begin
        dem := 0;
        Fillchar(kiemtra,sizeof(kiemtra),0);
        For i := 1 to ((p - 1) div 2) do
            Begin
                g1 := b[i];
                g2 := b[p - i];
                g := g1 XOR g2;
                inc(dem);
                c[dem] := g;
                kiemtra[g] := 1;
            end;
        For i := 0 to maxint do
            IF kiemtra[i] <> 1 then
                Begin
                    b[p] := i;
                    break;
                end;
    end;

Procedure Xuli;
    Var
        i : integer;
    Begin
        a[1] := 0;
        a[2] := 0;
        b[1] := 0;
        b[2] := 0;
        for i := 3 to n do
            cal(i);
    end;

Procedure Inkq;
    Var
        i : integer;
    Begin
        For i := 1 to n do
            Write(b[i],' ');
    end;

Begin
    Readln(n);
    Xuli;
    Inkq;
end.
```

Chương trình C++

```
#include<fstream>
#include<iostream>
using namespace std;
int n,b[1000],a[1000];
ifstream fi ("grundy.inp");
ofstream fo ("grundy.out");

void nhap()
    {fi >> n;
    }

void cal(int p)
    {int i,thu,dem,g,c[1000],danhdau[1000];
    dem = 0;
    for (i = 0; i <= 999 ; i++)
        danhdau[i] = 0;
    for (i = 1; i <= ((p - 1)/2) ; i++)
        {g = b[i]^b[p - i];
        dem++;
        c[dem] = g;
        danhdau[g] = 1;}
    for (i = 0 ; i<=999 ; i++)
        if (danhdau[i] != 1)
            {b[p] = i;
            break;
            }
    }

void xuli()
    {a[1] = 0;
    a[2] = 0;
    b[1] = 0;
    b[2] = 0;
    for (int i = 3; i <=n ; i++)
        cal(i);
    }

void inkq()
    {for (int i = 1; i <= n ; i++)
        fo << b[i] << " ";
    }

int main()
    {nhap();
    xuli();
    inkq();
    fi.close();
    fo.close();
    }
```

Có nhiều cách tổ chức bảng phương án. Thông thường có 2 loại chính:

- Bảng phương án phục vụ cho *điều khiển tiến trình*,
- Bảng phương án phục vụ *triển khai cài đặt* chương trình, phục vụ *nhận dạng*.

Bảng phương án ở hình trên thuộc loại I.

Với các bài toán trò chơi, bảng phương án loại II thường dùng để giải quyết yêu cầu: với trò chơi đã cho, người đi trước có thắng được hay không, nước đi đầu tiên cần thực hiện để thắng, số cách đi thắng khác nhau có thể thực hiện ở nước đầu tiên.

Bảng phương án cho Trò chơi Omaks:

Bảng phương án B cũng được xây dựng theo nguyên tắc tương tự.

Lưu ý là khi số điểm $n = 2 \times k$, người đi đầu luôn luôn có cách đi thắng bằng cách đưa

về tình huống $p = q = \frac{n-2}{2}$. Ta có 2 nhóm con giống nhau, tương ứng với 2 trò chơi độc lập như nhau.

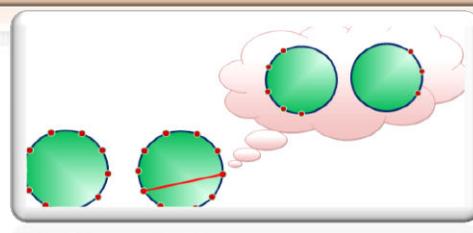
Việc điều khiển tiếp theo có thể thực hiện theo nguyên tắc đối xứng: đối phương đi như thế nào ta đi như thế đó ở bài toán con tương ứng. Số đồ thị con luôn chẵn và từng cặp giống nhau. Giá trị hàm Sprague – Grundy đương nhiên sẽ bằng 0!.

Như vậy chỉ cần chỉ cần khảo sát, xây dựng B với trường hợp $n = 2 \times k + 1$ ($B_n = 0 \forall n = 2 \times k$).

Để điều khiển trò chơi – cần có thêm các mảng hỗ trợ.

Trò chơi trên ma trận

Cho n điểm khác nhau trên đường tròn. Có 2 người chơi. Mỗi người khi đến lượt mình tạo một dây cung mới bằng cách nối 2 trong số các điểm đã cho sao cho dây cung mới không có điểm chung với bất kỳ dây cung nào đã có trước đó. Ai đến lượt mình đi không tạo được dây cung mới là thua.



Phần lớn các trò chơi này có hàm mục tiêu là giá trị nhận được của mỗi người chơi hoặc giá trị của hàm thời gian. Như đã nói ở trên, giải thuật giải những bài toán này thường liên quan tới Lý thuyết minimax.

Ví dụ: Bài toán **TRÒ CHƠI** (IOI 1996)

Xét trò chơi giữa 2 người. Trên bàn chơi có các số nguyên dương được viết thành một dãy. Hai người lần lượt đi. Khi đến lượt mình, người chơi chọn một số ở đầu bên trái hoặc cuối bên phải của dãy. Số được chọn sẽ bị xóa khỏi bảng. Trò chơi kết thúc khi tất cả các số đều được chọn. Người thứ nhất thắng nếu tổng các số đã chọn của mình không nhỏ hơn tổng các số đã chọn của người thứ 2. Người thứ 2 biết cách đi tốt nhất cho mình.

Người thứ nhất đi trước.

Nếu ban đầu trên bảng có một số lượng chẵn các số thì người thứ 1 luôn có chiến lược thắng. Bạn hãy viết chương trình hiện thực hóa chiến lược đi cho người chơi thứ 1. Chương trình của máy sẽ là người chơi thứ 2. Cả 2 người chơi đối thoại với nhau thông qua các thủ tục đã được cung cấp: Đó là các thủ tục StartGame, MyMove và YourMove. Người chơi thứ 1 phải khởi động trò chơi và lấy các tham số của trò chơi bằng cách gọi thủ tục StartGame. Nếu người chơi thứ nhất muốn chọn số ở cuối trái của dãy thì thực hiện lời gọi MyMove('L'). Tương tự như vậy, lời gọi MyMove('R') được gửi để thông báo việc chọn số ở đầu phải của dãy. Người chơi thứ 2, tức là máy, sẽ thực hiện nước đi của mình ngay lập tức và bạn có thể biết sự lựa chọn của máy bằng cách thực hiện lệnh gọi YourMove(C), trong đó C – biến kiểu ký tự (trong C/C++ cần thực hiện lời gọi YourMove(&C)). Giá trị của C là ‘L’ hoặc ‘R’ phụ thuộc vào việc máy chọn số ở bên trái hay bên phải của dãy.

Dữ liệu INPUT:

Dòng đầu tiên chứa số nguyên N (N – chẵn, $2 \leq N \leq 200$),

Mỗi dòng trong N dòng sau chứa một số nguyên dương, có giá trị không vượt quá 200, theo trình tự từ trái sang phải của dãy.

Dữ liệu OUTPUT:

Khi trò chơi kết thúc, bạn phải ghi ra file OUTPUT.TXT một dòng chứa 2 số nguyên, số thứ nhất là tổng các số được chọn cả người chơi thứ nhất, số thứ hai là

tổng các số được chọn cả người chơi thứ hai. Chương trình của bạn phải thực hiện quá trình chơi và đưa ra kết quả phù hợp với diễn biến chơi.

Ví dụ: một bộ dữ liệu vào và một kết quả có thể của trò chơi.

INPUT.TXT	OUTPUT.TXT
6	
4	
7	
2	
9	
5	
2	

Đây là loại bài toán tương tác, đối thoại giữa người và máy. Điều đáng tiếc, do quy cách tổ chức thi ở nước ta, các loại bài toán này không thể đưa vào các kỳ thi. Ở đây chúng ta chỉ xem xét ở khía cạnh giải thuật điều khiển trò chơi.

Ở trò chơi này, nói N chẵn, người thứ 2 luôn luôn thực hiện nước đi cuối cùng. Nhưng điều đó không quan trọng. Vấn đề ở tổng các số đã được chọn.

Giải thuật: Ta gọi chỉ số của mỗi số trong dãy là vị trí của nó. Sau khi người thứ nhất thực hiện nước đi thì trên bảng chỉ còn lại dãy số mà các số ở hai đầu hoặc cùng ở vị trí chẵn hoặc cùng ở vị trí lẻ. Như vậy người thứ nhất sẽ chọn tất cả các số ở vị trí chẵn hoặc tất cả các số ở vị trí lẻ phụ thuộc vào tổng của chúng.

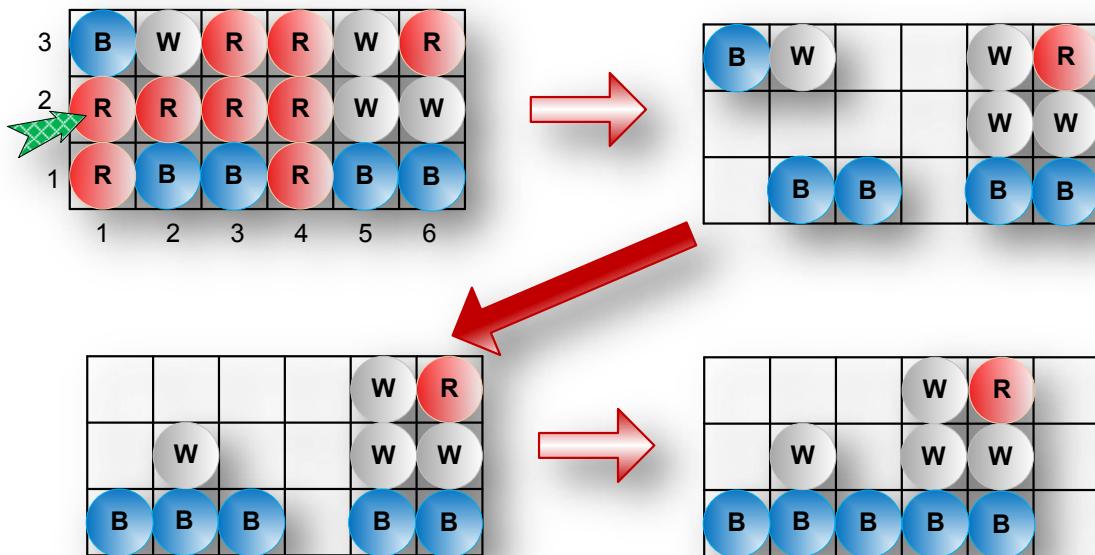
Bài toán trò chơi trên ma trận có thể là loại bài toán trò chơi một người. Tuy không có đối kháng nhưng cấu hình bảng sẽ thay đổi sau mỗi lựa chọn và tác động đến điểm số của người chơi.

Ví dụ: TRÒ CHƠI 3 MÀU

Cho bảng kích thước $m \times n$ ô ($2 \leq m, n \leq 20$). Các dòng của bảng được đánh số từ 1 đến m từ dưới lên trên, các cột được đánh số từ 1 đến n từ trái qua phải. Mỗi ô của bảng có một viên bi thuộc một trong 3 màu Xanh (B), Trắng (W) hoặc Đỏ (R). Các viên bi cùng màu ở các ô kề cạnh tạo thành một miền liên thông. Người chơi ở mỗi lượt đi, nhấp chuột vào ô tùy chọn bất kỳ của miền liên thông có diện tích không lớn hơn 2 ô. Nếu miền liên thông này có m viên bi thì điểm thưởng người chơi nhận được là $(m-2)^2$. Các viên bi trong miền liên thông bị kích hoạt sẽ biến mất. Nhữn

viên bị ở trên ô trống sẽ rơi xuống cho đến khi chạm vào viên bi ở dưới hay chạm vào cạnh đáy, các cột ở bên phải sẽ chuyển động tịnh sang trái, lấp vào các ô trống (nếu có). Trò chơi kết thúc không còn miền liên thông để chọn kích hoạt.

Hãy tìm cách đi để nhận được tổng điểm thưởng lớn nhất.



Dữ liệu: Vào từ file văn bản WBR.INP:

- Dòng đầu tiên chứa 2 số nguyên m và n,
- Mỗi dòng trong m dòng sau chứa xâu độ dài n từ tập các ký tự {W, B, R} mô tả bảng từ trên xuống dưới.

Kết quả: Đưa ra file văn bản WBR.OUT biên bản chơi mô tả các lần đi, thông tin về mỗi lần đi đưa ra trên một dòng theo trình tự xuất hiện và có quy cách:

$$C \ i \ j \ m \ k,$$

Trong đó:

- C – ký tự chỉ màu của các viên bi trong miền liên thông được chọn,
- i, j – tọa độ ô kích hoạt,
- m – diện tích miền liên thông,
- k – điểm thưởng,

Cuối cùng là dòng chứa 2 số nguyên S và r, trong đó S là tổng điểm thưởng đạt được, r – số viên bi còn lại trong bảng.

Ví dụ:

WBR.INP
10 15
RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWW
BBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBB
RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWW

WBR.OUT
W 1 1 30 784
R 1 1 30 784
B 1 1 30 784
W 1 1 30 784
R 1 1 30 784
4920 0

Lưu ý: Kết quả này hợp lệ, nhưng chưa phải là tối ưu.

Các loại bài toán này thông thường đòi hỏi kỹ thuật *tổ chức dữ liệu*, *tổ chức loang và duyệt* trên bảng.

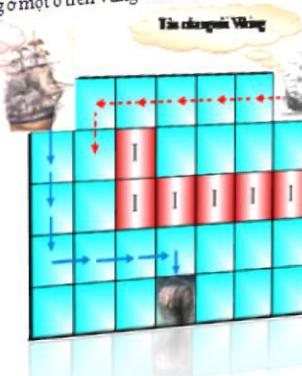
Ví dụ, giải thuật cho bài toán NGƯỜI VIKING dựa cở sở trên việc loang theo chiều rộng (BFS).

Bằng cách loang theo chiều rộng ta biết được sau bao nhiêu bước tàu Viking sẽ kiểm soát một ô nào đó của vùng biển. Cũng bằng kỹ thuật BFS có thể xác định được từ một ô của vùng biển ta cần đi theo chiều dọc hay ngang để đảm bảo an toàn. Giải thuật này có độ phức tạp $O(nm(n+m))$.

Nếu ta ghi nhận các đoạn ngang các ô biển (giới hạn bởi đảo hoặc biên của bản đồ), ghi nhận thời điểm đầu tiên tàu Viking kiểm soát

NGƯỜI VIKING
Bạn đang có trong tay hải đồ vùng biển hình chữ nhật kích thước $n \times m$ ô, một số ô là đảo, còn lại là biển. Trên hải đồ có đánh dấu một ô mà dưới mặt nước ở đó có một kho báu vô giá. Tàu của bạn đang ở một ô trên vùng biển này, ngoài ra còn có một tàu của cướp biển Viking cũng đang trên vùng biển này.

Mỗi bước, tàu của bạn chuyển động sang ô biển kề cạnh và tàu của người Viking di sang một ô kề cạnh tùy chọn. Sau một bước đó, nếu tàu của bạn ở cùng hàng hoặc cột với tàu của Viking và ở giữa 2 tàu không có ô đảo nào chắn, bạn sẽ bị bắt chém. Nếu bạn tới được ô chứa kho báu trước – kho báu sẽ là của bạn!



Hãy xác định xem có tồn tại hai trình để chiếm được kho báu hay không.

Đữ liệu: Vào từ file văn bản VIKINGS.INP:

- Đầu tiên chứa 2 số nguyên n và m ($1 \leq n, m \leq 700$).
- Mỗi dòng trong n dòng sau chứa xâu độ dài m từ tập ký tự {., I, Y, V, T}, ký tự “.” ứng với ô biển, “I” – ô đảo, “Y” – vị trí ban đầu của tàu bạn, “V” – vị trí tàu của Viking, “T” – nơi có kho báu. Các ký tự “Y”, “V” và “T” chỉ xuất hiện đúng một lần.

Kết quả: Đưa ra file văn bản VIKINGS.OUT thông báo “YES” hoặc “NO”.

Trên đây là bài toán tìm đường đi ngắn nhất trên một bảng 2D. Mình đã giải quyết bài toán này bằng cách sử dụng thuật toán Dijkstra. Tuy nhiên, bài toán này có một số đặc điểm riêng:

- Đầu tiên, chúng ta cần xác định các đỉnh là các ô biển (không có ký tự “.”).
- Thứ hai, chúng ta cần xác định các kề của một ô biển là ô biển lân cận (không có ô đảo “I”).
- Thứ ba, chúng ta cần xác định các ô biển kề của ô biển kề (ô biển lân cận nhất).
- Thứ tư, chúng ta cần xác định các ô biển kề của ô biển kề kề (ô biển lân cận thứ ba).

Để giải quyết bài toán này, ta cần áp dụng thuật toán Dijkstra như sau:

- Tạo một mảng $dist$ có kích thước $n \times m$ và khởi tạo tất cả các giá trị là ∞ . Giá trị $dist[i][j]$ sẽ lưu trữ khoảng cách từ ô biển (i, j) đến ô biển $(0, 0)$.
- Khởi tạo $dist[0][0] = 0$.
- Chạy vòng lặp Dijkstra cho đến khi không còn ô biển nào có giá trị $dist$ là ∞ .
- Kết quả cuối cùng là giá trị $dist[n-1][m-1]$.

được đoạn này, ghi nhận các đoạn dọc của biển và các thời điểm những đoạn này bị kiểm soát, việc loang để tìm hành trình lấy kho báu sẽ nhanh hơn và độ phức tạp sẽ là $O(mn)$.

Loang theo chiều rộng (BFS) là điều khá quen thuộc với mọi người trong lập trình. Phần lớn các thủ tục loang trong một bài toán đều tương tự nhau. Thủ tục loang cho tàu Viking ở Giả ngôn ngữ (để dễ dàng chuyển sang hiện thực hóa trên PASCAL hoặc C++) có dạng như sau:

```
Read input: water[i][j] ⇔ water at field (i,j)
Viking[i][j]: Num rounds before Viking can see (i,j). At first ∞
//BFS for Viking
Insert pos=V-position,distance=0 into bfsQueue
while bfsQueue is not empty:
    retrieve pos, distance from queue and remove afterwards
    updateVikingVertical(pos,distance)
    updateVikingHorizontal(pos,distance)
    add unvisited neighbours with distance=distance+1 to queue
procedure updateVikingVertical(pos=(i,j),distance):
for k = i downto 1:
    if not water[i][j]: break
    Viking[i][j] = min(Viking[i][j],distance)
for k = i to n:
    if not water[i][j]: break
    Viking[i][j] = min(Viking[i][j],distance)
```

Thư viện chuẩn của các hệ thống lập trình thường trang bị nhiều công cụ hỗ trợ tổ chức và các phép xử lý dữ liệu trên vec tơ, ma trận (Thư viện **Matrix** của FPC, thư viện **Vector** của C++). Việc tìm hiểu và khai thác các thư viện hỗ trợ lập trình là hết sức cần thiết. Để có được năng suất lập trình cao và xây dựng chương trình hiệu quả cần phải biết cách **sử dụng hợp lý** tất cả sức mạnh của các công cụ mà mình được trang bị.

Nhiều bài toán không được phát biểu tường minh như một trò chơi, nhưng dựa vào bản chất của vấn đề chúng được xếp vào lớp bài toán trò chơi (ví dụ như bài *Người Viking* nói trên).

BÀI TẬP

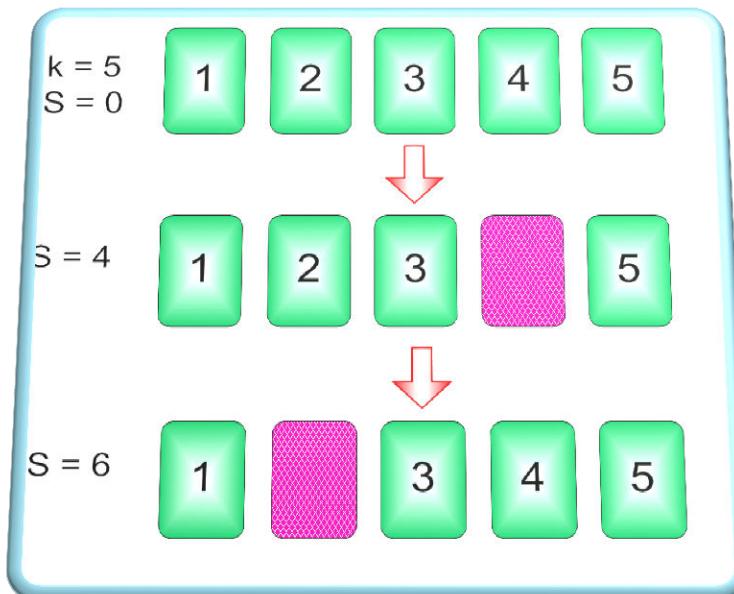
Bài 1. ỦP BÀI

Có k quân bài đánh số từ 1 đến k ($2 \leq k \leq 10$). Ban đầu mọi quân bài đều ở trạng thái ngửa. Có hai người chơi với tổng s ban đầu bằng 0. Hai người lần lượt đi. Người thứ nhất đi trước. Mỗi người khi đến lượt mình, úp một con bài tùy chọn đang ở trạng thái ngửa, cộng số ở quân bài này vào s và lật quân bài bị úp trước đó. Riêng nước đi đầu tiên của người thứ nhất không phải lật bài vì chưa có quân nào bị úp. Ai đến lượt mình đi làm cho tổng s vượt quá số n cho trước là người đó thua.

Yêu cầu: cho k và n . Hãy xác định người thứ nhất có bao nhiêu cách úp bài khác nhau để thắng và chỉ ra các quân bài có thể úp ở nước đi đầu tiên (theo thứ tự tăng dần của các quân bài có thể úp).

Dữ liệu: Vào từ file văn bản CARDS.INP:

- Dòng đầu tiên chứa số nguyên T – số lượng tests ($1 \leq T \leq 10^5$),
- Mỗi dòng trong T dòng sau chứa thông tin về 1 test bao gồm 2 số nguyên n và k ($0 < n \leq 10^6$).



Kết quả: Đưa ra file văn bản CARDS.OUT, kết quả mỗi test đưa ra trên một dòng gồm số nguyên m – số cách úp khác nhau. Nếu $m > 0$ thì tiếp theo là m số nguyên xác định các quân bài có thể úp.

Ví dụ:

CARDS.INP	CARDS.OUT
2	2 3 5
10 5	1 4
15 5	

Bài 2. XÓA SỐ

Cho lưới ô vuông $n \times n$ ($2 \leq n \leq 10$). Trên mỗi ô của lưới có điền một số nguyên trong phạm vi từ 1 đến n . Có 2 người chơi với tổng s ban đầu bằng 0. Mỗi người, khi đến lượt mình đi, xóa một số tùy chọn chưa được xóa trong bảng, cộng số bị xóa vào s . Ai đến lượt mình đi làm s lớn hơn m là người đó thua. Người thứ nhất đi trước.

Đã có k lần xóa kể từ đầu trò chơi. Biên bản trò chơi được ghi lại dưới dạng dãy số $\mathbf{A} = (a_1, a_2, \dots, a_k)$, $1 \leq a_i \leq n$, $i = 1 \div k$, trong đó a_i là số người thứ nhất xóa nếu i lẻ và là người thứ 2 xóa nếu là i chẵn. Việc xóa diễn ra hợp lệ, tức là không

có trường hợp xóa số không còn tồn tại. Trò chơi chưa kết thúc, tức là $\sum_{i=1}^k a_i \leq m$.

Một nước đi gọi là sai lầm nếu đang có cách đi thắng mà người đi chọn sai số để gạch và làm cho đối phương có khả năng thắng. Cả 2 người chơi đều có thể phạm sai lầm và sai lầm nhiều lần.

Hãy xác định số nước đi sai và chỉ ra số thứ tự của các nước đi đó (theo thứ tự tăng dần của số thứ tự).

Dữ liệu: Vào từ file văn bản DELNUM.INP:

- Dòng đầu tiên chứa 3 số nguyên n, m và k
- Mỗi dòng trong n dòng sau chứa n số nguyên của bảng,
- Dòng cuối cùng chứa k số nguyên a_1, a_2, \dots, a_k .

Kết quả: Đưa ra file văn bản DELNUM.OUT:

- Dòng đầu tiên chứa số nguyên p – số nước đi sai.
- Nếu $p > 0$ thì dòng thứ 2 chứa p số nguyên xác định số thứ tự của các nước đi sai.

Ví dụ:

DELNUM.INP
3 8 3
1 2 3
2 3 1
3 1 2
1 1 3

DELNUM OUT
2
2 3

Bài 3. DI CHUYỂN SỎI

Tên chương trình: MABLE.???

Jimmy và Rôn thích các trò chơi chuyển sỏi trên đồ thị và thường nghĩ ra các trò chơi mới về đề tài này. Một hôm, vào ngày nghỉ cuối tuần Jimmy gọi Rôn tới và giới thiệu một trò chơi mới. Đầu tiên, Jimmy giới thiệu về 2 quy tắc cơ bản: di chuyển sỏi và xóa cạnh.

Jimmy vẽ một đồ thị có hướng n đỉnh, từ mỗi đỉnh có không quá 1 cung đi ra tới đỉnh khác. Nếu ta đặt một viên sỏi vào đỉnh x thì nó sẽ di chuyển theo cung đi ra lần lượt tới các đỉnh khác và sẽ dừng lại ở đỉnh không có cung ra. Viên sỏi cũng có thể chuyển động vô hạn nếu nó rơi vào một chu trình.

Để xem Rôn đã nắm được quy tắc hay chưa Jimmy đưa ra một chuỗi q thao tác thuộc 2 loại:

- Loại I: $1 \ x$ – đặt viên sỏi ở đỉnh x , nó sẽ dừng lại ở đâu hay rơi vào chu trình?
- Loại II: $2 \ x$ – xóa cung đi ra từ đỉnh x .

Nếu áp dụng lần lượt các thao tác này trên đồ thị đã cho thì đáp án cho từng thao tác loại I sẽ là gì? Nếu viên sỏi rơi vào chu trình, nó sẽ chuyển động vô hạn và cần đưa ra câu trả lời là “**CIKLUS**”.

Dữ liệu: Vào từ file văn bản MABLE.INP:

- Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 3 \times 10^5$),
- Dòng thứ 2 chứa n số nguyên p_1, p_2, \dots, p_n , trong đó p_i cho biết có cung đi từ i tới p_i , nếu $p_i = 0$ – từ i không có cung ra,
- Dòng thứ 3 chứa số nguyên q ($1 \leq q \leq 3 \times 10^5$),
- Mỗi dòng trong q dòng sau chứa 2 số nguyên k và x , trong đó $k=1$ hoặc 2 – dạng thao tác. Với thao tác loại II, dữ liệu đảm bảo còn có cung để xóa.

Kết quả: Đưa ra file văn bản MABLE.OUT đáp án cho các thao tác loại I theo trình tự xuất hiện, mỗi đáp án trên một dòng.

Ví dụ:

MABLE.INP	MABLE.OUT
5	1
0 3 5 3 4	CIKLUS
6	4
1 1	3
1 2	
2 4	
1 2	
2 3	
1 2	

Bài 4. NGƯỜI MÁY HỦY DIỆT

Tên chương trình: KILLBOTS.???

Người máy hủy diệt là một trò chơi đơn giản. Ai đó đã tạo ra những rô bốt chuyên đi phá phách, tiêu diệt những gì chúng gặp trên đường đi. Cũng may là người tạo ra chúng thiên về số lượng hơn chất lượng, vì vậy độ thông minh của các rô bốt này không cao. Trò chơi diễn ra trên lưới ô vuông hình chữ nhật. Mỗi ô của lưới có thể là trống hoặc chứa một trong số các đối tượng sau:

Siêu nhân (@): là khắc tinh của rô bốt. Nếu siêu nhân và rô bốt gặp nhau ở một ô nào đó thì rô bốt sẽ bị tiêu diệt. Trong trò chơi nguyên bản, người chơi điều khiển siêu nhân di chuyển. Trong phạm vi bài này – siêu nhân đứng yên tại chỗ,

Rô bốt (+): Những con vật kim loại, chúng chưa bao giờ nghe đến “Ba luật của Rô bốt”, tiêu diệt người và vật chúng gặp trên đường đi. Rô bốt luôn đi tìm siêu nhân để tiêu diệt. Mỗi bước, rô bốt đi sang được ô kè cạnh hoặc đỉnh, *hướng về phía siêu nhân*. Nếu ở cùng hàng hay cột với siêu nhân chúng sẽ dọc theo hàng hoặc cột đó, trong trường hợp ngược lại – đi sang ô kè đỉnh.

Rô bốt nhanh (#): Hoạt động như rô bốt, nhưng mỗi bước đi được 2 ô, có thể đổi hướng khi gặp hàng hoặc cột có siêu nhân.

Đóng sắt vụn (*): Khi 2 hay nhiều rô bốt (thường hoặc nhanh) gặp nhau tại một ô nào đó, chúng sẽ tiêu diệt nhau và biến thành đóng sắt vụn. Các loại rô bốt cũng bị biến thành sắt vụn khi gặp ô chứa sắt vụn.

Cho bảng trạng thái các ô, những ô trống được đánh dấu bằng ký tự “.”. Hãy xác định số lượng rô bốt (thường và nhanh) bị Siêu nhân tiêu diệt.

Dữ liệu: Vào từ file văn bản KILLBOTS.INP gồm nhiều tests, mỗi test cho dưới dạng bảng ký tự khác rỗng, độ dài mỗi xâu không quá 1000. Cuối mỗi bảng là dòng chứa ký tự “\$”.

Kết quả: Đưa ra file văn bản KILLBOTS.OUT, kết quả mỗi test đưa ra trên một dòng dưới dạng số nguyên.

Ví dụ:

KILLBOTS.INP
...*....
.....#
....*..
....+...
.+..@..*
.....+
.*....+..
.+....#.
\$
...
..@
.*.
.+.
\$

KILLBOTS.OUT
4
1

Bài 5. TRÒ CHƠI 5×5

Tên chương trình: GAME25.???

Trò chơi đặt sỏi trên bảng 5×5 là hình thức giải trí giết thời gian phổ biến thứ 3 trên Ardenia (chỉ dùng sau trò giải toán đố và nói xáu sau lưng lảng giềng). Hai người lần lượt đặt sỏi của mình trên bàn cờ kích thước 5×5 ô vuông. Chúng ta không quan tâm đến luật đi, chỉ cần biết rằng mục tiêu của mỗi người chơi là khi kết thúc (bàn cờ đã đặt kín sỏi, mỗi ô một viên), mỗi quân của mình đều thuộc một dãy 3 quân của mình liên tiếp theo đường ngang, dọc hoặc chéo. Nếu cả hai người đều cùng đạt được mục đích hay cùng không đạt mục đích thì ván cờ coi như hòa. Nếu chỉ một người đạt mục đích thì người đó thắng.

Hai anh em Arthur và Bruce không thích trò chơi này lắm, nhưng mẹ cứ bắt phải chơi. Arthur đặt các viên đá có ký hiệu **A**, còn Bruce – các viên có ký hiệu **B**.

Đã 2 năm trôi qua bỗng nhiên hai người lại nhìn thấy ảnh chụp ván cờ họ đã chơi và muốn tính lại xem người đi viên sỏi **A** thắng hay người đi viên sỏi **B** thắng hoặc có thể đây là ván hòa.

Yêu cầu: Cho 5 xâu ký tự độ dài 5 từ tập **{A, B}** mô tả bàn cờ. Hãy đưa ra kết quả dưới dạng thông báo "**A wins**" hoặc "**B wins**" hoặc "**draw**".

Dữ liệu: Vào từ file văn bản GAME25.INP:

- Dòng đầu tiên chứa số nguyên **t** – số lượng tests ($1 \leq t \leq 10^5$),
- Mỗi test cho trên một nhóm 5 dòng mô tả bàn cờ, mỗi dòng một xâu độ dài 5.

Kết quả: Đưa ra file văn bản GAME25.OUT các xâu thông báo kết quả, mỗi xâu trên một dòng.



Ví dụ:

GAME25.INP	GAME25.OUT
2 AABBA BAAAB AAABA ABAAB BAAAB AAAAA AAAAA BAAAA ABAAA AABAA	A wins draw

Bài 6. CHIA BÁNH

Tên chương trình: GOODIES.???

Petra và Jan được tặng một hộp bánh, trong đó có n chiếc bánh khác nhau. Hai bạn quyết định chia nhau. Nhưng chia như thế nào cho công bằng không phải là chuyện dễ vì mỗi người có một sở thích riêng vì vậy cùng một chiếc bánh nhưng mỗi người có thể có một cách đánh giá riêng. Chiếc bánh thứ i theo cách nhìn của Petra có giá trị là p_i , còn theo Jan – giá trị là j_i .

Hai bạn quyết định lần lượt chọn, mỗi lần đến lượt mình sẽ lấy một trong số các chiếc bánh còn lại trong hộp. Ai chọn trước sẽ được quyết định bằng cách tung đồng xu.

Mỗi bạn có một chiến lược chọn riêng của mình. Khi đến lượt mình Petra luôn chọn chiếc bánh có giá trị lớn nhất (theo cách nhìn của Petra). Nếu có nhiều bánh có cùng giá trị lớn nhất Petra cẩn thận chọn chiếc bánh mà Jan ít thích nhất, tức là có giá trị nhỏ nhất đối với Jan (hai người rất thân nhau nên hiểu cách đánh giá của nhau).

Chiến lược của Jan là làm sao có tổng giá trị các bánh được chọn là lớn nhất. Khi có nhiều khả năng cùng cho giá trị lớn nhất thì Jan chọn chiếc bánh sao cho Petra cũng có tổng giá trị cuối cùng là lớn nhất có thể.

Yêu cầu: Cho n , kết quả tung xu ban đầu (“**Petra**” hoặc “**Jan**”), các giá trị p_i và j_i ($1 \leq n \leq 1000$, $0 \leq p_i, j_i \leq 1000$, $i = 1 \div n$). Hãy cho biết tổng giá trị bánh của mỗi người sau khi chia xong (theo cách đánh giá riêng tương ứng).

Dữ liệu: Vào từ file văn bản GOODIES.INP:

- Dòng đầu tiên chứa số nguyên t – số lượng tests ($1 \leq t \leq 100$),
- Mỗi test cho trên một nhóm dòng:
 - Dòng đầu tiên trong nhóm chứa số nguyên n ,
 - Dòng thứ 2 chứa tên người chọn trước,
 - Mỗi dòng trong n dòng tiếp theo chứa 2 số nguyên p_i và j_i .

Kết quả: Đưa ra file văn bản GOODIES.OUT, kết quả mỗi test đưa ra trên một dòng gồm 2 số nguyên – tổng giá trị bánh của Petra và tổng giá trị bánh của Jan.

Ví dụ:

GOODIES.INP
3
4
Petra
100 80
70 80
50 80
30 50
4
Petra
10 1
1 10
6 6
4 4
7
Jan
4 1
3 1
2 1
1 1
1 2
1 3
1 4

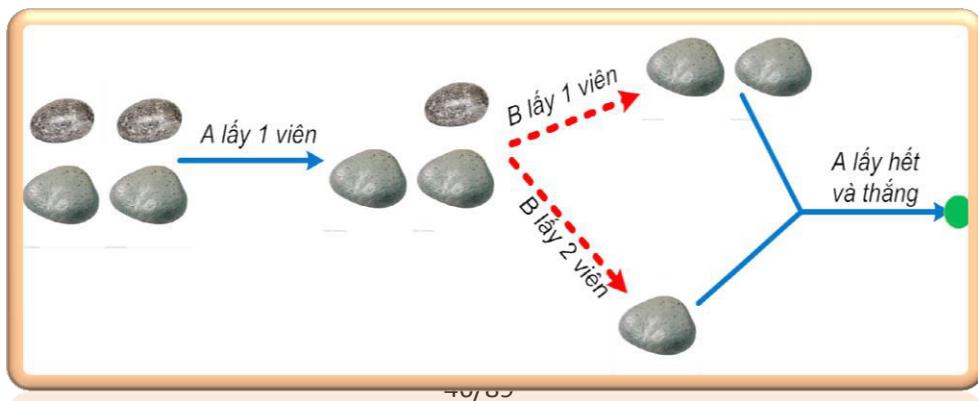
GOODIES.OUT
170 130
14 16
9 10

Bài 7. BÓC SỎI

Tên chương trình: PEBBLES.???

A và **B** chơi trò bóc sỏi trong thời gian ở trại hè. Ban đầu đống sỏi có n viên ($2 \leq n \leq 10^{15}$). **A** đi trước và ở nước đi đầu tiên này được quyền bóc một số lượng sỏi bất kỳ trong phạm vi từ 1 đến n . Sau đó **B** và **A** lần lượt đi. Mỗi người khi đến lượt mình được quyền bóc một số lượng sỏi lớn hơn hoặc bằng 1 và không vượt quá số sỏi mà người kia đã bóc ở nước đi trước. Dĩ nhiên, không ai có thể bóc quá số sỏi hiện có trong đống. Ai bóc được viên cuối cùng trong đống là người thắng. **A** luôn luôn có thể thắng vì ở ngay nước đi đầu tiên có quyền bóc cả n viên, nhưng như vậy không gì hấp dẫn. **A** muốn trò chơi kéo dài và mình vẫn thắng và vì vậy cần xác định số lượng sỏi tối thiểu phải bóc ở nước đi đầu tiên.

Ví dụ, với $n = 4$, **A** bóc 1 viên ở nước đi đầu tiên và có cách thắng dù **B** có bóc kiểu nào đi nữa ở các nước đi sau.



Yêu cầu: Cho n . Hãy xác định số lượng sỏi ít nhất mà A cần bốc ở nước đi đầu tiên để cuối cùng sẽ thắng.

Dữ liệu: Vào từ file văn bản PEBBLES.INP gồm một dòng chữ số nguyên n .

Kết quả: Đưa ra file văn bản PEBBLES.OUT một số nguyên – số lượng sỏi ít nhất mà A cần bốc ở nước đi đầu tiên.

Ví dụ:

PEBBLES.INP
4

PEBBLES.OUT
1

Bài 8. TRÒ CHƠI 10

Tên chương trình: GAME10.???

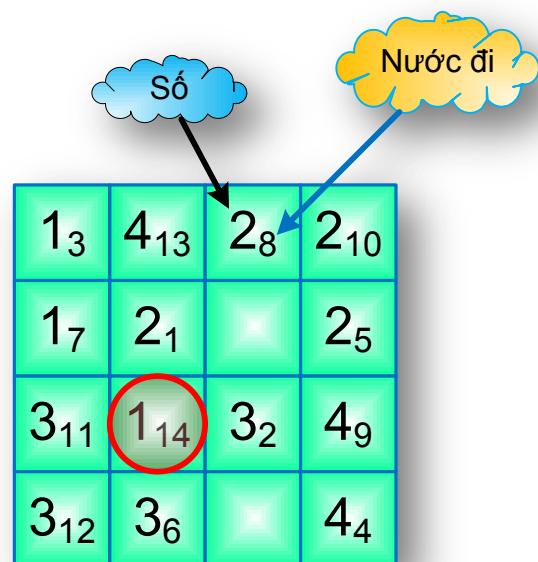
Xét trò chơi giữa 2 người: cho bảng ô vuông kích thước 4×4 . Ban đầu các ô đều trống. Hai người lần lượt đi. Ở mỗi nước đi người chơi chọn một ô trống và điền vào đó một số nguyên trong phạm vi từ 1 đến 4. Ai đến lượt mình làm cho một hàng hoặc cột được điền đủ 4 ô và có tổng các ô đó bằng 10 là thắng cuộc. Nếu tất cả các ô đều được điền số, nhưng không có cột hay hàng nào có tổng bằng mười là hòa.

Yêu cầu: Lập trình với vai trò người chơi thứ 2, đối thoại giữa người chơi thứ nhất với máy và đảm bảo chương trình luôn luôn thắng. Chương trình đọc nước đi của người thứ nhất từ file INPUT chuẩn và đưa ra file OUTPUT chuẩn nước đi của mình. Mỗi nước đi có quy cách

$r \ c \ k$

cho biết điền số k vào ô ở dòng r cột j .

Sau nước đi thắng có thêm thông báo “WIN” trên cùng dòng.



Ví dụ:

INPUT	OUTPUT
2 2 2	3 3 3
1 1 1	4 4 4
2 4 2	4 2 3
2 1 1	1 3 2
3 4 4	1 4 2
3 1 3	4 1 3
1 2 4	3 2 1 WIN

Bài 9. LIÊN MINH

Tên chương trình: ALLIANCE.???

Trò chơi online “Cuộc chiến giữa các hiệp sĩ” thu hút nhiều bạn trẻ tham gia. Trò chơi diễn ra trên lưới ô vuông kích thước $m \times m$ ô. Các dòng được đánh số từ trên xuống dưới bắt đầu từ 1, các cột – đánh số từ 1 từ trái sang phải. Mỗi người chơi là một hiệp sĩ và đứng ở một ô nào đó. Mỗi nước đi hiệp sĩ có thể thực hiện không quá k bước, mỗi bước hiệp sĩ có thể chuyển sang ô kề cạnh hoặc kè đỉnh. Khi hai người gặp nhau ở cùng một ô, họ sẽ chiến đấu cho đến khi một người bị thua và bật ra khỏi cuộc chơi.

Để trò chơi trở nên hấp dẫn hơn, có cơ chế thiết lập “Liên minh chiến lược” giữa 3 hiệp sĩ. Khi một người nào đó trong liên minh bị tấn công thì những người còn lại trong liên minh có trách nhiệm hợp sức chống trả. Tuy vậy, liên minh chỉ có thể xác lập với 3 hiệp sĩ mà bằng không quá một nước đi họ có thể tới các ô của người khác

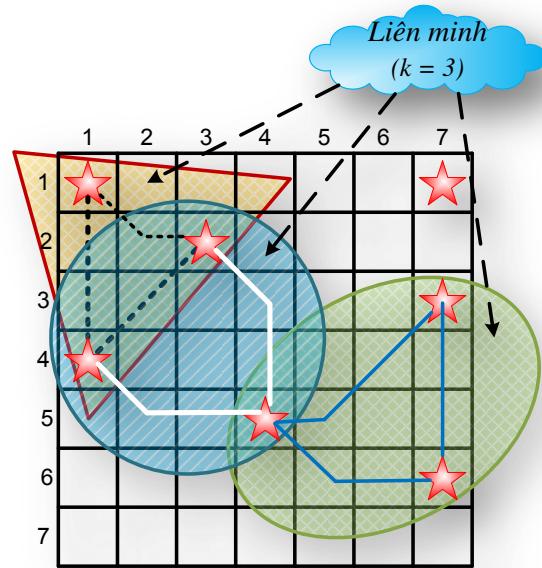
trong liên minh. Mỗi người có thể tham gia vào nhiều liên minh khác nhau. Hai liên minh gọi là khác nhau nếu tồn tại ít nhất một người thuộc liên minh này, nhưng không thuộc liên minh kia. Ở thời điểm hiện tại có n người đang chơi.

Yêu cầu: Cho n , m , k và tọa độ (x_i, y_i) của mỗi người chơi ($1 \leq n \leq 5000, 1 \leq m, k \leq 10^9, 1 \leq x_i, y_i \leq m, i = 1 \div n$). Hãy xác định số lượng liên minh khác nhau có thể xác lập.

Dữ liệu: Vào từ file văn bản ALLIANCE.INP:

- Dòng đầu tiên chứa 3 số nguyên n , m và k ,
- Dòng thứ i trong n dòng sau chứa 2 số nguyên x_i và y_i .

Kết quả: Đưa ra file văn bản ALLIANCE.OUT một số nguyên – số liên minh khác nhau có thể xác lập.



Ví dụ:

ALLIANCE.INP	ALLIANCE.OUT
7 7 3 1 1 4 1 6 7 5 4 1 7 2 3 3 7	3

Bài 10. BENDER

Tên chương trình: BENDER.???

Rô bốt Bender quyết định biểu diễn một trò chơi cá cược để cải thiện tình hình tài chính của mình. Nội dung của trò chơi là có một dãy k cốc giống nhau đặt úp thành một hàng ở các vị trí từ 1 đến k . Một viên bi được dấu dưới một trong số các cốc. Sau đó Bender tráo nhanh n lần vị trí của các cốc, mỗi lần tráo đổi vị trí một cặp cốc. Kết thúc quá trình tráo đổi vị trí, người chơi phải chỉ ra viên bi ở vị trí nào.

Bender là một rô bốt, vì vậy mọi thao tác đều thực hiện theo chương trình. Bender xây dựng dãy số nguyên \mathbf{x}_i theo quy tắc $\mathbf{x}_1 = \mathbf{c}$, $\mathbf{x}_i = \mathbf{x}_{i-1} \times \mathbf{a} + \mathbf{b}$, $i \geq 2$.

Ở bước thứ i Bender hoán đổi các cốc ở vị trí $\mathbf{x}_i \bmod k + 1$ và $(\mathbf{x}_i + 1) \bmod k + 1$.

Ban đầu viên bi ở vị trí \mathbf{p} . Bender muốn khi hoán đổi vị trí các cốc, bi phải ở vị trí \mathbf{q} .

Yêu cầu: Cho n , k , p và q ($1 \leq n \leq 10^5$, $2 \leq k \leq 10$, $1 \leq p, q \leq k$). Hãy xác định các giá trị a , b , c (giá trị cần tìm không vượt quá 1000). Nếu không tồn tại các giá trị cần tìm thì đưa ra thông báo “**Impossible**”.

Dữ liệu: Vào từ file văn bản BENDER.INP gồm một dòng chứa 4 số nguyên n , k , p và q .

Kết quả: Đưa ra file văn bản BENDER.OUT trên một dòng 3 số nguyên a , b và c hoặc thông báo “**Impossible**”.

Ví dụ:

BENDER.INP	BENDER.OUT
3 4 2 4	2 0 1

Bài 11. TRÒ CHƠI 11

Tên chương trình: GAME11.???

Do điều kiện thời tiết, việc tiếp tế tới đảo GreenIce bị gián đoạn, dầu mõi cho các rô bốt sắp hết. Mọi người quyết định tổ chức một cuộc thi đấu mà phần thưởng dành cho rô bốt thông minh và may mắn là một toa dầu mõi.

Cuộc chơi diễn ra theo quy tắc đấu loại trực tiếp giữa từng cặp rô bốt. Ban giám khảo đưa ra một số nguyên dương n đủ lớn, hai rô bốt lần lượt đi. Ở nước đi đầu tiên, người chơi thứ nhất phải viết một chữ số khác không, ở các bước tiếp theo, khi đến lượt mình người chơi viết một chữ số tùy chọn vào bên phải số đã viết. Ai đến lượt mình đi tạo được số lớn hơn hoặc bằng n là thắng cuộc.

Yêu cầu: Cho số nguyên n ($1 \leq n \leq 10^{100\,000}$). Hãy xác định rô bốt thắng cuộc và đưa ra thông báo “**First**” hoặc “**Second**”.

Dữ liệu: Vào từ file văn bản GAME11.INP gồm một dòng chứa số nguyên n .

Kết quả: Đưa ra file văn bản GAME11.OUT thông báo xác định rô bốt thắng cuộc.

Ví dụ:

GAME11.INP	GAME11.OUT
22	First



Bài 12. TRÒ CHƠI 12

Tên chương trình: GAME12.???

Rôn và Hermione chơi trò chơi xếp chữ. Từ một xâu nguồn độ dài n chỉ chứa các ký tự latin thường hai người lần lượt đi. Đến lượt ai người đó chọn một ký tự hiện có trong xâu ghi vào cuối xâu riêng của mình và xóa ký tự được chọn trong xâu nguồn. Trò chơi kết thúc khi xâu nguồn rỗng. Ban đầu xâu riêng của 2 người đều rỗng. Từ của ai có thứ tự từ điển nhỏ hơn là người ấy thắng. Nếu hai xâu riêng giống nhau thì coi là cả hai cùng thua. Rôn là người đi trước.

Rôn chơi tốt hơn và thường thắng, vì vậy muốn bạn mình không chán Rôn quyết định luôn luôn chỉ chọn ký tự cuối của xâu nguồn. Hermione nhanh chóng nhận ra chiến lược chơi của Rôn và cố gắng tìm cách giành chiến thắng.

Yêu cầu: Cho n ($2 \leq n \leq 10^5$) và xâu nguồn. Hãy xác định xem Hermione có thắng được hay không và đưa ra câu trả lời **YES** hoặc **NO** tương ứng. Trong mọi trường hợp – đưa ra từ riêng mà Hermione nhận được khi chơi theo chiến lược tối ưu.

Dữ liệu: Vào từ file văn bản GAME12.INP:

- Dòng đầu tiên chứa số nguyên n ,
- Dòng thứ 2 chứa xâu độ dài n chỉ chứa các ký tự la tinh thường.

Kết quả: Đưa ra file văn bản GAME12.OUT:

- Dòng đầu tiên chứa thông báo **YES** hoặc **NO**,
- Dòng thứ 2 chứa xâu riêng của Hermione.

Ví dụ:

GAME12.INP	GAME12.OUT
8 cokolada	YES acko

Bài 13. QUÂN TỐT

Tên chương trình: PAWNS.???

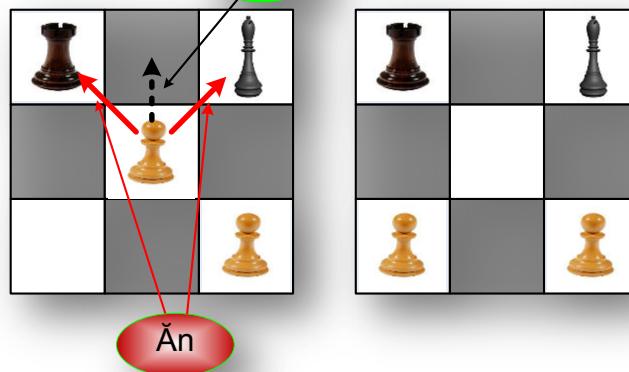
Jimmy học lớp một và mới được học bài đầu tiên về cách chơi cờ. Jimmy chỉ mới biết quân tốt: mỗi bước quân tốt được đi thẳng về phía trước theo cột sang ô kề cạnh nếu ô này trống, có thể ăn quân của đối phương ở ô theo đường chéo trên trái hoặc trên phải. Jimmy cũng chưa biết luật con tốt có thể biến thành quân hậu.

Jimmy tự nghĩ cho mình cách chơi cờ. Bàn cờ có kích thước $n \times m$ ô, các hàng đánh số từ 1 đến n từ dưới lên trên, các cột đánh số từ 1 đến m từ trái qua phải. Ở dòng 1 Jimmy đặt p quân tốt trắng, quân thứ i ở cột v_i . Trên bàn cờ không còn quân trắng nào khác. Ngoài ra còn có k quân đen mà Jimmy không biết tên, quân thứ j ở dòng x_j và cột c_j . Chỉ có quân trắng được đi và mục tiêu là ăn hết quân đen. Khi ăn, quân tốt sẽ chiếm vị trí quân đen, quân bị ăn sẽ bỏ ra ngoài bàn cờ. Nếu quân đen bị ăn hết thì ván cờ được coi là thắng.

Yêu cầu: Cho n, m, p, k và vị trí các quân trên bàn cờ ($1 \leq n, m \leq 100, 0 \leq p \leq m, 1 \leq k \leq 1000, k \leq (m-1) \times n$). Hãy xác định xem Jimmy có thể thắng được hay không và đưa ra thông báo “**YES**” hoặc “**NO**”.

Dữ liệu: Vào từ file văn bản PAWNS.INP:

- Dòng đầu tiên chứa 4 số nguyên n, m, p và k ,
- Dòng thứ 2 chứa p số nguyên v_1, v_2, \dots, v_p ,
- Dòng thứ j trong k dòng sau chứa 2 số nguyên x_j và c_j .



Kết quả: Đưa ra file văn bản PAWNS.OUT thông báo “**YES**” hoặc “**NO**”.

Ví dụ:

PAWNS.INP
3 3 2 2
1 3
3 1
3 3

PAWNS.OUT
NO

Bài 14. THI BẮN CUNG

Tên chương trình: SHOOTING.???

Steve tìm thấy trong đống giấy tờ cũ một biên bản ghi kết quả cuộc thi bắn cung trong có bô mình tham gia. Đáng tiếc là tờ giấy đã quá cũ, bị hoen ô nhiều chỗ, phần tên người dự thi không thể đọc được, chỉ còn rõ phần điểm số, ghi theo trình tự vào bắn của mỗi người. Khi Steve hỏi bô về cuộc thi này ông cũng đã quên nhiều thứ, chỉ nhớ rằng số điểm mình đạt có chữ số tận cùng là 5, một trong số những người thắng cuộc vào bắn trước, còn người bạn thân thì vào bắn ngay sau ông và đạt kết quả không tốt bằng ông.

Steve cầm tờ giấy và suy nghĩ xem bô mình có thể chiếm vị trí cao nhất là bao nhiêu.

Một người được coi là đứng thứ k nếu có đúng $k-1$ người có điểm cao hơn. Thắng cuộc là những người có điểm số cao nhất.



Yêu cầu: Cho n ($3 \leq n \leq 10^5$) và điểm số của từng người theo trình tự bắn. Điểm số là các số nguyên dương và có giá trị không vượt quá 1000. Hãy xác định vị trí cao nhất có thể của bô Steve. Nếu dữ liệu mâu thuẫn thì đưa ra số 0.

Dữ liệu: Vào từ file văn bản SHOOTING.INP:

- Dòng đầu tiên chứa số nguyên n ,
- Dòng thứ 2 chứa n số nguyên – điểm của các người dự thi theo trình tự bắn.

Kết quả: Đưa ra file văn bản SHOOTING.OUT một số nguyên – vị trí cao nhất có thể hoặc số 0.

Ví dụ:

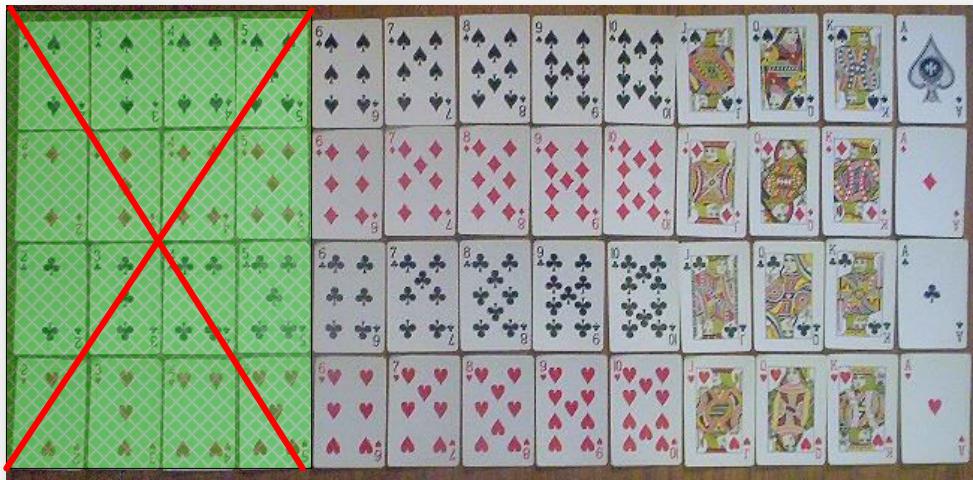
SHOOTING.INP
7
10 20 15 10 30 5 1

SHOOTING.OUT
6

Bài 15. CHƠI BÀI

Tên chương trình: CARDS.???

Bốn người chơi bài với cỗ bài tú lơ khơ chỉ có các quân từ 6 đến Át, mỗi quân có 4 chất. Như vậy bộ bài có 36 quân. Mỗi người được chia 6 quân, một quân trong phần còn lại của cỗ bài được lật lên xác định chất chủ, phần còn lại của cỗ bài được đặt úp.



Mỗi người phải nói một số từ 6 đến 14 cho biết quân chủ nhỏ nhất mình đang cầm trên tay, số lớn hơn 10 tương ứng với các bài J,Q, K hoặc A. Nếu trong tay không có chủ thì nói số 0. Ai có quân chủ nhỏ nhất sẽ được đi trước.

Nhưng người chơi cũng có thể nói dối!

Yêu cầu: Cho biết số do mỗi người chơi nói. Hãy xác định số lượng người chắc chắn nói dối.

Dữ liệu: Vào từ file văn bản CARDS.INP gồm một dòng chứa 4 số nguyên t6 đến 14 hoặc 0.

Kết quả: Đưa ra file văn bản CARDS.OUT một số nguyên – số lượng người chắc chắn nói dối.

Ví dụ:

CARDS.INP
6 10 10 14

CARDS.OUT
1

Bài 16. RUBIC 2 CHIỀU

Tên chương trình: RUBIC.???

Cho lưới ô vuông kích thước 4×4 , mỗi ô trên lưới chứa một số nguyên khác nhau trong phạm vi từ 1 đến 16.

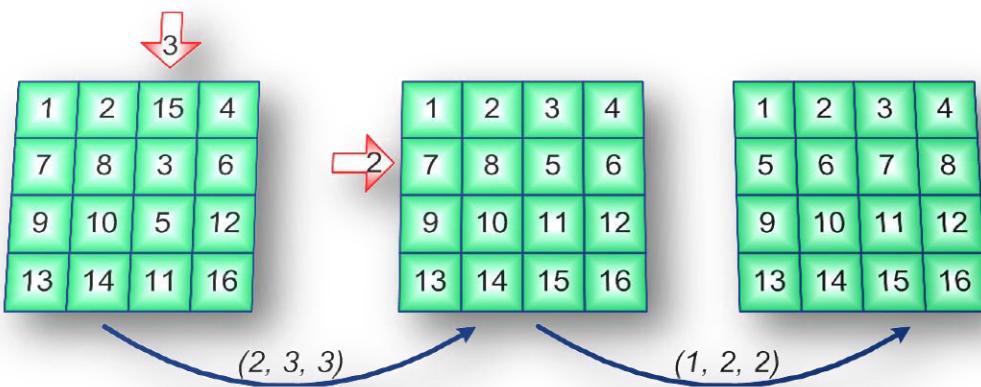
Tồn tại 2 phép biến đổi áp dụng đối với lưới:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

- Phép **1 $i \ k$** – đẩy vòng tròn các ô ở dòng i từ trái sang phải k vị trí ($1 \leq i \leq 4, 1 \leq k \leq 3$),
- Phép **2 $j \ k$** – đẩy vòng tròn các ô ở cột j từ trên xuống dưới k vị trí ($1 \leq j \leq 4, 1 \leq k \leq 3$).

Yêu cầu: Hãy xác định số phép biến đổi ít nhất và chỉ ra các phép biến đổi đó để đưa lưới về trạng thái ở hình bên.

Ví dụ:



Dữ liệu: Vào từ file văn bản RUBIC.INP gồm 4 dòng, mỗi dòng chứa 4 số nguyên xác định trạng thái ban đầu của lưới.

Kết quả: Đưa ra file văn bản RUBIC.OUT:

- Dòng đầu tiên chứa số nguyên n – số phép biến đổi,
- Mỗi dòng trong n dòng tiếp theo chứa 3 số nguyên xác định một phép biến đổi.

Ví dụ:

RUBIC.INP
1 2 15 4
7 8 3 6
9 10 5 12
13 14 11 16

RUBIC.OUT
2
2 3 3
1 2 2

Bài 17. TRÒ CHƠI NGỌT NGÀO

Tên chương trình: SWEET.???

Hermione ngưỡng mộ các số nguyên tố còn Rôn thì sùng báy các số palindrome. Một hôm hai bạn phải chia nhau n chiếc kẹo ($1 \leq n \leq 10^6$). Chia đều kẹo cho nhau là một chuyện quá nhảm chán, hai bạn quyết định chia theo cách sau: đánh số các viên kẹo từ 1 đến n . Mỗi người khi đến lượt mình đi sẽ lấy một viên kẹo. Hermione lấy viên kẹo có số nguyên tố, còn Rôn lấy viên kẹo có số là palindrome. Ai đến lượt mình mà không lấy được kẹo thì người kia sẽ lấy tất cả chỗ còn lại. Hermione sẽ là người lấy kẹo trước.

Yêu cầu: Cho số nguyên n . Hãy xác định số kẹo Hermione lấy được nếu cả hai bạn đều biết cách đi tối ưu.

Dữ liệu: Vào từ file văn bản SWEET.INP gồm một dòng chứa số nguyên n .

Kết quả: Đưa ra file văn bản SWEET.OUT một số nguyên – số kẹo Hermione lấy được.

Ví dụ:

SWEET. INP	SWEET. OUT
20	6

Bài 18. CHIA HẾT 3

Tên chương trình: GAME_3.???

Thu xếp lại đồ đạc cũ trên gác xép, Steve tình cờ tìm thấy một cỗ bài n quân, trên mỗi quân bài có một số nguyên dương và biên bản ghi lại diễn biến một số ván chơi. Nghiên cứu biên bản, Steve xác định được luật của trò chơi này là hai người lần lượt chọn và lấy ra một quân bài cho đến khi hết bài. Kết quả là hòa nếu tổng các số ở những quân bài lấy được mỗi người đều chia hết cho 3 hoặc đều không chia hết 3. Trong trường hợp ngược lại, ai có tổng chia hết cho 3 là người đó thắng.

Có lẽ đây là một trò chơi phổ biến ở một thời nào đó vì Steve nhớ ra là đã không dưới một lần được nghe bạn bè kể về trò chơi này với những cỗ bài khác nhau. Điều này làm Steve băn khoăn, với một cỗ bài cho trước và cả hai người chơi đều biết chiến lược tối ưu thì kết quả ra sao, hòa hay ai đó thắng?

Yêu cầu: Cho n ($1 \leq n \leq 50$) và danh sách n số nguyên dương, mỗi số không vượt quá 1000. Hãy xác định kết quả của trò chơi và đưa ra thông báo **FIRST** nếu người thứ nhất thắng, **SECOND** nếu người thứ 2 thắng, **DRAW** trong trường hợp hòa.

Dữ liệu: Vào từ file văn bản GAME_3.INP:

- Dòng đầu tiên chứa số nguyên n ,
- Dòng thứ 2 chứa n số nguyên.

Kết quả: Đưa ra file văn bản GAME_3.OUT thông báo xác định được.

Ví dụ:

GAME_3.INP	GAME_3.OUT
2	FIRST
1 3	

Bài 19. ZUMA 2.0

Tên chương trình: ZUMA.???

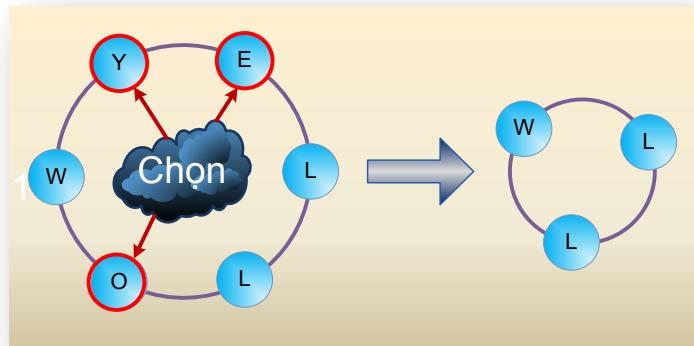
Trò chơi ZUMA đã hớp hồn của Vi chi a. Làm xong bài vở là Vi chi a lại lao vào bắn bi, hủy các đoạn bi cùng màu trong dây bi chạy trên màn hình, hết toán lên khi qua bài và vò đầu bứt tai lúc bị thua. Nhưng bài vở ở năm cuối của bậc phổ thông quá nhiều, Vi chi a phải dồn hết tâm trí vào thi tốt nghiệp phổ thông, rồi thi vào đại học. Trò chơi này bị roi vào quên lãng và sau đó bị xóa mất trong những lần cập nhật, nâng cấp hệ thống.

Thời gian trôi qua, Vi chi a tốt nghiệp đại học với bằng xuất sắc, bảo vệ thành công luận án tiến sĩ và bây giờ đã là chủ nhiệm Bộ môn Công nghệ phần mềm.

Tai họa xảy ra ở những lúc ít ai ngờ tới nhất. Trò chơi Zuma cải tiến ra đời với những luật chơi đa dạng hơn và khó hơn. Ở mỗi mức có một luật riêng. Ai đó đã cài vào máy của Bộ môn và nó, một lần nữa, lại đưa Vi chi a, ôi, xin lỗi, bây giờ đã là Victor Xec gây ê vic, trở lại những năm tháng sôi nổi của thời niên thiếu. Tuy vậy, tinh thần trách nhiệm vẫn chiến thắng. Victor Xecgâyevic đã bị loại ở mức áp chót và dừng lại ở đó. Victor Xecgaayevic quyết định lập trình để tự động hóa quá trình chơi.

Ở mức áp chót này có n viên bi xếp thành một vòng tròn. Trên mỗi viên bi có một chữ cái latin in hoa. Người chơi lựa chọn 5 viên liên tiếp bất kỳ. Với chuỗi 5 viên bi này có thể thực hiện một trong 2 thao tác sau (nếu đủ liệu cho phép):

- Hủy 2 viên bi tùy chọn có các ký tự giống nhau,
- Hủy 3 viên bi tùy chọn trên đó ký tự là nguyên âm (tức là thuộc tập $\{A, E, I, O, U, Y\}$).



Sau khi bị bót bi, các viên khác sẽ nhóm lại thành vòng tròn mới, bé hơn. Người chơi qua mức nếu số viên bi còn lại ít hơn 5 và còn càng ít bao nhiêu thì điểm số càng cao bấy nhiêu. Nếu còn từ 5 viên bi trở lên và không có cách hủy bi thì người chơi bị thua.

Yêu cầu: Cho trạng thái vòng tròn dưới xâu s độ dài trong phạm vi từ 5 đến 20, chứa các ký tự latin in hoa. Hãy xác định số ký tự còn lại nếu người chơi đi tối ưu.

Dữ liệu: Vào từ file văn bản ZUMA.INP gồm một dòng chứa xâu s .

Kết quả: Đưa ra file văn bản ZUMA.OUT một số nguyên – số ký tự còn lại.

Ví dụ:

ZUMA.INP
YELLOW

ZUMA.OUT
3

Ghi chú:

- Vi Chi a tên thân mật, Victor Xecgâyêvic – cách gọi kính trọng;
- Trên thực tế trong tiếng Anh Y vừa là nguyên âm vừa là bán nguyên âm.

Bài 20. BẮT CÁ

Tên chương trình: FISH.???

Bắt cá là trò chơi phổ biến trên máy tính và điện thoại di động. Ở đây chúng ta xét phương án đơn giản của trò chơi. Có n tảng băng lập phương xếp thành một hàng đánh số từ 1 đến n từ trái sang phải. Trên tảng băng thứ i có a_i con cá ($1 \leq a_i \leq 10^6$, $i = 1 \dots n$, $4 \leq n \leq 10$). Có 2 người chơi, mỗi người có 2 con chim cánh cụt. Người thứ nhất chọn một tảng băng và đặt con chim cánh cụt thứ nhất của mình lên đó. Người thứ 2 chọn tảng băng chưa có chim và đặt chim cánh cụt thứ nhất của mình lên đó. Tương tự như vậy người thứ nhất và sau đó là người thứ 2 đặt nốt chim cánh cụt thứ 2 của mình lên tảng băng. Sau đó cuộc thu gom cá bắt đầu. Người thứ nhất đi trước, sau đó đến người thứ 2. Đến lượt mình đi, người chơi gom hết cá nơi chim cánh cụt của mình đứng rồi nhảy nhảy sang tảng băng khác tùy chọn ở bên phải hay bên trái với bước bất kỳ. Tảng băng chim vừa nhảy đi sẽ bị đẩy trôi ra khỏi hàng tạo thành khoảng trống. Chim không được phép nhảy qua khoảng trống, không được nhảy qua tảng băng có chim. Người thứ nhất cho chim nhảy tới khi không còn cách đi thì thu các chim cánh cụt của mình về (cùng với cá trên đó). Các tảng băng nơi chim bị thu cũng sẽ trôi ra ngoài. Tương tự như vậy người thứ 2 thu gom hết cá của mình chừng nào còn đi được. Cả hai người đều biết đi một cách tối ưu.

Yêu cầu: Gọi số cá người thứ nhất gom được là f_1 , số cá người thứ 2 gom được là f_2 . Hãy tính $\max\{f_1-f_2\}$.

Dữ liệu: Vào từ file văn bản FISH.INP:

- Dòng đầu tiên chứa số nguyên n ,
- Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n .

Kết quả: Đưa ra file văn bản FISH.OUT một số nguyên – kết quả tìm được.

Ví dụ:

FISH.INP
7
1 3 2 2 3 2 1

FISH.OUT
0

Bài 20. TRÒ CHƠI 21

Tên chương trình: GAME21.???

Cho một sơ đồ trên đó có n thành phố đánh số từ 1 đến n . Có m đường, mỗi đường nối 2 thành phố khác nhau ($1 \leq n, m \leq 10^5$). Từ một thành phố ta có thể tới được thành phố khác bằng mạng đường này. Một trong các thành phố là thủ đô. Các đường có độ dài khác nhau. Nếu đường có độ dài c thì trên đường đó có $c-1$ làng ở các vị trí cách đều nhau ($1 \leq c \leq 10^9$). Các làng đầu tiên và cuối cùng trên đường cách thành phố gần nhất là 1.

Có 2 người chơi trò lái ô tô về thủ đô. Người thứ nhất xuất phát từ thành phố $s1$, người thứ 2 – từ thành phố $s2$. Mỗi người, đến lượt mình đi – chuyển tới thành phố hoặc làng kè cận trên đường đi qua chỗ mình đang đứng hoặc có quyền dừng. Nếu ai đó đang ở một làng thì người kia không được vào làng nào trong số các làng của đường nối 2 thành phố của làng đó.

Yêu cầu: Hãy xác ai là người tới thủ đô trước và đưa ra thông báo **First** hoặc **Second**.

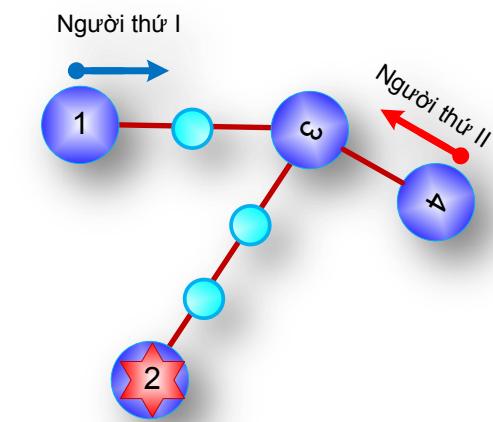
Dữ liệu: Vào từ file văn bản GAME21.INP:

- Dòng đầu tiên chứa 2 số nguyên n và m ,
- Mỗi dòng trong m dòng sau chứa 3 số nguyên a , b và c , cho biết đường nối 2 thành phố a và b có độ dài c ,
- Dòng cuối cùng chứa 3 số nguyên $s1$, $s2$ và t , trong đó t là thủ đô.

Kết quả: Đưa ra file văn bản GAME21.OUT thông báo **First** hoặc **Second**.

Ví dụ:

GAME21.INP
4 3
1 3 2
3 2 3
4 3 1
1 4 2



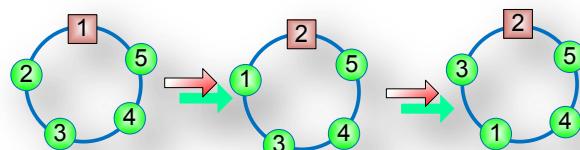
GAME21.OUT
Second

Bài 22. NGƯỜI BÊN CẠNH

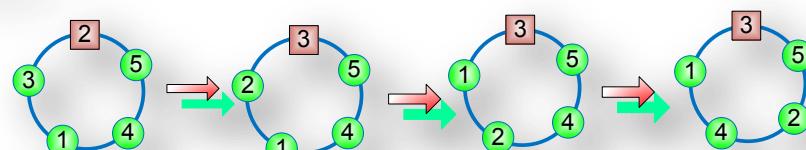
Tên chương trình: NEIGHBOR.???

Trong giờ nghỉ giải lao của một cuộc hội thảo toán học Ban tổ chức đề nghị mọi người giải trí bằng trò chơi “*Xác định láng giềng*”. Hãy tưởng tượng ta có một vòng tròn lớn, trên đó vẽ một hình vuông và $n-1$ vòng tròn nhỏ. Các vòng tròn và hình vuông nằm cách đều nhau. Người thứ nhất đứng vào hình vuông, $n-1$ người còn lại đứng vào các vòng tròn con, quay mặt vào tâm đường tròn lớn. Mọi người được đánh số từ 1 đến n bắt đầu từ người đứng ở hình vuông và theo chiều ngược kim đồng hồ. Trò chơi diễn ra k vòng. Ở vòng thứ i , người ta xác định số nguyên tố thứ i (gọi là p_i), sau đó người đang đứng ở ô vuông lần lượt đổi chỗ với người bên phải mình p_i lần. Ví dụ ở hình dưới tương ứng với $n = 5$ và $k = 3$.

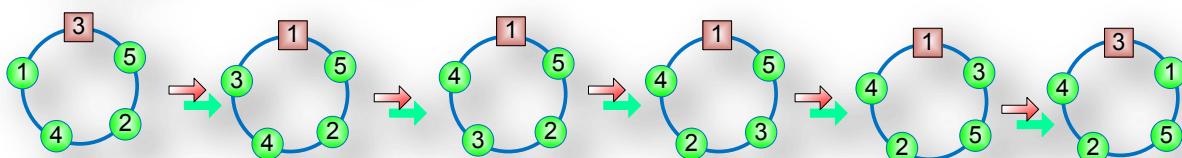
Vòng I



Vòng II



Vòng III



Yêu cầu: Cho n , k và a ($3 \leq n \leq 5 \times 10^6$, $1 \leq k \leq 5 \times 10^5$, $1 \leq a \leq n$). Hãy xác định 2 người đứng cạnh người a sau khi trò chơi kết thúc.

Dữ liệu: Vào từ file văn bản NEIGHBOR.INP gồm một dòng chứa 3 số nguyên n , k và a .

Kết quả: Đưa ra file văn bản NEIGHBOR.OUT 2 số nguyên – người bên phải và người bên trái của a .

Ví dụ:

NEIGHBOR.INP
5 3 1

NEIGHBOR.OUT
3 5

Bài 23. BẢNG SỐ

Tên chương trình: TABLE.???

Trò chơi bảng số sử dụng bảng n dòng và m cột ($2 \leq n, m \leq 100$) chứa $n \times m$ số nguyên có giá trị tuyệt đối không quá 100. Ở mỗi bước người chơi chọn một hàng hay cột và đổi dấu tất cả các số trong hàng hay cột được chọn.

Mục tiêu của trò chơi là đưa bảng về dạng có tổng các số ở mỗi hàng và ở mỗi cột đều không âm hoặc xác định được là không thể đưa về dạng đó.

Yêu cầu: Cho n, m và các số trong bảng. Hãy xác định số lượng phép biến đổi và chỉ ra các phép biến đổi dưới dạng $\mathbf{x} \ i$, trong đó \mathbf{x} là ký tự C nếu chọn cột i và là R nếu chọn dòng i . Không cần cực tiểu hóa số phép biến đổi, nhưng số lượng phép biến đổi không được vượt quá 20 000. Nếu bảng là không thể biến đổi được thì đưa ra số -1.

Dữ liệu: Vào từ file văn bản TABLE.INP:

- Dòng đầu tiên đưa ra 2 số nguyên n và m ,
- Dòng thứ i trong n dòng sau chứa m số nguyên xác định dòng thứ i của bảng.

Kết quả: Đưa ra file văn bản TABLE.OUT:

- Dòng đầu tiên đưa ra số nguyên k – số phép biến đổi,
- Nếu $k \neq -1$ thì mỗi dòng trong k dòng sau ghi một phép biến đổi.

Ví dụ:

TABLE.INP	TABLE.OUT
3 2 -1 -1 -1 -1 -1 -1	2 C 1 C 2

Bài 24. LỌC SỐ

Tên chương trình: FILTR.???

Trò chơi truyền hình “Lọc số” thu hút được rất nhiều người tham gia vì họ có thể ngồi ở nhà, dùng máy tính của mình hỗ trợ để chơi. Người điều khiển chương trình chọn và mở một phong bì trên bàn, thông báo hai số nguyên L và R ghi trong phong bì ($2 \leq L \leq R \leq 10^9$). Bảng chứa các số nguyên từ L đến R với các ô màu xanh.

Bước 1 người dẫn chương trình đưa ra số 2. Người chơi phải gạch hết các số trong bảng chia hết cho 2. Các ô chứa những số này sẽ trở thành màu đỏ. Những ai trả lời đúng và nhanh nhất sẽ được chơi tiếp. Ở bước thứ i người dẫn chương trình đưa ra số $i+1$. Người chơi phải gạch tất cả các số ở những ô xanh còn lại và chia hết cho $i+1$. Cũng có thể ở một số bước nào đó không có

số nào bị gạch. Trò chơi kết thúc khi không có người chơi nào trả lời đúng hoặc khi tất cả các số đều bị gạch.

Điều chúng ta quan tâm là với L, R nhận được trò chơi sẽ diễn ra tối đa là bao nhiêu bước. Ví dụ, với $L = 2, R = 10$, trò chơi sẽ diễn ra tối đa trong 6 bước.

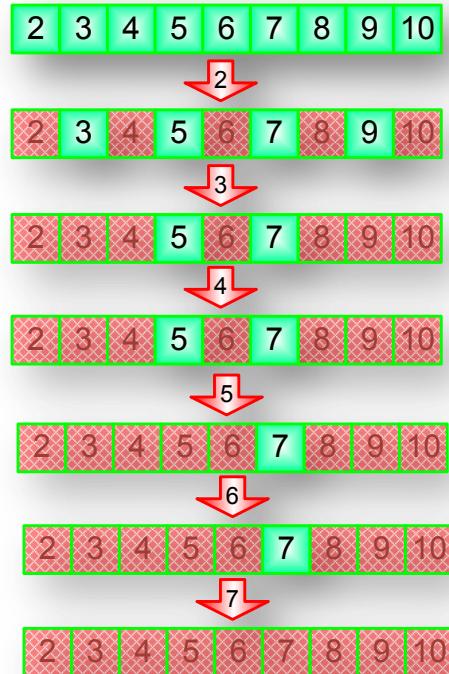
Yêu cầu: Cho 2 số nguyên L, R . Hãy xác định số bước tối đa của trò chơi.

Dữ liệu: Vào từ file văn bản FILTR.INP gồm một dòng chứa 2 số nguyên L và R .

Kết quả: Đưa ra file văn bản FILTR.OUT một số nguyên – số bước tối đa của trò chơi.

Ví dụ:

FILTR.INP	FILTR.OUT
2 10	6



Bài 25. LINE 2010

Tên chương trình: LINE2010.???

Sự ra đời của máy tính bảng iPAD đã làm rộ lên phong trào thiết kế trò chơi cài đặt trên iPAD. Một nhóm bạn trẻ đã đề xuất trò chơi LINE 2010 như sau.

Cho bảng ô vuông kích thước $m \times n$ (m hàng và n cột). Các hàng được đánh số từ trên xuống dưới bắt đầu từ 1, các cột được đánh số bắt đầu từ 1 từ trái sang phải. Ô (i, j) là ô nằm ở giao giữa hàng i với cột j . Một số ô trên bảng bị đánh dấu cấm truy nhập. Các ô còn lại được gọi là các ô tự do. Mỗi ô tự do có thể để trống hoặc có một viên bi.

Cho trước trạng thái đầu và trạng thái đích của bảng, ở mỗi trạng thái tại một số ô tự do có bi. Nhiệm vụ của người chơi là bằng cách phép biến đổi nêu dưới đây đưa bảng từ trạng thái đầu về trạng thái đích *bằng chi phí nhỏ nhất có thể với khả năng của bạn*.

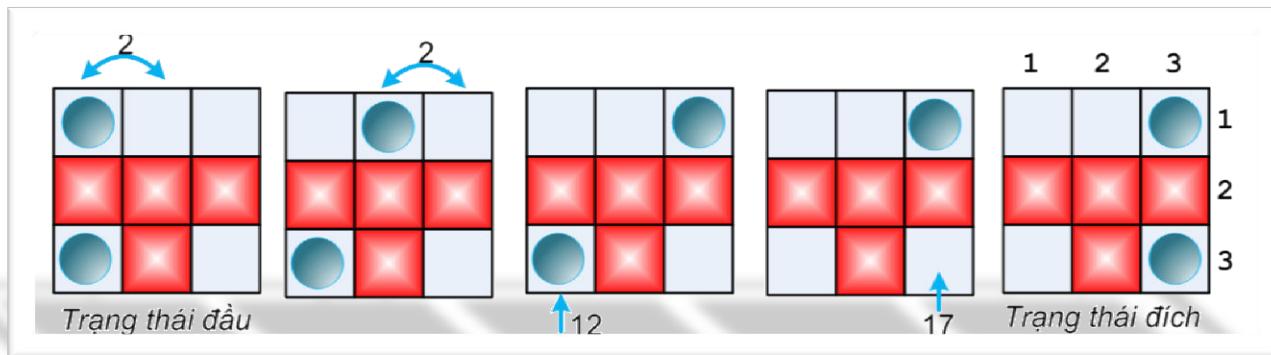
Các phép biến đổi có thể áp dụng là:

- *Biến đổi 1:* khi chọn phép biến đổi này, nếu bạn kích hoạt vào ô tự do còn trống, một viên bi sẽ xuất hiện ở ô này với chi phí là \mathbf{A} ,
- *Biến đổi 2:* khi chọn phép biến đổi này, nếu bạn kích hoạt vào ô có bi, viên bi sẽ biến mất với chi phí là \mathbf{B} ,

- *Biến đổi 3:* khi chọn phép biến đổi này, nếu bạn kích hoạt vào cắp ô tự do kề cạnh, trong đó có một ô trống và một ô có bi thì bi sẽ lăn sang ô trống với chi phí là C .

Mỗi phép biến đổi có thể được sử dụng nhiều lần hoặc không sử dụng lần nào.

Trạng thái của bảng được cho dưới dạng m xâu ký tự độ dài n . Xâu thứ i xác định trạng thái dòng i của bảng, mỗi ký tự của xâu nhận một trong 3 giá trị từ tập $\{\star, \#, .\}$, ký tự “ \star ” cho biết đó là ô tự do và có bi, ký tự “ $\#$ ” cho biết đó là ô cấm truy nhập, ký tự “ $.$ ” cho biết đó là ô tự do và không có bi.



Ví dụ, với $n = m = 3$, $A = 17$, $B = 12$, $C = 2$, các trạng thái đầu và cuối của bảng là:

$$\begin{array}{ll}
 \star \dots & \dots \star \\
 \#\#\# & \#\#\# \\
 *\# . & . \# *
 \end{array}
 \quad
 \begin{array}{ll}
 \dots & \star \\
 \#\#\# & \#\#\# \\
 . \# * &
 \end{array}$$

Trạng thái đầu Trạng thái cuối

Việc biến đổi tối ưu có thể được thực hiện theo cách nêu trên hình vẽ với chi phí nhỏ nhất là $2+2+12+17 = 33$.

Yêu cầu: Cho các số nguyên m , n , A , B , C ($1 \leq m, n \leq 100$, $1 \leq A, B, C \leq 10^5$), trạng thái đầu và trạng thái cuối của bảng. Hãy chỉ ra với chi phí nhỏ nhất có thể cách đưa bảng từ trạng thái đầu về trạng thái cuối: số bước biến đổi và bắn thân các phép biến đổi cần thực hiện.

Dữ liệu: Vào từ file văn bản LINE10.INP:

- Dòng đầu tiên chứa 2 số nguyên m và n ghi cách nhau ít nhất một dấu cách,
- Dòng thứ 2 chứa 3 số nguyên A , B và C , các số trên một dòng ghi cách nhau ít nhất một dấu cách
- Dòng thứ i trong m dòng tiếp theo chứa xâu P độ dài n mô tả dòng i của bảng ở trạng thái ban đầu ($i = 1 \div m$),
- Dòng thứ $m+3$: dòng trống,
- Dòng thứ j trong m dòng tiếp theo chứa xâu Q độ dài n mô tả dòng j của bảng ở trạng thái cuối ($j = 1 \div m$).

Kết quả: Đưa ra file văn bản LINE10.OUT:

- Dòng đầu tiên là số nguyên k – số phép biến đổi cần thực hiện,
- Mỗi dòng trong k dòng sau mô tả một phép biến đổi theo quy cách:
 - Với phép biến đổi 1: 1 x y, trong đó (x, y) - tọa độ ô tác động,
 - Với phép biến đổi 2: 2 x y, trong đó (x, y) - tọa độ ô tác động,
 - Với phép biến đổi 3: 1 x y u v trong đó (x, y) và (u,v) - tọa độ cặp ô tác động.

Các phép biến đổi ghi theo trình tự thực hiện.

Ví dụ:

LINE10.INP	LINE10.OUT
<pre> 3 3 17 12 2 *.. ### *#. ..* *** .#* </pre>	<pre> 4 3 1 1 1 2 3 1 2 1 3 2 3 1 1 3 3 </pre>

Chấm điểm:

- Nếu vi phạm một trong các lỗi sau bạn sẽ nhận được điểm 0:
 - Câu lệnh không hợp lệ: tác động lên ô cấm, đổi chỗ 2 ô không có bi hay cùng có bi, cặp ô không kề nhau, thực hiện phép biến đổi 1 với ô đã có bi, thực hiện phép biến đổi 2 với ô chưa có bi, . . .
 - Viết câu lệnh không đúng quy tắc,
 - Số phép biến đổi vượt quá 65 535,
 - Trạng thái bảng sau khi thực hiện các câu lệnh không trùng với trạng thái cuối đã cho.
- Nếu không vi phạm các điều nói trên thì điểm của bạn sẽ được tính theo công thức:

$$\frac{10 \times Ans}{S}$$

Trong đó:

- S – chi phí tính theo cách biến đổi bạn nêu trong file output,
- Ans – chi phí tối ưu đối với test đang xét.

Điểm sẽ được đưa ra với 2 chữ số sau dấu chấm thập phân (không làm tròn).

Ví dụ, với test nêu trong đầu bài, nếu file output của bạn có dạng:

4
2 1 1

1 1 3
 2 3 1
 1 3 3

Chi phí S = $12 + 17 + 12 + 17 = 58$.

Điểm của bạn sẽ là $\frac{10 \times 33}{58} = 5.68$

Bài 26. CỜ CA RÔ

Tên chương trình: CARO.???

Các bạn trẻ tìm thấy một cuốn vở nhỏ ai đó để quên trên ghế đá công viên. Trong cuốn vở này chỉ có các dòng số khó hiểu. Ban đầu các bạn của chúng ta nghĩ đây là thông báo mã hóa của một điệp viên nào đó định chuyển cho một người khác và đang tìm cách xử lý, nhưng may quá, người chủ cuốn vở đã quay lại tìm và giải thích đây chỉ là ghi chép các nước đi của một ván đấu trong trò chơi ca rô giữa 2 người trên trường ca rô vô hạn. Người thứ nhất dùng dấu chéo, người thứ hai – chữ o, mỗi người đến lượt mình đi sẽ đánh dấu của mình vào một ô trống tùy chọn. Ai tạo được trước dãy từ 5 ký hiệu trở lên của mình theo chiều dọc hoặc ngang hay chéo thì sẽ thắng.

Các bạn trẻ bị thu hút bởi ván đấu và muốn xác định xem ai đã thắng.

Yêu cầu: Cho n – số nước đi của ván đấu ($0 \leq n \leq 10\,000$), tọa độ các ô của mỗi nước đi, cho theo trình tự thực hiện nước đi. Hãy các định xem ai thắng ván đấu này hoặc kết quả là hòa, đưa ra thông báo tương ứng **First player won** hay **Second player won** hoặc **Draw**. Nếu sau một nước đi nào đó đã xác định được người thắng nhưng người chơi vẫn được tiếp tục thực hiện nước đi thì thông báo **Inconsistent**.

Dữ liệu: Vào từ file văn bản CARO.INP:

- Dòng đầu tiên chứa số nguyên n ,
- Mỗi dòng trong n dòng sau chứa 2 số nguyên x và y xác định tọa ô được đi ($|x|, |y| \leq 10^9$).

Kết quả: Dưa ra file văn bản CARO.OUT dòng thông báo xác định được.

Ví dụ:

CARO.INP	CARO.OUT
<pre>9 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5</pre>	<pre>First player won</pre>

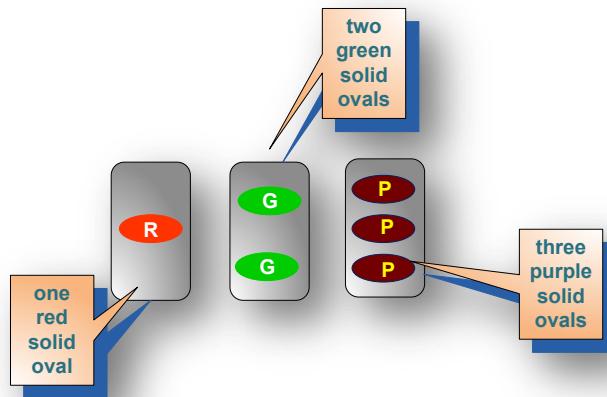
Bài 27. TRÒ CHƠI SET

Tên chương trình: SET.???

SET là trò chơi rèn luyện kỹ năng quan sát. Trò chơi sử dụng bộ bài đặc biệt 81 quân. Hình in trên mỗi quân bài khác nhau theo 4 dấu hiệu:

- Số lượng hình: trên mỗi quân bài có thể có in một (**ONE**), hai (**TWO**) hoặc ba (**THREE**) hình,
- Màu: màu của hình có thể là đỏ (**RED**), xanh (**GREEN**) hoặc tím (**PURPLE**),
- Cách tô màu: không tô (**OPEN**), tô đặc (**SOLID**) hoặc gạch chéo (**STRIPED**),
- Kiểu hình: Ô van (**OVAL**), thoi (**DIAMOND**) hoặc sợi xoắn (**SQUIGGLE**).

Người chơi phải tìm trong những quân bài mình có các tập và đặt chúng xuống bàn. Mỗi tập là một bộ 3 quân bài, theo mỗi dấu hiệu hoặc các bài đều giống nhau hoặc khác nhau từng đôi một. Hình bên thể hiện một tập.



Yêu cầu: Cho n quân bài. Hãy xác định số cách chọn ra một tập. Hai cách chọn khác nhau có thể có chứa quân bài chung.

Dữ liệu: Vào từ file văn bản SET.INP:

- Dòng đầu tiên chứa số nguyên n ($3 \leq n \leq 81$),
- Mỗi dòng trong n dòng sau mô tả một quân bài, chứa 4 xâu ký tự theo trình tự nêu trên. Số kiểu hình có thể ở dạng số ít hay số nhiều tùy theo hoàn cảnh.

Kết quả: Đưa ra file văn bản SET.OUT một số nguyên – số cách chọn.

Ví dụ:

SET.INP	SET.OUT
<pre> 3 one red solid oval two green solid ovals three purple solid ovals </pre>	<pre> 1 </pre>

Bài 28. XÓA CỘT

Tên chương trình: DELCOL.???

Cho bảng kích thước $2 \times n$ ô (2 dòng, n cột). Ô ở cột i dòng thứ nhất chứa số nguyên dương a_i , ô ở cột i dòng thứ hai chứa số nguyên dương b_i ($1 \leq a_i, b_i \leq 3000$, $1 \leq n \leq 300000$).

a_1	a_2	a_n
b_1	b_2	b_n

Xét trò chơi giữa 2 người với tổng s_1 và s_2 tương ứng ban đầu bằng 0. Mỗi người đến lượt mình đi chọn một cột i nào đó chưa được đánh dấu, cộng a_i vào s_1 (nếu là người thứ nhất đi) hoặc cộng b_i vào s_2 (nếu là người thứ hai đi), sau đó đánh dấu xóa cột này. Trò chơi kết thúc khi tất cả các cột đều bị đánh dấu xóa. Ban đầu chưa có cột nào bị đánh dấu xóa. Người thứ nhất đi trước. Sau khi trò chơi kết thúc, nếu $s_1 > s_2$ thì người thứ nhất thắng và giá trị thắng sẽ là $s_1 - s_2$. Nếu $s_2 > s_1$ thì người thứ 2 thắng với giá trị là $s_2 - s_1$. Giá trị của trò chơi được xác định bằng giá trị $s_1 - s_2$, người thứ nhất có găng cực đại hóa nó, người thứ 2 – có găng cực tiểu hóa. Cả hai người đều biết cách đi tối ưu.

Yêu cầu: Xác định giá trị lớn nhất có thể đạt được đối với người chơi thứ nhất.

Dữ liệu: Vào từ file văn bản DELCOL.INP:

- Dòng đầu tiên chứa số nguyên n ,
- Dòng thứ i trong n dòng sau chứa 2 số nguyên a_i và b_i .

Kết quả: Đưa ra file văn bản DELCOL.OUT một số nguyên – giá trị trò chơi.

Ví dụ:

DELCOL.INP	DELCOL.OUT
<pre> 3 1 2 3 4 5 6 </pre>	<pre> 2 </pre>

Bài 29. TRÒ CHƠI 29

Tên chương trình: GAME29.???

Ghép từ là trò chơi phổ biến của những người làm máy tính. Đây là trò chơi 2 người. Nội dung trò chơi như sau: bắt đầu từ một từ rỗng hai người lần lượt đi, mỗi người đến lượt mình phải ghép vào từ hiện có một ký tự tùy chọn vào đầu hoặc cuối từ để nhận được từ mới là xâu con các ký tự liên tiếp nhau của một từ trong danh sách các từ cho trước. Ai làm cho từ hiện có trở thành từ trong danh sách các từ cho trước nói trên là thua.

Ví dụ, “**comp**” là từ trong danh sách từ cho trước.

- Người thứ nhất: “**m**”,
- Người thứ hai: “**mp**”,
- Người thứ nhất: “**omp**”,
- Người thứ hai: “**comp**”. Người thứ hai thua!

Yêu cầu: Cho biết danh sách từ. Hãy xác định ai sẽ là người thắng nếu cả hai đều đi một cách tối ưu. Nếu người thứ nhất thắng thì hãy chỉ ra danh sách các ký tự mà người thứ nhất có thể bắt đầu.

Dữ liệu: Vào từ file văn bản GAME29.INP:

- Dòng đầu tiên chứa số nguyên **n** – số từ trong danh sách ($1 \leq n \leq 200$),
- Mỗi dòng trong **n** dòng sau chứa một từ độ dài không quá 50 ký tự, mỗi ký tự có mã ASCII trong phạm vi từ 33 đến 126, chữ hoa và thường là khác nhau.

Kết quả: Đưa ra file văn bản GAME29.OUT thông báo **First** hoặc **Second** xác định người thắng. Nếu người thứ nhất thắng thì ở dòng thứ hai – đưa ra danh sách các ký tự mà người thứ nhất có thể bắt đầu trò chơi.

Ví dụ:

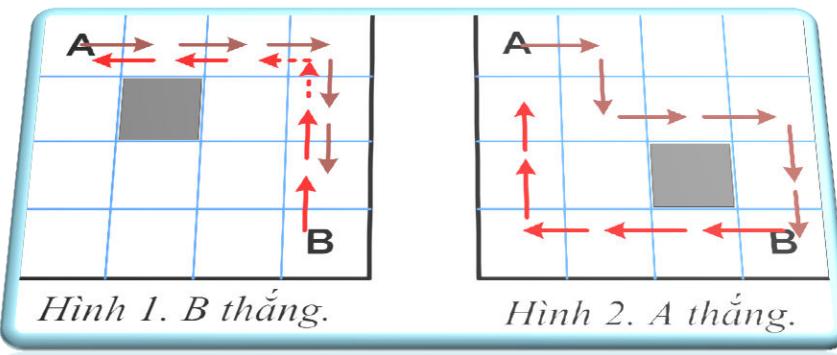
GAME29.INP	GAME29.OUT
10 one two three four five six seven eight nine ten	First fu

Bài 30. TRÒ CHƠI 30

Tên chương trình: GAME30.???

Trò chơi giữa 2 người A và B diễn ra trên bảng lưới ô vuông kích thước $n \times n$ ($2 \leq n \leq 300$). Một số ô của bảng có màu trắng, những ô khác – màu đen. Người chơi chỉ được đi trên các ô màu trắng. A và B mỗi người có một vị trí xuất phát riêng của mình, không trùng nhau và thuộc các ô màu trắng. Hai người lần lượt đi, bắt đầu từ người A. Quy tắc đi là như sau:

- Từ vị trí của mình đi sang ô trống kề cạnh của bảng,
- Nếu ô đi tới đang có quân của đối phương thì được quyền đi tiếp một nước nữa (theo hướng tùy chọn, có thể khác với hướng nước đi trước).



Ai đến được vị trí xuất phát của đối phương trước là người đó thắng. Mỗi bảng, với các vị trí xuất phát cho trước sẽ tồn tại chiến lược đi thắng cho A hoặc B. Với bảng ở hình 1 B có chiến lược đi thắng, ví dụ, nếu A đi sang phải thì B đi lên trên. Sẽ tồn tại thời điểm B có thể đi theo quy tắc 2 và về đích sớm hơn A. Với bảng ở hình 2 A có chiến lược đi thắng: hai nước đi đầu tiên là sang phải rồi xuống dưới.

Yêu cầu: Cho n và thông tin về các ô trong bảng. Hãy xác định ai có chiến lược đi thắng.

Đữ liệu: Vào từ file văn bản GAME30.INP gồm nhiều tests:

- Dòng đầu tiên chứa số nguyên t – số lượng tests,
- Mỗi test cho trên một nhóm dòng:
 - Dòng đầu tiên chứa số nguyên n ,
 - Mỗi dòng trong n dòng sau chứa ký tự độ dài n mô tả một dòng của bảng, các ký tự ‘A’, ‘B’ chỉ vị trí xuất phát, ký tự ‘.’ – ô trống, ‘#’ – ô đen.

Kết quả: Đưa ra file văn bản GAME30.OUT, kết quả mỗi test đưa ra trên một dòng dưới dạng ký tự ‘A’ hoặc ‘B’ cho biết ai có chiến lược thắng.

Ví dụ:

GAME30.INP	GAME30.OUT
2 4 A # B 4 A #. ... B	B A

Bài 31. THÁP XU

Tên chương trình: COINS.???

Alice và Bob chơi trò bốc xu từ tháp được xây dựng bởi N đồng xu. Hai bạn chọn hai số nguyên dương khác nhau K và L. Hai người lần lượt đi. Alice đi trước. Mỗi người, khi đến lượt mình, được bốc khỏi tháp 1, K hoặc L xu. Ai bốc được đồng xu (hoặc các đồng xu) cuối cùng là thắng. Sau rất nhiều lần chơi, Alice nhận thấy rằng có những trường hợp mình chắc chắn thắng không phụ thuộc vào cách đi của Bob, ngược lại, có trường hợp dù đi thế nào Bob vẫn thắng. Trước ván chơi mới Alice nóng lòng muốn biết mình có thắng được hay không.

Yêu cầu: Cho N, K và L. Hãy xác định Alice hay Bob sẽ thắng.

Dữ liệu: Vào từ file văn bản COINS.INP:

- Dòng đầu tiên chứa 3 số nguyên K, L và m, trong đó m – số ván chơi ($1 < K < L < 10$, $3 < m < 50$),
- Dòng thứ 2 chứa m số nguyên N_1, N_2, \dots, N_m , trong đó N_i – số xu trong tháp ở ván chơi thứ i ($1 \leq N_i \leq 10^6$, $i = 1 \div m$).

Kết quả: Đưa ra file văn bản COINS.OUT xâu m ký tự từ tập {A, B}, ký tự thứ i là A nếu Alice thắng được và bằng B nếu Bob thắng.

Ví dụ:

COINS.INP	COINS.OUT
2 3 5 3 12 113 25714 88888	ABAAB

Bài 32. \sqrt{Nim}

Tên chương trình : NIMSQRT.???

\sqrt{Nim} là trò chơi bốc sỏi giữa hai người. Ban đầu trên bàn có một đống n viên sỏi. Hai người lần lượt bốc khỏi đống sỏi một số lượng viên theo quy tắc sau: nếu trước đi đống sỏi còn k viên thì người đi được bốc một số lượng viên tùy chọn không vượt quá \sqrt{k} . Ví dụ, nếu trên bàn còn 10 viên thì người đi được quyền bốc 1, 2 hoặc 3 viên. Ai bốc được viên cuối cùng thì người đó thắng.

Yêu cầu: Cho biết n ($1 \leq n \leq 10^{12}$). Hãy xác định người thứ nhất (người đi trước) có thể thắng được hay không nếu cả hai người chơi đều biết chiến lược tối ưu.

Dữ liệu: Vào từ file văn bản NIMSQRT.INP gồm nhiều tests, mỗi test cho trên một dòng chứa một số nguyên n .

Kết quả: Đưa ra file văn bản NIMSQRT.OUT, kết quả mỗi test đưa ra trên một dòng dưới dạng thông báo **WIN** hoặc **LOSE**.

Ví dụ:

NIMSQRT.INP	NIMSQRT.OUT
3	WIN
5	LOSE

Bài 33. TRÒ CHƠI 33

Tên chương trình: GAME33.???

Ba người giải trí bằng trò chơi xây dựng dãy số. Người thứ nhất nghĩ ra một số nguyên N_1 và báo cho người thứ hai. Người thứ hai phải tìm số nguyên N_2 , sao cho với nó tồn tại hai số nguyên A_1 và B_1 thoả mãn các điều kiện:

- $A_1 \geq 2$ và $B_1 \geq 2$,
- $A_1 + B_1 = N_2$, $A_1 \times B_1 = N_1$.

Số N_2 được báo cho người thứ ba. Người này phải tìm số N_3 với các tính chất tương tự và báo cho người thứ nhất, . . . Nếu ở bước i một người chuyển cho người tiếp theo số N_i và người tiếp theo này không thể tìm ra các số A_i , B_i thì người đó thua cuộc và trò chơi kết thúc. Nếu dãy số có thể kéo dài vô hạn thì kết quả trò chơi là hoà.

Yêu cầu: Cho số nguyên N - số mà người thứ nhất chọn. Hãy xác định người thua hay đưa ra số 0 trong trường hợp hoà. Giả thiết cả ba người đều thực hiện nước đi một cách tối ưu, tức là nếu có cách đi thắng thì họ sẽ đi theo cách đó, nếu không thắng được và có nước đi dẫn đến hoà thì người đi sẽ chọn số đảm bảo hoà, trong trường hợp còn lại – người đi chọn số lớn nhất trong các số phù hợp.

Dữ liệu: Vào từ file văn bản GAME33.INP gồm nhiều tests, mỗi test cho trên một dòng chứa một số nguyên N .

Kết quả: Đưa ra file văn bản GAME33.OUT, kết quả mỗi test đưa ra trên một dòng, chứa một số nguyên trong phạm vi từ 0 đến 3.

Ví dụ:

GAME33.INP	GAME33.OUT
5	1
16	3
4	0

Bài 34. GOM BI

Tên chương trình: COLLECT.PAS

Bác Tôm làm nhiệm vụ coi kho giữ các thiết bị máy tính cũ. Công việc hàng ngày cũng không nhiều lăm. Để tiêu khiển giết thời gian, bác gom các viên bi chuột vào các hòm đựng bi, rồi xúc ra ba rổ bi, mỗi rổ chứa bi một màu. Kết quả là bác Tôm có trong tay A viên màu đỏ, B viên màu vàng và C viên màu xanh. Sau đó bác lấy ra hai viên bi, mỗi viên một màu và thay chúng bằng một viên bi màu thứ ba. Nếu sau một số lần thay thế như vậy chỉ còn lại đúng một viên bi thì coi như thắng. Với kinh nghiệm chơi nhiều năm, khi biết được A , B , C , bác Tôm biết ngay là có thể thắng được hay không. Còn bạn thì sao?

Yêu cầu: Cho ba số nguyên A , B , C ($0 \leq A, B, C \leq 2^{63}-1$). Hãy xác định, bác Tôm có thể thắng hay không.

Dữ liệu: Vào từ file văn bản COLLECT.INP:

- Dòng đầu tiên chứa số nguyên T – số lượng Tests,
- Mỗi dòng trong số T dòng sau chứa 3 số nguyên A, B, C .

Kết quả: Đưa ra file văn bản COLLECT.OUT, kết quả mỗi test đưa ra trên một dòng dưới dạng thông báo **YES** hoặc **NO**.

Ví dụ:

COLLECT.INP	COLLECT.OUT
2	YES
1 0 0	NO
1 1 1	

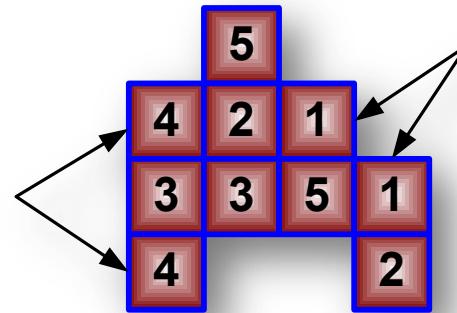
Bài 35. TRÒ CHƠI MAHJONG

Tên chương trình: MAHJONG.???

Steva tìm được trên mạng trò chơi Mahjong và rất hứng thú với trò chơi mới này. Để chơi Mahjong cần có một bảng chữ nhật kích thước $m \times n$ ô và một bộ quân các màu, trong đó mỗi màu có đúng hai quân. Ban đầu các quân được đặt một cách ngẫu nhiên trong các ô trên bảng, mỗi ô có không quá một quân. Sau đó người ta bắt đầu đi nhặt các quân cùng màu. Ở mỗi bước ta có thể nhặt ra khỏi bảng một cặp quân cùng màu nếu chúng thỏa mãn ít nhất một trong hai điều kiện:

- Đầu là quân trái nhất trong hàng ngang của mình,
- Đầu là quân trên nhất trong hàng dọc của mình.

Với cấu hình nêu ở hình bên, bước đầu tiên ta có thể nhặt hai quân 4 hoặc hai quân 1. Khi nào không thể nhặt được quân nữa thì trò chơi kết thúc. Mục tiêu của trò chơi là phải tìm cách đi được càng nhiều nước càng tốt.



Yêu cầu: Cho m, n ($1 \leq m, n \leq 300$, ít nhất một trong hai số đó là chẵn) và màu quân ở mỗi ô. Màu được đánh số từ 1 đến $m \times n / 2$. Hãy xác định số nước đi tối đa có thể thực hiện.

Dữ liệu: Vào từ file văn bản MAHJONG.INP:

- Dòng đầu tiên chứa hai số nguyên $m n$,
- Dòng thứ i trong m dòng sau: chứa n số nguyên xác định màu các quân trên dòng thứ i .

Kết quả: Đưa ra file văn bản MAHJONG.OUT:

- Dòng thứ nhất chứa số nguyên k - số nước đi tối đa có thể thực hiện,
- Dòng thứ 2 chứa k số nguyên xác định màu các quân cần nhặt theo trình tự các nước đi.

Nếu có nhiều cách đi thì đưa ra cách có thứ tự từ điển nhỏ nhất.

Ví dụ:

MAHJONG.INP	MAHJONG.OUT
3 4 1 2 3 4 1 2 3 4 5 5 6 6	6 1 2 3 4 5 6

Bài 36. QUÂN HẬU

Tên chương trình: QUEEN.???

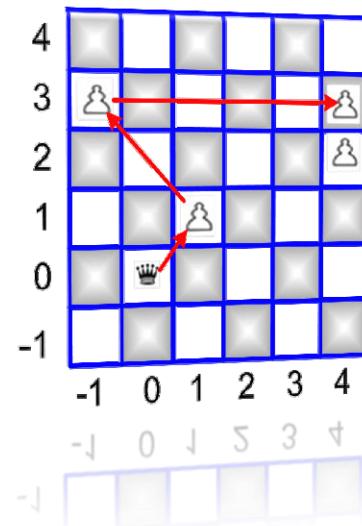
Tại ô $(0, 0)$ của một bàn cờ vô hạn có một quân hậu đen. Quân hậu có thể đi ngang, dọc hoặc chéo, nhưng không được đi xuống dưới, tức là ở mỗi nước đi, ô mới tới phải có toạ độ y lớn hơn hoặc bằng toạ độ y ở ô cũ.

Trên bàn cờ có n quân tốt ($1 \leq n \leq 50\,000$). Quân tốt thứ i đứng ở ô (x_i, y_i) , ($y_i >= 0$). Các quân tốt không di chuyển.

Yêu cầu: Hãy chỉ ra cách đi cho con hậu để ăn được nhiều tốt nhất. Mỗi nước đi phải ăn một quân và không được nhảy qua ô đang có quân.

Dữ liệu: Vào từ file văn bản QUEEN.INP:

- Dòng đầu tiên chứa số nguyên n ,
- n dòng sau: mỗi dòng chứa 2 số nguyên x_i y_i ($0 \leq |x_i|, y_i \leq 10^9$).



Kết quả: Đưa ra file văn bản QUEEN.OUT:

- Dòng đầu tiên đưa ra số nguyên k – số quân tốt có thể ăn được,
- Dòng thứ 2 đưa ra k số nguyên – trình tự các quân cần ăn.

Ví dụ:

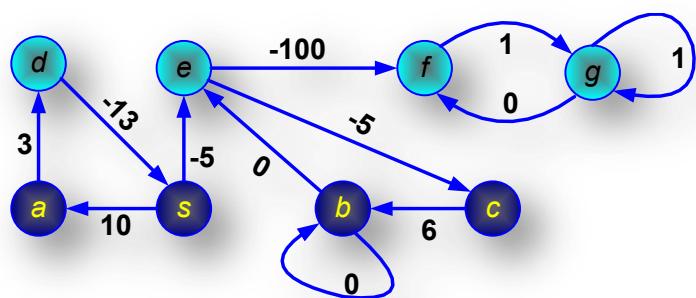
QUEEN.INP	QUEEN.OUT
<pre>4 1 1 4 3 -1 3 4 2</pre>	<pre>3 1 3 2</pre>

Bài 37. TRÒ CHƠI CHI PHÍ TRUNG BÌNH

Tên chương trình: MEAN.???

Đây là trò chơi 2 người trên đồ thị có hướng. Đỉnh của đồ thị được chia thành 2 tập: A và B . Tập đỉnh A thuộc người chơi thứ nhất, tập đỉnh B – thuộc người thứ 2. Mỗi cung uv của đồ thị được gắn với một số nguyên w_{uv} .

Trò chơi diễn ra như sau: Ban đầu quân cờ được đặt ở một đỉnh s nào đó của đồ thị. Sau đó hai người chơi bắt đầu di chuyển quân cờ. Nếu quân ở đỉnh thuộc tập A thì người thứ 1 được quyền đi, trong trường hợp ngược lại – người



thứ 2 đi. Hai người chơi có một bộ đếm z chung. Ban đầu $z = 0$. Sau mỗi nước đi, giá trị w_{uv} được tích lũy vào z . Trò chơi kéo dài vô hạn.

Mục tiêu của người thứ nhất là làm cho $z \rightarrow +\infty$. Mục tiêu của người thứ 2 là làm cho $z \rightarrow -\infty$. Nếu không ai đạt được mục tiêu của mình – kết quả hòa.

Ở hình nêu trên, những đỉnh tó đậm thuộc tập A , những đỉnh còn lại – tập B . Ban đầu quân cờ ở đỉnh s . Di chuyển sang đỉnh e , người thứ nhất sẽ thắng. Nếu người thứ 2 đi sang f , anh ta sẽ bị quẩn trong chu trình $f \rightarrow g \rightarrow f$, sau mỗi lần như vậy z tăng lên 1. Còn nếu người thứ 2 đi sang c , thì người thứ nhất đi sang b và trở về e , điều này cũng làm z tăng lên 1.

Có thể chứng minh được rằng, nếu một người chơi có thể thắng thì tồn tại *chiến lược tiền định*, tức là ở mỗi đỉnh thuộc người chơi này, nếu có quân cờ thì có thể chỉ ra cung di chuyển, không phụ thuộc vào giá trị hiện tại của z cũng như quá trình đi trước đó.

Rôn và Harry định thi đấu với nhau. Rôn là người chơi thứ nhất. Hai bạn vẽ đồ thị và xác định đỉnh xuất phát. Rôn cho rằng mình có thể thắng và xác định chiến lược đi thắng. Tuy vậy Rôn chưa hoàn toàn tin tưởng là mình đã tính đúng.

Yêu cầu: Cho n - số đỉnh đồ thị, m - số cung ($1 \leq n \leq 2000$, $1 \leq m \leq 10000$), danh sách các đỉnh thuộc Rôn (đánh dấu là ‘ A ’) và các đỉnh thuộc Harry (đánh dấu là ‘ B ’), chiều và giá trị các cung (theo giá trị tuyệt đối không vượt quá 10^5). Mỗi đỉnh có ít nhất một cung ra. Cho biết chiến lược thắng của Rôn dưới dạng dãy n số nguyên a_i ($i = 1 \div n$, $a_i = 0$ nếu đỉnh thuộc Harry, trong trường hợp ngược lại – cách đi của Rôn ($i \rightarrow a_i$) nếu quân cờ ở đỉnh i). Nếu chiến lược này đúng thì đưa ra thông báo **WIN**. Nếu theo cách đi của Rôn, Harry có thể thắnh thì đưa ra thông báo **LOSE**, trong trường hợp hòa – đưa ra thông báo **DRAW**. Trong hai trường hợp cuối – đưa ra chiến lược đi của Harry (tương tự như cách Rôn nêu).

Dữ liệu: Vào từ file văn bản MEAN.INP gồm nhiều tests, mỗi test cho trên một nhóm dòng:

- Dòng đầu tiên chứa 2 số nguyên n và m ,
- Dòng thứ 2 chứa xâu n ký tự từ tập $\{A, B\}$,
- Dòng thứ 3 chứa số thứ tự đỉnh xuất phát,
- Mỗi dòng trong m dòng sau chứa 3 số nguyên u v w_{uv} ,
- Dòng cuối cùng chứa n số nguyên mô tả chiến lược đi của Rôn.

Kết quả: Đưa ra file văn bản MEAN.OUT, kết quả của mỗi test đưa ra trên 1 dòng (nếu là **WIN**) hoặc 2 dòng trong các trường hợp còn lại:

- Dòng thứ nhất chứa thông báo **LOSE** hoặc **DRAW**,
- Dòng thứ 2 chứa n số nguyên mô tả chiến lược của Harry.

Ví dụ:

MEAN.INP	MEAN.OUT
8 12 AAAAABBBB 22 1 10 3 3 0 4 3 6 1 5 3 5 2 -13 2 6 -5 3 6 0 6 4 -5 6 7 -100 7 8 1 8 7 0 8 8 1 4 6 7 3 0 0 0 0 8 12 AAAAABBBB 22 1 10 3 3 0 4 3 6 1 5 3 5 2 -13 2 6 -5 3 6 0 6 4 -5 6 7 -100 7 8 1 8 7 0 8 8 1 4 1 7 3 0 0 0 0	WIN DRAW 0 0 0 0 5 8 10 11

Bài 38. TRÒ CHƠI KÝ TỰ

Tên chương trình: STRGAME.PAS

Đã là ngày thứ ba công việc ở Viện nghiên cứu Xử lý thông tin văn bản bị đình trệ. Mọi người bị cuốn hút vào một trò chơi hấp dẫn mới. Trường xử lý là xâu S nối vòng cả hai đầu. Ban đầu quân cờ được đặt ở vị trí 1 của xâu S. Mỗi người đến lượt mình đi có quyền di chuyển quân cờ trong phạm vi **R** vị trí sang phải hay **L** vị trí bên trái với điều kiện không được tới ô có ký tự trùng với ký tự ở ô mà quân cờ đang đứng hay đứng trước đó một nước hoặc 2 nước đi. Ai đến lượt mình không đi được nữa là người đó thua.

Ví dụ, 3 ký tự tương ứng với 3 nước đi cuối cùng là ‘*a*’, ‘*b*’, và ‘*d*’. Hiện tại quân cờ ở vị trí thứ 4 trong xâu ‘*abadbac*’, $L = 5$ và $R = 1$. Nước đi duy nhất là chuyển sang trái 4 ô, tới vị trí ký tự ‘*c*’.

Không hài lòng với sự trì trệ công việc, ban lãnh đạo Viện quyết định giao cho một nhân viên lập trình phân tích trò chơi, xác định người đi đầu sẽ thắng, thua hay hoà. Một chương trình như vậy sẽ làm các nhân viên mất hứng thú với trò chơi và quay trở lại công việc thường ngày của mình.

Yêu cầu: Cho xâu S độ dài không quá 10 000 ký tự và chỉ chứa các ký tự từ ‘*a*’ đến ‘*h*’, các giá trị L và R ($1 \leq L, R \leq 50$). Hãy xác định người đi đầu tiên thắng, thua hay hoà và đa ra thông báo tương ứng **WIN**, **LOSE** hoặc **DRAW**.

Dữ liệu: Vào từ file văn bản STRGAME.INP:

- Dòng đầu tiên chứa xâu S ,
- Dòng thứ hai chứa 2 số nguyên $L M$.

Kết quả: Đưa ra file văn bản STRGAME.OUT thông báo kết quả khảo sát.

Ví dụ:

STRGAME.INP
<i>abadbac</i>
5 1

STRGAME.OUT
DRAW

STRGAME.INP
<i>abacaba</i>
2 2

STRGAME.OUT
LOSE

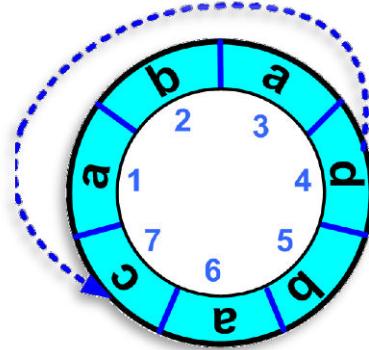
STRGAME.INP
<i>aabacab</i>
1 1

STRGAME.OUT
WIN

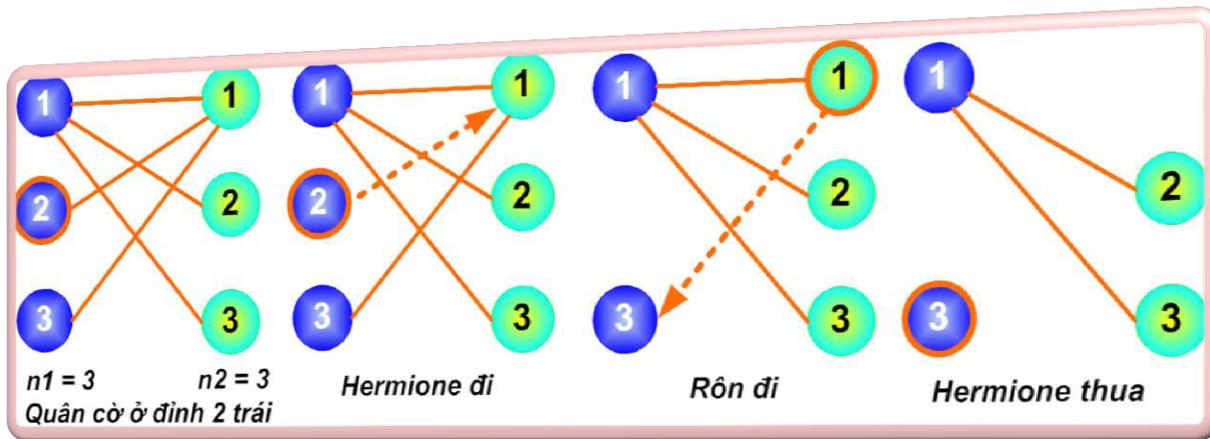
Bài 39. TRÒ CHƠI 39

Tên chương trình: GAME39.PAS

Trong giờ nghỉ giải lao giữa các tiết Hermione và Rôn thường hay chơi trò chuyển quân trên đồ thị hai phía. Các bạn vẽ một đồ thị hai phía với các cung vô hướng. Quân cờ được đặt ở một trong các đỉnh của đồ thị. Hai người lần lượt đi. Mỗi người, khi đến lượt mình đi - chuyển quân



cờ tới đỉnh khác theo cung hiện có trên đồ thị, xoá khỏi đồ thị đỉnh nơi quân cờ vừa rời khỏi



cùng các cung xuất phát từ nó. Ai đến lượt mình không đi được nữa là thua. Hermione đi trước.

Yêu cầu: Cho $n1, n2, m$ và m cặp (i, j) , trong đó $n1$ - số đỉnh bên trái, $n2$ - số đỉnh bên phải, m - số cung, i - đỉnh bên trái, j - đỉnh bên phải ($1 \leq n1, n2 \leq 500, 0 \leq m \leq 50\,000$). Hãy xác định với từng vị trí ban đầu có thể của quân cờ ai sẽ là người thắng : Hermione (**H**) hay là Rôn (**R**).

Dữ liệu: Vào từ file văn bản GAME39.INP:

- Dòng đầu tiên chứa 3 số nguyên $n1\ n2\ m$,
- m dòng sau: mỗi dòng chứa 2 số nguyên $i\ j$.

Kết quả: Đưa ra file văn bản GAME39.OUT 2 dòng, dòng thứ nhất chứa xâu ký tự độ dài $n1$, dòng thứ 2 chứa xâu ký tự độ dài $n2$, ký tự thứ i trong mỗi xâu là **H** hoặc **R** xác định người thắng nếu ban đầu quân cờ đặt ở vị trí i . Xâu thứ nhất ứng với cột trái và xâu thứ hai - cột phải.

Ví dụ:

GAME39.INP		
3	3	5
1	1	
1	2	
1	3	
2	1	
3	1	

GAME39.OUT		
HRR		
HRR		



118

Bài 40. GOM SỎI

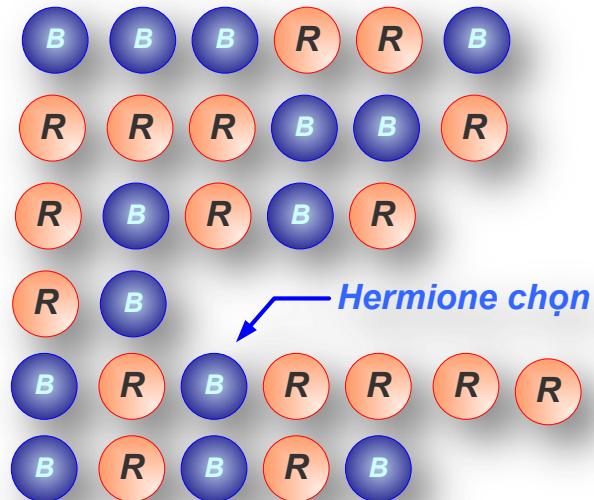
Tên chương trình: STONE.PAS

Hermione có một số viên sỏi xanh và Rôn có một số viên màu đỏ. Hai bạn bỏ chung sỏi vào một hộp rồi lấy ngẫu nhiên các viên sỏi ra khỏi hộp và sắp chúng thành các hàng. Số lượng sỏi ở các hàng không nhất thiết phải như nhau. Sau đó hai người lần lượt bốc sỏi ra khỏi hàng. Mỗi người,

khi đến lượt mình chọn một viên sỏi của mình (Hermione – màu xanh, còn Rôn – màu đỏ) ở hàng và vị trí tuỳ chọn, bốc ra khỏi hàng viên sỏi đó cùng với tất cả các viên sỏi ở bên phải của nó trong hàng, không phụ thuộc các viên đó có màu gì. Ai đến lượt đi mà không còn sỏi màu của mình thì người đó thua. Hermione được đi trước.

Ví dụ, trong cấu hình dưới đây, Hermione chọn quân ở vị trí nêu trong hình và sẽ thắng trong ván này bằng cách áp dụng chiến lược “đối xứng”: Nếu Rôn đi tiếp ở hàng 1 thì Hermione lặp lại nước đi đó ở hàng 2 và ngược lại; nếu Rôn đi ở hàng 3 thì Hermione lặp nước đi đó ở hàng 6, . . .

Yêu cầu: Cho biết n – số hàng sỏi ($1 \leq n \leq 100$) và cấu hình của mỗi hàng dưới dạng các xâu ký tự R , B , mỗi xâu có độ dài không quá 20. Hãy đưa ra tên người thắng. Nếu Hermione thắng thì cần chỉ ra nước đi đầu tiên dưới dạng $i\ j$, trong đó i – hàng, j – vị trí viên cần chọn kể từ trái sang (bắt đầu bằng 1).



Dữ liệu: Vào từ file văn bản STONE.INP:

- Dòng đầu tiên chứa số nguyên n ,
- Các dòng sau: mỗi dòng chứa một xâu cấu hình.

Kết quả: Đưa ra file văn bản STONE.OUT:

- Dòng thứ nhất: Đưa ra tên người thắng,
- Nếu người thắng là **Hermione** thì dòng thứ 2 chứa 2 số nguyên $i\ j$.

Ví dụ:

STONE.INP
6
BBBBRB
RRRBBR
RBRBR
RB
BRBRRRR
BRBRB

STONE.OUT
Hermione
5 3

Bài 41. ĐOÁN SỐ

Tên chương trình: GUESS.PAS

Hermione và Rôn chơi trò đoán số. Hermione nghĩ trong đầu một số trong phạm vi từ 1 đến n ($1 \leq n \leq 10^5$). Rôn phải đoán số đó bằng cách đưa ra các câu hỏi dạng: “Số đó có nằm trong tập S không?”, trong đó S là tập các số bất kỳ trong phạm vi từ 1 tới n . Mỗi lần có câu trả lời khẳng định, Rôn phải đưa cho Hermione 2 cái kẹo, trong trường hợp ngược lại – đưa một cái.

Yêu cầu: Hãy xác định số kẹo tối thiểu Rôn cần chuẩn bị để chắc chắn đoán được số Hermione nghĩ.

Dữ liệu: Vào từ file văn bản GUESS.INP gồm một số nguyên n .

Kết quả: Đưa ra file văn bản GUESS.OUT số kẹo cần chuẩn bị.

Ví dụ:

GUESS.INP	GUESS.OUT
6	5
2	2



j27

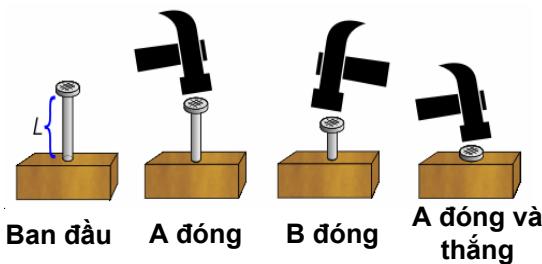
Bài 42. ĐÓNG ĐINH

Tên chương trình: NAIL.???

A và B thi tài khéo léo quả mình qua trò chơi đóng đinh. Ban đầu đinh được cắm vào súc gỗ và còn nhô ra một đoạn độ dài L . Quy tắc đóng như sau:

- Khi $L > A_{max}$, A chỉ được đóng cho đinh lún thêm x: $A_{min} \leq x \leq A_{max}$,
- Khi $L > B_{max}$, B chỉ được đóng cho đinh lún thêm x: $B_{min} \leq x \leq B_{max}$,
- Khi $L \leq A_{max}$, nếu đến lượt A đi, A sẽ đóng ngập đinh và thắng,
- Khi $L \leq B_{max}$, nếu đến lượt B đi, B sẽ đóng ngập đinh và thắng.

X – nguyên.



Yêu cầu: Cho $L, A_{max}, A_{min}, B_{max}, B_{min}$ ($1 \leq L, A_{max}, A_{min}, B_{max}, B_{min} \leq 2^{30}$, $A_{min} \leq A_{max}$, $B_{min} \leq B_{max}$). A là người đi trước. Hãy xác định ai sẽ thắng, nếu cả hai đều biết cách đi tối ưu.

Dữ liệu: Vào từ file văn bản NAIL.INP:

- Dòng đầu tiên chứa số nguyên T - số lượng Tests,
- Mỗi dòng trong số T dòng tiếp theo chứa 5 số nguyên $L A_{min} A_{max} B_{min} B_{max}$.

Kết quả: Đưa ra file văn bản NAIL.OUT mỗi kết quả đưa ra trên một dòng gồm một ký tự (A hoặc B) xác định người thắng.

Ví dụ:

NAIL.INP	NAIL.OUT
4	A
4 5 7 1 20	A
5 1 3 1 3	B
5 2 2 1 3	B
1000 1 3 1 3	

Bài 43. ĐI TRÊN LƯỚI

Tên chương trình: GRID.PAS

Cho lưới ô vuông kích thước $N \times N$. Ô trên trái có tọa độ (1,1), ô dưới phải có tọa độ (N,N). Trên mỗi ô có ghi một số nguyên. Bạn phải đi từ ô trên trái xuống ô dưới phải theo các quy tắc sau:

- ✚ Không được ra ngoài lưới,
- ✚ Chỉ được đi sang phải, trái hoặc xuống dưới theo các ô kề cạnh,
- ✚ Mỗi ô đi qua không quá một lần,
- ✚ Không được đi qua ô chứa giá trị âm.

Yêu cầu: Cho N, K ($0 < N \leq 75, 0 \leq K \leq 5$) và giá trị các ô trên lưới (số nguyên 16 bit). Hãy xác định tổng lớn nhất của đường đi theo quy tắc trên. Nếu không có đường đi thỏa mãn thì đưa ra thông báo NO.

Dữ liệu: Vào từ file văn bản GRID.INP, gồm nhiều test, mỗi test bao gồm:

- Dòng đầu tiên chứa 2 số nguyên $N K$,
- N dòng sau: mỗi dòng chứa N số nguyên mô tả một dòng của lưới, lần lượt từ trên xuống dưới.

Kết quả: Đưa ra file văn bản GRID.OUT, mỗi kết quả đưa ra trên một dòng.

Ví dụ:

GRID.INP	GRID.OUT
4 1	11
1 2 3 -5	NO
-10 6 0 -1	
-10 -10 -10 2	
0 0 0 1	
4 0	
1 2 3 -5	
-10 6 0 -1	
-10 -10 -10 2	
0 0 0 1	

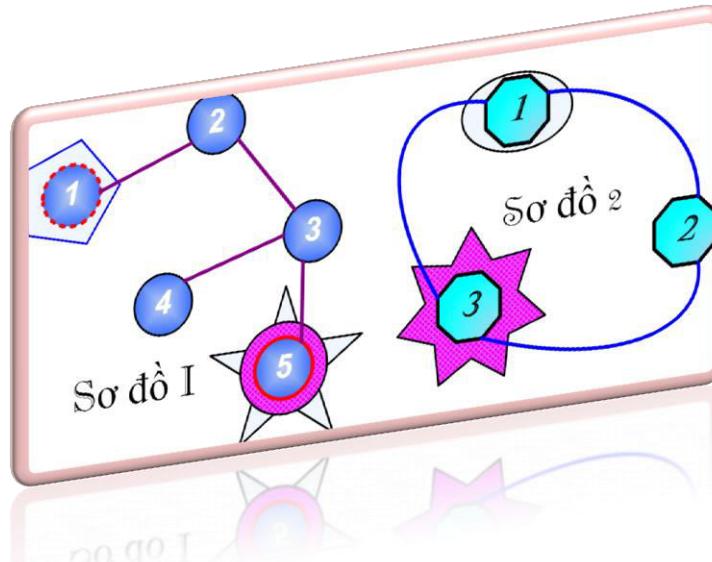
Bài 44. CHUYỂN QUÂN

Tên chương trình: PUZZLE.PAS

Trò chơi di chuyển quân theo một số luật nhất định trên sơ đồ cho trước vẽ trên giấy là khá phổ biến trong những năm trước đây.

Rõn tìm thấy trên gác xếp một tập các sơ đồ như vậy. Đáng tiếc là không thấy một chỉ dẫn nào cả kèm theo các sơ đồ. Rõn chỉ đoán nhận được các ô tròn là vị trí quân đi có thể đặt, các cung nối là đường có thể di chuyển và theo hình vẽ - đoán được đâu là ô xuất phát và đâu là ô đích. Có những ô được gắn với nhiều đường dẫn. Đường dẫn cho phép đi cả hai chiều.

Bởi vì Rõn không biết quy tắc chơi nên phải tự mình nghĩ ra quy tắc riêng. Rõn sử dụng đồng thời tất cả các sơ đồ. Ở mỗi sơ đồ chỉ đặt đúng một quân, bắt đầu từ vị trí xuất phát. Mỗi bước đi là một lần di chuyển quân trên từng sơ đồ sang ô có thể đi được. Ngay cả khi quân đã về tới ô đích, nhưng nếu cần thiết vẫn phải đi sang ô khác. Rõn muốn biết, tối thiểu sau bao nhiêu nước đi quân trên tất cả các sơ đồ đồng thời tới đích. Biết chắc chắn rằng ở mỗi sơ đồ, giữa hai ô bất kỳ bao giờ cũng có đường đi (trực tiếp hoặc qua các ô trung gian).



Yêu cầu: Cho k – số lượng bản đồ, n_i – số lượng ô trong sơ đồ thứ i , m_i – số cung trên sơ đồ thứ i ($1 \leq k \leq 10$, $2 \leq n_i \leq 50$, $1 \leq m_i \leq 1500$). Các đỉnh trong sơ đồ thứ i được đánh số từ 1 đến n_i , trong đó 1 – ô xuất phát, n_i – ô đích ($i = 1 \div k$). Hãy xác định số bước đi tối thiểu để đưa quân trên các sơ đồ đồng thời về ô đích.

Dữ liệu: Vào từ file văn bản PUZZLE.INP:

- Dòng đầu tiên chứa số nguyên k ,
- k nhóm dòng sau; mỗi nhóm mô tả một sơ đồ, trong đó:
 - Dòng đầu tiên chứa 2 số nguyên n_i m_i ,
 - Mỗi dòng trong m_i dòng tiếp theo chứa 2 số nguyên, cho biết 2 ô có cung nối trực tiếp.

Kết quả: Đưa ra file văn bản PUZZLE.OUT số nguyên r – số nước đi tối thiểu. $r = -1$ nếu không thể đưa các quân đồng thời về ô đích.

Ví dụ:

PUZZLE.INP	PUZZLE.OUT
<pre> 2 5 4 1 2 2 3 3 4 3 5 3 3 1 2 2 3 3 1 </pre>	<pre> 3 </pre>

Bài 45. ĐÃY QUÂN

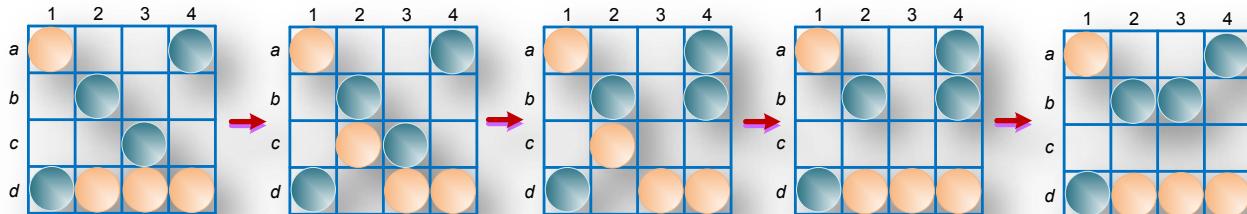
Tên chương trình:MOVE.PAS

Trò chơi *Dao* do Jeff Pickering và Ben van Burkirk đề xuất năm 1999. Đó là trò chơi 2 người. Ở đây chúng ta xét phương án cài tiến thành trò một người chơi. Xét bảng kích thước 4×4 ô, trên đó có 4 quân cờ trắng và 4 quân cờ đen. Ban đầu các quân cờ được đặt một cách ngẫu nhiên. Người chơi phải đưa bảng các quân cờ về trạng thái cuối cho trước theo các quy tắc di chuyển sau:

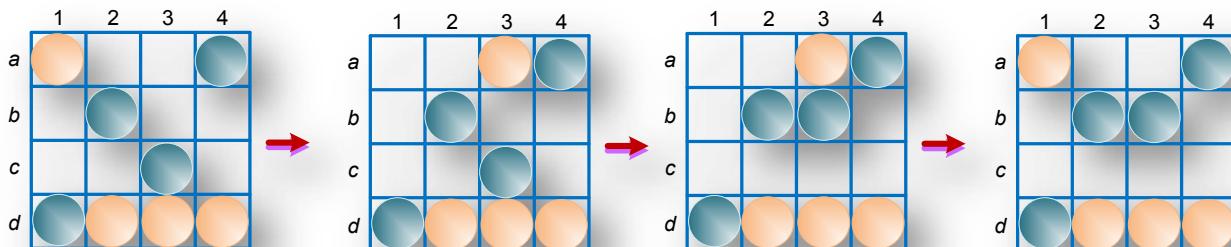
- Di lần lượt quân trắng- quân đen, bắt đầu từ quân trắng,
- Quân có thể di theo hàng ngang, hàng dọc hoặc chéo cho tới khi bị chặn bởi quân đã có hay mép bàn cờ, không được ăn hoặc nhảy qua quân khác,
- Ở mỗi bước phải di chuyển quân màu tương ứng, không được bỏ qua nước đi.

Ví dụ, các nước đi sau là hợp lệ:

Trong trường hợp này, 4 nước đi đã được thực hiện.



Tuy vậy, ta cũng có thể nhận được kết quả cuối như trên chỉ với 3 nước đi:



Yêu cầu: Cho trạng thái đầu và cuối của bàn cờ. Hãy xác định số bước đi tối thiểu để từ trạng thái đầu nhận được trạng thái cuối.

Dữ liệu: Vào từ file văn bản MOVE.INP:

- Dòng chứa số nguyên t - số lượng Tests,
- Thông tin về mỗi Test được đưa trên 8 dòng, 4 dòng đầu mô tả trạng thái đầu của bảng, 4 dòng sau – mô tả trạng thái cuối. Các bảng được mô tả theo dòng, từ trên xuống dưới, mỗi dòng được mô tả bằng một xâu 4 ký tự, trong đó w chỉ quân trắng, b – quân đen và * - vị trí trống.

Kết quả: Đưa ra file văn bản MOVE.OUT đưa ra số lượng bước đi tối thiểu cho từng Test, mỗi kết quả đưa ra trên một dòng.

Ví dụ:

MOVE.INP	MOVE.OUT
w***b	1
wb	
bw	
b**w	
w**b	
wb	
bw	
bw**	
w**b	
*b**	
**b*	
bwww	
w**b	
bb	

bwww	

Bài 46. SỐ XU

Tên chương trình: NUMCOINS.PAS

Có N cọc tiền xu, cọc thứ i có P_i đồng, ($1 \leq P_i \leq 1000$, $2 \leq N \leq 100$). Các cọc tiền được xếp thành một dãy từ trái sang phải. Hai người lần lượt đi, mỗi người khi đến lượt mình được lấy một số tuỳ ý các cọc xu bên trái, ít nhất là một cọc và nhiều nhất là số cọc người trước đã lấy. quá trình đi kết thúc khi các cọc xu bị lấy hết. Người đi đầu không được lấy quá K cọc. Mỗi người đều cố gắng làm thế nào để lấy được nhiều tiền nhất.

Yêu cầu: Xác định số xu tối đa mà người đi trước có thể lấy được ứng cách đi tối ưu của người thứ 2.

Dữ liệu: Vào từ file văn bản NUMCOINS.INP:

- Dòng đầu tiên chứa 2 số nguyên N và K,

- Dòng thứ 2 chứa N số nguyên P_1, P_2, \dots, P_N .

Kết quả: Đưa ra file văn bản NUMCOINS.OUT:

- Dòng thứ nhất chứa số xu tối đa người đi đầu lấy được,
- Dòng thứ 2 chứa các số nguyên Q_1, Q_2, \dots, Q_M - số cọc xu mà hai người lần lượt lấy.

Chỉ cần nêu một trong số các cách đi

Ví dụ:

NUMCOINS.INP
5 2
3 5 4 4 6

NUMCOINS.OUT
13
1 1 1 1 1 1

Bài 47. BỐC SỎI

Tên chương trình: STONE.PAS

Có N đồng sỏi, mỗi đồng có 2 viên và M đồng sỏi, mỗi đồng có 3 viên. Hai người chơi trò chơi bốc sỏi. Hai người lần lượt đi. Mỗi người đến lượt mình đi có quyền bốc một số lượng sỏi bất kỳ trong một đồng. Ai bốc được viên cuối cùng trong đồng cuối cùng thì người đó thắng.

Yêu cầu: Cho N và M ($0 < M, N \leq 100$). Hãy xác định xem người thứ nhất có cách đi thắng hay không. Nếu có thì chỉ ra nước đi đầu tiên của người thứ nhất.

Dữ liệu: Vào từ file văn bản STONE.INP gồm 2 số nguyên N M trên một dòng.

Kết quả: Đưa ra file văn bản STONE.OUT:

- Dòng đầu tiên chứa thông báo YES hoặc NO ứng với trường hợp người thứ nhất thắng hoặc thua,
- Nếu dòng thứ nhất là YES thì đưa ra tiếp thông tin trên các dòng sau:
 - Dòng thứ 2: số nguyên K - số cách đi khác nhau ban đầu (việc bốc cùng một số lượng sỏi ở đồng này hay đồng khác có số lượng giống nhau không coi là cách đi khác nhau),
 - Mỗi dòng trong K dòng sau đưa ra 2 số nguyên: loại đồng sỏi (2 hoặc 3) và số lượng sỏi trong đồng cần bốc.

Ví dụ:

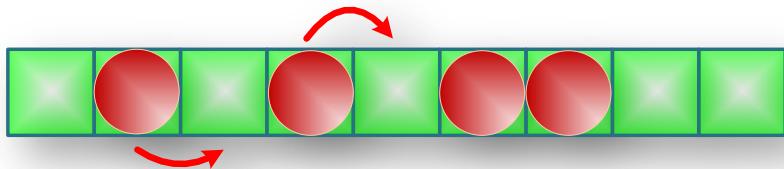
STONE.INP
1 2

STONE.OUT
YES
2
2 2
3 2

Bài 48. CHUYỂN QUÂN

Tên chương trình STEP.PAS

Trên một băng N ($1 < N \leq 200$) vị trí đánh số từ 1 đến N từ trái sang phải ở M vị trí có đặt các viên bi ($1 \leq M \leq N-1$). Mỗi vị trí chứa không quá một viên bi và vị trí N không có bi. Hai người lần lượt di chuyển bi theo quy tắc sau: mỗi người, đến lượt mình, lấy một viên bi tùy chọn và đặt vào vị trí trống gần nhất bên phải của viên bi đó. Ai đặt được bi vào vị trí N sẽ thắng. Ở hình bên với $N = 9$, $M = 4$, bi được đặt tại các vị trí 2, 4, 6, 7. Với cấu hình này, người thứ nhất có thể thắng được và nước đi duy nhất có thể thắng đọc là chuyển viên bi ở vị trí 2 sang vị trí 3.



Yêu cầu: Cho N , M và vị trí các viên bi. Hãy xác định số lượng cách đi nước đi đầu tiên của người thứ nhất để đảm bảo phần thắng sẽ thuộc về người thứ nhất. Số lượng cách đi này có thể bằng 0 nếu người thứ nhất không có cách thắng.

Dữ liệu: Vào từ file văn bản STEP.INP:

- Dòng đầu tiên chứa 2 số nguyên N M ,
- Dòng thứ 2 chứa M số nguyên xác định vị trí các viên bi.

Kết quả: Đưa ra file văn bản STEP.OUT một số nguyên - số lượng cách đi nước đi đầu tiên của người thứ nhất để đảm bảo phần thắng sẽ thuộc về người thứ nhất.

Ví dụ:

STEP.INP
9 4
2 4 6 7

STEP.OUT
2

Bài 49. VUA VÀ HẬU

Tên chương trình: QUEEN.PAS

Cho bàn cờ kích thước $2*N$, trên đó có không quá M vách ngăn ở một số ô ($2 \leq N \leq 50$, $1 \leq M \leq 300$). Các vách ngăn nằm ngang hoặc đứng. Có quân vua và quân hậu, mỗi quân ở một đầu của bàn cờ, quân vua ở ô $(1, 1)$, quân hậu ở ô $(2, N)$. Các quân này chuyển động theo quy tắc của quân cờ tương ứng. Trong quá trình di chuyển chúng không được đi vào các nước ăn nhau. Các quân không được vượt qua vách chắn. Không được phép đi chéo khi vách ngăn chạm vào đường đi. Ai không thể đi được khi đến lượt mình là thua. Quân vua đi trước.

Hãy xác định quân nào thắng và tổng số nước đi của cả hai quân trong trường hợp cả hai đều đi một cách tối ưu.

Dữ liệu: Vào từ file văn bản QUEEN.INP:

- Dòng đầu tiên chứa 2 số nguyên N M ,
- M dòng sau: mỗi dòng chứa 4 số nguyên X_1 Y_1 X_2 Y_2 xác định toạ độ các ô giữ chúng có vách ngăn ($1 \leq X_1, X_2 \leq 2, 1 \leq Y_1, Y_2 \leq N$). Mỗi vách ngăn có thể được mô tả nhiều lần.

Kết quả: Đưa ra file văn bản QUEEN.OUT:

- Dòng thứ nhất chứa một trong số 3 thông báo QUEEN WINS, KING WINS hoặc DRAW (hoà),
- Trong trường hợp không hoà thì có dòng thứ 2 chứa tổng số nước đi.

Ví dụ:

QUEEN.INP
2 1
1 2 2 2
2 2
1 2 2 2
2 2 2 1
2 2
1 2 2 2
1 1 2 1

QUEEN.OUT
queen wins
2
king wins
1
draw

Bài 50. TÍCH LŨY TIỀN

Tên chương trình: MONEY.???

Phần chơi giành cho khán giả giữa một chương trình truyền hình có nội dung như sau: một khán giả được chọn làm người chơi và được tặng n đồng ($1 \leq n \leq 100\,000$). Nội dung trò chơi giữa khán giả được chọn (người chơi) với người dẫn chương trình như sau: Gọi số tiền hiện tại người chơi đang có là k đồng. Nếu k chẵn thì người chơi phải đưa cho người dẫn chương trình một nửa số tiền mình có, trong trường hợp ngược lại người chơi nhận được thêm $2k+1$ đồng. Sau mỗi lần, người chơi quyết định có tiếp tục chơi hay dừng trò chơi. Trò chơi cũng kết thúc khi người chơi chỉ còn 1 đồng.

Yêu cầu: Hãy xác định số tiền lớn nhất người chơi có thể nhận được nếu biết cách dừng trò chơi đúng lúc.

Dữ liệu: Vào từ file văn bản MONEY.INP gồm nhiều bộ dữ liệu, mỗi bộ dữ liệu là một số nguyên ghi trên một dòng.

Kết quả: Đưa ra file văn bản MONEY.OUT, kết quả ứng với mỗi bộ dữ liệu ghi trên một dòng dưới dạng số nguyên.

Ví dụ:

MONEY.INP
11
27

MONEY.OUT
52
9232

Bài 51. TRÒ CHƠI EULER

Tên chương trình: EULER.PAS

Xét trò chơi giữa 2 người trên bàn cờ 4×4 . Ban đầu bàn cờ rỗng. Mỗi người chơi có một số lượng không hạn chế các que độ dài 1, 2 và 3. Người chơi có thể cắm que từ cạnh bàn cờ và que đó sẽ chiếm một số lượng ô đúng bằng độ dài que. Que cũng có thể đặt dựng vuông góc với mặt bàn cờ và nó sẽ chiếm chỗ đúng một ô. Các que không được phép gác lên nhau hay đặt lên ô đã bị chiếm chỗ. Người nào không còn nước đi nữa sẽ thua. Ở trạng thái trên hình bên, nếu đi đúng người thứ nhất sẽ thắng.

Giả thiết hai người thông minh như nhau. Do số nước đi hạn chế, nên sớm hay muộn trò chơi sẽ kết thúc và ai đó sẽ thắng, không có trường hợp hoà.

Yêu cầu: Cho trạng thái hiện tại của bàn cờ. Hãy xác định người đi nước tiếp theo có thể thắng được hay không.

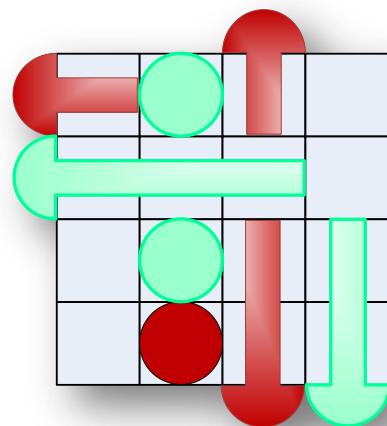
Dữ liệu: Vào từ file văn bản EULER.INP:

- Dòng đầu tiên chứa số nguyên K - số lượng Tests,
- Các dòng sau mô tả các Test, mỗi Test bắt đầu bằng một dòng trống, sau đó là 4 dòng, mỗi dòng một xâu 4 ký tự. Các vị trí trống được đánh dấu bằng ‘.’, các vị trí còn lại – ký tự ‘x’.

Kết quả: Đưa ra file văn bản EULER.OUT: Mỗi Test đưa ra một kết quả YES/NO cho biết người đi tiếp theo thắng hay thua.

Ví dụ:

EULER.INP	EULER.OUT
3	NO
xxx.	YES
xxx.	NO
.xxx	
.xxx	
xxxx	
...x	
xx.x	
xx.x	
....	
....	
....	
....	



Bài 52. ĐOÁN TỪ

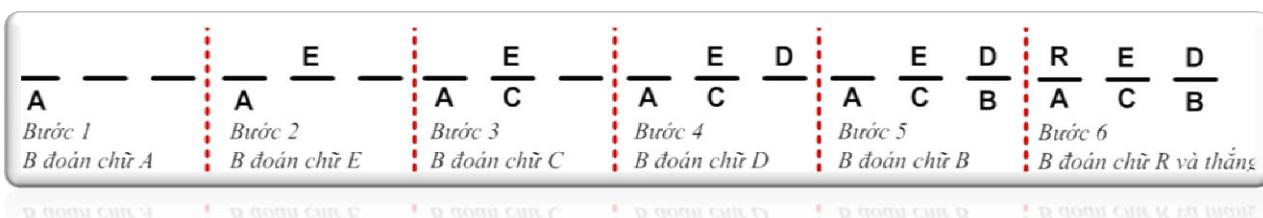
Tên chương trình: GUESS.???

Đoán từ là trò chơi 2 người rất phổ biến trong học sinh, sinh viên Iran. Gọi 2 người chơi là A và B. A đi trước, chọn một từ trong danh sách các từ mà cả hai đều biết, ghi nhớ từ đó trong đầu và vạch trên đường thẳng n dấu vạch ngang – đúng bằng số ký tự của từ mình đã chọn. Sau đó B bắt đầu đoán từng chữ cái một trong từ. Ở mỗi lượt đi, B chọn một chữ cái và nói cho A. Sẽ có các trường sau xảy ra:

Chữ cái B chọn có trong từ của A, khi đó A sẽ viết chữ cái đó bên trên dấu gạch ngang ở vị trí tương ứng với vị trí chữ cái trong từ. Nếu đoán được hết các chữ cái B sẽ là người thắng.

Trong trường hợp ngược lại A sẽ viết chữ cái đã nói dưới gạch ngang trái nhất mà dưới đó còn trống. Nếu không còn chỗ viết (B đoán sai n+1 lần) thì A thắng.

Ví dụ A chọn từ RED (có trong danh sách). B lần lượt đoán các chữ cái **A**, **E**, **C**, **D**, **B** và **R** thì B thắng và biên bản trò chơi có dạng:



Tuy vậy, nếu ở lượt đi cuối cùng, thay vì **R** B đoán là **S** thì sẽ thua!

Aidin là người hâm mộ cuồng nhiệt trò chơi này. Aidin nhận thấy rằng nếu danh sách các từ đủ lớn và các từ có quan hệ với nhau đủ mạnh thì người chơi A có thể có hành động không đẹp là thay đổi từ trong quá trình chơi, bởi từ đã chọn chỉ ở trong đầu A chứ không ghi lại ở bất kỳ một chỗ nào khác và A có thể thay đổi từ để biên bản đã ghi vẫn phù hợp. Ví dụ, danh sách các từ là **RED**, **BED**, **LED** và **TED**, thì sau bước 4 A chắc chắn sẽ thắng: A luôn luôn có thể viết chữ cái **B** đoán xuống dưới và sau mỗi lượt đi sẽ bị mất thêm một từ trong tập **{RED, BED, LED, TED}**, rồi cuối cùng chỉ cần thông báo: “*Tù tớ đã nghĩ là, ở ..., ở ...*”.

Aidin thấy rằng với danh sách từ tốt người chơi A trong một số trường hợp đảm bảo chắc chắn là người đi đầu. Ví dụ với các từ độ dài 2 thì danh sách từ **{ME, MD, DE, ED, AS, IS, AI, SI}** đảm bảo người đi đầu chắc chắn thắng. Bạn hãy tự mình tìm lấy chiến lược chơi!

Cho một danh sách từ, hãy xác định xem A có chắc chắn thắng được không mà không phụ thuộc vào chiến lược đoán của B.

Dữ liệu: Vào từ file văn bản GUESS.INP:

- Dòng đầu tiên chứa số nguyên **c** – số danh sách từ ($1 < c \leq 20$),

- Mỗi danh sách từ cho trên 2 dòng: dòng đầu tiên chứa số nguyên k – số từ trong danh sách, dòng thứ 2 chứa các từ, mỗi từ có độ dài nhỏ hơn 7, chỉ chứa các chữ cái la tinh in hoa, trong một từ không có 2 ký tự nào giống nhau, các từ các nhau bởi các dấu cách, dấu Tab hoặc dấu xuống dòng.
- Thông tin về các danh sách cách nhau bởi một dòng trống.
- Đảm bảo kích thước file input không quá 500KB.

Kết quả: Đưa ra file văn bản GUESS.OUT thông báo “**Yes**” hoặc “**No**” với mỗi danh sách từ, mỗi thông báo trên một dòng. Thông báo “**Yes**” ứng với trường hợp A có cách chọn từ để chắc chắn thắng.

Lưu ý là trong trường hợp A thắng thì phải nói từ đã chọn để B kiểm tra lại tính đúng đắn của các bước trong quá trình chơi.

Ví dụ:

GUESS.INP	GUESS.OUT
<pre>2 12 SI ME AND AI ARE MD AS WHEN ED IS DE HAPPY 5 A B AB AC AD</pre>	<pre>Yes No</pre>

Tổng quan về các bài toán trò chơi đối kháng

Nguyễn Duy Khuong

Các trò chơi đối kháng giữa hai người đã được hình thành từ lâu. Và những người chơi luôn cố gắng tìm mọi cách để mình giành được phần thắng. Và bạn có biết rằng các trò chơi đã được đoán trước là thắng, thua hay hoà không? Ý tôi muốn nói rằng, nếu một trò chơi cho trước vị trí ban đầu thì kết quả tốt nhất mà người chơi đầu tiên đạt được đã được biết từ trước (ở đây tôi giả thiết cả hai người chơi đều chơi tối ưu). Vẫn đề là các trò chơi thường quá phức tạp lên không có một ai có thể đảm bảo rằng mọi nước đi của mình là tối ưu. Do vậy cho đến nay, chỉ một số lượng nhỏ bài toán đó đã được giải quyết. Và trong bài viết này tôi xin giới thiệu một cách khá đầy đủ về trò chơi đối kháng hai người. Bài toán đó được phát biểu tổng quát dưới dạng đồ thị như sau:

Cho đồ thị có hướng $G=(V,E)$ (Đồ thị G có tập đỉnh V , tập cạnh là E). Với mỗi đỉnh $v \in V$, ta định nghĩa $E(v) = \{ u \mid (v,u) \in E \}$

Một trò chơi hai người được định nghĩa là một đồ thị có hướng $G = (V, E)$ trong đó mỗi trạng thái chơi tương ứng với một đỉnh của đồ thị, hàm $E(v)$ là qui tắc chơi tức là $E(v)$ chứa các đỉnh hay trạng thái chơi mà từ v có thể đi đến. Hai người luôn phiên nhau đi, ở thế chơi u người chơi chỉ có thể đi sao cho nước v nhận được thoả mãn $v \in E(u)$. Trò chơi kết thúc khi đến lượt đầu mà không thể đi tiếp được nữa. (Thông thường thì người không thể đi tiếp là người thua cuộc).

Tôi xin chia bài toán này thành hai loại bài toán: loại thứ nhất là, mỗi trạng thái chơi chỉ có một đối tượng mỗi đối tượng là một đỉnh của đồ thị. Loại thứ hai là mỗi trạng thái chơi có nhiều đối tượng. (Sự khác nhau căn bản các bạn sẽ được rõ ở phần sau).

I. Loại thứ nhất:

P1. Xét bài toán cụ thể - GAME

Một trò chơi đối kháng giữa hai người A và B diễn ra như sau: Hai người luôn phiên nhau điều khiển một con tốt theo một số con cho trước. Một người có thể di chuyển con tốt từ vị trí u đến v nếu có một đường nối trực tiếp có hướng từ u đến v . Trò chơi kết thúc không thể tiếp tục di chuyển. Người không thể tiếp tục đi là người thua cuộc. Hỏi nếu cho trước vị trí ban đầu và danh sách các đường nối hỏi người đi trước thắng hay người đi sau thắng hay hoà? Giả hai người này rất thông minh các bước đi của họ là tối ưu (tức học không bao giờ đi các nước không có loại cho mình).

Input: Game.In

- Dòng đầu ghi số N là số vị trí con tốt có thể dừng, và số M là số đường đi (có hướng) mà con tốt có thể đi ($1 \leq N \leq 200$, $1 \leq M \leq N^*(N-1)$).
- Dòng thứ hai ghi u là trạng thái bắt đầu.
- M dòng tiếp theo mỗi dòng ghi hai số u, v mô tả một đường đi từ u đến v .

Output: Game.Out

- Ghi một số duy nhất 1, 2, hoặc 0. 1 nghĩa là người 1 thắng, 2 là người 2 thắng, 0 là hoà.

Nhận xét:

- Những vị trí không có đường ra thì chắc chắn sẽ thua.
- Những vị trí nào có một đường ra nối với vị trí chắc chắn thua thì chắc chắn thua.
- Những vị trí nào tất đường ra nối với các vị trí chắc chắn thắng thì chắc chắn thua.
- Những vị trí nào mà trạng thái thua không thể xác định thì là vị trí hoà.- Bài toán có trạng thái hoà: VD: có các đường nối $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 1$, $1 \rightarrow 4$, $4 \rightarrow 5$. Các vị trí 1,2,3 sẽ hòa, 5 thua, 4 thắng.

Thuật toán:

- Lúc đầu coi tất cả các vị trí v đều hoà gán giá trị đỉnh $F[v] = 0$. Tìm các vị trí không có đường ra thì gán lại

$F[v] = 2$ (tức là nếu người chơi ở vị trí này sẽ thua). - Khi thay trạng thái một vị trí từ hoà sang thắng hoặc thua thì kiểm tra các vị trí có đường đi đến nó: Những vị trí u nào có một đường ra nối với vị trí v chắc chắn thua ($F[v] = 2$) sẽ thì chắc chắn thua (thay $F[u] = 1$); Những vị trí u nào tất đường ra nối với các vị trí v ($F[v] = 1$) chắc chắn thắng thì chắc chắn thua (thay $F[u] = 2$).

- Quá trình này ngừng khi không có sự chuyển trạng nào nữa.

Chương trình mô tả thuật toán:

Procedure gan_nhan (u: byte);

Var td, v : byte;

Begin

td := 0;

If Noi_dinh_thuau) then td := 1 Else

If Noi_toan_dinh_thang_hoac_khong_co_dinh_ra (u) Then td := 2;

F[u] := td;

If td <> 0 Then

For v := 1 to N do

If F[v] = 0 Then

If C[v, u] Then gan_nhan (td);

End;

Procedure Main;

Var u : Integer;

Begin

Fillchar (F, sizeof (F), 0);

For u := 1 to N do

If Khong_Co_Canh_Ra (u) Then Gan_nhan(u);

End;

P2. (Bài tập tự giải) LGAME (BOI 2002) Cho một bảng kích thước 4×4 ô vuông, trên đó đặt hai thanh thước thợ hòn L kích thước 4 ô vuông và hai hình tròn như hình vẽ, các hình này nằm trên bảng và không được đè lên nhau. Hình kẻ ca rô là của người chơi A, hình kẻ sọc của người chơi B. Hai người sẽ chơi luôn phiên, tại mỗi nước đi, một người sẽ phải nhắc thanh hòn L của mình lên, xoay, lật tuỳ ý và di chuyển đến vị trí mới (khác ít nhất một ô so với vị trí ban đầu), như vậy hình đầu tiên có hai cách di chuyển. Và người chơi có thể thực hiện thêm một bước đi không bắt buộc là di chuyên một ô tròn đến một ô mới.

Trò chơi kết thúc khi không thể di chuyển được nữa, người không thể di chuyển được sẽ thua cuộc. Tuy nhiên, trò chơi vẫn có thể hoà vì trong trạng đó cả hai người đều không muốn thua.

Yêu cầu: Cho một trạng thái trò chơi, hỏi trò chơi đó sẽ kết thúc như thế, (hoà, A thắng hay B thắng, ở đây A là người đi trước)

Input: Lgame.In

- Gồm 4 dòng mỗi dòng ghi 4 ký tự, ‘.’ thể hiện ô trống(có 6 ô), ‘x’ là ô chứa miếng hình tròn (2 ô), ‘#’ biều thị ô bị miếng hình L của người chơi A đặt lên (có 4 ô), còn lại bốn ô biều thị ô bị miếng hình L của người chơi

B đặt lên.

Output:Lgame.out

- Có ba trường hợp:

+ A thắng: ghi trạng thái sau khi A đi nước đi đầu tiên dẫn đến trạng thái đó.

+ A thua: ghi ra xâu “No winning move Losing”.

+ Hoà: ghi ra xâu “No winning Draw”.

Gợi ý: Có không quá 18 000 trạng thái, giải bằng Freepascal.

Bỏ xung:Đôi khi không phải lúc nào cũng có thể lưu được tất cả các trạng thái vì có một số bài toán có số trạng thái rất lớn. Vì vậy, thay vì tính trạng thái thắng thua hiện thời ta thay bằng trạng thái tương đương có cùng tính chất thắng thua.

Khái niệm: trạng thái A được gọi là tương đương với B khi và chỉ khi A và B có cùng thắng, cùng thua hoặc cùng hoà.

Để hiểu sâu hơn ta xét một bài toán cụ thể:

Stones (ACM)

Một trò chơi bốc sỏi diễn ra trên một bảng ngang kích thước $1 \times N$ ô vuông. Trên một số ô có đặt một số viên sỏi. Tại một bước đi người cầm một viên sỏi ở một ô và di chuyển viên sỏi sang bên trái một hoặc hai ô với điều kiện là ô di chuyển tới phải không có sỏi và đường di chuyển không được qua ô có sỏi. Người nào không di chuyển được sẽ là người thua cuộc. Cho trước trạng thái ban đầu hỏi người di trước có bao nhiêu nước đi đầu tiên mà người thứ luôn thua với giả thiết cả hai người đều chơi tối ưu.

Input: Stones.in

- Dòng đầu ghi số $N(1 \leq N \leq 50)$.

- Dòng thứ hai ghi một xâu gồm N ký tự thể hiện trạng thái lúc bắt đầu trò chơi, ‘.’ thể hiện ô trống, ‘X’ thể hiện có sỏi (số viên sỏi không vượt quá 10).

Output: Stones.out

- Ghi một số là số đi mà có thể thắng.

Nhận xét:- Nếu coi mỗi trạng thái là một đỉnh đồ thị rõ ràng bài toán theo lý thuyết có thể tính được kết quả cần tính. Nhưng trên thực tế số trạng thái rất lớn(có thể lên đến Tỷ hợp chap 10 của 50 phần tư). Như vậy

bài toán không thể lập trình được vì thiếu bộ nhớ và tốc độ tính toán rất chậm. - Vì vậy người ta đã nghĩ ra một cách giảm số lượng trạng thái đang xét xuống. Đầu tiên ta thấy trạng thái của người chơi được đặc trưng bởi tập có thứ tự ở đầu trước các ô tự do của mỗi viên sỏi Ví dụ: xâu "...XX.X" $\leftrightarrow \{3, 0, 1\}$. Nếu cứ để như vậy thì không giải quyết được và thay vì xét sự thắng thua của dãy đó ta xét sự thắng thua của dãy khi lấy đồng dư 3 của tất cả các phần tử trong dãy: $\{3, 0, 1\} \leftrightarrow \{0, 0, 1\}$, vì ta có thể chứng minh được hai dãy này là tương. **Chứng minh:**

Gọi dãy ban đầu là A, dãy sau khi giảm ước là B=f(A) (f là hàm rút gọn).

Vì B là dãy giảm ước của A nên với mọi B đi một nước đến B' thì A cũng đi một nước đến A' (cùng vị trí và số ô) sao cho f(A') = f(B'). (I)

Ví dụ: B{0,0,1} sau một nước đi vị trí 3 với số ô đi bằng 1 đến B'{0,0,0} thì A cũng đi tại 3 với số ô bằng 1 đến A'{3,0,0}. Lúc đó ta có: f(A') = f(B') = {0,0,0}.

Vì mọi bước chơi của đối thủ hỏng có lợi cho mình. Nếu người chơi thứ nhất thực hiện một nước đi từ A đến A' hỏng thay đổi sự thua \rightarrow thắng (vốn theo lý thuyết là xác định), tức f(A) thua, f(A') thua mà B = f(A), suy ra B không đi được đến B' (vì B=f(A) suy ra B thua, B' cũng thua) suy ra người chơi đã thực hiện trên một ô có số ô tự do ở đầu trước lớn hơn bằng 3, suy tiếp ra người thức hai có thể đi tiếp một nước trên cùng ô đấy với số ô bằng (3 - số ô người một đã đi). Suy ra người 1 vẫn ở vị trí f(A'') thua. (II)

(I)(II) \Rightarrow người chơi trạng thái cuối.

Thuật ngữ:

- Mỗi trạng thái chơi hay mỗi đỉnh của đồ thị là một số viết trong hệ cơ số 3, sau mỗi một bước đi thì chơi đến một trạng thái chơi khác, ta làm động tác rút gọn lấy modun 3 thì lại được một trạng thái khác được biểu diễn dưới dạng cơ số ba khác. Ta tính sự thắng thua trên đồ thị này. Ví dụ: {0, 0, 1} chỉ đi đến {0,0,0}; {1,2,1} nếu ta đi viên sỏi thứ hai sang trái hai ô ta đến trạng thái {1,0,3} $\leftrightarrow \{1,0,0\}$. (lưu ý: nếu biểu diễn theo này ta chỉ đi đến trạng thái có giá trị cơ số 3 nhỏ hơn, trong đó vị trí có giá trị lớn nhất nằm bên phải).

- Nếu một trạng thái chơi mà thắng khi chỉ khi trạng thái tương đương là thắng.

Chương trình mô tả

```

Var ketqua : array [0..59060] of byte;
Procedure Thang_thua (x : longint); {0<= x <=59049 = 3^10}
Var thang, i : byte;
a, b : array [0..10] of byte;
y : longint;
Begin
Doi_x_sang_co_so_3 (x, a);
/* x=16 => a[0]=3(số chữ số trong hệ cơ số 3 của x);
/*a[1] = 1, a[2]=2, a[3]=1;
For i := 1 to a[0] do
For ci:= 1 to 2 do
If (a[i] >= ci) then
Begin
Thuc_hien_buoc_di_o_vị_trí_i(i, ci, a, b);
/* có thể có tới hai cách, ci =1 hoặc 2
/* ví dụ: i=2, ci=2, a={3, 1, 2, 1}
/* b={3,1,0,3}
Rut_gon_b(b);
/*b={3,1,0,0}
Doi_b_sang_y(b);
/* đổi sang cơ số 10
/* y=9
If (ketqua[y] = 0) then

```

```

Begin
Ketqua[x] := 1;
Exit;
End;
End;
Ketqua[x] := 0;
End;

Procedure Chuong_trinh;
Var a, b : array [0..10] of byte;
Xau : string[50];
x : longint;
dem : byte;
Begin
Dem := 0;
Nhaph_N_va_xau( N, Xau);
Doi_xau_sang_co_so_3(Xau, a);
Vong lap: Di_cac_buoc_di_thu_(a, b)
Doi_b_sang_x (b, x);
If ketqua[x] = 0 then inc (dem);
Ket_thuc_vong_lap;
Print (dem);
End;

```

(Bài tập tự giải) đề thi thử ioicamp.com lần 2:

Trò chơi chuyển đá

Nguồn: Topcoder – Sưu tầm: Nguyễn Văn Hiếu

Vào một ngày đẹp trời, A nghĩ ra một trò chơi và rủ B cùng tham gia. Có n ô, mỗi ô chứa một số viên đá. Các ô được đánh số từ 0 đến n-1. Để thực hiện một nước đi, A/B chọn 3 ô với chỉ số i, j, k thoả mãn $i < j, j \leq k$ và ô i chứa ít nhất 1 viên đá, sau đó bỏ đi 1 viên đá ở ô i đồng thời thêm hai viên đá vào ô j và ô k (mỗi ô một viên). Chú ý là j có thể bằng k, và sau mỗi bước tổng số viên đá luôn tăng lên 1. Ai không thể thực hiện nước đi coi như bị thua. A đi trước. Nhiệm vụ của bạn là xác định xem A có thể chiến thắng hay không? (giả sử B chơi tối ưu). Nếu có thể hãy in ra 3 số i, j, k mô tả nước đi đầu tiên của A. Nếu có nhiều kết quả hãy in ra kết quả có i nhỏ nhất, nếu vẫn có hơn một kết quả chọn kết quả có j nhỏ nhất, nếu vẫn có hơn một kết quả chọn kết quả có k nhỏ nhất.

Input: STONES.INP

- Dòng đầu gồm số nguyên n là số ô.
- Dòng thứ hai gồm n số, số thứ i thể hiện số viên đá ở ô i.

Output: STONES.OUT

- Nếu A thắng thì in ra 3 số i, j, k trên một dòng duy nhất.
- Nếu A thua thì in ra một số -1 duy nhất.

Giới hạn:

- Kích thước:
 - + $1 \leq n \leq 15$
 - + Số viên đá ở một ô không vượt quá 1000.
- Thời gian: 1 s/test
- Bộ nhớ: 1 MB

Gợi ý: Trạng tương đương là trạng thái rút lấy modun cho 2, như vậy có nhiều nhất là 2^{15} trạng thái.

Tóm lại: Tư tưởng của lại trò chơi này rất đơn giản, bước đi tốt nhất của mình là bước đi dồn đối thủ đến tình trạng xấu nhất hay có lợi cho mình nhất: $F(v) = \max\{G(v) - F(u); u | (v,u) \in E\}$. Trong đó: $F(v)$ là giá trị tốt nhất tại đỉnh v của người đi từ đỉnh đây, $G(v)$ là giả trị của đỉnh v .