

HỒ SĨ ĐÀM (Chủ biên)
ĐỖ ĐỨC ĐÔNG - LÊ MINH HOÀNG - TRẦN ĐỖ HÙNG
NGUYỄN THANH HÙNG

TÀI LIỆU CHUYÊN TIN HỌC

BÀI TẬP

QUYỂN 3



NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

HỒ SĨ ĐÀM (Chủ biên)
ĐỖ ĐỨC ĐÔNG - LÊ MINH HOÀNG - TRẦN ĐỖ HÙNG
NGUYỄN THANH HÙNG

TÀI LIỆU CHUYÊN TIN HỌC
BÀI TẬP
QUYỂN 3

(Tái bản lần thứ nhất)

LỜI NÓI ĐẦU

Bộ sách *Tài liệu chuyên Tin học - Bài tập Quyển 1, 2, 3* được viết kèm với bộ *Tài liệu chuyên Tin học - Quyển 1, 2, 3* tương ứng đã được xuất bản. Các tác giả tham gia biên soạn bộ sách là những thầy giáo đã và đang dạy ở các trường chuyên, lớp chọn hoặc tham gia các khóa bồi dưỡng thi tin học quốc tế, bồi dưỡng giáo viên tin cho các trường chuyên theo chương trình của Bộ Giáo dục và Đào tạo, mong muốn xây dựng được các tài liệu có tính hệ thống phục vụ tốt các đối tượng thuộc lĩnh vực chuyên tin học.

Các cuốn *Tài liệu chuyên Tin học - Bài tập* đều có cấu trúc như nhau, gồm hai phần:

Phân I - Bài tập bao gồm tất cả các bài tập trong những chuyên đề của sách *Tài liệu chuyên Tin học* tương ứng và các bài tập bổ sung, được sắp xếp từ dễ đến khó, từ đơn giản đến phức tạp.

Phân II - Hướng dẫn giải bài tập có thể là những hướng dẫn chi tiết để giúp bạn đọc tìm được lời giải hoặc chỉ là đoạn chương trình chính giúp bạn đọc hiểu và tìm được lời giải hoặc chương trình hoàn chỉnh để tham khảo. Đối với một số bài tập thì có thể chỉ là đáp án hay hướng dẫn ngắn gọn.

Hai bộ sách *Tài liệu chuyên Tin học* và *Tài liệu chuyên Tin học - Bài tập* tạo thành hệ thống tài liệu khá hoàn chỉnh theo định hướng Chương trình các chuyên đề chuyên tin học đã được Bộ Giáo dục và Đào tạo ban hành. Do vậy cùng với bộ sách *Tài liệu chuyên Tin học*, bộ sách *Tài liệu chuyên Tin học - Bài tập* sẽ là tài liệu thiết thực phục vụ cho giáo viên, học sinh các trường chuyên, lớp chọn cả Trung học phổ thông và Trung học cơ sở. Ngoài ra, bộ sách còn là tài liệu tham khảo bổ ích cho việc tập huấn sinh viên các trường Đại học, Cao đẳng tham gia các kì thi Olympic Tin học Sinh viên Toàn quốc và Kì thi lập trình viên Quốc tế.

Lưu ý khi sử dụng bộ sách: Các bài tập trong bộ sách này được đánh số như trong sách lí thuyết; các bài tập bổ sung được để ở mục riêng và đánh số tiếp theo.

Mặc dù các tác giả và Ban biên tập đã cố gắng hoàn thiện nhưng chắc chắn bộ sách còn nhiều thiếu sót, các tác giả mong nhận được nhiều ý kiến đóng góp để sách sẽ hoàn thiện hơn, phục vụ bạn đọc được hiệu quả hơn. Các góp ý xin gửi về:

Ban Toán-Tin, Công ty cổ phần Dịch vụ xuất bản Giáo dục Hà Nội-
Nhà xuất bản Giáo dục Việt Nam, 187B, Giảng Võ, Hà Nội.

Các tác giả

đã được chấp nhận. Cố gắng và nỗ lực của các nhà khoa học Việt Nam trong việc nghiên cứu và ứng dụng công nghệ sinh học để cải thiện chất lượng nông nghiệp và nông sản là rất đáng khen ngợi. Tuy nhiên, cần phải có sự đổi mới và sáng tạo trong cách tiếp cận để đạt được kết quả tốt hơn.

Trong thời gian qua, Việt Nam đã có những bước tiến đáng kể trong việc áp dụng công nghệ sinh học vào nông nghiệp. Điều này đã mang lại hiệu quả rõ rệt, giúp nâng cao năng suất và chất lượng nông sản.

Tuy nhiên, vẫn còn một số vấn đề cần giải quyết để tiếp tục phát triển nông nghiệp theo hướng bền vững.

Đầu tiên, cần phải tăng cường đầu tư cho nghiên cứu và phát triển công nghệ sinh học. Điều này sẽ giúp tạo ra những sản phẩm nông nghiệp mới và chất lượng cao.

Thứ hai, cần phải tăng cường hợp tác между các nhà khoa học và nông dân. Điều này sẽ giúp đưa ra những giải pháp phù hợp với điều kiện thực tế của nông nghiệp Việt Nam.

Thứ ba, cần phải tăng cường quản lý và giám sát chất lượng nông sản. Điều này sẽ giúp đảm bảo rằng nông sản được sản xuất theo quy trình an toàn và chất lượng cao.

Thứ tư, cần phải tăng cường giáo dục và truyền thông về công nghệ sinh học. Điều này sẽ giúp nâng cao nhận thức và kỹ năng của nông dân.

Thứ năm, cần phải tăng cường hợp tác quốc tế. Điều này sẽ giúp trao đổi kinh nghiệm và công nghệ sinh học giữa các nước.

Thứ sáu, cần phải tăng cường quản lý và giám sát chất lượng nông sản. Điều này sẽ giúp đảm bảo rằng nông sản được sản xuất theo quy trình an toàn và chất lượng cao.

Thứ bảy, cần phải tăng cường giáo dục và truyền thông về công nghệ sinh học. Điều này sẽ giúp nâng cao nhận thức và kỹ năng của nông dân.

Thứ tám, cần phải tăng cường hợp tác quốc tế. Điều này sẽ giúp trao đổi kinh nghiệm và công nghệ sinh học giữa các nước.

8
T
t
I
x
th
C
p
V
C
C
8.
Tr
In
că
đô
O
ph
Ví
0
4
(*)
đượ

Phần I

Bài tập

CHUYÊN ĐỀ 8. HÌNH HỌC TÍNH TOÁN

8.1. Tìm độ dài đoạn thẳng nằm trong tam giác

Trên mặt phẳng cho tam giác ABC và đoạn thẳng DE. Tính độ dài của phần đoạn thẳng DE nằm trong tam giác.

Input: Tệp BAI01_C8.INP gồm hai dòng: Dòng thứ nhất có sáu số thực $x_A, y_A, x_B, y_B, x_C, y_C$ lần lượt từng cặp là toạ độ tương ứng của ba đỉnh A, B và C. Dòng thứ hai là bốn số thực x_D, y_D, x_E, y_E lần lượt từng cặp là toạ độ hai điểm D và E. (*)

Output: Tệp BAI01_C8.OUT chỉ có một dòng duy nhất ghi số thực là độ dài phần đoạn thẳng DE nằm trong tam giác ABC.

Ví dụ:

BAI01_C8.INP	BAI01_C8.OUT
0 2 2 4 4 2 0 2 3 3	3.16228

8.2. Tìm diện tích phần chung của hai tam giác

Trên mặt phẳng cho hai tam giác. Tính diện tích phần chung của hai tam giác đó.

Input: Tệp BAI02_C8.INP gồm hai dòng, mỗi dòng có sáu số thực lần lượt từng cặp là toạ độ tương ứng của ba đỉnh của một tam giác. Các toạ độ có giá trị tuyệt đối không vượt quá 10^3 .

Output: Tệp BAI02_C8.OUT chỉ có một dòng duy nhất ghi số thực là diện tích phần chung của hai tam giác này (chính xác đến 10^{-2}).

Ví dụ:

BAI02_C8.INP	BAI02_C8.OUT
0 6 8 6 4 0 4 8 8 2 0 2	16.00

(*) Nếu không chỉ định gì thêm thì trong sách này xin được hiểu rằng các số trên cùng dòng được ghi cách nhau ít nhất một dấu cách.

8.3. Tính diện tích phần phủ bởi N hình chữ nhật

Cho hai hình chữ nhật với các cạnh song song với các trục toạ độ. Mỗi hình chữ nhật xác định bởi toạ độ hai đỉnh đối. Tính diện tích phần mặt phẳng được phủ bởi hai hình chữ nhật và diện tích phần chung của chúng.

Từ bài toán này ta suy ra bài toán tổng quát tính diện tích phần phủ bởi N hình chữ nhật như sau:

Trên mặt phẳng toạ độ xOy , cho N hình chữ nhật có các cạnh song song với các trục toạ độ.

- a) Tính diện tích phần mặt phẳng được phủ bởi N hình chữ nhật này.
- b) Tính diện tích phần chung của N hình chữ nhật này.

Input: Tệp BAI03_C8.INP có dòng đầu ghi số nguyên N ($1 \leq N \leq 10000$). Trong N dòng tiếp theo, dòng thứ i có bốn số nguyên $x_{i1}, y_{i1}, x_{i2}, y_{i2}$ thể hiện $(x_{i1}; y_{i1})$ là toạ độ đỉnh trái-dưới và $(x_{i2}; y_{i2})$ là toạ độ đỉnh phải-trên của hình chữ nhật thứ i ($0 \leq x_{i1} < x_{i2} \leq 30000, 0 \leq y_{i1} < y_{i2} \leq 30000$).

Output: Tệp BAI03_C8.OUT có hai dòng tương ứng ghi kết quả câu a) và câu b); các số ghi từ đầu dòng.

Ví dụ:

BAI03_C8.INP	BAI03_C8.OUT
0 1 2 3	10
1 0 3 2	1.
1 0 3 4	

8.4. Tìm hai điểm xa nhau nhất trong N điểm

Trên mặt phẳng cho N điểm. Tìm hai điểm xa nhau nhất trong N điểm đó.

Gợi ý: Hai điểm cần tìm là hai đỉnh của đa giác bao lồi.

Input: Tệp BAI04_C8.INP có dòng đầu ghi số nguyên N ($1 \leq N \leq 10000$); trong N dòng tiếp theo, dòng thứ i có hai số nguyên x_i, y_i là toạ độ điểm thứ i ($0 \leq x_i, y_i \leq 30000$).

Output: Tệp BAI04_C8.OUT ghi khoảng cách hai điểm xa nhau nhất chính xác tới 10^{-4} .

Ví dụ:

BAI04_C8.INP	BAI04_C8.OUT
4	5.6569
0 0	
0 2	
3 0	
4 4	

8.5. Phân hoạch N điểm thành K tập có đường kính lớn nhất đạt min

Trên mặt phẳng cho N điểm. Hãy phân hoạch N điểm thành K tập S_1, S_2, \dots, S_K sao cho đường kính lớn nhất của K tập là nhỏ nhất, trong đó đường kính của tập S_i là khoảng cách lớn nhất giữa các cặp điểm thuộc S_i .

8.6. Trên mặt phẳng cho N điểm đôi một khác nhau. Hãy tìm hai trong N điểm sao cho đường thẳng đi qua hai điểm đó chia mặt phẳng thành hai phần mà số điểm thuộc hai nửa mặt phẳng chênh lệch nhau ít nhất.

Input: Tệp BAI06_C8.INP có dòng đầu ghi số nguyên N ($1 \leq N \leq 10000$). Trong N dòng tiếp theo, dòng thứ i có hai số nguyên x_i, y_i là tọa độ điểm thứ i ($0 \leq x_i, y_i \leq 30000$).

Output: Tệp BAI06_C8.OUT ghi tọa độ hai điểm tìm được, mỗi điểm trên một dòng.

Ví dụ:

BAI06_C8.INP	BAI06_C8.OUT
4	0 0
0 0	4 4
4 0	
0 4	
4 4	

8.7. Trên mặt phẳng cho N điểm. Hãy tìm tập ít nhất các đường thẳng sao cho mỗi điểm trong N điểm đã cho thuộc ít nhất một đường thẳng.

Input: Tệp BAI07_C8.INP có dòng đầu ghi số nguyên N ($1 \leq N \leq 10000$). Trong N dòng tiếp theo, dòng thứ i có hai số nguyên x_i, y_i là tọa độ điểm thứ i ($0 \leq x_i, y_i \leq 30000$).

Output: Tệp BAI07_C8.OUT có dòng đầu tiên ghi số k là số đường thẳng ít nhất thỏa mãn yêu cầu đề bài. Tiếp theo là k dòng, mỗi dòng ghi số hiệu các điểm cùng thuộc một đường thẳng.

Ví dụ:

BAI07_C8.INP	BAI07_C8.OUT
4	2
0 0	1 2
4 0	3 4
0 4	
4 4	

8.8. Nối N điểm thành mạng liên thông

Trên mặt phẳng cho N điểm. Hãy nối N điểm thành một mạng liên thông bằng các đoạn thẳng nối các cặp điểm sao cho tổng độ dài các đoạn thẳng được nối là nhỏ nhất.

8.9. Trên mặt phẳng cho N hình chữ nhật có các cạnh song song với hai trục tọa độ. Hình thứ i xác định bởi hai đỉnh đối $(x_i; y_i)$ và $(z_i; t_i)$. Toạ độ các điểm là nguyên, có giá trị tuyệt đối không vượt quá 10^4 . Khi tô các hình chữ nhật này bằng màu xanh, ta nhận được một số phần mặt phẳng được tô xanh. Tính tổng độ dài các đường bao quanh các vùng màu xanh.

Input: Tệp BAI09_C8.INP.

Output: Tệp BAI09_C8.OUT.

Ví dụ:

BAI09_C8.INP	BAI09_C8.OUT
7 0 -2 5 0 -2 -1 1 3 5 1 6 6 0 2 2 5 1 4 5 6 3 0 4 5 1 1 5 3	48

8.10. Tầm nhìn

Trên mặt phẳng cho N đoạn thẳng ($1 \leq N \leq 1000$). Toạ độ các đầu mút của các đoạn thẳng là các số nguyên không âm không vượt quá 20000. Các đường thẳng thu được bằng cách kéo dài các đoạn thẳng đã cho luôn cắt hai trục tọa độ và hai giao điểm cùng gốc tọa độ tạo thành một tam giác vuông cân. Không có hai đoạn thẳng nào giao nhau.

Ta nói một đoạn thẳng là nhìn thấy được từ gốc tọa độ O, nếu tìm ra điểm X trên nó sao cho đoạn thẳng OX không cắt bất cứ đoạn nào trong số các đoạn thẳng đã cho.

Hãy viết chương trình đếm số đoạn thẳng nhìn thấy từ gốc tọa độ.

Input: Tệp BAI10_C8.INP. Dòng đầu tiên chứa số đoạn thẳng N. Mỗi một trong số N dòng tiếp theo chứa bốn số nguyên không âm X_1, Y_1, X_2, Y_2 , phân cách bởi dấu cách, trong đó $(X_1; Y_1)$ là toạ độ của đầu mút thứ nhất còn $(X_2; Y_2)$ là toạ độ của đầu mút thứ hai của đoạn thẳng tương ứng.

Output: Tệp BAI10_C8.OUT ghi số đoạn thẳng nhìn thấy được từ gốc toạ độ.

Ví dụ:

BAI10_C8.INP	BAI10_C8.OUT
4 3 13 11 5 14 1 10 5 10 14 20 4 5 6 10 1	3

8.11. Khôi phục đa giác

Bờm vẽ trên mặt phẳng một hình đa giác tổng quát (đường gấp khúc khép kín không tự cắt) với các cạnh song song với các trục toạ độ và các đỉnh có toạ độ nguyên. Sau đó, vô ý Bờm đã xoá mất tất cả các cạnh thẳng đứng (cạnh song song với trục tung) của đa giác. Bạn hãy tìm cách giúp Bờm tính diện tích của đa giác đã vẽ ban đầu từ những thông tin còn lại.

Input: Tệp văn bản BAI11_C8.INP. Dòng đầu tiên chứa N là số cạnh nằm ngang (cạnh song song với trục hoành) của đa giác đã cho ($N \leq 1000$; Mỗi dòng trong số N dòng tiếp theo chứa thông tin về một cạnh nằm ngang của đa giác bao gồm bốn số nguyên x, y, u, v được ghi cách nhau bởi dấu cách, trong đó (x; y) và (u; v) là hai cặp toạ độ của hai đầu mút của cạnh nằm ngang. Giả thiết rằng các toạ độ là các số nguyên có giá trị tuyệt đối không vượt quá 100.

Output: Tệp văn bản BAI11_C8.OUT. Dòng đầu tiên ghi diện tích đa giác. Dòng thứ i trong số $2*N$ dòng tiếp theo chứa toạ độ đỉnh thứ i của đa giác được liệt kê theo thứ tự đi vòng quanh đa giác theo chiều kim đồng hồ (đỉnh bắt đầu được chọn tùy ý).

Ví dụ:

BAI11_C8.INP	BAI11_C8.OUT
2 1 1 3 1 1 3 3 3	4 1 1 1 3 3 3 3 1

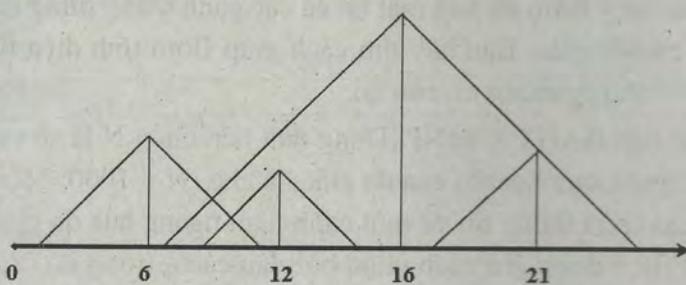
8.12. Đỉnh núi

Đỉnh núi có thể nhìn thấy được ở chân trời chỉ khi nó không bị che khuất bởi một ngọn núi khác hay là đường viền của nó không bị ẩn trên nền một ngọn núi khác. Cho biết tất cả các dốc núi đều có góc nghiêng là 45° và cho biết toạ độ và độ cao đỉnh của tất cả các ngọn núi. Xác định số lượng đỉnh núi nhìn thấy được.

Input: Tệp văn bản BAI12_C8.INP: Dòng đầu tiên chứa số nguyên N ($1 \leq N \leq 10000$). Mỗi dòng trong N dòng tiếp theo chứa hai số nguyên x, y được ghi cách nhau bởi dấu cách, lần lượt là toạ độ và độ cao của đỉnh núi ($0 \leq x \leq 30000$, $0 \leq y \leq 10000$).

Output: Tệp văn bản BAI12_C8.OUT ghi số lượng đỉnh núi nhìn thấy được.

BAI12_C8.INP	BAI12_C8.OUT
4 6 5 12 3 16 9 21 4	2



8.13. Đài phun nước

Để làm đẹp cảnh quan trụ sở, ban Giám đốc một công ty quyết định xây dựng ở sân tiền sảnh trụ sở công ty một đài phun nước. Đài phun nước có dạng một hình tròn với kích thước lớn nhất có thể được. Nhà thiết kế được biết là sân tiền sảnh của công ty có dạng hình chữ nhật kích thước $X \times Y$. Tuy nhiên, khi lựa chọn vị trí cho đài phun nước, nhà thiết kế gặp phải một vấn đề phức tạp: Trong sân tiền sảnh có N cột hình trụ tròn xoay không được phép di chuyển. Vì vậy vấn đề đặt ra cho nhà thiết kế là cần đặt đài phun nước ở vị trí nào để nó có bán kính lớn nhất có thể được, đồng thời không được có diện tích chung khác 0 với các cột. Bạn hãy lập trình giúp nhà thiết kế giải quyết vấn đề trên.

Input: Tệp văn bản BAI13_C8.INP:

- Dòng đầu tiên chứa hai số thực X, Y ($1 \leq X, Y \leq 10^4$). Giả thiết rằng sân tiền sảnh là hình chữ nhật có toạ độ các đỉnh là (0; 0), (X; 0), (X; Y) và (0; Y).
- Dòng thứ hai chứa số nguyên N ($0 \leq N \leq 10$) là số lượng cột trong sân tiền sảnh.

- Dòng thứ i trong số N dòng tiếp theo chứa ba số thực X_i , Y_i và R_i cho biết toạ độ tâm và bán kính của cột thứ i ($R_i \leq X_i \leq X - R_i$, $R_i \leq Y_i \leq Y - R_i$, $0.1 \leq R_i \leq \min\left(\frac{X}{2}, \frac{Y}{2}\right)$, $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \geq R_i + R_j$ với mọi $i \neq j$).

Các số trên một dòng trong tệp dữ liệu được ghi cách nhau bởi dấu cách.

Output: Tệp văn bản BAI13_C8.OUT ghi ba số thực X_F , Y_F , R_F là toạ độ và bán kính của đài phun nước.

Chú ý: Đài phun nước phải được đặt trong sân, được phép tiếp xúc với tường của sân hoặc cột nhưng không được có diện tích chung khác 0 với các cột. Nếu có nhiều vị trí cùng cho bán kính lớn nhất chỉ cần đưa ra một trong số chúng.

Ví dụ:

BAI13_C8.INP	BAI13_C8.OUT
10 20 0	5.000 5.000 5.000
20 20 4 2 2 2 18 2 2 2 18 2 18 18 2	10.000 10.000 9.314
20 20 4 2 2 2 18 2 2 3 17 2 16 16 4	9.510 7.054 7.053

BÀI TẬP BỔ SUNG

8.14. Tính diện tích phần phủ của N tam giác

Trên mặt phẳng toạ độ xOy , cho N tam giác. Tính diện tích phần mặt phẳng được phủ bởi N tam giác này.

Input: Tệp BAI14_C8.INP có dòng đầu tiên ghi số nguyên dương N ($2 \leq N \leq 10$). Tiếp theo là N dòng, mỗi dòng có sáu số thực lần lượt từng cặp là toạ độ tương ứng với ba đỉnh của một tam giác. Các toạ độ có giá trị tuyệt đối không vượt quá 10^3 .

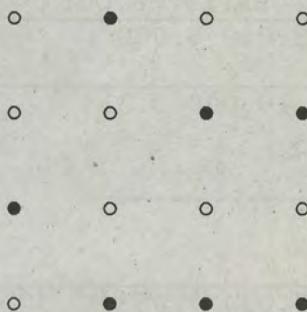
Output: Tệp BAI14_C8.OUT chỉ có một dòng duy nhất ghi số thực là diện tích được phủ bởi N tam giác này (chính xác đến 10^{-2}).

Ví dụ:

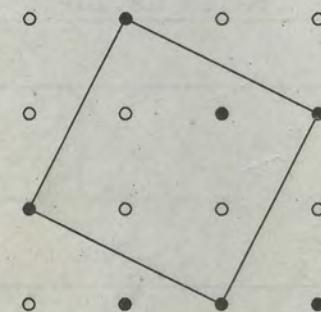
BAI14_C8.INP	BAI14_C8.OUT
2 0 2 2 4 4 2 0 3 2 5 4 3	7.00

8.15. Tính số hình vuông

Xét tổng thể một khu đất hình vuông A có kích thước $N \times N$ (đơn vị) được chia thành lưới ô vuông đơn vị bởi nhiều hàng rào. Giả sử các cột mốc của hàng rào được sơn màu trắng hoặc đen. Yêu cầu xác định số lượng các hình vuông trong khu A có các đỉnh là các cột mốc cùng màu. Ví dụ, khu đất 4×4 như trong hình 1 chỉ có một hình vuông như được chỉ ra trong hình 2.



Hình 1



Hình 2

Input: Tệp BAI15_C8.INP. Dòng đầu tiên chứa số N là cạnh của khu đất A ($1 \leq N \leq 50$). N dòng sau, mỗi dòng chứa N số thuộc tập hợp $\{0, 1\}$, thể hiện các cột mốc hàng rào trong khu đất. Nếu điểm có tọa độ (i, j) là cột màu trắng thì số thứ j của dòng i bằng 0, còn nếu là cột mốc đen thì bằng 1.

Output: Tệp BAI15_C8.OUT, ghi số lượng hình vuông tìm được.

Ví dụ:

BAI15_C8.INP	BAI15_C8.OUT
4 0 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1	1

8.16. Chia đất

Một người cha, khi mất đi, để lại một mảnh đất có hình dạng một đa giác lồi làm của thừa kế cho hai người con của mình. Trong di chúc ông yêu cầu rằng hai

người con phải chia mảnh đất này thành hai phần có diện tích bằng nhau theo một đường ranh giới thẳng dọc theo phương Nam-Bắc. Bạn là người được giao giúp hai người con thực hiện bản di chúc này. Hãy viết chương trình tìm cách chia.

Giả sử mảnh đất là đa giác lồi với các đỉnh là $A_1A_2\dots A_n$ nằm trên mặt phẳng toạ độ có trục Oy nằm theo hướng Nam-Bắc, trục Ox theo hướng Tây-Đông.

Input: Tệp BAI16_C8.INP: Dòng đầu tiên ghi N là số đỉnh của đa giác ($N \leq 5000$). Trong N dòng tiếp theo, dòng thứ i ghi hai số nguyên x_i, y_i lần lượt là hoành độ và tung độ của điểm thứ i trong số N đỉnh đa giác (các đỉnh của đa giác được liệt kê theo chiều xuôi hoặc ngược kim đồng hồ).

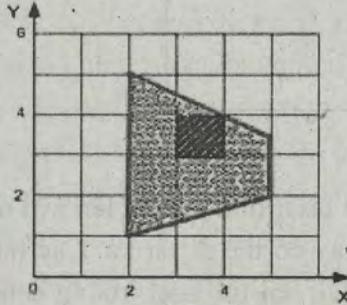
Output: Tệp văn bản BAI16_C8.OUT ghi một số thực x_0 với ý nghĩa đường ranh giới dùng để chia đất là đường thẳng $x = x_0$ (x_0 viết với bốn chữ số phần thập phân).

Ví dụ:

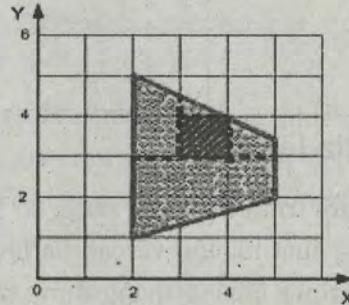
BAI16_C8.INP	BAI16_C8.OUT
4	1.0000
0 0	
2 0	
2 2	
0 2	

8.17. Khuôn thép (Đề thi học sinh giỏi 2005- Bảng A)

Để chuẩn bị cho lễ hội kỉ niệm 30 năm ngày Chiến dịch Hồ Chí Minh toàn thắng, giải phóng miền Nam, thống nhất đất nước, người ta cần gia công các loại khuôn thép có dạng đa giác lồi M đỉnh. Mỗi khuôn thép được thiết kế trên một tấm thép cũng có dạng đa giác lồi N đỉnh, không có cạnh nào của khuôn thép nằm gọn trên một cạnh của tấm thép. Để tiện cho việc gia công, khuôn thép được vẽ sao cho hai đường thẳng chứa hai cạnh không kề nhau của nó không cắt nhau ở bên trong tấm thép.



Hình 1



Hình 2

Công việc chính cần làm trong quá trình gia công là sử dụng máy cắt để cắt được khuôn thép từ tấm thép ra. Rõ ràng là cần phải thực hiện M nhát cắt. Mỗi nhát cắt được thực hiện bằng cách chọn một cạnh nào đó của khuôn thép và cắt theo đường thẳng chứa cạnh ấy chia tấm thép thành hai phần, một phần chứa khuôn thép cần gia công. Chi phí cắt khuôn thép là tổng chiều dài của các đường cắt.

Trên hình 1 và 2, tấm thép là tứ giác được tô nhạt, khuôn thép là hình vuông được tô bằng các gạch đậm. Các nét gạch đứt là các đường cắt với tổng chi phí bằng 6.5 đơn vị.

Yêu cầu: Cho biết hình dạng tấm thép và khuôn thép cần gia công. Hãy tìm phương án cắt khuôn thép có chi phí nhỏ nhất.

Input: Tệp văn bản BAI17_C8.INP: Dòng đầu ghi số N ($3 \leq N \leq 2000$) là số đỉnh của tấm thép; N dòng tiếp theo, mỗi dòng ghi hai số thực x và y ($-10^4 < x, y < 10^4$), là toạ độ N đỉnh của tấm thép được liệt kê theo chiều kim đồng hồ. Dòng tiếp theo ghi số M ($3 \leq M \leq 2000$) là số đỉnh của khuôn thép. M dòng tiếp theo, mỗi dòng ghi hai số thực x và y ($-10^4 < x, y < 10^4$) là toạ độ M đỉnh của khuôn thép được liệt kê theo chiều kim đồng hồ. Các số trên một dòng cách nhau ít nhất một dấu cách.

Output: Tệp văn bản BAI17_C8.OUT ghi chi phí nhỏ nhất tìm được với độ chính xác tới 4 chữ số sau dấu chấm thập phân.

Ví dụ:

BAI17_C8.INP	BAI17_C8.OUT
4	6.5000
2 1	
2 5	
5 3.5	
5 2	
4	
3 3	
3 4	
4 4	
4 3	

8.18. Các tia laser

Trong một buổi trình diễn ánh sáng, có N tia laser được chiếu lên trời nhờ các đèn chiếu có công suất rất lớn và các tia laser này có thể đi rất xa. Các tia laser nằm trên cùng một mặt phẳng thẳng đứng nên nếu hai tia laser không song song với nhau thì sẽ cắt nhau. Bạn tổ chức buổi trình diễn muốn nhờ bạn cho biết với các

tia laser được chiếu như trong kế hoạch thì hai tia laser nào cắt nhau tại điểm cao nhất.

Các tia laser được biểu diễn bằng ba số nguyên: Một số là toạ độ x ở dưới đất của ngọn đèn chiếu, hai số còn lại là toạ độ của một điểm nào đó thuộc tia laser này. Biết rằng không có hai tia laser nào song song và cũng không có tia laser nào trùng với mặt đất (mặt đất được coi là đường thẳng $y = 0$).

Input: Tệp văn bản BAI18_C8.INP:

Dòng đầu tiên ghi N là số tia laser ($2 \leq N \leq 10000$). Tiếp theo là n dòng, mỗi dòng ghi ba số nguyên x_i, z_i, t_i với ý nghĩa tia laser thứ i đi qua hai điểm $(x_i, 0)$ và (z_i, t_i) ($|x_i|, |z_i| \leq 10^6; 0 < t_i \leq 10^6$) ($i = 1, 2, \dots, N$).

Output: Tệp văn bản BAI18_C8.OUT như sau: Trong trường hợp không có hai tia laser nào cắt nhau thì ghi duy nhất một số -1 . Nếu tồn tại hai tia laser cắt nhau thì dòng thứ nhất ghi số độ cao lớn nhất (được ghi với độ chính xác bốn chữ số sau dấu thập phân) của giao điểm của hai tia laser cắt nhau. Dòng thứ hai ghi hai số nguyên là thứ tự (theo thứ tự trong tệp dữ liệu vào) của hai tia laser cắt nhau có độ cao lớn nhất đó.

Ví dụ:

BAI18_C8.INP	BAI18_C8.OUT	BAI18_C8.INP	BAI18_C8.OUT
<pre>2 -1 -2 4 2 3 4</pre>	-1	<pre>4 -1 -2 4 0 1 1 3 0 4 4 4 1</pre>	<pre>8.0 1 3</pre>

8.19. Cho N đa giác lồi thỏa mãn các tính chất sau: Với hai đa giác bất kì luôn có một đa giác mà mọi điểm của nó nằm trong đa giác kia; các cạnh của chúng không có điểm chung.

Bài toán đặt ra là: Với mỗi đa giác i , có bao nhiêu đa giác trong N đa giác nói trên bao nó (đa giác i nằm trong bao nhiêu đa giác)?

Input: Tệp văn bản BAI19_C8.INP: Dòng đầu tiên ghi số tự nhiên N ($3 \leq N \leq 10000$). Dòng thứ $i + 1$ ghi thông tin về đa giác thứ i ; bao gồm S_i là số đỉnh của đa giác; S_i cặp số nguyên tiếp theo lần lượt là hoành độ và tung độ các đỉnh của đa giác. Các số trên cùng dòng cách nhau ít nhất một dấu cách.

Output: Tệp văn bản BAI19_C8.OUT gồm N dòng, dòng thứ i ghi số lượng các đa giác bao đa giác i .

Ví dụ:

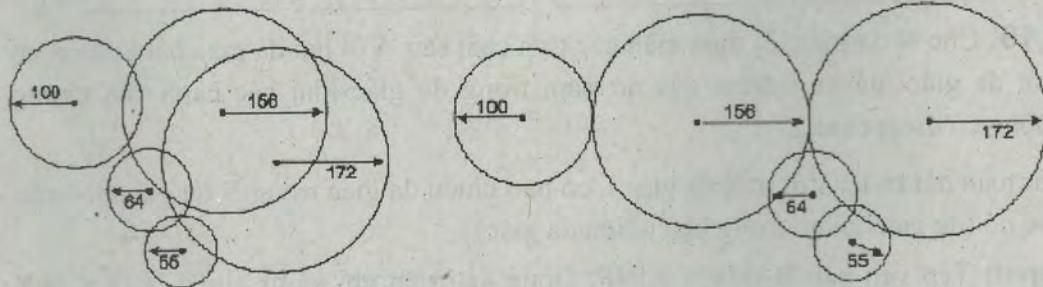
BAI19_C8.INP	BAI19_C8.OUT
4	0
4 1 1 15 1 15 8 1 8	2
4 9 3 9 6 4 6 4 3	1
4 3 2 11 2 11 7 3 7	3
3 8 4 8 5 6 8	

8.20. Bộ vòng ảo thuật

Bộ vòng Trung Quốc là một dụng cụ ảo thuật cổ điển. Nhà ảo thuật giới thiệu cho khán giả xem xét một số vòng thép đặc lồng vào nhau tưởng chừng không thể tháo rời chúng ra. Sau đó, nhà ảo thuật liên tiếp nới và tháo các vòng trước sự ngạc nhiên của khán giả. Những cái vòng đặc xuất hiện có vẻ như từ từ lồng xuyên qua lẫn nhau. Cao điểm của trò ảo thuật là diễn viên tách rời tất cả các vòng thành từng chiếc một.

Trong bài toán này, bạn được cho một tập hợp những vòng tròn. Hai vòng cắt nhau được coi là liên kết lồng vào nhau. Chương trình của bạn cần xác định khoảng cách tối đa có thể đạt được bằng cách kéo hai vòng nào đó theo hai hướng ngược nhau. Do bạn không phải là một nhà ảo thuật nên các vòng vẫn còn liên kết với nhau và giả định rằng tất cả các vòng vẫn tiếp tục là những vòng tròn.

Với những cái vòng được cho như hình 1 có thể được kéo căng theo mô tả trên được khoảng cách tối đa là 856 (hình 2).



Input: Tệp BAI20_C8.INP ghi số N (số vòng liên kết với nhau). Mỗi dòng trong N dòng sau là ba số nguyên mô tả một vòng. Hai số đầu trong ba số này là toạ độ tâm vòng tròn, số thứ ba là bán kính của nó. Các số cách nhau ít nhất một dấu cách.

Biết rằng sẽ có không quá 500 vòng và tất cả toạ độ và bán kính trong phạm vi 1 đến 1000. Dữ liệu vào bảo đảm không có vòng nào nằm trong vòng khác. Hơn nữa, bảo đảm rằng toàn bộ các vòng là nối với nhau. Khoảng cách tối đa được hạn chế là số nguyên.

Output: Tập BAI20_C8.OUT ghi khoảng cách tối đa đạt được.

Ví dụ:

BAI20_C8.IN	BAI20_C8.OUT
5	856
680 417 64	
872 374 172	
727 511 55	
797 297 156	
569 282 100	

CHUYÊN ĐỀ 9. LÍ THUYẾT TRÒ CHƠI

9.1. Trò chơi Lấy bớt quân cờ

“Trên bàn có C quân cờ. Hai người chơi lần lượt, mỗi lần người chơi đến lượt sẽ lấy ra khỏi bàn từ 1 đến M quân cờ. Người thắng là người lấy được quân cờ cuối cùng”. Viết chương trình thực hiện trò chơi “Lấy bớt quân cờ” nhưng với quy định người có lượt lấy cuối cùng thì thua. Chương trình nhập hai số dương C và M từ bàn phím ($1 \leq C \leq 10^4$, $1 \leq M \leq C$) và tạo giao diện trên màn hình quá trình người chơi với máy (ưu tiên cho người đi trước) khi máy áp dụng thuật toán giành thắng. Cuối cùng thông báo người thắng.

9.2. Trò chơi trừ dần

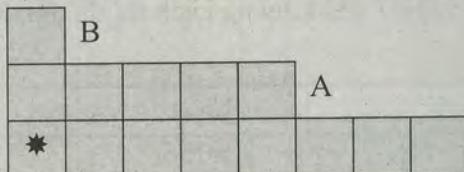
Tìm tập hợp các vị trí P đối với trò chơi trừ dần, nếu tập trừ là:

- a) $S = \{1, 3, 5, 7\}$.
- b) $S = \{1, 3, 6\}$.
- c) $S = \{1, 2, 4, 8, 16, \dots\}$ là tập các luỹ thừa của 2.

9.3. Trò chơi Chomp! (Bẻ miếng sô-cô-la)

Cần di chuyển những ô vuông trong bảng hình chữ nhật gồm $m \times n$ ô vuông. Mỗi phép di chuyển là lấy đi một ô vuông và các ô vuông ở bên phải nó cùng các ô vuông phía trên chúng. Hai người chơi lần lượt di chuyển, người nào chọn phải ô

(1, 1) thì thua cuộc (ô này coi như bị nhiễm độc). Ví dụ chơi với bảng 3×8 . Người A lấy đi ô (6, 2) cùng các ô bên phải và các ô phía trên (lấy đi một hình chữ nhật có tất cả sáu ô). Sau đó người B lấy ô (2, 3) và các ô bên phải (lấy đi hình chữ nhật có tất cả bốn ô). Khi đó các ô vuông còn lại trên bảng như hình dưới:



- a) Tìm ra một phép chuyển tiếp theo giành chiến thắng cho A.
- b) Có thể chứng minh A có thuật toán giành thắng trong bảng tổng quát $m \times n$ ô vuông hay không?

9.4. Trò chơi Phép trừ động

Có thể mở rộng thêm lớp trò chơi trừ dần bằng cách tạo ra tập trừ S phụ thuộc vào lần di chuyển trước của đối thủ.

Có một cọc N thẻ, người chơi đầu tiên có thể lấy bao nhiêu thẻ tùy ý nhưng ít nhất là một thẻ và không được lấy tất cả. Sau đó, đến lượt người thứ hai và hai người thay nhau lần lượt chơi. Mỗi người chơi không được phép lấy số thẻ nhiều hơn số thẻ mà đối thủ vừa lấy.

- a) Người thứ nhất di chuyển như thế nào để thắng nếu số thẻ $N = 44$?
- b) Với giá trị nào của N thì người chơi sau thắng?

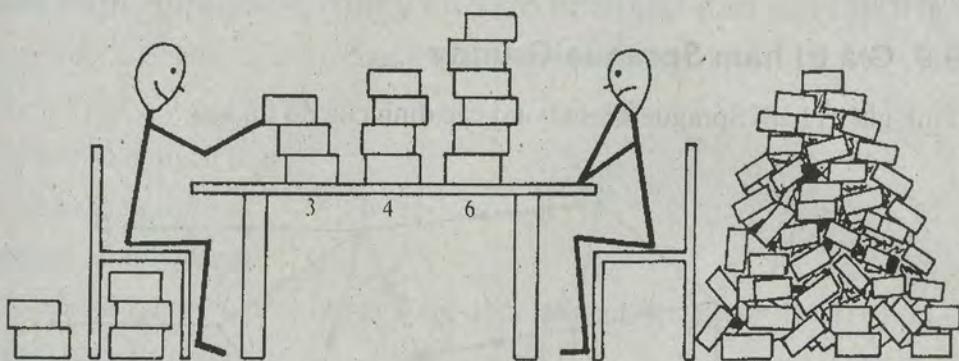
9.5. Trò chơi Fibonacci Nim

Có một cọc N thẻ ($1 < N \leq 100$), người chơi đầu tiên có thể lấy bao nhiêu thẻ tùy ý nhưng ít nhất là một thẻ và không được lấy tất cả. Hai người chơi lần lượt, mỗi người chơi có thể lấy số thẻ nhiều nhất bằng hai lần số thẻ mà đối thủ lấy trong lượt đi trước. Lập trình trò chơi giữa người và máy, áp dụng chiến thuật giành thắng cho máy, biết máy đi trước.

9.6. Trò chơi Porker Nim

Đây là trò chơi với các cọc là các ch่อง hộp gỗ. Cũng như trò Nim chuẩn, ngoại trừ một điều là có thể giảm kích thước của một vài cọc và phép di chuyển là có thể giảm hoặc tăng thêm kích thước của một vài cọc bằng các hộp gỗ đã chiếm

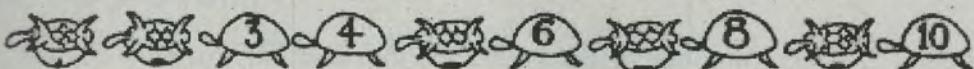
được trong các bước di chuyển trước. Trong hình vẽ dưới đây là trò chơi ở thời điểm ba chồng hộp gỗ có kích thước là 3, 4 và 6 sau khi hai người chơi đã di chuyển một số bước và thu được một số hộp (để bên cạnh họ). Bây giờ đến lượt đấu thủ bên trái (gọi là A). Vậy chiến thuật của A như thế nào để thắng (nếu trò chơi không cho phép kéo dài mãi).



9.7. Trò chơi Lật rùa (Turning Turtles)

Trên đường thẳng nằm ngang có n con rùa đang bò hoặc đang nằm ngửa. Một phép chuyển là lật một con rùa đang bò thành nằm ngửa và nếu muốn còn có thể lật một con rùa khác ở bên trái nó (từ bò thành nằm ngửa hoặc từ nằm ngửa thành bò). Người chơi nào có phép di chuyển cuối cùng thì thắng.

- a) Chúng tôi rằng trò chơi này là Nim nếu mỗi con rùa đang bò ở vị trí n là một cọc có n quân.
- b) Nếu a) đúng, tìm chiến thuật giành thắng cho trạng thái nêu ở hình vẽ sau:



9.8. Trò chơi Moore's Nim_k

Có n cọc với số quân tương ứng là x_1, x_2, \dots, x_n và quá trình chơi như trong Nim nhưng một phép chuyển là người chơi có thể chuyển số quân tùy ý từ k cọc bất kì (k là số cố định cho trước) và ít nhất phải có một quân chuyển đi từ một cọc nào đó. Trò chơi Nim_k với $k = 1$ chính là trò chơi Nim chuẩn với n cọc.

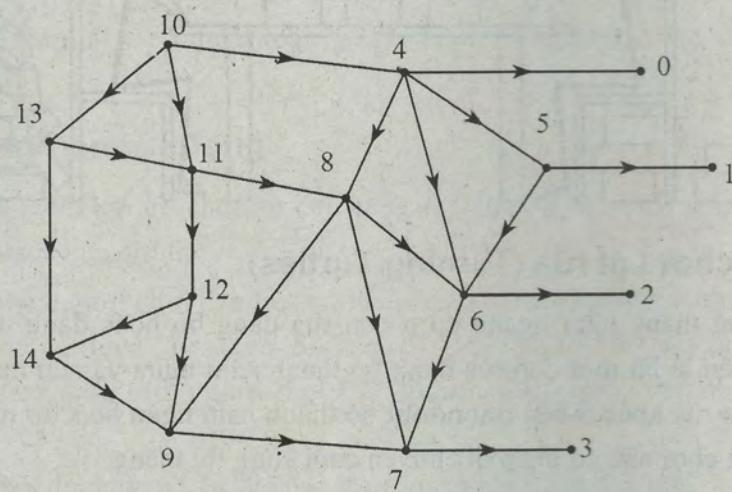
Hãy lập trình trò chơi Nim_k giữa máy và người, cho máy đi trước.

Input: Tập BAI08_C9.INP: mỗi dòng cho một cấu hình của trò chơi: đầu dòng là số k nguyên dương sau đó là n số nguyên dương thể hiện số quân trên n cọc ($1 \leq k \leq n$).

Output: Tập BAI08_C9.OUT có số dòng tương ứng. Đầu mỗi dòng ghi số 1 hoặc 0 thể hiện máy có thể đạt chiến thắng hoặc không chắc thắng. Nếu ghi số 1 thì sau đó ghi ra n số thể hiện số quân còn lại trên các cọc 1, 2, ..., n sau bước đi thứ nhất của máy.

9.9. Giá trị hàm Sprague-Grundy

Tính giá trị hàm Sprague-Grundy tại các đỉnh của đồ thị sau:



Input: Tập BAI09_C9.INP mô tả đồ thị trên theo dạng sau:

BAI09_C9.INP
15
4 4 0 5 6 8
5 2 1 6
6 2 2 7
7 1 3
8 3 6 7 9
9 1 7
10 3 4 11 13
11 2 8 12
12 2 9 14
13 2 11 14
14 1 9

Dòng đầu ghi số đỉnh $N = 15$. Mỗi dòng sau: số thứ nhất là số hiệu đỉnh x , số thứ hai là số n_x đó là số lượng các đỉnh y có cung từ đỉnh x tới đỉnh y , n_x số tiếp theo là số hiệu các đỉnh y .

Output: Tệp BAI09_C9.OUT, gồm N dòng, mỗi dòng ghi hai số x và $g(x)$ tương ứng là số hiệu đỉnh và giá trị hàm Sprague-Grundy của đỉnh đó.

9.10. Tìm hàm Sprague-Grundy cho trò chơi trừ dần với tập trừ

Tìm hàm Sprague-Grundy cho trò chơi trừ dần với tập trừ sau đây :

- a) $S = \{1, 3, 4\}$.
- b) S là tập các số nguyên tố lẻ.
- c) S là tập các số nguyên tố.
- d) S là tập các số Fibonacci.
- e) S là tập các số Lucas (tập số Lucas được định nghĩa đệ quy: $L_1 = 1$, $L_2 = 3$, $L_{i+2} = L_{i+1} + L_i$ ($i = 1, 2, \dots, 100$).

9.11. Trò chơi At-Most-Haft

Trò chơi At-Most-Haft là trò chơi trên một cọc với quy luật: phải chuyển ít nhất là một quân và nhiều nhất là nửa số quân trên cọc (tất nhiên vị trí kết thúc là 0 hoặc 1). Viết chương trình xây dựng hàm Sprague-Grundy với trò chơi này.

Input: Tệp BAI11_C9.INP gồm 10 dòng, mỗi dòng ghi một giá trị của N ($1 \leq N \leq 10000$).

Output: Tệp BAI11_C9.OUT gồm 10 dòng, mỗi dòng là giá trị hàm Sprague-Grundy của N tương ứng với dòng input.

9.12. Trò chơi Dim⁺

Trò chơi Dim⁺ là trò chơi trên một cọc chứa n quân, hai người chơi lần lượt. Mỗi lượt chơi loại bỏ đi c quân từ cọc với điều kiện c là ước của n (kể cả 1 và n). Tính hàm Sprague-Grundy của các vị trí trong trò chơi này với $n = 20$ và ghi kết quả vào tệp BAI12_C9.OUT.

9.13. Trò chơi Chắn-Lê

Cho một cọc chứa n quân. Quy tắc chuyển quân khỏi cọc là: có thể bớt đi một số chắn quân nhưng không phải là toàn bộ quân trên cọc hoặc có thể bớt đi toàn bộ quân trên cọc nếu cọc có số quân lẻ.

Chứng minh quy nạp công thức: $g(2k) = k - 1$; $g(2k - 1) = k$ với mọi $k \geq 1$.

9.14. Trò chơi At-Least-Haft

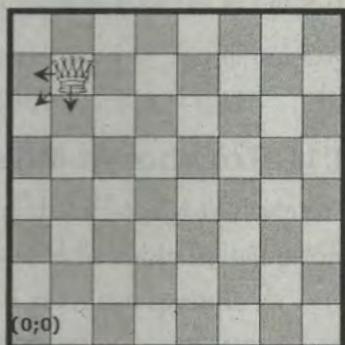
Trò chơi chỉ có một cọc với quy luật chơi là phải di chuyển ít nhất là một nửa số thẻ trên cọc. Vị trí kết thúc duy nhất là 0.

Chứng minh (quy nạp) giá trị hàm Sprague-Grundy $g(x) = \min\{k: 2^k > x\}$.

Lưu ý: Người thứ nhất trong trò chơi này dành chiến thắng rất dễ dàng (đó là lấy hết mọi quân ngay từ phép di chuyển đầu tiên). Nhưng việc tính hàm Sprague-Grundy có ý nghĩa phục vụ cho trò chơi chính mà At-Least-Haft là một trò chơi thành phần.

9.15. Trò chơi Wythoff

Trên bàn cờ vua kích thước 8×8 đặt một quân hậu. Hai người lần lượt đi, mỗi lượt đi sẽ di chuyển quân hậu với số bước tùy ý theo hướng thẳng xuông, ngang sang trái, hoặc đi chéo xuông góc trái-dưới. Khi con hậu đi đến góc trái-dưới thì trò chơi kết thúc (đỉnh kết thúc). Viết chương trình tính giá trị hàm Sprague-Grundy của các vị trí trong trò chơi ghi vào tệp BAI15_C9.OUT theo khuôn dạng: ghi 8 dòng, mỗi dòng 8 số.



Dòng thứ nhất lần lượt là các giá trị $g(7; 0), g(7; 1), \dots, g(7, 7)$.

Dòng thứ nhì lần lượt là các giá trị $g(6; 0), g(6; 1), \dots, g(6, 7)$.

...
Dòng cuối cùng lần lượt là các giá trị $g(0; 0), g(0; 1), \dots, g(0, 7)$.

9.16.

Tìm phép chuyển tối ưu từ vị trí ban đầu của trò chơi tổng ba trò chơi sau:

- Trò chơi chẵn-lẻ với cọc 18 quân;
- Trò chơi At-Least-Haft với cọc 17 quân;
- Trò chơi Nim chuẩn một cọc 7 quân.

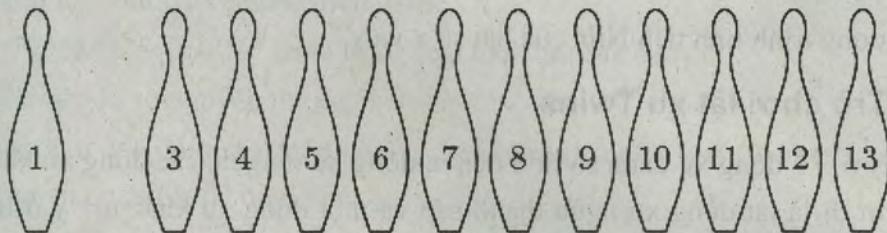
9.17. Trò chơi Grundy

Trò chơi này có phép chuyển hợp lệ là chia một cọc thành hai cọc có kích thước khác nhau. Người nào có phép chuyển cuối cùng thì thắng.

- Viết chương trình tính giá trị hàm Sprague-Grundy cho trò chơi này với một cọc ban đầu có kích thước n quân (với $n = 1, 2, 3, \dots, 13$).
- Xét trò chơi Grundy với ba cọc ban đầu có số quân là 5, 8 và 13. Tìm các phép chuyển tối ưu đầu tiên nếu có.

9.18. Trò chơi Kayler

Có 13 ki (kegen) trong trò chơi bowling sắp thành một hàng, nhưng ki thứ hai đã bị hạ đổ (xem hình vẽ).

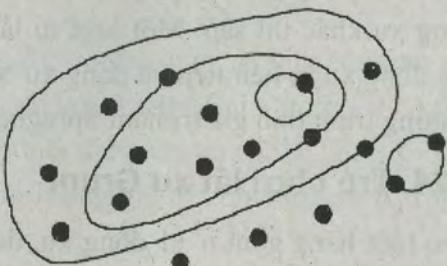


- Chứng tỏ vị trí này là N.
- Tìm phép chuyển giành chiến thắng. Khi đó ki nào sẽ bị hạ?

9.19. Trò chơi Rims

Một vị trí của trò chơi Rims là một tập hữu hạn chấm tròn trên mặt phẳng có thể bị ngăn cách bởi một vài đường vòng kín không cắt nhau. Một phép di chuyển là vẽ một vòng kín đi qua một vài chấm điểm nào đó (ít nhất là một điểm) nhưng không cắt các vòng đã có.

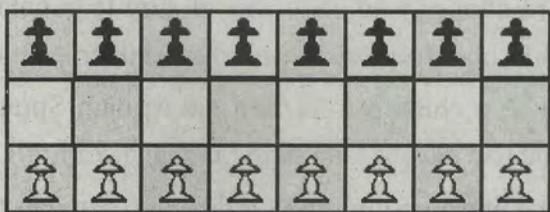
Hai người chơi lần lượt thực hiện di chuyển, người có phép chuyển cuối cùng là người chiến thắng.



- Chứng tỏ trò chơi này là một dạng của Nim.
- Với vị trí cho như hình vẽ, tìm phép di chuyển giành chiến thắng nếu có.

9.20. Cờ Dawson

Có hai hàng quân tốt đen và trắng đối diện nhau, mỗi hàng có n quân, cách nhau một hàng trống (hình bên). Một quân tốt chỉ có thể đi lên một ô trống phía trước hoặc buộc phải bắt một quân của đối phương đã đứng ở ô kè chéo bên trái hoặc bên phải của ô chứa quân tốt này.



- Viết chương trình tìm giá trị hàm Sprague-Grundy của trò chơi Dawson.
- Hãy tìm bước chuyển chiến thắng khi $n = 18$.

9.21. Tích Nim

Lập chương trình tính tích Nim của hai số x và y .

9.22. Trò chơi lật xu Twins

Cho dãy $n + 1$ đồng xu đánh số từ 0 đến n , đồng xu n ngửa, các đồng xu khác sấp. Một lượt đi là lật đồng xu ngửa thành sấp và một đồng xu khác tuỳ ý ở bên trái đang sấp thành ngửa. Xây dựng hàm Sprague-Grundy cho các vị trí của trò chơi này.

9.23. Trò chơi lật xu Ruler

Cho một hàng gồm n đồng xu, đánh số từ 1 đến n , đồng xu n đang ngửa, còn các đồng xu khác thì sấp. Một lượt đi là lật đồng thời một số đồng xu với điều kiện các đồng xu là liên tiếp và đồng xu bên phải nhất đang ngửa bị lật thành sấp. Viết chương trình tính giá trị hàm Sprague-Grundy cho các vị trí trong trò chơi này.

9.24. Trò chơi lật xu Grunt

Cho một hàng gồm $n + 1$ đồng xu, đánh số từ 0 đến n , đồng xu n đang ngửa, còn các đồng xu khác thì sấp. Một lượt đi là lật bốn đồng xu có vị trí đối xứng qua vị trí chính giữa của đồng xu bên trái nhất và đồng xu bên phải nhất.

Viết chương trình tính giá trị hàm Sprague-Grundy cho các vị trí trong trò chơi này.

9.25. Trò chơi Turning Corners

Bàn cờ vuông kích thước $n \times n$ được chia thành các ô vuông và đánh toạ độ như sau: ô góc trên-phải có toạ độ $(0, 0)$, hoành độ tăng dần từ phải qua trái, tung độ tăng dần từ trên xuống dưới. Trên mỗi ô vuông có thể đặt một đồng xu (ngửa hoặc sấp). Hai người chơi lần lượt. Một lượt đi là chọn một hình chữ nhật nhỏ trong bàn cờ có góc phải-dưới là (x, y) mà đồng xu tại ô này ngửa và ba đồng xu ở ba góc còn lại của hình chữ nhật nhỏ này úp. Sau đó người chơi cần lật đồng xu đang ngửa thành thành úp và lật ba đồng xu ở ba góc còn lại của hình chữ nhật thành ngửa. Người chơi nào úp được đồng xu cuối cùng thì thắng cuộc.

- Lập trình tính các giá trị hàm Sprague-Grundy của các vị trí trong trò chơi.
- Lập trình cho biết trạng thái ban đầu của bàn cờ là N hay P. Nếu trạng thái là N, hãy tìm một bước đi giành chiến thắng.

Input: Tệp BAI25_C9.INP chứa trạng thái ban đầu của bàn cờ:

Dòng đầu tiên là số nguyên dương n

Tiếp theo là $n + 1$ dòng thể hiện bàn cờ, mỗi dòng có $n + 1$ số 0 hoặc 1 (số 0 thể hiện đồng xu úp, số 1 thể hiện đồng xu ngửa).

Output: Ghi vào tệp BAI25_C9.OUT. Dòng đầu ghi số 1 nếu trạng thái ban đầu của bàn cờ là N, ghi số 0 nếu trạng thái ban đầu là P. Nếu trạng thái ban đầu là N thì dòng thứ hai ghi toạ độ (x, y) ở góc phải-dưới của hình chữ nhật nhỏ trong bước đi chiến thắng. Hạn chế: $n < 21$.

9.26. Truy tìm tội phạm

Tội phạm B lên tàu từ Hà Nội về Hải Phòng để trốn sự truy nã của công an A. Công an A có thể lên tàu nhanh, bắt B tại Hải phòng. Tại ga Hải Dương, B có thể xuống để tàu thoát. Tuy nhiên công an A cũng biết điều này và có thể cũng xuống tàu tại Hải Dương để truy tìm. Sau đây là bảng lượng giá trị cho công an A trong kế hoạch truy tìm tội phạm B:

		Tội phạm B xuống ga	
		Hải Dương	Hải Phòng
Công an A xuống ga	Hải Dương	100	-50
	Hải Phòng	0	100

Tìm chiến thuật hỗn hợp tối ưu cho A và B, giá trị trò chơi bằng bao nhiêu?

9.27. Điểm yên ngựa. Trò chơi ma trận 2×2

a) Lập trình tìm điểm yên ngựa của ma trận 2×2 .

b) Cho trò chơi ma trận có ma trận lượng giá là $A = \begin{pmatrix} 0 & 2 \\ t & 1 \end{pmatrix}$ với t là số thực tùy ý.

Chạy chương trình ở câu a kiểm nghiệm: Khi $t \leq 0$ thì điểm yên ngựa là $A[1][1]$, khi $0 < t \leq 1$ thì điểm yên ngựa là $A[2][1]$; khi $t > 1$ thì A không có điểm yên ngựa. Từ đó tìm hàm $v(t)$ là giá trị trò chơi (phụ thuộc t).

9.28. Giảm ước ma trận

Giải trò chơi trên các ma trận sau bằng cách giảm bớt các hàng và cột bị chi phối đưa về ma trận 2×2 .

$$(a) \begin{pmatrix} 5 & 4 & 1 & 0 \\ 4 & 3 & 2 & -1 \\ 0 & -1 & 4 & 3 \\ 1 & -2 & 1 & 2 \end{pmatrix}; \quad (b) \begin{pmatrix} 10 & 0 & 7 & 1 \\ 2 & 6 & 4 & 7 \\ 6 & 3 & 3 & 5 \end{pmatrix}.$$

9.29. Kiểm nghiệm định lí Minmax

Người ta chứng minh được kết luận sau : “Với một trò chơi ma trận đã cho, các chiến thuật (p_1, p_2, \dots, p_m) và (q_1, q_2, \dots, q_n) tương ứng là các chiến thuật tối ưu cho người thứ nhất và người thứ hai nếu giá trị trung bình nhận được ít nhất của người thứ nhất bằng giá trị trung bình phải trả nhiều nhất của người thứ hai”.

Chứng tỏ rằng chiến thuật $p = \left(\frac{6}{37}; \frac{20}{37}; 0; \frac{11}{37} \right)$ và $q = \left(\frac{14}{37}; \frac{4}{37}; 0; \frac{19}{37}; 0 \right)$

tương ứng là chiến thuật tối ưu cho người thứ nhất và người thứ hai trong trò chơi ma trận như sau:

$$\begin{pmatrix} 5 & 8 & 3 & 1 & 6 \\ 4 & 2 & 6 & 3 & 5 \\ 2 & 4 & 6 & 4 & 1 \\ 1 & 3 & 2 & 5 & 3 \end{pmatrix}$$

Tìm giá trị của trò chơi.

9.30. Trò chơi Mendelsohn

Hai người chơi đồng thời chọn mỗi người một số nguyên không vượt quá mức nào đó. Chẳng hạn: Hai người chọn số nguyên trong phạm vi từ 1 đến 100. Nếu hai số bằng nhau thì giá trị trả về là 0, nếu chọn lớn hơn đối thủ 1 đơn vị thì thắng 1, nếu chọn lớn hơn đối thủ 2 đơn vị hoặc nhiều hơn thì thua 2. Bảng giá trị trả về cho người thứ nhất (chọn dòng) như hình bên.

Hãy tìm cách loại bỏ những dòng và cột bị chia phôi, sau đó sử dụng định lí cân bằng tìm chiến thuật tối ưu cho hai người chơi.

	1	2	3	4	5	...
1	0	-1	2	2	2	...
2	1	0	-1	2	2	...
3	-2	1	0	-1	2	...
4	-2	-2	1	0	-1	...
5	-2	-2	-2	1	0	...
...

9.31. Ma trận nghịch đảo

Viết chương trình tìm ma trận nghịch đảo của ma trận khả đảo.

9.32. Trò chơi ma trận

Xét trò chơi có ma trận $A = \begin{pmatrix} -2 & 2 & -1 \\ 1 & 1 & 1 \\ 3 & 0 & 1 \end{pmatrix}$.

- Kiểm tra xem ma trận A có điểm yên ngựa hay không.
- Chứng tỏ tồn tại ma trận nghịch đảo của A.
- Chứng tỏ người thứ hai có chiến thuật tối ưu với trọng số dương cho mỗi cột.
- Vì sao trong trò chơi này không dùng đẳng thức $q = \left(\frac{A^{-1} \cdot 1}{1^T \cdot A^{-1} \cdot 1} \right)$ để tìm chiến thuật tối ưu của người thứ hai.

9.33. Trò chơi đoán số

Người chơi thứ hai chọn một số $j \in \{1, 2, 3, 4\}$ và người chơi thứ nhất cố gắng phòng đoán số mà người thứ hai đã chọn. Nếu đoán đúng thì người chơi thứ nhất thắng 2^j điểm do người thứ hai trả, ngược lại không phải trả gì. Lập ma trận trò chơi và giải trò chơi này.

9.34. Chiến thuật Baye

Xét trò chơi trên ma trận $A = \begin{pmatrix} 0 & 7 & 2 & 4 \\ 1 & 4 & 8 & 2 \\ 9 & 3 & -1 & 6 \end{pmatrix}$. Qua kinh nghiệm chơi trò chơi này,

người thứ nhất tin rằng người thứ hai sẽ dùng chiến thuật $q = \left(\frac{1}{5}; \frac{1}{5}; \frac{1}{5}; \frac{2}{5}\right)$.

- a) Tìm chiến thuật Baye phản ứng lại chiến thuật q nêu trên.
- b) Giả sử người thứ hai dự đoán đúng rằng người thứ nhất sẽ dùng chiến thuật Baye để chống lại chiến thuật $q = \left(\frac{1}{5}; \frac{1}{5}; \frac{1}{5}; \frac{2}{5}\right)$ của mình. Hãy hướng dẫn cho người thứ hai tìm chiến thuật Baye chống lại người thứ nhất khi người thứ nhất phản ứng lại chiến thuật $q = \left(\frac{1}{5}; \frac{1}{5}; \frac{1}{5}; \frac{2}{5}\right)$.

9.35. Thuật toán đơn hình

Giải trò chơi ma trận $A_{3 \times 3} = \begin{pmatrix} 0 & 5 & -2 \\ -3 & 0 & 4 \\ 6 & -4 & 0 \end{pmatrix}$ bằng thuật toán đơn hình.

9.36. Trò chơi với đồng bạc

Người thứ hai chọn một trong hai phòng để giấu một đồng bạc. Người thứ nhất không biết phòng giấu đồng bạc và thử tìm. Nếu đồng bạc giấu ở phòng I và

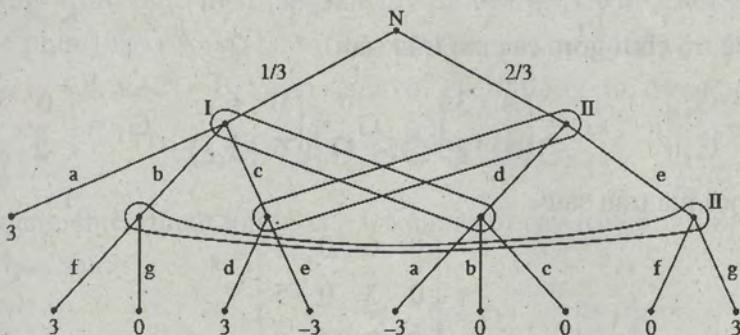
người thứ nhất tìm ở phòng I thì xác suất tìm thấy là $\frac{1}{2}$. Nếu đồng bạc giấu ở

phòng II và người thứ nhất tìm ở phòng II thì xác suất tìm thấy là $\frac{1}{3}$. Nếu tìm

được đồng bạc thì người thứ nhất được đồng bạc đó, ngược lại thì người thứ hai được. Nhưng người thứ nhất tìm lần thứ nhất không thành công và thực hiện tìm lần thứ hai độc lập với lần thứ nhất (coi như người thứ hai lại dấu đồng bạc lần thứ hai mà người thứ nhất không biết giấu ở phòng nào). Hãy vẽ cây trò chơi và giải trò chơi.

9.37. Cây trò chơi

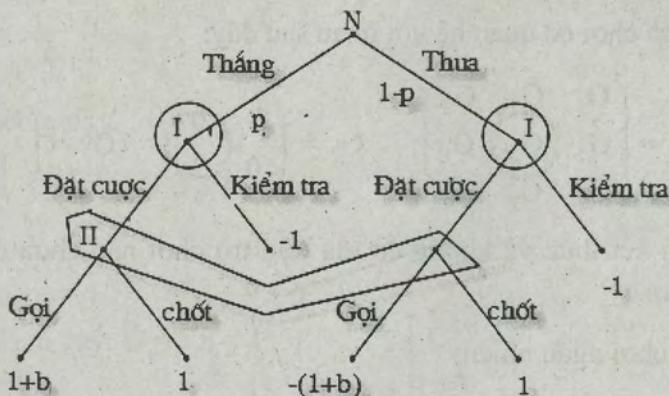
Tìm dạng chiến thuật tối ưu của trò chơi với cây trò chơi sau:



Sau đó giải trò chơi này.

9.38. Ván cơ bản kết thúc trò chơi bài

Tổng quát hoá ván cơ bản kết thúc trò chơi bài là cách đặt xác suất cho quân bài thắng là một số tuỳ ý p ($0 \leq p \leq 1$) và cho tiền đặt cược là một số tuỳ ý b , $b > 0$ (hình vẽ).



Hãy viết chương trình tìm giá trị trò chơi và chiến thuật tối ưu cho từng người.

Lưu ý: Với $p \geq \frac{2+b}{2+2b}$ thì có điểm yên ngựa. Khi kết thúc chú ý rằng $p < \frac{2+b}{2+2b}$, chiến thuật tối ưu của người thứ hai không phụ thuộc vào p .

Input: Tệp BAI38_C9.INP chỉ có một dòng ghi hai số p và b ($0 \leq p \leq 1$).

Output: Tệp BAI38_C9.OUT có ba dòng: dòng thứ nhất là giá trị trò chơi trả về cho người thứ nhất. dòng thứ hai ghi chiến thuật của người thứ nhất. dòng thứ ba ghi chiến thuật của người thứ hai.

9.39. Giải hệ trò chơi gồm các ma trận sau:

$$a) G = \begin{pmatrix} 0 & G_1 \\ G_2 & G_3 \end{pmatrix}; \quad G_1 = \begin{pmatrix} 4 & 3 \\ 1 & 2 \end{pmatrix}; \quad G_2 = \begin{pmatrix} 0 & 6 \\ 5 & 1 \end{pmatrix}; \quad G_3 = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}.$$

b) Giải trò chơi ma trận sau:

$$G = \begin{pmatrix} 0 & 6 & 0 & 1 \\ 0 & 3 & 0 & 5 \\ 5 & 0 & 2 & 0 \\ 1 & 0 & 4 & 0 \end{pmatrix}.$$

9.40. Trò chơi đê quy

Giải trò chơi : $G = \begin{pmatrix} G & 2 \\ 0 & 1 \end{pmatrix}$. Q, trong đó quy ước Q là giá trị trả về khi trò chơi kéo dài vô hạn.

9.41. Xét ba trò chơi có quan hệ với nhau sau đây:

$$G_1 = \begin{pmatrix} G_1 & G_2 & G_3 \\ G_2 & G_3 & G_1 \\ G_3 & G_1 & G_2 \end{pmatrix}; \quad G_2 = \begin{pmatrix} G_1 & 0 \\ 0 & 2 \end{pmatrix}; \quad G_3 = \begin{pmatrix} G_2 & 1 \\ 1 & 0 \end{pmatrix}.$$

Giả sử trò chơi kết thúc và không có ma trận trò chơi nào chứa điểm yên ngựa. Hãy giải trò chơi.

9.42. Giải trò chơi ngẫu nhiên:

$$G = \begin{pmatrix} 4 & 1 + \frac{1}{3}G \\ 0 & 1 + \frac{2}{3}G \end{pmatrix}.$$

9.43. Cho trò chơi ngẫu nhiên có hai trạng thái sau đây:

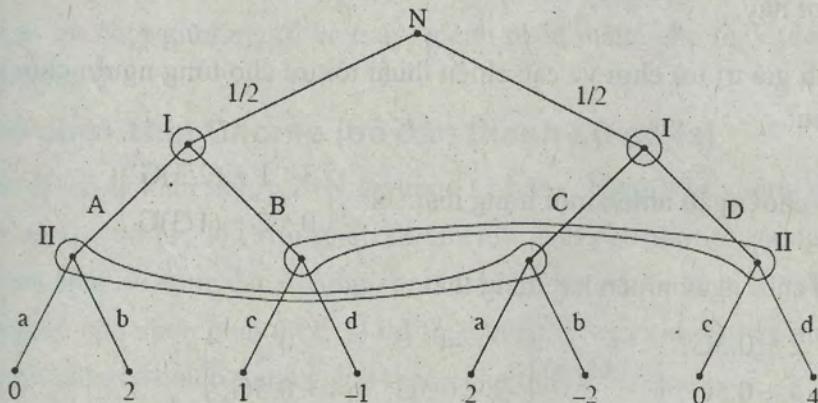
$$G^{(1)} = \begin{pmatrix} 2 & 2 + 0.5G^{(2)} \\ 0 & 4 + 0.5G^{(2)} \end{pmatrix}; \quad G^{(2)} = \begin{pmatrix} -4 & 0 \\ -2 + 0.5G^{(1)} & -4 + 0.5G^{(1)} \end{pmatrix}.$$

a) Dùng định lí Shapley tìm chính xác các giá trị $v(1)$ và $v(2)$ của hai trò chơi $G^{(1)}$ và $G^{(2)}$ nếu giả sử khi kết thúc các trò chơi này trả về được các giá trị đó.

b) Viết chương trình thực hiện lặp Shapley để tìm giá trị trò chơi ở giai đoạn k (nhập từ bàn phím) là $v_k = (v_k(1); v_k(2))$, bắt đầu với dự đoán cho giai đoạn khởi trị là $v_0 = (v_0(1) = 0; v_0(2) = 0)$ và so sánh với giá trị đúng tìm được từ câu a).

BÀI TẬP BỔ SUNG

9.44. Tìm dạng chiến thuật tối ưu của trò chơi với cây trò chơi tính giá trị trả về cho người I như sau:



Sau đó giải trò chơi này.

9.45.

$$\text{Giải trò chơi } G \text{ sau đây: } G = \begin{pmatrix} 3 & 0 & 5 & 2 & 2 \\ 1 & 3 & 5 & 2 & 2 \\ 4 & 4 & 1 & 2 & 2 \\ 1 & 1 & 1 & 6 & 3 \\ 1 & 1 & 1 & 4 & 7 \end{pmatrix}$$

9.46.

$$a) \text{ Giải trò chơi } G = \begin{pmatrix} G & 1 & 0 \\ 1 & 0 & G \\ 0 & G & 1 \end{pmatrix}, Q;$$

b) Giải trò chơi $G_0 = \begin{pmatrix} G_0 & 5 \\ 1 & 0 \end{pmatrix}, Q$;

Trong đó quy ước Q là giá trị trả về cho người I nếu trò chơi được kéo dài vô hạn.

9.47. Cho ba trò chơi liên kết:

$$G_1 = \begin{pmatrix} G_2 & 0 \\ 0 & G_3 \end{pmatrix}, Q; \quad G_2 = \begin{pmatrix} G_1 & 1 \\ 1 & 0 \end{pmatrix}, Q; \quad G_3 = \begin{pmatrix} G_1 & 2 \\ 2 & 0 \end{pmatrix}, Q.$$

Q là giá trị trả lại của các trò chơi này khi được chơi vô hạn. Hãy giải đồng thời các trò chơi này.

9.48. Tính giá trị trò chơi và các chiến thuật tối ưu cho từng người chơi trong các trò chơi sau:

a) Cho trò chơi ngẫu nhiên một trạng thái: $G = \begin{pmatrix} 3 & 1 + (2/3)G \\ 0 & 2 + (1/3)G \end{pmatrix}$

b) Cho trò chơi ngẫu nhiên hai trạng thái :

$$G_1 = \begin{pmatrix} 2 & 2 + 0.5G_2 \\ 0 & 4 + 0.5G_2 \end{pmatrix} \quad G_2 = \begin{pmatrix} -4 & 0 \\ -2 + 0.5G_1 & -4 + 0.5G_1 \end{pmatrix}$$

không có điểm yên ngựa khi trò chơi tiếp diễn.

9.49. Viết chương trình tìm giá trị trò chơi ở giai đoạn k của trò chơi gồm hai trò chơi G_1 và G_2 có quan hệ bởi các ma trận sau :

$$G_1 = \begin{pmatrix} 4 + 0.3G_1 & 0 + 0.4G_2 \\ 1 + 0.4G_2 & 3 + 0.5G_1 \end{pmatrix}; \quad G_2 = \begin{pmatrix} 0 + 0.5G_1 & -5 \\ -4 & 1 + 0.5G_2 \end{pmatrix}$$

Input: Tệp BAI49_C9.INP chỉ có một dòng duy nhất ghi số k.

Output: Tệp BAI49_C9.OUT có dạng gồm nhiều nhóm dòng. Nhóm mô tả kết quả giai đoạn 1 của trò chơi nêu trong đề bài cần ghi theo khuôn dạng sau:

Giai đoạn 1 :

v1=2 p=(0.333333 , 0.666667) q=(0.5 , 0.5)

v2=-2 p=(0.5 , 0.5) q=(0.6 , 0.4)

Các nhóm còn lại tương tự. Hai nhóm dòng liên tiếp cách nhau một dòng trống.

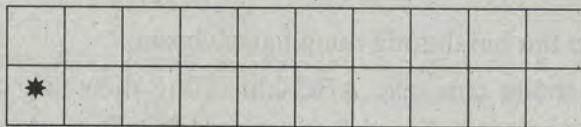
9.50. Trò chơi “Ba_mươi_mốt”

Trò chơi Ba_mươi_mốt như sau: Từ mỗi loại chất của cỗ bài tú lơ khơ, lấy ra các quân: A (quân át), 2, 3, 4, 5 và 6 được 24 quân bài đặt ngửa trên bàn. Hai người chơi lần lượt lật úp một quân trong các quân bài đang ngửa và cộng dồn số trên quân bài này vào tổng chung (A được tính là 1). Người nào tạo ra tổng lớn hơn 31 đầu tiên là thua.

- Nếu bạn là người đi đầu tiên và áp dụng chiến thuật bạn luôn chọn quân mang số a ($a = 1, 2, \dots, 6$) sao cho sau nước đi của bạn thì (31-a) chia hết cho 7 (như thuật toán giành thắng của trò chơi lấy bớt quân cờ) thì bạn có thể thắng hay không?
- Lập trình trò chơi giữa người và máy, giành phần thắng cho máy (đóng vai trò người đi trước).

9.51. Trò chơi Thin Chomp (bẻ dần thanh sô-cô-la)

Cho thanh sô-cô-la kích thước $2 \times N$ ô vuông (2 hàng, N cột). Ô vuông ở góc trái-dưới là ô có tọa độ (1; 1). Mỗi phép di chuyển là lấy đi một ô vuông và các ô vuông ở bên phải nó cùng các ô vuông phía trên chúng. Hai người chơi lần lượt di chuyển, người nào chọn phải ô (1, 1) thì thua cuộc (ô này coi như bị nhiễm độc). Chứng minh luôn có chiến thuật giành thắng cho người đi đầu.



9.52. Trò chơi 1D-SOS (The Y2K Game)

Trò chơi 1D-SOS được chơi trên một băng lưới kích thước $1 \times N$ gồm N ô vuông đơn vị. Hai người chơi A, B lần lượt viết chữ cái S hoặc O vào một ô lưới còn rỗng. Người chiến thắng là người đầu tiên tạo ra từ “SOS” trên ba ô liên tiếp. Nếu tất cả các ô đã điền đầy chữ mà chưa xuất hiện từ “SOS” thì hoà.

- Khi $N = 4$ và người chơi A đi trước đặt S vào ô vuông thứ nhất, chứng minh người chơi B sẽ chiến thắng.
- Chứng minh khi $N = 7$, người chơi A có thể chiến thắng.
- Chứng minh khi $N = 2000$, người chơi B có thể chiến thắng (Bài 5, Kì thi vô địch Toán của Mĩ năm 1999).
- Lập trình trò chơi 1D-SOS với $N = 16$, giữa người và máy, người đi trước (máy luôn thắng).

Phần III

Hướng dẫn giải bài tập

CHUYÊN ĐỀ 8

8.1. Thuật toán: Tìm tập điểm X gồm các giao điểm của đoạn thẳng DE với các cạnh của tam giác ABC và các điểm đầu mút của đoạn thẳng DE nằm trong tam giác ABC. Nếu tập điểm X có đúng hai điểm thì khoảng cách giữa hai điểm này là độ dài phần đoạn thẳng DE nằm trong tam giác ABC; ngược lại độ dài này coi như bằng 0. Nên dùng lớp vector để lưu giữ tập điểm X nêu trên. Vậy để giải bài toán này cần chuẩn bị hai bài toán cơ sở:

- a) Bài toán cơ sở thứ nhất là: Tìm giao điểm của hai đoạn thẳng (trang 19-20, Tài liệu chuyên Tin học, Quyển 3);
- b) Bài toán cơ sở thứ hai là: Xác nhận vị trí một điểm có ở trong tam giác hay không?

Giải bài toán cơ sở thứ hai thường dùng hai cách sau:

Cách 1. Điểm H thuộc tam giác ABC khi: Tổng diện tích các tam giác HAB, HBC, HAC không vượt quá diện tích tam giác ABC. Điều này tương đương với:

$$\text{Abs}(\text{tichcheo}(\overrightarrow{HA}, \overrightarrow{HB})) + \text{Abs}(\text{tichcheo}(\overrightarrow{HB}, \overrightarrow{HC})) + \text{Abs}(\text{tichcheo}(\overrightarrow{HC}, \overrightarrow{HA})) \leq \text{Abs}(\text{tichcheo}(\overrightarrow{AB}, \overrightarrow{AC})).$$

Cách 2. Sử dụng hàm *ccw* để định hướng đi $H \rightarrow A \rightarrow B$, $H \rightarrow B \rightarrow C$, $H \rightarrow C \rightarrow A$. Nếu ba định hướng cùng dấu hoặc bằng 0 thì H ở trong tam giác ABC.

Chương trình:

```
#include <iostream>
#include <math.h>
#include <vector>

#define tichcheo(u, v) (u.x*v.y - u.y*v.x) // Định nghĩa tích chéo của 2 vector u, v
#define vocung 1.0E+30
using namespace std;
struct TVector { // cấu trúc một vector
    double x, y;
```

```

    } u, v, t;      //Khai báo các vector  $\vec{u}$ ,  $\vec{v}$ ,  $\vec{t}$ 
#define TPoint Tvector //Định nghĩa kiểu của Điểm (như cấu trúc vector là một cặp số)
vector <TPoint> res; //Khai báo res là đối tượng thuộc lớp mẫu vector
TPoint A, B, C, D, E, M, N, P; //Các điểm cần thiết
ifstream fin ("bai01_c8.in");
ofstream fout ("bai01_c8.out");
void read_input(){ //Đọc dữ liệu vào
    fin >> A.x >> A.y >> B.x >> B.y >> C.x >> C.y;
    fin >> D.x >> D.y >> E.x >> E.y;
    fin.close();
}

TVector SolveSLE(TVector u, TVector v, TVector t) {
//Xác định hệ số p và q trong đẳng thức  $\vec{t} = p \cdot \vec{u} + q \cdot \vec{v}$ 
    TVector T;
    T.x = vocung; T.y = vocung;
    double D;
    D = tichcheo(u, v);
    if (D!=0) {
        T.x = tichcheo(t,v)/D; //là p
        T.y = tichcheo(u,t)/D; //là q
    }
    return T;
}

bool is_in(double z) { //Hàm xác nhận số thực z thuộc đoạn [0; 1]
    return ((z>=0) && (z<=1));
}

bool Cross(TPoint A, TPoint B, TPoint E, TPoint D, TPoint &M) {
//Xác nhận đoạn thẳng AB và DE có cắt nhau không, nếu có thì xác định giao điểm M của chúng
    u.x = B.x-A.x; u.y = B.y-A.y;
    v.x = E.x-D.x; v.y = E.y-D.y;
    t.x = E.x-A.x; t.y = E.y-A.y;
    TVector p = SolveSLE(u,v,t);
    bool kq = is_in(p.x) && is_in(p.y);
    if (kq) { //p và q thuộc [0; 1] thì tồn tại giao điểm M của đoạn thẳng AB và DE
        M.x = A.x + p.x * (B.x - A.x);
        M.y = A.y + p.x * (B.y - A.y);
    }
    return (kq);
}

double Distance_(TPoint M, TPoint N) { //Tính khoảng cách MN
    return (sqrt( (M.x-N.x)*(M.x-N.x)+(M.y-N.y)*(M.y-N.y) ) );
}

bool is_inTriangle(TPoint H) { //Xác nhận điểm H có trong miền tam giác hay không
    TVector v1,v2,v3,v4,v5;
    v1.x = H.x - A.x; v1.y = H.y-A.y;
    v2.x = H.x - B.x; v2.y = H.y-B.y;

```

```

v3.x = H.x - C.x; v3.y = H.y-C.y;
v4.x = B.x - A.x; v4.y = B.y-A.y;
v5.x = C.x - A.x; v5.y = C.y-A.y;
double t1= tichcheo(v1,v2);
if (t1<0) t1 = -t1;
double t2= tichcheo(v1,v3);
if (t2<0) t2 = -t2;
double t3= tichcheo(v3,v2);
if (t3<0) t3 = -t3;
double t4= tichcheo(v4,v5);
if (t4<0) t4 = -t4;
return ( t1+t2+t3 <= t4);
}
bool is_notSame(TPoint H) { //Xác nhận điểm H không trùng với các điểm đã tìm được
int i=0;
while ( i<res.size()) {
    if ( (res[i].x==H.x) && (res[i].y==H.y) )
        return false;
    i++;
}
return true;
}
int main() {
    double dis=0;
    read_input();
    res.reserve(5);
    if (Cross(A,B,D,E,M) ) //Nếu có M là giao của các đoạn thẳng AB và DE
        res.push_back(M); //thì lưu M vào vector res
    if (Cross(B,C,D,E,N) ) { //Nếu có N là giao của các đoạn thẳng BC và DE
        if ( is_notSame(N) ) //và N không trùng với các điểm đã lưu
            res.push_back(N); //thì lưu N vào res
    }
    if (Cross(C,A,D,E,P) ){ //Nếu có P là giao của các đoạn thẳng CA và DE
        if ( is_notSame(P) ) //và P không trùng với các điểm đã lưu
            res.push_back(P); //thì lưu P vào res
    }
    if (is_inTriangle(D)) //Nếu D thuộc miền tam giác
        if (is_notSame(D) ) //và không trùng với các điểm đã lưu
            res.push_back(D); //thì lưu D vào res
    if (is_inTriangle(E)) //Tương tự với E
        if ( is_notSame(E) )
            res.push_back(E);
    if (res.size()==2) { //Nếu res chỉ có 2 điểm
        dis = Distance_(res[0], res[1]); //Độ dài cần tìm
        cout << dis << endl;
    }
    else cout << 0 << endl;
    fout.close();
    return 0;
}

```

5.2. Thuật toán:

- Tìm tập X gồm các điểm sau: giao điểm của hai đoạn thẳng mà mỗi đoạn thẳng là một cạnh thuộc hai tam giác khác nhau; các đỉnh của tam giác này thuộc tam giác kia.
- Tìm bao lồi B của tập X (thực chất trong tình huống bài tập này là chỉ định thứ tự các đỉnh liên tiếp theo một chiều nào đó).
- Tính diện tích đa giác là bao lồi B.

Vậy cần sử dụng bốn bài toán cơ sở gồm: hai bài toán cơ sở đã nêu trong phần hướng dẫn của bài 8.1; bài toán cơ sở thứ ba là bài toán tìm bao lồi, bài toán cơ sở thứ tư là tính diện tích đa giác (theo công thức ở trang 23, Tài liệu chuyên Tin học, Quyển 3).

Bao lồi được xây dựng theo ý tưởng “bọc gói” N điểm:

Giả sử cho tập điểm A_1, A_2, \dots, A_N .

- Tìm hai điểm ban đầu bao đảm mọi điểm phải ở cùng một phía của đường thẳng nối hai điểm này. Trong hai điểm này, điểm thứ nhất chọn là điểm có hoành độ lớn nhất trong các điểm có tung độ thấp nhất.
- Sau đó kết nạp dần các điểm khác vào bao lồi. Cách kết nạp điểm mới phải bao đảm: mọi điểm phải ở cùng một phía của đường thẳng nối điểm mới với điểm cuối cùng đưa vào bao lồi ở thời điểm trước khi kết nạp điểm mới. Cụ thể: Từ hai điểm vừa kết nạp là i, j ta tìm thêm điểm k sao cho điểm i và mọi điểm l ($l \neq k$) luôn cùng phía đường thẳng qua k và j .
- Cuối cùng duyệt lại đường bao, loại bỏ những điểm ở giữa trong ba điểm thẳng hàng trên đường bao.

Một cách thực hiện cụ thể ý tưởng “bao gói” này là:

- Tìm điểm có hoành độ lớn nhất trong các điểm có tung độ nhỏ nhất và chọn làm điểm thứ nhất của bao gọi là điểm A_m . Tráo đổi dữ liệu của điểm A_1 và điểm A_m .
- Tìm đỉnh i chưa thuộc bao lồi là điểm tiếp theo của bao lồi, sao cho góc α tạo bởi vectơ $\overrightarrow{A_mA_i} = (x_i - x_m, y_i - y_m) = (d_x, d_y)$ với vectơ $\vec{e} = (1; 0)$ đạt giá trị nhỏ nhất (đó là góc $\overline{A_mA_i}$ nghiêng so với chiều dương trực hoành). Nghĩa là mọi điểm sẽ cùng thuộc một phía của đường thẳng A_mA_i . Có thể thay việc tính góc α bằng tính góc tê-ta nhờ hàm theta() như sau:

```

float theta(Tpoint p1, Tpoint p2)
{
    float dx, dy, ax, ay, t;
    dx=p2.x-p1.x;      dy=p2.y-
p1.y;
    ax=dx; if (ax<0) ax=-ax;
    ay=dy; if (ay<0) ay=-ay;
    if ((dx==0) && (dy==0))
t= 0;
    else t= dy/(ax+ay);
    if (dx<0) t=2-t;
    else if (dy<0) t=4+t;
    return t*90;
}

```

Góc *tê-ta* này có giá trị từ 0° đến 360° . Như vậy không phải sử dụng hàm *arctan* (thường chạy chậm).

- Kết nạp đỉnh i vừa tìm được ở bước trên vào bao lồi bằng các tráo đổi dữ liệu của đỉnh i vừa tìm được với đỉnh m + 1, nghĩa là bây giờ đỉnh thứ m+1 của bao lồi chính là đỉnh i.
- Giai đoạn cuối cùng là lược bỏ những điểm nằm trên cạnh của bao lồi (không kể đỉnh bao lồi) nếu quá trình trên chưa loại bỏ.

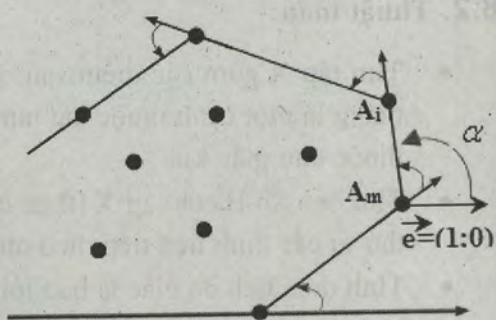
Một kĩ thuật khác để tìm bao lồi: sắp xếp lại các điểm để tạo đa giác đơn, sau đó đi trên đa giác đơn tìm các đỉnh của bao lồi (xem chương trình dưới đây).

Chương trình:

```

#include <iostream>
#include <fstream>
#include <math.h>
#include <iomanip>
#define tichcheo(u, v) (u.x*v.y-u.y*v.x)
#define vocung 1.0E+30;
struct TVector {
    double x;
    double y;
} TG[2][3], u,v,t,M;
struct TVector B[10000], X[10000]; // Lưu bao lồi B, lưu tập giao điểm X
int nx, nb; // Chi số cuối cùng của phần tử mảng X, mảng B
using namespace std;
ifstream fin ("bai02_c8.inp");
ofstream fout ("bai02_c8.out");
TVector SolveSLE(TVector u, TVector v, TVector t) {
    TVector T;

```



```

T.x = vocung; T.y = vocung;
double D;
D = tichcheo(u, v);
if (D!=0) {
    T.x = tichcheo(t,v)/D;
    T.y = tichcheo(u,t)/D;
}
return T;

bool is_in(double z) {
    return ((z>=0) && (z<=1));

bool Cross(TVector A, TVector B, TVector E, TVector D, TVector &M)

arctan
u.x = B.x-A.x; u.y = B.y-A.y;
v.x = E.x-D.x; v.y = E.y-D.y;
t.x = E.x-A.x; t.y = E.y-A.y;
TVector p = SolveSLE(u,v,t);
bool kq = is_in(p.x) && is_in(p.y);
if (kq) {
    M.x = A.x + p.x * (B.x - A.x);
    M.y = A.y + p.x * (B.y - A.y);
}
return (kq);

sau đó
void read_input() { //Đọc tọa độ các đỉnh hai tam giác
    fin >> TG[0][0].x >> TG[0][0].y >> TG[0][1].x >> TG[0][1].y >>
TG[0][2].x >> TG[0][2].y ;
    fin >> TG[1][0].x >> TG[1][0].y >> TG[1][1].x >> TG[1][1].y >>
TG[1][2].x >> TG[1][2].y ;

    bool is_notSame(TVector H) { //Xác nhận điểm H có trùng điểm nào của tập X không
        int i=0;
        while (i<nx) {
            if ( (X[i].x==H.x) && (X[i].y==H.y) )
                return false;
            i++;
        }
        return true;
    }

    bool is_inTriangle(TVector A, TVector B, TVector C, TVector H) {
        //Xác định điểm H có trong tam giác ABC hay không dựa vào diện tích HAB, HBC, HCA và ABC
        TVector v1,v2,v3,v4,v5;
        v1.x = H.x - A.x; v1.y = H.y-A.y;
        v2.x = H.x - B.x; v2.y = H.y-B.y;
        v3.x = H.x - C.x; v3.y = H.y-C.y;
        v4.x = B.x - A.x; v4.y = B.y-A.y;

```

```

v5.x = C.x - A.x; v5.y = C.y-A.y;
double t1= tichcheo(v1,v2);
if (t1<0) t1 = -t1;
double t2= tichcheo(v1,v3);
if (t2<0) t2 = -t2;
double t3= tichcheo(v3,v2);
if (t3<0) t3 = -t3;
double t4= tichcheo(v4,v5);
if (t4<0) t4 = -t4;
return ( t1+t2+t3 <= t4);
}

void find_X() {
// Tìm tập điểm X gồm: tập giao điểm các cạnh tam giác, tập đỉnh tam giác này nằm trong tam giác kia
    nx=-1;
    for (int i=0; i!=2; i++)
        for (int j=i+1; j!=3; j++)
            for (int k=0; k!=2; k++)
                for (int h=k+1; h!=3; h++)
                    if (Cross(TG[0][i], TG[0][j], TG[1][h], TG[1][k], M) )
                        if (is_notSame(M) ){
                            nx++;
                            X[nx]=M;
                        }
                    for (int j=0; j!=3; j++)
                        if (is_inTriangle(TG[0][0], TG[0][1], TG[0][2], TG[1][j]))
                            if ( is_notSame(TG[1][j]) ){
                                nx++;
                                X[nx]=TG[1][j];
                            }
                    for (int j=0; j!=3; j++)
                        if (is_inTriangle(TG[1][0], TG[1][1], TG[1][2], TG[0][j]))
                            if (is_notSame(TG[0][j])){
                                nx++;
                                X[nx]=TG[0][j];
                            }
}
double tichcheo2 (TVector a1, TVector a2, TVector b1, TVector b2)
{
    return ( (a2.x-a1.x)*(b2.y-b1.y)-(b2.x-b1.x)*(a2.y-a1.y) );
}
double kc(TVector a1, TVector a2) {
    return ((a2.x-a1.x)*(a2.x-a1.x) + ((a2.y-a1.y)*(a2.y-a1.y)));
}
}

```

```

void baoloi() {
    // Tìm bao lồi của tập điểm X bằng thuật toán bọc gói, lưu vào mảng B
    // Tìm điểm tung độ thấp nhất, hoành độ lớn nhất làm điểm đầu tiên của bao lồi gọi là B0
    int m = 0;
    for (int i=1; i!=nx; i++)
        if (X[i].y<X[m].y) m=i;
        else
            if ((X[i].y==X[m].y) && (X[i].x>X[m].x)) m=i;
    B[0]=X[m];
    X[m]=X[0]; X[0]=B[0];
    /* Sắp xếp lại các điểm của X để tạo đa giác đơn, khoá sắp xếp là góc quay ( $\overrightarrow{B_0X_j}, \overrightarrow{B_0X_{j+1}}$ ) < 0
    for (int i=nx; i>1; i--)
        for (int j=1; j<=i-1; j++) {
            double tc=tichcheo2(X[0], X[j], X[0], X[j+1]);
            double kc1,kc2;
            kc1 = kc(X[0], X[j]);
            kc2 = kc(X[0], X[j+1]);
            if ((tc<0) || ((tc==0) && (kc1>kc2))) {
                B[nb-1]=X[j];
                X[j] = X[j+1];
                X[j+1] = B[nb-1];
            }
        }
    // Chọn điểm thứ hai, thứ ba của bao lồi là B[1], B[2]
    int i=1;
    while (tichcheo2(X[0],X[i+1], X[0], X[1])==0) i++;
    B[1]= X[i];
    B[2]= X[i+1];
    // Chọn các điểm tiếp theo của bao lồi
    nb=2;
    for (int j=i+2; j<=nx; j++) {
        while (tichcheo2(B[nb-1], B[nb], B[nb], X[j])<=0) nb--;
        nb++;
        B[nb] = X[j];
    }
}

double dientich() { // Tính diện tích bao lồi B (tọa độ đỉnh M cũng là tọa độ vectơ  $\overrightarrow{OM}$ )
double S=0;
b2)
nb++;
B[nb]=(B[0]);
) );
for (int i=0; i!=nb; i++)
    S += tichcheo(B[i],B[i+1]);
if (S<0) S=-S;
S = S/2;
fout << fixed << setprecision(2)<< S << endl;
.y))

```

```

}
int main() {
    read_input();
    find_X();
    if (nx>1) {
        baoloi();
        dientich();
    }
    else
        fout << 0 << endl;
    return 0;
}

```

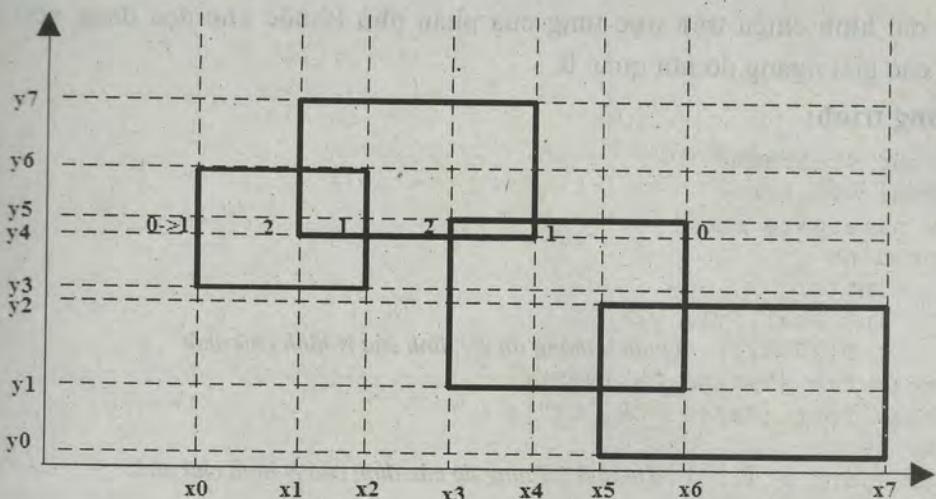
8.3. Câu a) Có hai thuật toán thường được đề cập khi giải bài toán này:

Thuật toán 1 (với độ phức tạp trung bình là $O(N^2)$): Tạo hai mảng D và T tương ứng chứa hoành độ và tung độ của $2N$ đỉnh cho trong tệp dữ liệu vào. Sắp tăng hai mảng này. Sau khi sắp tăng, các tung độ của $2N$ đỉnh tạo thành dãy tăng $T = \{y_0, y_1, \dots, y_{2N-1}\}$. Trên mặt phẳng toạ độ, các đường thẳng $y = y_i$ ($0 \leq i \leq 2N - 1$) tạo thành các dải nằm ngang.

Với mỗi dải nằm giữa đường thẳng $y = y_i$ và $y = y_{i+1}$ ($0 \leq i \leq 2N - 2$), xét các đường thẳng $x = x_j$ ($0 \leq j \leq 2N - 1$) với x_j tăng dần. Nếu trên đường thẳng này chứa cạnh bên trái của một hình chữ nhật đi qua cả dải thì tăng biến đếm 1 đơn vị. Ngược lại nếu trên đường thẳng này chứa cạnh bên phải của một hình chữ nhật đi qua cả dải này thì giảm biến đếm 1 đơn vị. Khi tăng biến đếm từ 0 lên 1, ghi nhận giá trị x_j vào biến h_1 , khi biến đếm giảm từ 1 về 0 thì ghi nhận giá trị x_j vào biến h_2 . Rõ ràng hình chữ nhật con nằm kẹp trong dải $(y_i; y_{i+1})$ và khe $(x_1; x_2)$ là một phần bị phủ; do đó cộng thêm diện tích hình chữ nhật con này là $(x_2 - x_1)*(y_{i+1} - y_i)$ vào diện tích phủ đang tìm.

Lưu ý rằng, với cách làm trên, các hình chữ nhật con không thể đè lên nhau, nên việc cộng thêm diện tích của chúng vào kết quả là hợp lý.

Khi lập trình, mảng D cần xây dựng thành mảng có dữ liệu thuận tiện hơn như sau: Mỗi phần tử của D là một bản ghi đầy đủ thông tin cho một đỉnh của hình chữ nhật, vì vậy chúng ta xây dựng mỗi phần tử là một cấu trúc có bốn trường: hoành độ của đỉnh này, loại đỉnh (1: trái-dưới, 2: phải-trên), tung độ đỉnh này và tung độ đỉnh đối diện với nó.



Hình minh họa xét dải giữa $y = y_4$ và $y = y_5$: Khi đi từ x_0 đến x_7 , tại x_0 biến đếm bắt đầu tăng từ 0 lên 1 nên ghi ta nhận $h_1 = x_0$, tại x_1 biến đếm tăng thành 2, tại x_2 biến đếm giảm thành 1, tại x_3 biến đếm tăng thành 2, tại x_4 biến đếm giảm thành 1, tại x_6 biến đếm giảm từ 1 thành 0 nên ta ghi nhận $h_2 = x_6$. Vậy hình chữ nhật nhỏ bị giới hạn bởi các đường thẳng $y = y_4$, $y = y_5$, $x = x_0$, $x = x_6$ là một phần của phủ đang tìm. Phần độ dài từ x_0 đến x_6 chính là độ dài hình chiếu của phủ trong dải $(y_4; y_5)$ chiếu trên trực hoành.

Thuật toán 2 (Độ phức tạp trung bình là $O(N \log N)$): Tương tự ý tưởng của thuật toán 1, chương trình dẫn ra cho thuật toán 2 chỉ là cải tiến của thuật toán 1 bằng cách tổ chức dữ liệu trên cây khoảng (Interval Tree). Lợi dụng những lưu trữ sẵn có trong chương trình cho thuật toán 1 (là vector T và mảng D). Để thuận tiện cho tổ chức cây khoảng thì ta không tìm phần phủ đọc theo các dải ngang tạo bởi các đường thẳng $y = y_i$ ($0 \leq i \leq 2N - 1$) mà tìm các phần phủ nằm trên các khe đọc tạo bởi các đường thẳng $x = x_j$ ($0 \leq j \leq 2N - 1$), nghĩa là tìm hình chiếu của các phần phủ trong các khe đọc chiếu lên trực tung:

Xây dựng cây nhị phân đầy đủ quản lí các dải nằm ngang tạo bởi các đường thẳng $y = y_i$ ($0 \leq i \leq 2N - 1$), mà nút gốc có số hiệu là 1. Mỗi nút i của cây có hai nút con: nút con trái mang số hiệu $2*i$, nút con phải mang số hiệu $2*i + 1$. Thông tin trên mỗi nút của cây khoảng gồm:

- + Số hiệu nút, số hiệu các dải ngang được nút này quản lí. Cụ thể: Nút 1 (gốc) quản lí toàn bộ $2N - 1$ dải từ $T[0]$ đến $T[2n - 1]$, nút 2 quản lí từ dải $T[0]$ đến dải $T[N/2]$, nút 3 quản lí từ dải $T[N/2]$ đến $T[2N - 1]$,

+ Độ dài hình chiếu trên trục tung của phần phủ (thuộc khe đọc đang xét) nằm trong các giải ngang do nút quản lí.

Chương trình:

```
#include <fstream>
#include <vector>
using namespace std;
struct dinh {
    int x;
    int loai, y1, y2;
} D[20002]; //Quản lý thông tin 2N đỉnh của N hình chữ nhật
ifstream fin ("BAI03_C8.INP");
ofstream fout ("BAI03_C8.OUT");
int n;
vector < int > T; //Quản lý 2N tung độ các đỉnh của N hình chữ nhật
int x1, x2, y1, y2;
int sosanh (const void* pt1, const void* pt2) {
    dinh t1, t2;
    t1 = *(dinh*) pt1;
    t2 = *(dinh*) pt2;
    return t1.x-t2.x;
}
void read_input() {
    fin >> n;
    T.reserve(2*n);
    for (int i=0; i!=n; i++) {
        fin >> x1 >> y1 >> x2 >> y2;
        T.push_back(y1); T.push_back(y2);
        D[i].loai=1; D[i].x=x1; D[i].y1=y1; D[i].y2=y2;
        D[i+n].loai=2; D[i+n].x=x2; D[i+n].y1=y1; D[i+n].y2=y2;
    }
    sort(T.begin(), T.end()); //Sắp tăng các tung độ
    qsort(&D, 2*n, sizeof(dinh), sosanh); //Sắp tăng theo hoành độ các đỉnh
}
void caculate() {
    int h1, h2;
    long long area=0;
    for (int i=0; i!=T.size()-1; i++) { //Duyệt theo các dài nằm ngang
        int dem = 0;
        for (int j=0; j!=2*n; j++) { //Duyệt theo các cạnh đọc của các hcn
            if ((D[j].y1 <= T[i]) && (T[i+1] <= D[j].y2)) {
                if (D[j].loai==1) {
                    dem++;
                    if (dem==1) h1=D[j].x; //Ghi nhận biên trái của phần phủ
                }
                else

```

) năm

```
        if (D[j].loai==2) {
            dem--;
            if (dem==0) {
                h2=D[j].x; //Ghi nhận biên phải của phần phù
                area += (h2-h1)*(T[i+1]-T[i]); //thêm diện tích phù
            }
        }
    }
}
fout << area << endl;
fout.close();
}

int main() {
    read_input();
    caculate();
    return 0;
}
```

Chương trình 2 (Dùng Interval Tree)

```
#include <iostream>
#include <vector>
using namespace std;
struct dinh {
    int x;
    int loai, y1, y2;
} D[20000];
ifstream fin ("BAI03_C8.INP");
ofstream fout ("BAI03_C8.OUT");
int n;
vector<int> T;
int Count[80000], Dy[80000];
int sosanh (const void* pt1, const void* pt2) {
    dinh t1, t2;
    t1 = *(dinh*) pt1;
    t2 = *(dinh*) pt2;
    return t1.x-t2.x;
}
void read_input() {
    int x1, x2, y1, y2;
    fin >> n;
    T.reserve(2*n);
    for (int i=0; i!=n; i++) {
        int j=2*i;
        fin >> x1 >> y1 >> x2 >> y2;
        T.push_back(y1); T.push_back(y2);
        D[j].loai=1; D[j].x=x1; D[j].y1=y1; D[j].y2=y2;
        D[j+1].loai=2; D[j+1].x=x2; D[j+1].y1=y1; D[j+1].y2=y2;
    }
}
```

```

    }
    sort(T.begin(), T.end());
    qsort(&D, 2*n, sizeof(dinh), sosanh);
}

void Vao(int y1, int y2, long k, long i, long j) {
// Cạnh trái hình chữ nhật có tung độ từ y1 đến y2 xử lí trên nút k quản lí các dài ngang từ T[i] đến T[j]
    long mid;
    // Không xử lí vì đoạn [y1; y2] không đi qua các dài ngang do nút quản lí
    if ((y1>=T[j]) || (y2<=T[i])) return;
    // Xử lí khi đoạn [y1; y2] đi qua các dài ngang do nút quản lí
    if ((y1<=T[i]) && (T[j]<=y2)) {
        Count[k]++;
        // Tăng biến đếm số cạnh hcn được chiếu lên trực tung
        Dy[k] = T[j]-T[i];
        // Độ dài hình chiếu trên trực tung được nút k quản lí
        return;
    }
    if (i+1>j) return; // Đến lá của cây thì thoát
    mid=(i+j)/2; // Biên chia tách thành hai nút con
    Vao(y1, y2, 2*k, i, mid); // Duyệt con trái
    Vao(y1, y2, 2*k+1, mid, j); // Duyệt con phải
/* Gặp cạnh phải của hcn, thì xác nhận lại độ dài hình chiếu của phần phủ trên trực tung do nút k quản lí bằng tổng độ dài hình chiếu này do hai nút con của k quản lí */
    if (Count[k]==0) Dy[k]=Dy[2*k]+Dy[2*k+1];
}

void Ra(int y1, int y2, long k, long i, long j) {
// Cạnh phải hình chữ nhật có tung độ từ y1 đến y2 xử lí trên nút k quản lí các dài ngang từ T[i] đến T[j]
    long mid;
    // Không xử lí vì đoạn [y1; y2] không đi qua các dài ngang do nút quản lí
    if ((y1>=T[j]) || (y2<=T[i])) return;
    // Xử lí khi đoạn [y1; y2] đi qua các dài ngang do nút quản lí
    if ((y1<=T[i]) && (T[j]<=y2)) {
        Count[k]--;
        // Giảm biến đếm số cạnh hcn được chiếu lên trực tung
        if (Count[k]>0) return; // Nếu còn trong phần phủ, thoát
        Dy[k]=Dy[2*k]+Dy[2*k+1];
        // Đã hết một phần phủ thì cập nhật độ dài hình chiếu
        return;
    }
    if (i+1>j) { // Đến lá của cây thì khởi trị lại độ dài hình chiếu bằng 0, thoát
        Dy[k]=0;
        return;
    }
    mid=(i+j)/2;
    Ra(y1, y2, 2*k, i, mid); // Duyệt xuống con trái
    Ra(y1, y2, 2*k+1, mid, j); // Duyệt xuống con phải
    if (Count[k]==0) Dy[k]=Dy[2*k]+Dy[2*k+1]; // Cập nhật độ dài hình chiếu
}
void Solve() {
    long long area=0;
    Vao(D[0].y1, D[0].y2, 1, 0, 2*n-1); // Duyệt khe đứng thứ nhất, bắt đầu từ gốc cây
}

```

```

        for (int i=1; i!=2*n; i++) { //Duyệt các khe đứng tiếp theo
            int dx=D[i].x-D[i-1].x; //Độ rộng khe đứng
            area += dx*Dy[1]; //Diện tích phần phủ trong khe đứng
            if (D[i].loai==1) //Gặp đỉnh trái-dưới của một hình chữ nhật thì gọi xử lý Vao()
                Vao(D[i].y1,D[i].y2,1,0,2*n-1);
            else //Gặp đỉnh phải-trên của một hình chữ nhật thì gọi xử lý Ra()
                Ra(D[i].y1,D[i].y2,1,0,2*n-1);
        }
        fout << area << endl;
        fout.close();
    }

int main() {
    read_input();
    Solve();
    return 0;
}

```

Câu b) Phần chung của N hình chữ nhật (có các cạnh trên các đường thẳng cùng phuơng với các trục toạ độ). Nếu tồn tại là hình chữ nhật có toạ độ đỉnh trái-dưới là $(x_1; y_1)$ và toạ độ đỉnh phải-trên là $(x_2; y_2)$ thì:

$x_1 = \text{Max}\{x_{1i}\}$, $x_2 = \text{Min}\{x_{2i}\}$, $y_1 = \text{Max}\{y_{1i}\}$, $y_2 = \text{Min}\{y_{2i}\}$ ($i = 1, 2, \dots, N$).

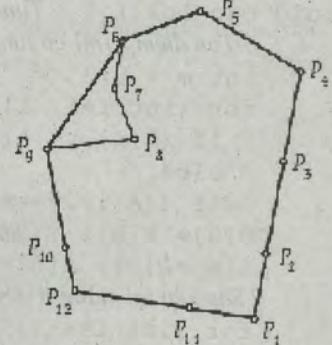
Nếu $x_1 \geq x_2$ hoặc $y_1 \geq y_2$ thì không tồn tại phần chung của N hình chữ nhật.

Chương trình: Học sinh tự lập trình.

8.4. Tìm bao lồi của tập N điểm đã cho. Sau đó xét các cặp hai đỉnh bao lồi, tính khoảng cách của chúng chọn ra khoảng cách xa nhất.

Sau đây là kĩ thuật quét Graham tìm bao lồi:

- Tìm một đa giác đơn $p_1p_2\dots p_n$ không tự cắt bằng cách sắp xếp các đỉnh tăng theo góc tạo bởi $\overrightarrow{Op_i}$ tạo với $\vec{e} = (1; 0)$ trong đó thường chọn O là gốc toạ độ.
- Đi trên đa giác đơn bắt đầu từ cạnh p_1p_2 , dựa vào hiện tượng thay đổi chiều $r\vec{e}$ (phải, trái) để loại bỏ những điểm trên đa giác đơn. Ví dụ trên cạnh đi từ đỉnh p_8 sang đỉnh p_9 là đổi chiều $r\vec{e}$ nên bỏ các cạnh nối p_6-p_7 , p_7-p_8 , p_8-p_9 thay bằng một cạnh p_6-p_9 .



Một thể hiện cụ thể khác của kĩ thuật này đã nêu trong phần tìm bao lồi ở chương trình của bài 8.2

Chương trình:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <math.h>
#include <iomanip>
using namespace std;
int n;
struct TPoint {
    double x, y;
} u,v, A[10000], B[10000];
ifstream fin ("bai04_c8.inp");
ofstream fout ("bai04_c8.out");
void read_input(){
    fin >> n;
    for (int i=0; i!=n; i++) {
        fin >> A[i].x >> A[i].y;
    }
    fin.close();
}
double tichcheo ( TPoint a1, TPoint a2, TPoint b1, TPoint b2) {
// Tích chéo của  $\vec{a_1a_2}$  và  $\vec{b_1b_2}$ 
    return ( (a2.x-a1.x)*(b2.y-b1.y)-(b2.x-b1.x)*(a2.y-a1.y) );
}
double kc(TPoint a1, TPoint a2) { // Khoảng cách hai điểm a1 và a2
    return ((a2.x-a1.x)*(a2.x-a1.x) + ((a2.y-a1.y)*(a2.y-a1.y)));
}
void bao_loi() { // Tìm bao lồi của tập điểm A[0], A[1], ...A[N-1]
// Tìm điểm A[m] có tung độ nhỏ nhất mà hành độ lớn nhất
    int m = 0;
    for (int i=1; i!=n; i++)
        if (A[i].y< A[m].y) m=i;
        else
            if ((A[i].y==A[m].y) && (A[i].x>A[m].x)) m=i;
    B[0]= A[m]; // Chọn A[m] làm điểm đầu tiên nạp vào bao lồi B
    A[m]=A[0]; A[0]=B[0]; // Chuyển đổi A[m] và A[0]
// Sắp xếp lại mảng A để đường gấp khúc đi từ A[0], A[1],... là đa giác đơn
    for (int i=n-1; i>1; i--)
        for (int j=1; j<=i-1; j++) {
            double tc=tichcheo(A[0], A[j], A[0], A[j+1]);
            double kc1,kc2;
            kc1 = kc (A[0], A[j]);
            kc2 = kc (A[0], A[j+1]);
            if ((tc<0) || ((tc==0) && (kc1>kc2))) {
                B[n-1] =A[j];
            }
        }
}
```

```

        A[j] = A[j+1];
        A[j+1] = B[n-1];
    }

    // Chọn điểm thứ hai và thứ ba vào bao lồi B
    int i=1;
    while (tichcheo(A[0], A[i+1], A[0], A[1]) == 0) i++;
    B[1] = A[i];
    B[2] = A[i+1];
    // Chọn các điểm tiếp theo
    int k=2;
    for (int j=i+2; j<n; j++) {
        while (tichcheo(B[k-1], B[k], B[k], A[j]) <= 0) k--;
        k++;
        B[k] = A[j];
    }
    n=k; // Số đỉnh của bao lồi
}

void xa_nhat() { // Tìm hai điểm xa nhau nhất trong tập điểm đã cho
    double max=-1, lx1, lx2, ly1, ly2;
    for (int i=0; i<=n-1; i++)
        for (int j=i+1; j<=n; j++) {
            double kcl= kc(B[i], B[j]);
            if (kcl>max) {
                max = kcl;
                lx1=B[i].x; lx2=B[j].x;
                ly1=B[i].y; ly2=B[j].y;
            }
        }
    cout << fixed << setprecision(2);
    cout << sqrt(max) << endl; // Khoảng cách hai điểm xa nhau nhất và toạ độ
    cout << lx1 << " " << ly1 << " " << lx2 << " " << ly2 <<
    endl;
    fout.close();
}

int main() {
    read_input();
    bao_loi();
    xa_nhat();
    return 0;
}

```

8.5. Bài toán thuộc loại NP-Hard, chỉ tiếp cận lời giải tối ưu khi kích thước N lớn.

Giả sử chọn một giá trị đường kính lớn nhất của các tập con là $R = R_0$. Dùng duyệt, lần lượt nạp (phân hoạch) các điểm vào các tập sao cho:

- Khoảng cách hai điểm bất kì trong cùng một tập con không vượt quá R_0 . (a)
- Không có điểm nào thuộc hai tập con khác nhau và mỗi điểm đều phải thuộc một tập con.
- Số tập con là K . Khi duyệt hết N điểm nếu số tập thỏa mãn điều kiện (a) chưa đủ K thì vẫn dừng duyệt vì từ những tập này dễ dàng tách thành đủ K tập thỏa mãn (a).

Nếu với R_0 phân hoạch N điểm thành K tập thỏa mãn thì giảm R_0 , nếu không phân hoạch được thì tăng R_0 . Quá trình này lặp bằng tìm kiếm nhị phân để có giá trị R_0 bé nhất.

Chương trình: Học sinh tự lập trình.

8.6. Thuật toán: Khi N không lớn, vét cạn mọi cặp điểm (A_i, A_k). Xét mọi cặp điểm, với mỗi cặp, tính tích chéo giữa $\overrightarrow{A_i A_k}, \overrightarrow{A_j A_k}$ đặt $c1$ là số lần tích chéo âm thì $c1$ bằng số điểm ở một phía nào đó của đường thẳng $A_i A_k$, đặt $c2$ là số lần tích chéo dương thì $c2$ bằng số điểm ở phía còn lại. Nếu hiệu số giữa n và số điểm thuộc đường thẳng $A_i A_k$ là chẵn và $c2 = c1$ thì dừng ngay quá trình vét cạn.

Nếu hiệu số giữa n và số điểm thuộc đường thẳng $A_i A_k$ là lẻ và $\text{abs}(c2 - c1) = 1$ thì cũng dừng ngay quá trình vét cạn.

Khi N khá lớn, giải gần đúng như sau: Gọi tập N điểm đã cho là tập $X = \{A_0, A_1, A_2, \dots, A_{N-1}\}$. Chọn điểm có tung độ nhỏ nhất mà hoành độ lớn nhất tráo đổi cho A_0 , sau đó xây dựng đa giác đơn $A_0B_1B_2\dots B_{N-1}$ sao cho các tia $A_0B_1, A_0B_2, \dots, A_0B_{N-1}$ lần lượt quay ngược chiều quay kim đồng hồ (các điểm B_1, B_2, \dots, B_{N-1} chính là các vị trí mới sau sắp xếp của A_1, A_2, \dots, A_{N-1} (như phần đầu của thuật toán tìm bao lồi trình bày trong bài 8.4).

Gọi $m = N/2$. Đường thẳng qua A_0A_m có nhiều triển vọng là đường thẳng cần tìm.

Sau đó đếm số điểm của tập X nằm trên đường thẳng A_0A_m . Gọi số điểm này là t , thì ở một phía của đường thẳng A_0A_m có $m + 1 - t$ điểm, phía kia là $n - t$ điểm, vậy cần điều chỉnh quay thêm tia A_0A_m theo chiều ngược chiều quay kim đồng hồ đến vị trí mới A_0A_i . Lại kiểm tra lại yêu cầu số điểm hai bên A_0A_i đã chênh nhau ít nhất chưa (số điểm còn lại không thuộc A_0A_i nếu lẻ thì chênh nhau 1, nếu chẵn thì bằng nhau thì dừng tìm kiếm).

Dùng tìm kiếm nhị phân để xác định điểm A_i nằm giữa A_m và A_{N-1} .

Chương trình:

```
Chương trình 1 (Vết cạn)
#include <fstream>
using namespace std;
int n, t;
struct TPoint {
    double x, y;
} u,v, tmp, A[10000], B[10000];
ifstream fin ("bai06_c8.inp");
ofstream fout ("bai06_c8.out");

void read_input(){
    double x, y;
    fin >> n;
    for (int i=0; i!=n; i++) {
        fin >> A[i].x >> A[i].y;
    }
    fin.close();
}

double tichcheo ( TPoint a1, TPoint a2, TPoint b1, TPoint b2) {
    return ( (a2.x-a1.x)*(b2.y-b1.y)-(b2.x-b1.x)*(a2.y-a1.y) );
}

int dem (int i, int k, int &t) {
    int c=0;
    t=0;
    for (int j=0; j<n; j++)
        if (tichcheo(A[0], A[i], A[0], A[j])>0)  c++;
        else if (tichcheo(A[0], A[i], A[0], A[j])<0)  c--;
        else t++;
    return c;
}

int main() {
    read_input();
    int li, lk, chenh=n;
    for (int i=0; i!=n-1; i++)
        for (int k=i+1; k!=n; k++) {
            int x=dem(i,k,t);
            if (x<chenh) {
                chenh=x; li =i; lk=k;
            }
            if (chenh==0) break;
        }
    fout << A[li].x << " " << A[li].y << endl;
    fout << A[lk].x << " " << A[lk].y << endl;
    fout.close();
    return 0;
}
```

0. (a)
đều phải
kiện (a)
hành đú
không
n để có
nơi cặp
héo âm
lần tích
số điểm
c1) = 1
A0, A1,
đổi cho
B2,...,
..., BN-1
ia thuật
àn tìm.
này là t,
t điểm,
đồng hồ
ã chênh
u 1, nếu

```

}

Chương trình 2 (giải gần đúng khi N lớn)
#include <iostream>
#include <fstream>
#include <vector> .
using namespace std;
int n, t;
struct TPoint {
    double x, y;
} u,v, tmp, A[10000], B[10000];
ifstream fin ("bai06_c8.inp");
ofstream fout ("bai06_c8.out");

void read_input(){
    double x, y;
    fin >> n;
    for (int i=0; i!=n; i++) {
        fin >> A[i].x >> A[i].y;
    }
    fin.close();
}
double tichcheo ( TPoint a1, TPoint a2, TPoint b1, TPoint b2) {
    return ( (a2.x-a1.x)*(b2.y-b1.y)-(b2.x-b1.x)*(a2.y-a1.y) );
}
double kc(TPoint a1, TPoint a2) {
    return ((a2.x-a1.x)*(a2.x-a1.x)+((a2.y-a1.y)*(a2.y-a1.y)))
};
}
void dagiacdon() {
    int m = 0;
    for (int i=1; i!=n; i++)
        if (A[i].y< A[m].y) m=i;
        else
            if ((A[i].y==A[m].y) && (A[i].x>A[m].x)) m=i;
    tmp= A[m];
    A[m]=A[0]; A[0]=tmp;
    for (int i=n-1; i>1; i--)
        for (int j=1; j<=i-1; j++) {
            double tc=tichcheo(A[0], A[j], A[0], A[j+1]);
            double kc1,kc2;
            kc1 = kc (A[0], A[j]);
            kc2 = kc (A[0], A[j+1]);
            if (((tc<0)||((tc==0)&&(kc1>kc2)))) {
                tmp =A[j];
                A[j] = A[j+1];
                A[j+1] = tmp;
            }
        }
}

```

```

    }

int dem (int i, int &t) {
    int c=0;
    t=0;
    for (int j=0; j<n; j++)
        if (tichcheo(A[0], A[i], A[0], A[j])>0) c++;
        else if (tichcheo(A[0], A[i], A[0], A[j])<0) c--;
        else t++;
    return c;
}

int main() {
    read_input();
    dagiacd();
    int i, left=0, right=n-1;
    while (left<right) {
        i= (left+right)/2;
        int x=dem(i, t);
        if ( ((n-t)%2==0) && (x==0) ) break;
        if ( ((n-t)%2==1) && ((x==1)|| (x==-1)) ) break;
        if (x<0) right=i+1;
        else left=i;
    }
    cout << A[0].x << " " << A[0].y << endl;
    cout << A[i].x << " " << A[i].y << endl; fout.close();
    return 0;
}

```

8.7. Đây là một bài toán còn mở, V. Kumar, S. Arya, and H. Ramesh xếp bài toán này vào loại APX-Hard. Vì vậy đến thời điểm này, chỉ có thể nói về thuật toán tiếp cận tối ưu.

Thuật toán:

- Xây dựng đồ thị G từ N điểm đã cho: Mỗi cặp điểm là một đỉnh của đồ thị G. Đỉnh A = {(x₁; y₁), (x₂; y₂)} có cạnh nối với đỉnh B = {(x₃; y₃), (x₄; y₄)}. Khi hai đỉnh A và B có chung một điểm và vectơ $\vec{u} = (x_2 - x_1; y_2 - y_1) = (u_1; u_2)$ cùng phương với vectơ $\vec{v} = (x_4 - x_3; y_4 - y_3) = (v_1; v_2)$, nghĩa là $u_1.v_2 - u_2.v_1 = 0$.
- Trên đồ thị G, xây dựng đồ thị hai phía tìm bộ phủ đường của G.
- Từ bộ phủ đường trên đồ thị G, với mỗi đường tìm ra các điểm thuộc một đường thẳng (gọi là “dẫn” đường). Quá trình này cần kết hợp với kĩ thuật tham để số đường thẳng tìm được là tương đối ít nhất (ví dụ dẫn những

đường có số đỉnh nhiều nhất trước sau đó cập nhật lại số điểm còn trên những đường còn lại của bộ phủ để chọn ra đường cần dẫn tiếp theo).

Một hướng tiếp cận khác là: Xây dựng đồ thị thích hợp và áp dụng luồng Min Cost. Theo hướng này cần kết hợp với tìm kiếm nhị phân để chọn trọng số W (lượng tối đa cho phép tải qua - là số đường thăng) cho cung đi từ đỉnh giả (đỉnh 0) tới các đỉnh của đồ thị và giá chi phí C trên các cung này bằng 0. Trên các cung khác W = 1, C = 1.

Chương trình: (Sử dụng thuật toán tìm bộ phủ đường + Greedy)

```
#include <iostream>
#include <fstream>
#define maxn 101
#define maxm 5000
#define e 1.0E-4;
using namespace std;
int nn, n, m, nList;
int Adj[maxm], Link[maxm]; // Tạo danh sách kè trên đồ thị G
int Head[maxn], A[maxn], B[maxn], List[maxn];
bool found, Avail[maxn];
struct diem {
    int x, y, sh;
} D[maxn]; // Quản lý N điểm đã cho
struct dinh {
    diem d1, d2;
} Dinh[maxm]; // Quản lý các đỉnh của đồ thị G
int L[maxm], L2[maxm]; // Lưu các tập đường trong bộ phủ đường

ifstream fin ("BAI07_C8.INP");
ofstream fout ("BAI07_C8.OUT");

bool chung (int x, int y) { // Hai đỉnh x và y của G có điểm nào chung hay không
    bool tmp;
    tmp = ( (Dinh[x].d1.x == Dinh[y].d1.x) &&
            (Dinh[x].d1.y == Dinh[y].d1.y) ) ;
    tmp = (tmp || ( (Dinh[x].d1.x == Dinh[y].d2.x) &&
                    (Dinh[x].d1.y == Dinh[y].d2.y) ) );
    tmp = (tmp || ( (Dinh[x].d2.x == Dinh[y].d1.x) &&
                    (Dinh[x].d2.y == Dinh[y].d1.y) ) );
    tmp = (tmp || ( (Dinh[x].d2.x == Dinh[y].d2.x) &&
                    (Dinh[x].d2.y == Dinh[y].d2.y) ) );
    return tmp;
}
bool cungphuong (int x, int y) { // Hai vectơ tương ứng với hai đỉnh x và y cùng phương
    double a1, b1, a2, b2;
```

```

còn trên
Dinh[x].d1.y;
Dinh[y].d1.y;
double tich=a1*b2-a2*b1;
if (tich<0) tich = -tich;
return tich<e;

bool thanghang(int x, int y) { //Có cạnh nối đỉnh x và đỉnh y của G
    return (chung(x,y) && cungphuong(x,y));
}

void Read_input() {
    int x, y;
    fin >> n;
    nn=n; //nn: lưu lại số điểm đã cho, để n nhận vai trò mới là số đỉnh của G
    for (int i=1; i<=n; i++) {
        fin >> D[i].x >> D[i].y;
        D[i].sh = i;
    }
    fin.close();
    //Tạo đồ thị G
    m=0;
    for (int i=1; i<n; i++)
        for (int j=i+1; j<=n; j++) {
            m++;
            Dinh[m].d1 = D[i];
            Dinh[m].d2 = D[j];
        }
    n=m; //n: số đỉnh của đồ thị G
    int k=0; //Tạo và đếm số cạnh của G, đồng thời xây dựng danh sách kề
    for (int i=1; i<m; i++)
        for (int j=i+1; j<=m; j++)
            if ( thanghang(i,j) ) {
                k++;
                Adj[k] = j;
                Link[k] = Head[i];
                Head[i] = k;
            }
    m=k; //m: số cạnh của G

}

void Init() {
    for (int i=1; i<=n; i++) { //Khởi tạo đồ thị hai phía (A, B) xây dựng từ đồ thị G
        A[i]=0; B[i]=0; //Màu các đỉnh là nhạt
    }
    for (int i=1; i<=n; i++)
        List[i] = i; //Tập phu đường ban đầu (gồm n đường)
    nList = n; //Số đường của bộ phu ban đầu
}

```

```

}

void Visit (int x) { // Thăm từ đỉnh x
    int i, j;
    i = Head[x];
    while (i!=0) { // duyệt mọi cung xuất phát từ x
        j = Adj[i]; // đỉnh kè
        if (Avail[j]) { // chưa thăm j
            Avail[j] = false; // đánh dấu đã thăm
            if (B[j]==0)
                found = true; // kết thúc một đường mở
            else Visit(B[j]); // còn không thăm tiếp từ B[j]
            if (found) { // Nếu có đường mở thì ghép x-y
                B[j] = x;
                A[x] = j;
                return;
            }
        }
        i = Link[i]; // chọn cung tiếp theo
    }
}

void FindAugmentingPath() { // Tìm đồng thời nhiều đường mở
    int old;
    do {
        old = nList; // Lưu lại số
        for (int i=1; i<=n; i++)
            Avail[i] = true; // các đỉnh chưa thăm
        for (int i=nList; i>=1; i--) {
            found = false;
            Visit(List[i]);
            if (found) { // nếu có đường mở thì loại bỏ i trong danh sách List
                List[i] = List[nList];
                nList--;
            }
        }
    } while (!(old==nList)); // danh sách không thể tăng
}

// Dẫn các đường của bộ phủ, tìm ra các đường thăng chứa N điểm đã cho
void Write_output() {
    // Tuỳ theo cách chọn thuật toán thăm, có thể tìm được số đường thăng gần với tối ưu
}

int main() {
    Read_input();
    Init();
    FindAugmentingPath();
    Write_output();
    return 0;
}

```

8.8. Xây dựng đồ thị đầy đủ với các đỉnh là các điểm trong tập điểm đã cho, trọng số của một cạnh là khoảng cách giữa hai điểm là hai đầu mút của cạnh đó. Sau đó dùng thuật toán tìm cây khung có trọng số nhỏ nhất của đồ thị.

Chương trình:

```
#include <iostream>
#include <fstream>
#include <math.h>
#define maxn 1000
#define vc 1.0E+30
using namespace std;
double a[maxn][maxn]; // Ma trận khoảng cách giữa các cặp điểm
int d[maxn]; // Đánh dấu đỉnh đã nạp vào cây khung
int d1[maxn], d2[maxn]; // Lưu 2 đầu mút cạnh được nạp vào cây khung
int t[maxn]; // Nhãn đỉnh trước
double v[maxn]; // Nhãn khoảng cách
int n, i0, dem; // n: số đỉnh, i0: đỉnh vừa nạp, dem: đếm số cạnh nạp vào cây khung
struct diem {
    double x, y;
} E[maxn]; // Quản lý N đỉnh đã cho
ifstream fin ("BAI08_C8.INP");
ofstream fout ("BAI08_C8.OUT");

double dist(int i, int j) { // Tính khoảng cách hai đỉnh
    return sqrt((E[i].x-E[j].x)*(E[i].x-E[j].x)+(E[i].y-E[j].y)*(E[i].y-E[j].y)));
}

void read_input() { // Đọc dữ liệu vào và xây dựng ma trận trọng số
    int i, j;
    fin >> n;
    for (int k=0; k!=n; k++) {
        fin >> E[k].x >> E[k].y;
    }
    for (int i=0; i!=n-1; i++)
        for (int j=i+1; j!=n; j++) {
            a[i][j] = dist(i, j);
            a[j][i] = a[i][j];
        }
    fin.close();
}

void step1() { // Khởi trị, nạp đỉnh đầu tiên vào cây khung là đỉnh 0
    d[0] = 1; i0 = 0; v[0] = 0; t[0] = -1;
    for (int i=1; i!=n; i++) {
        if (a[0][i] == 0) {
            v[i] = vc;
            t[i] = -1;
        }
    }
}
```

```

        }
    else {
        v[i] = a[0][i];
        t[i] = 0;
    }
}
}

void add() { //Thêm một đỉnh vào cây khung
    int li=0;
    double min = vc;
    for (int i=0; i!=n; i++)
        if (d[i]==0)
            if (v[i]<min) {
                min = v[i];
                li = i;
            }
    i0 = li;
    d[i0] = 1; //Nạp i0, có định nhãn i0
    dem++;
    d1[dem] = t[i0]; //Lưu cạnh vừa tạo thành
    d2[dem] = i0;
}

void repair() { //Sửa nhãn các đỉnh còn nhãn tự do kề với i0
    for (int i=0; i<n; i++)
        if (a[i0][i]>0)
            if ((v[i]>a[i0][i]) && (d[i]==0)) {
                v[i] = a[i0][i];
                t[i] = i0;
            }
}

void solve() { //Thuật toán tìm cây khung trọng số nhỏ nhất bằng Prim+Gán nhãn
    dem = 0;
    step1();
    do {
        add();
        repair();
    } while (dem<n-1); //Đến n-1 cạnh thì hoàn thành cây khung
}

void write_output() {
    double tong=0.0000;
    for (int i=1; i<n; i++)
        tong += a[d1[i]][d2[i]];
    fout << tong << endl;
    fout.close();
}

int main(){
    read_input();

```

```

solve();
write_output();
system("pause");
return 0;

```

8.9. Với kích thước N không lớn và toạ độ các đỉnh là số nguyên không vượt quá 10^4 , có thể dùng phương pháp phủ các ô vuông đơn vị của mặt phẳng toạ độ nằm trong các hình chữ nhật bởi màu xanh (số 1), các ô chưa được phủ là số 0. Sau đó đếm số ô số 0 kề cạnh mỗi ô số 1 sẽ được các đoạn đơn vị nằm trên đường bao của phủ các hình chữ nhật. Khi đếm các ô, chỉ cần quét trên hình chữ nhật có góc trái-dưới là $(xmin-1, ymin-1)$, góc phải-trên là $(xmax+1, ymax+1)$, trong đó $xmin = \text{Min}\{x_1, x_2\}$, $xmax = \text{Max}\{x_1, x_2\}$, $ymin = \text{Min}\{y_1, y_2\}$, $ymax = \text{Max}\{y_1, y_2\}$ mà $(x_1; y_1)$ và $(x_2; y_2)$ là các toạ độ đỉnh các hình chữ nhật đã cho. Cũng có thể đồng thời tích luỹ các ô 1 thành diện tích.

Với kích thước N lớn, có thể thay đổi thuật toán nêu ở bài 8.3 (tính diện tích phần phủ bởi N hình chữ nhật) khi thay một số lệnh liên quan đến tìm diện tích thành lệnh phục vụ tìm chu vi.

Chương trình:

```

#include <iostream>
#include <iostream>
#include <vector>
#define maxv 10000
using namespace std;
int H[maxv][maxv];
int n;
int xmin=20000, ymin=20000, xmax=-20000, ymax=-20000;
ifstream fin ("BAI09_C8.INP");
ofstream fout ("BAI09_C8.OUT");

int dem(int x, int y) {
    int d=0;
    if (H[x+1][y]==0) d++;
    if (H[x-1][y]==0) d++;
    if (H[x][y+1]==0) d++;
    if (H[x][y-1]==0) d++;
    return d;
}

void phu(int x1, int y1, int x2, int y2) {
    for (int i=x1; i!=x2; i++)
        for (int j=y1; j!=y2; j++) H[i][j]=1;
}

```

```

}

void read_input() {
    int x1, y1, x2, y2;
    fin >> n;
    for (int i=0; i!=n; i++) {
        fin >> x1 >> y1 >> x2 >> y2;
        if (x1>x2) { int x=x1; x1=x2; x2=x; }
        if (y1>y2) { int y=y1; y1=y2; y2=y; }
        x1 += maxv/2; y1 += maxv/2;
        x2 += maxv/2; y2 += maxv/2;
        if (xmin>x1) xmin=x1; if (xmin>x2) xmin=x2;
        if (ymin>y1) ymin=y1; if (ymin>y2) ymin=y2;
        if (xmax<x1) xmax=x1; if (xmax<x2) xmax=x2;
        if (ymax<y1) ymax=y1; if (ymax<y2) ymax=y2;
        phu(x1,y1,x2,y2);
    }
    xmin--; ymin--; xmax++; ymax++;
}

int main() {
    read_input();
    int chuvi=0;
    // int dientich=0;
    for (int i=xmin; i!=xmax; i++)
        for (int j=ymin; j!=ymax; j++)
            if (H[i][j]==1) {
                //dientich++;
                chuvi += dem(i,j);
            }
    cout << "Dien tich: " << dientich << endl;
    cout << "Chu vi: " << chuvi << endl;
    system("pause");
    return 0;
}

```

8.10. Xét tập G gồm các góc hợp bởi các tia OA_i (A_i là các đầu mút của các đoạn thẳng đã cho, $i = 0, 1, \dots, A_{2N-2}$). Gọi α là góc giữa hai tia liên tiếp, hai tia này tạo với trực hoành hai góc tương ứng là góc k_1 và k_2 mà $k_2 \geq k_1$. Đoạn thẳng thứ i là A_jA_{j+1} ($j = 2*i$) ứng với hai tia OA_j và OA_{j+1} , hai tia này tạo với trực hoành góc g_1 và g_2 mà $g_1 \geq g_2$. Đoạn thẳng A_jA_{j+1} phủ góc α khi $g_1 \geq k_2$ và $k_1 \geq g_2$. Từ nhận xét này, ta có thuật toán:

- Sắp tăng tập G;

- Lần lượt xét các góc tạo bởi hai tia liên tiếp, đánh dấu đoạn thẳng phủ hết góc, gần O nhất;
- Đếm lại những đoạn thẳng được đánh dấu.

Chương trình:

```

#include <iostream>
#include <fstream>
#include <vector>
#include <math.h>
#define MaxXY 20000
#define MaxN 1000

using namespace std;
int N;
struct TPoint {
    int x, y;
    int dt;
} O, A[2*MaxN]; // Tập đỉnh  $A_j$  của  $N$  đoạn thẳng

vector <double> G; // Tập các góc tạo bởi các tia  $\overrightarrow{OA_j}$  với trục hoành
double C[MaxN]; // Tập các hoành độ giao điểm của các đoạn thẳng kéo dài cắt Ox
int Res[MaxN]; // Lưu số hiệu các đoạn thẳng nhìn thấy
ifstream fin ("bai10_c8.inp");
ofstream fout ("bai10_c8.out");

double theta(TPoint p1, TPoint p2) {
    double dx, dy, ax, ay, t;
    dx=p2.x-p1.x; dy=p2.y-p1.y;
    ax=dx; if (ax<0) ax=-ax;
    ay=dy; if (ay<0) ay=-ay;
    if ((dx==0)&&(dy==0)) t= 0;
    else t= dy/(ax+ay);
    if (dx<0) t=2-t;
    else if (dy<0) t=4+t;
    return t*90;
}

void read_input(){
    O.x=0; O.y=0; G.reserve(2*MaxN);
    fin >> N;
    for (int i=0; i!=N; i++) {
        int j= 2*i;
        int x1, y1, x2, y2;
        fin >> x1 >> y1 >> x2 >> y2;
        if (x1>x2) {
    
```

```

        int tmp = x1; x1 = x2; x2=tmp;
        tmp=y1; y1=y2; y2=tmp;
    }
    A[j].x = x1; A[j].y = y1; A[j].dt = i;
    A[j+1].x = x2; A[j+1].y = y2; A[j+1].dt = i;
    double g= theta(O, A[j]); G.push_back(g);
    g= theta(O, A[j+1]); G.push_back(g);
    C[i] = (x1*y2 - x2*y1)/(y2-y1);
}
fin.close();
sort(G.begin(), G.end());
}

void Solve () {
    double k1, k2;
    vector< double >::iterator j;
    j=G.begin();
    while (j!=G.end()-1) { //Xét các cặp tia liên tiếp tạo với Ox góc k1 và k2
        k1 = *j;
        k2 = *(j+1);
        cout << k1 << " " << k2 << endl;
        j++;
        int Li;
        double minC=1000000000;
        for (int i=0; i!=N; i++) { //Xét mọi đoạn thẳng, tìm đoạn phù góc k1-k2
            int k=2*i;
            double g1=theta(O, A[k]), g2=theta(O, A[k+1]);
            if ((g1>=k2) && (k1>=g2) ) //Điều kiện phù
            if (C[i]<minC) { //Tìm đoạn phù gần gốc toạ độ nhất
                minC= C[i];
                Li = i;
            }
        }
        Res[Li]++;
    } //Lưu đoạn nhín thấy vào mảng Res
}
int count = 0;
for (int i=0; i!=N; i++)
    if (Res[i]>0) {
        count++;
    }
fout << count << endl;
fout.close();
}

int main() {
    read_input();
    Solve();
    return 0;
}

```

8.11. Do các cạnh bị xoá là những đoạn song song với trục tung nên hai điểm đầu mút của nó phải có hoành độ bằng nhau. Tại một đầu mút A($x_i; y_i$), ta thấy đầu mút B($x_k; y_k$) phải có tung độ gần tung độ của A nhất khi xét các điểm cùng hoành độ với A, nghĩa là: $x_i = x_k$ và y_k gần y_i nhất.

Dựa vào kết quả của bài toán kiểm tra vị trí của một điểm với miền đa giác, có thể suy ra một trong hai cách nối A - B như sau:

- Nếu số điểm có hoành độ bằng x_i và tung độ lớn hơn y_i là số lẻ thì nối A với B($x_k; y_k$) có $x_i = x_k$ và y_k lớn hơn và gần y_i nhất.
- Nếu số điểm có hoành độ bằng x_i , có tung độ lớn hơn y_i là số chẵn thì nối AB mà B($x_k; y_k$) có $x_i = x_k$ và y_k nhỏ hơn và gần y_i nhất.

Chương trình:

```
I và k2
#include <iostream>
#include <fstream>
#include <vector>
#define tichcheo(u, v) (u.x*v.y-u.y*v.x)
#define MaxN 1001
#define MaxXY 10
using namespace std;
vector < int > X[2*MaxXY];
//Là mảng các vector, mỗi vector X[x] chứa các tung độ các điểm có hoành độ bằng x
int N;
struct diem {
    int x, y; //tọa độ đỉnh
    int sh; //Số hiệu ban đầu của các đỉnh
    int sh_new; //Số hiệu của các đỉnh theo thứ tự các đỉnh liên tiếp trên đa giác
} D[2*MaxN]; //Đa giác chứa các điểm đầu mứt của N đoạn nằm ngang
int SH[2*MaxXY][2*MaxXY]; //Chứa SH[x][y] là số hiệu của đỉnh (x; y)
ifstream fin ("BAI11_C8.INP");
ofstream fout ("BAI11_C8.OUT");
void read_input() {
    for (int i=0; i!=2*MaxXY; i++)
        X[i].reserve(2*MaxXY);
    fin >> N;
    for (int i=0; i!=N; i++) {
        int j=2*i;
        int x1, y1, x2, y2;
        fin >> x1 >> y1 >> x2 >> y2;
        if (y2<y1) {int tmp = y1; y1=y2; y2=tmp;}
        x1 += MaxXY; y1 += MaxXY; //Thêm vào các tọa độ để các tọa độ không âm
        x2 += MaxXY; y2 += MaxXY;
        D[j].x = x1; D[j].y = y1;
        D[j+1].x = x2; D[j+1].y = y2;
```

```

        D[j].sh = j; D[j+1].sh = j+1;
        X[x1].push_back(y1);
        X[x2].push_back(y2);
        SH[D[j].x][D[j].y]=j;
        SH[D[j+1].x][D[j+1].y]=j+1;
    }
    fin.close();
}
int cmp (const void* pt1, const void* pt2) { //So sánh theo số hiệu mới
    diem t1, t2;
    t1 = *(diem*) pt1;
    t2 = *(diem*) pt2;
    return t1.sh_new-t2.sh_new;
}
double dientich() { //Tính diện tích đa giác
    double S=0;
    int nb = 2*N;
    D[nb]= D[0];
    for (int i=0; i!=nb; i++)
        S += tichcheo(D[i],D[i+1]);
    if (S<0) S=-S;
    S = S/2;
    return S;
}
void Solve() { //Thuật toán
    int i = 0; //Bắt đầu tìm từ đỉnh có số hiệu cũ là 0
    D[0].sh_new = 0; //Xác nhận số thứ tự các đỉnh bắt đầu từ 0
    int shNew=0;
    do {
        int x=D[i].x, y= D[i].y;
        vector<int>::iterator j;
        sort(X[x].begin(), X[x].end()); //Sắp tăng các tung độ trong vector X[x]
        j=X[x].begin();
        while ((j!=X[x].end()) && (*j < y) ) j++;
        int so_y_lon_hon = X[x].end()-j-1; //Số tung độ lớn hơn tung độ điểm
        int next;
        if (so_y_lon_hon % 2 == 1)
            next.= SH[x][*(j+1)]; //next : số hiệu đỉnh tiếp theo (ở phía trên)
        else
            next = SH[x][*(j-1)]; //số hiệu đỉnh tiếp theo (ở phía dưới)
        shNew++; //Số hiệu mới của đỉnh next
        D[next].sh_new = shNew; //Xác nhận số hiệu mới
        if (next%2==0) next++; //Số hiệu đỉnh tiếp theo (sang phải)
        else next--; //Số hiệu đỉnh tiếp theo (sang trái)
        shNew++; //Số hiệu mới của đỉnh next
        D[next].sh_new = shNew; //Xác nhận số hiệu mới
        i = next; //Chuyển tiếp sang điểm tiếp theo trong 2N điểm
    }
}

```

```

    } while (i!=0); //Duyệt hết đa giác (quay về đỉnh 0)
    //Sắp tăng theo số hiệu mới sẽ được thứ tự các đỉnh của D là liên tiếp
    qsort(& D, 2*N, sizeof(diem), cmp);
    fout << dientich() << endl; //Xuất ra diện tích đa giác của Bờm
    //Tim cách xuất các đỉnh trong D (đã có thứ tự liên tiếp) theo đúng chiều quay kim đồng hồ
    //Tim đỉnh li có tung độ thấp nhất, mà hoành độ lớn nhất (đỉnh thấp nhất, bên phải)
    int li, lj, ymin=3*MaxXY;
    for (int i=0; i!=2*N; i++)
        if ((ymin> D[i].y) || ((ymin== D[i].y) && (D[li].x < D[i].x))) {
            ymin = D[i].y;
            li = i;
        }
    lj = li-1; //Điểm lj là điểm ngay trước li theo thứ tự liên tiếp
    if (lj<0) lj=2*N-1; //Để phòng trường hợp li=0
    int tc = D[li].x - D[lj].x; //Chênh lệch hoành độ của li và lj
    if (tc == 0) { //Chiều thứ tự trong D cũng là chiều thuận kim đồng hồ
        for (int i=0; i!=2*N ; i++)
            fout << D[i].x-MaxXY << " " << D[i].y-MaxXY << endl;
    }
    else // Chiều trong D ngược chiều kim đồng hồ
        for (int i=2*N-1; i>=0 ; i--)
            fout << D[i].x-MaxXY << " " << D[i].y-MaxXY << endl;
    fout.close();
}

int main() {
    read_input();
    Solve();
    system("pause");
    return 0;
}

```

$X[x]$
điểm i
trên)

8.12. Đường chân trời coi như trục hoành, hình chiếu của mỗi ngọn núi thứ i ($i = 1, 2, \dots, N$) xuống trục hoành là một đoạn thẳng $[x_{1i}; x_{2i}]$. Sắp xếp tăng theo các x_{1i} của đầu mút trái các đoạn thẳng này. Lần lượt xét từng đoạn thẳng i (i từ 1 đến N), nếu đầu mút phải x_{2i} của nó không vượt quá đầu mút phải của các đoạn đã xét là $\text{Max}\{x_{2k} \text{ với mọi } k < i\}$ thì không nhìn thấy được đoạn thẳng i.

Nên tính sẵn $\text{Max}\{x_{2k} \text{ với mọi } k < i\}$ với mọi $i = 1, 2, \dots, N$.

Chương trình:

```

#include <iostream>
#include <fstream>
#define MaxN 10000
using namespace std;
int N;

```

```

struct nui {
    int x1, x2;
    int right;
} A[MaxN];
ifstream fin ("BAI12_C8.INP");
ofstream fout ("BAI12_C8.OUT");

int cmp (const void* pt1, const void* pt2) {
    nui t1, t2;
    t1 = *(nui*) pt1;
    t2 = *(nui*) pt2;
    return t1.x1-t2.x1;
}

int main() {
    int MaxR=0, x, h;
    fin >> N;
    for (int i=0; i!=N; i++) {
        fin >> x >> h;
        A[i].x1 = x-h;
        A[i].x2 = x+h;
    }
    fin.close();
    qsort(&A, N, sizeof(nui), cmp); //Sắp tăng theo hoành độ đầu mút trái đoạn thẳng
    for (int i=0; i!=N; i++) { //Tìm biên phải hoành độ Max của các đoạn thẳng trước
        if (A[i].x2> MaxR) MaxR= A[i].x2;
        A[i].right=MaxR;
    }
    // Đếm số ngọn núi nhìn thấy được
    int view=1;
    for (int i=1; i!=N; i++)
        if (A[i].x2>A[i-1].right) view++;
    fout << view << endl;
    fout.close();
    return 0;
}

```

8.13.

Giả sử đáy của đài phun nước là hình tròn (I_f ; R_f) có tâm là I_f và bán kính là R_f nằm trong sân tiền sảnh $X \times Y$. Nếu chưa kể đến các cột trụ trong sân tiền sảnh thì I_f có thể nằm hoàn toàn tùy ý trong một hình chữ nhật ABCD có toạ độ góc trái-dưới là $(x_1; y_1) = (m; m)$ và toạ độ góc phải-trên là $(x_2; y_2) = (X - m; Y - m)$ với $m \geq R_f$.

Giá trị lớn nhất của m không vượt quá $right = \text{Min}\{X, Y\}$, giá trị nhỏ nhất của m có thể coi là $left = 0$.

Do tồn tại các cột trụ trong sân tiền sảnh không được vi phạm nên cần thu hẹp hình chữ nhật ABCD (bằng cách tăng m), nghĩa là cần chọn một giá trị $m \geq R_f$ thích hợp nào đó làm giá trị gần đúng của R_f .

Khi m càng tăng tiến tới right thì hình chữ nhật ABCD càng thu hẹp về một điểm. Do tâm I_f chỉ chọn trong hình chữ nhật nhỏ bé này, nên lúc đó coi hình chữ nhật ABCD là đại diện cho tâm I_f . Vậy bốn đỉnh của hình chữ nhật ABCD lúc này phải cách tâm đáy các cột trụ tròn một khoảng không nhỏ hơn $R_i + m$ (R_i là bán kính đáy cột trụ tròn thứ i). Khi m đã đạt được giá trị lớn nhất mà hình chữ nhật ABCD có thể chấp nhận được thì $I_f = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$ và coi $R_f \approx M$ vì khi đó I_f rất gần đỉnh (m; m).

Ta xây dựng thủ tục kiểm tra hình chữ nhật ABCD có được chấp nhận hay không. Hình chữ nhật ABCD được chấp nhận khi bốn đỉnh của nó (coi như đại diện cho vị trí tâm của đài phun nước) phải cách tâm đáy của mọi cột trụ i một khoảng không nhỏ hơn tổng $R_i + m$. Nếu không được chấp nhận thì phải giảm m, nếu được chấp nhận thì tăng m. Vậy nên tìm kiếm nhị phân giá trị m với giá trị bên trái ban đầu của khoảng tìm kiếm giá trị cho m là left = 0 và giá trị bên phải ban đầu cho khoảng này là right = Min{X, Y}.

Ngoài ra để tăng hiệu suất, khi kiểm tra hình chữ nhật ABCD (tâm I), ta kiểm tra cả bốn hình chữ nhật con của ABCD được tạo thành do hai đường thẳng qua I song song với cạnh hình chữ nhật này chia ABCD.

Chương trình:

```
#include <iostream>
#include <fstream>
#include <math.h>
#include <iomanip>
#define maxn 1000
#define e 1.0E-4
#define min (a,b) (ab);
using namespace std;
double x[maxn], y[maxn], r[maxn];
double xg, yg, xf, yf, m;
int n;
ifstream fin ("BAI13_C8.INP");
ofstream fout ("BAI13_C8.OUT");
void read_input(){
    fin >> xg >> yg;
```

```

        fin >> n;
        for (int i=0; i!=n; i++)
            fin >> x[i] >> y[i] >> r[i];
        fin.close();
    }
    double dis(double x1, double y1, double x2, double y2) {
        return sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
    }
    double abs(double x) {
        if (x<0) x = -x;
        return x;
    }
    bool check(double x1, double y1, double x2, double y2) {
        if ( (abs(x2-x1)<e) && (abs(y1-y2)<e) ) {
            xf = (x1+x2)/2;
            yf = (y1+y2)/2;
            return false;
        }
        for (int i=0; i!=n; i++) {
            double d = r[i] + m;
            if ( (dis(x1,y1,x[i],y[i]) <= d) &&
                (dis(x1,y2,x[i],y[i]) <= d) &&
                (dis(x2,y1,x[i],y[i]) <= d) &&
                (dis(x2,y2,x[i],y[i]) <= d) )
                return true;
        }
        return check (x1,y1,(x1+x2)/2,(y1+y2)/2) &&
            check ((x1+x2)/2,y1,x2,(y1+y2)/2) &&
            check (x1,(y1+y2)/2,(x1+x2)/2,y2) &&
            check ((x1+x2)/2,(y1+y2)/2,x2,y2);
    }
    void solve() {
        double left, right;
        left=0;
        if (xg<yg) right = xg/2;
        else right=yg/2;
        while (abs (right-left) > e) {
            m = (left + right)/2;
            if (check(m,m,xg-m, yg-m)) right =m;
            else left = m;
        }
        cout << fixed << setprecision(3);
        cout << xf << " " << yf << " " << m << endl;
        fout << fixed << setprecision(3);
        fout << xf << " " << yf << " " << m << endl;
        fout.close();
    }
}

```

```

int main() {
    read_input();
    solve();
    system("pause");
    return 0;
}

```

8.14. Có thể dùng phương pháp “Vi phân hình học”: Đầu tiên tìm một hình chữ nhật nhỏ nhất phủ kín mọi tam giác. Sau đó chia nhỏ dần hình chữ nhật này thành bốn hình chữ nhật bằng hai đường trung bình của nó. Quá trình chia nhỏ là một quá trình đệ quy, cho dừng khi bốn đỉnh của những hình chữ nhật con tạo ra đã rất sát nhau. Trong quá trình chia nhỏ, hình chữ nhật con nào lọt vào một trong n tam giác thì được coi là thuộc phần phủ của N tam giác, diện tích của hình chữ nhật này được cộng thêm vào diện tích phủ đang tìm của n tam giác.

Một hình chữ nhật con thuộc vào một tam giác khi và chỉ khi tất cả bốn đỉnh của hình chữ nhật con này thuộc tam giác. Vậy cần dùng bài toán cơ sở: “Kiểm tra một điểm M thuộc tam giác ABC hay không”. Giải bài toán cơ sở này có thể dựa vào khái niệm diện tích và tích chéo như đã nêu trong hướng dẫn của bài 8.1. Cũng có thể giải theo cách sử dụng hàm ccw để định hướng đi $M \rightarrow A \rightarrow B$, $M \rightarrow B \rightarrow C$, $M \rightarrow C \rightarrow A$ nếu ba định hướng cùng dấu hoặc bằng 0 thì M ở trong tam giác ABC.

Chương trình:

```

#include <iostream>
#include <fstream>
#include <vector>
#include <iomanip>
#define e 1.0E-4 //Đo độ sát gần nhau của các điểm
using namespace std;
int n;
vector <long double> xA, xB, xC, yA, yB, yC;
long double area; //Diện tích vùng phủ bởi các tam giác
long double xmin, ymin, xmax, ymax;
ifstream fin ("bai14_c8.inp");
ofstream fout ("bai14_c8.out");
void read_input() {
    fin >> n;
    xA.reserve(n); yA.reserve(n);
    xB.reserve(n); yB.reserve(n);
    xC.reserve(n); yC.reserve(n);
    long double x1,x2,x3,y1,y2,y3;
    for (int i=0; i!=n; i++) { //Đọc tọa độ các đỉnh của N tam giác

```

```

        fin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
        // Nạp tọa độ các đỉnh tam giác thứ i vào các vector xA, yA, xB, yB, xC, yC
        xA.push_back(x1); yA.push_back(y1);
        xB.push_back(x2); yB.push_back(y2);
        xC.push_back(x3); yC.push_back(y3);
    }
    fin.close();
}
int ccw(long double x1, long double y1, long double x2, long double
y2,
        long double x3, long double y3) {
    // A(x1; y1), B(x2; y2), C(x3; y3). Hàm định hướng đi từ A đến B đến C
    long double dx1, dy1, dx2, dy2, t;
    dx1 = x2 - x1; dy1 = y2 - y1;
    dx2 = x3 - x1; dy2 = y3 - y1;
    t = dx1 * dy2 - dx2 * dy1;
    if (t > 0) return 1; // A đến B đến C ngược chiều kim đồng hồ
    else
        if (t == 0) return 0; // A, B, C thẳng hàng
        else return -1; // A đến B đến C thuận chiều kim đồng hồ
}
bool in_triangle(long double x, long double y, int k) {
    // Điểm M(x; y) có trong tam giác thứ k hay không
    int t12, t23, t31;
    t12 = ccw(x, y, xA[k], yA[k], xB[k], yB[k]); // chiều M → A → B
    t23 = ccw(x, y, xB[k], yB[k], xC[k], yC[k]); // chiều M → B → C
    t31 = ccw(x, y, xC[k], yC[k], xA[k], yA[k]); // chiều M → C → A
    return ((t12 >= 0) && (t23 >= 0) && (t31 >= 0)) ||
           ((t12 <= 0) && (t23 <= 0) && (t31 <= 0));
}
bool in_rectangle(long double x, long double y, long double x1,
long double y1, long double x2, long double y2) {
    // Điểm M(x; y) có trong hình chữ nhật xác định bởi góc trái-dưới (x1; y1) và góc phải-trên (x2; y2)?
    return (x >= x1) && (x <= x2) && (y >= y1) && (y <= y2);
}
bool cross(long double x1, long double y1, long double x2, long
double y2,
           long double x3, long double y3, long double x4, long double
y4) {
    // A(x1; y1), B(x2; y2), C(x3; y3), D(x4; y4), đoạn thẳng AB có cắt đoạn thẳng CD hay không?
    return (ccw(x1, y1, x3, y3, x4, y4) * ccw(x2, y2, x3, y3, x4, y4) < 0) && (ccw(x3, y3, x1, y1, x2, y2) * ccw(x4, y4, x1, y1, x2, y2) < 0);
}
int check(long double x1, long double y1, long double x2, long double
y2, int k) {

```

* Kiểm tra hình chữ nhật có góc trái-dưới là $(x1; y1)$ và góc phải-trên $(x2; y2)$ có vị trí thế nào so với tam giác thứ k hay không? */

```
if ((x2-x1<ε) || (y2-y1<ε)) //HVN con đã đủ nhỏ, không chia nữa
    return 2;
```

//HVN còn nằm trong tam giác k

```
if ( (in_triangle(x1, y1, k)) && (in_triangle(x2, y1, k)) &&
    (in_triangle(x1, y2, k)) && (in_triangle(x2, y2, k)) )
    return 2;
```

//Tam giác thứ k nằm hoàn toàn trong HVN, còn phải chia nhỏ HVN

```
if ((in_rectangle(xA[k], yA[k], x1, y1, x2, y2)) &&
    (in_rectangle(xB[k], yB[k], x1, y1, x2, y2)) &&
    (in_rectangle(xC[k], yC[k], x1, y1, x2, y2)) )
    return 1;
```

//Tam giác có một cạnh cắt một cạnh của hình chữ nhật, còn phải chia nhỏ HVN

```
if (cross(xA[k], yA[k], xB[k], yB[k], x1, y1, x2, y2) ||
    cross(xA[k], yA[k], xB[k], yB[k], x2, y1, x1, y2) ||
    cross(xA[k], yA[k], xB[k], yB[k], x2, y2, x1, y2) ||
    cross(xA[k], yA[k], xB[k], yB[k], x1, y2, x1, y1) ||
    cross(xB[k], yB[k], xC[k], yC[k], x1, y1, x2, y1) ||
    cross(xB[k], yB[k], xC[k], yC[k], x2, y1, x1, y2) ||
    cross(xB[k], yB[k], xC[k], yC[k], x2, y2, x1, y1) ||
    cross(xB[k], yB[k], xC[k], yC[k], x1, y2, x1, y1) ||
    cross(xC[k], yC[k], xA[k], yA[k], x1, y1, x2, y1) ||
    cross(xC[k], yC[k], xA[k], yA[k], x2, y1, x1, y2) ||
    cross(xC[k], yC[k], xA[k], yA[k], x1, y2, x1, y1) ||
    cross(xC[k], yC[k], xA[k], yA[k], x2, y2, x1, y1) )
    return 1;
```

```
return 0;
```

```
void add(long double x1, long double y1, long double x2, long double y2) {
```

```
    bool next=false; //Cờ báo: còn phải chia nhỏ HVN
```

```
    int k;
```

```
    long double x, y;
```

```
    for (int i=0; i!=n; i++) { //Duyệt các tam giác
```

```
        k = check(x1, y1, x2, y2, i); //Kiểm tra vị trí của HVN và tam giác i
```

```
        if (k==2) { //HVN nằm trong tam giác
```

```
            area += (x2-x1)*(y2-y1); //Cộng diện tích HVN vào vùng phủ
```

```
            return ;
```

```
}
```

```
    next = next || (k==1); //Xác nhận lại giá trị của cờ báo
```

```
}
```

```
if (!next) return ; //Không cần chia nhỏ nữa thì dừng
```

```
    x = (x1+x2)/2; //Toạ độ tâm HVN là (x,y)
```

```
    y=(y1+y2)/2;
```

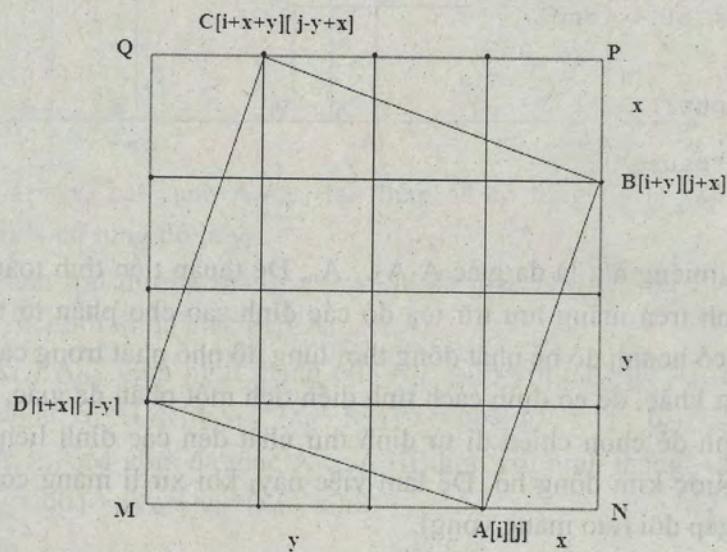
```
    add(x1, y1, x, y); //Đề quy với HVN đã chia nhỏ ở góc trái-dưới
```

```

        add(x, y1, x2, y); //Đệ quy với HCN đã chia nhỏ ở góc phải-dưới
        add(x1, y, x, y2); //Đệ quy với HCN đã chia nhỏ ở góc trái-trên
        add(x, y, x2, y2); //Đệ quy với HCN đã chia nhỏ ở góc phải-trên
    }
    void caculate() {
        //Tim HCN nhỏ nhất phủ mọi tam giác
        xmax = -1001; ymax=-1001;
        xmin = 1001; ymin=1001;
        for (int i=0; i!=n; i++) {
            if (xA[i]<xmin) xmin = xA[i];
            if (xB[i]<xmin) xmin = xB[i];
            if (xC[i]<xmin) xmin = xC[i];
            if (yA[i]<ymin) ymin = yA[i];
            if (yB[i]<ymin) ymin = yB[i];
            if (yC[i]<ymin) ymin = yC[i];
            if (xA[i]>xmax) xmax = xA[i];
            if (xB[i]>xmax) xmax = xB[i];
            if (xC[i]>xmax) xmax = xC[i];
            if (yA[i]>ymax) ymax = yA[i];
            if (yB[i]>ymax) ymax = yB[i];
            if (yC[i]>ymax) ymax = yC[i];
        }
        // Thực hiện thuật toán vi phân hình học bằng đệ quy chia nhỏ dần HCN ban đầu
        add(xmin, ymin, xmax, ymax);
    }
    int main() {
        read_input();
        caculate();
        fout << fixed << setprecision(2) << area << endl;
        fout.close();
        return 0;
    }
}

```

8.15. Giả sử ABCD là hình vuông có bốn đỉnh cùng màu. Vẽ các đường thẳng song song với các trục toạ độ và qua bốn đỉnh hình vuông ABCD, chúng tạo thành hình vuông MNPQ. Để thấy bốn tam giác vuông thuộc MNPQ và nằm ngoài ABCD là bằng nhau. Gọi điểm A thuộc dòng i, cột j và đặt $AN = x$, $MA = y$ thì B thuộc dòng $i + y$, cột $j + x$, C thuộc dòng $i + x + y$, cột $j - y + x$, D thuộc dòng $i + x$, cột $j - y$ với điều kiện bốn đỉnh A, B, C, D thuộc phạm vi lưới ô vuông (nghĩa là: $0 \leq i, j < n$, $j - y \geq 0$, $j + x < n$, $i + x + y \leq n$, $x \geq 0$, $y \geq 0$). Bằng cách duyệt mọi giá trị của i, j, x, y thoả mãn điều kiện này sẽ tìm được số hình vuông có bốn đỉnh cùng màu.



Chương trình:

```
#include <iostream>
#include <fstream>
#define Max 51
using namespace std;
int A[Max][Max];
int n;
long long sl;
ifstream fin ("BAI15_08.inp");
ofstream fout ("BAI15_08.OUT");
void read_input() {
    fin >> n;
    for (int i=0; i!=n; i++)
        for (int j=0; j!=n; j++)
            fin >> A[i][j];
    fin.close();
}
```

```

}

void solve() {
    sl=0;
    for (int i=0; i!=n; i++)
        for (int j=0; j!=n; j++)
            for (int y=1; y<=j-1; y++)
                for (int x=0; x<=n-y-i; x++)
                    if (j+x <= n)
                        if ( (A[i][j]==A[i+x][j-y]) &&
                            (A[i][j]==A[i+x+y][j-y+x]) &&
                            (A[i][j]==A[i+y][j+x]) )
                            sl++;
    fout << sl << endl;
}
int main() {
    read_input();
    solve();
    system("pause");
    return 0;
}

```

8.16. Giả sử miếng đất là đa giác $A_1A_2\dots A_n$. Để thuận tiện tính toán cần đổi lại thứ tự các đỉnh trên mảng lưu trữ toạ độ các đỉnh sao cho phần tử thứ nhất của mảng là đỉnh có hoành độ bé nhất đồng thời tung độ nhỏ nhất trong các điểm cùng hoành độ. Mặt khác, để cố định cách tính diện tích một phần đa giác, cũng đổi lại thứ tự các đỉnh để chọn chiều đi từ đỉnh thứ nhất đến các đỉnh liên tiếp sau nó theo chiều ngược kim đồng hồ. Để làm việc này, khi xử lý mảng có thể kéo dài mảng toạ độ gấp đôi (tạo mảng vòng).

$$\text{Diện tích đa giác } A_1A_2\dots A_n \text{ là } S = \sum_{i=1}^n \frac{(y_{i+1} + y_i)(x_{i+1} - x_i)}{2}$$

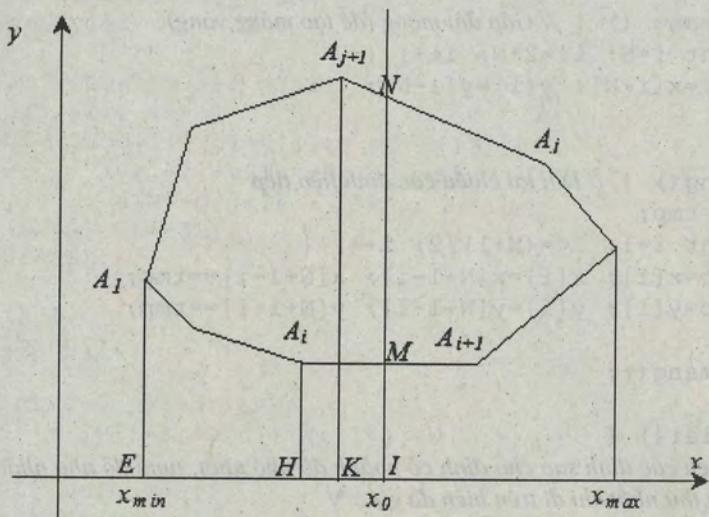
Xây dựng mảng một chiều $d(N)$ mà $d[1] = 0$ và có:

$$d[i] = \sum_{k=1}^{i-1} \frac{(y_{k+1} + y_k)(x_{k+1} - x_k)}{2} \text{ thì diện tích đa giác } A_1A_2\dots A_n \text{ là } S = |d[n+1]|.$$

Gọi hoành độ nhỏ nhất là x_{\min} , hoành độ lớn nhất là x_{\max} . Bằng tìm kiếm nhị phân, tìm giá trị x_0 nằm giữa x_{\min} và x_{\max} sao cho đường thẳng $x = x_0$ chia đa giác thành hai phần có diện tích xấp xỉ nhau.

Ta tính diện tích phần bên trái, nếu nhỏ hơn $S/2$ thì tăng x_{\min} lên x_0 , ngược lại nếu diện tích phần bên trái lớn hơn $S/2$ thì giảm x_{\max} xuống x_0 .

Vấn đề còn lại là tính diện tích phần bên trái như thế nào? Chúng ta quan sát hình vẽ sau:



Đường thẳng $x = x_0$ cắt cạnh A_iA_{i+1} tại điểm M có tung độ là y_M và cắt cạnh A_jA_{j+1} tại điểm N có tung độ là y_N .

Phần đa giác bên trái đường thẳng $x = x_0$ là đa giác $NMA_{i+1}...A_1...A_{j+1}$ bằng đa giác $NIEA_1...A_{j+1}$ bớt đi đa giác $MIEA_1...A_i$.

Đa giác $NIEA_1...A_{j+1}$ gồm hình thang $NIKA_{j+1}$ hợp với đa giác $A_{j+1}KEA_1...A_{j+1}$ nên có diện tích là: $S_1 = (y_{j+1} + y_N) * (x_0 - x_{j+1}) / 2 + \text{abs}(d[n+1] - d[j+1])$.

Đa giác $MIEA_1...A_iM$ gồm đa giác $A_1...A_iHE$ hợp với hình thang A_iMIH nên có diện tích là: $S_2 = d[i] + (y_M + y_i) * (x_0 - x_i) / 2$.

Chương trình:

```
#include <iostream>
#include <fstream>
#define maxN 10001
#define e 1.0E-6
using namespace std;
double x[maxN], y[maxN], d[maxN];
int N;
double S, yy;
double x0, xmin, xmax;
ifstream fin ("BAI16_08.INP");
ofstream fout ("BAI16_08.OUT");
void Read_input() {
    fin >> N;
    for (int i=0; i!=N; i++) // Đọc tọa độ các đỉnh liên tiếp của mảnh đất đa giác
        fin >> x[i] >> y[i];
    fin.close();
```

```

void keodaimang () { //Gấp đôi mảng (để tạo mảng vòng)
    for (int i=N; i!=2*N; i++) {
        x[i]=x[i-N]; y[i]=y[i-N];
    }
}
void doihiuong() { //Đổi lại chiều các đỉnh liên tiếp
    double tmp;
    for (int i=1; i<=(N+1)/2; i++) {
        tmp=x[i]; x[i]=x[N+1-i]; x[N+1-i]==tmp;
        tmp=y[i]; y[i]=y[N+1-i]; y[N+1-i]==tmp;
    }
    keodaimang();
}
void danhsolai() {
/* Đánh lại số hiệu các đỉnh sao cho đỉnh có hành độ nhỏ nhất, tung độ nhỏ nhất ở bên trái nhá
đóng vai trò đỉnh thứ nhất khi đi trên biên đa giác */
    //Đổi lại số hiệu các đỉnh
    int u=0;
    for (int i=1; i!=N; i++)
        if ( (x[i]<x[u]) ||
            ((x[i]==x[u]) && (y[i]<y[u]))) u=i;
    //Tìm xmin
    xmin=x[u];
    for (int i=0; i!=N; i++) {
        x[i]=x[u+i]; y[i]=y[u+i];
    }
    //Tìm xmax
    xmax=x[0];
    for (int i=1; i!=N; i++)
        if (xmax<x[i]) xmax=x[i];
    }
    bool chuacat(int k){
//Tim chi so k de duong thang x=x0 cat canh A_k A_{k+1}
        if ( ((x[k]<x0)&&(x[k+1]<x0)) ||
            ((x[k]>x0)&&(x[k+1]>x0)) ) return true;
        if (x[k+1]==x0) return true;
        return false;
    }
    void Giao(int k, double & yy){
//Tim tung do yy cua giao diem giua duong thang A_k A_{k+1} voi duong thang x=x0
        yy=(y[k+1]-y[k])*(x0-x[k])/(x[k+1]-x[k]) + y[k];
    }
    double S_Left() { //Tinh dien tich phan da giác bên trái đường thẳng x=x0
        int i,j;
        double y1,y2;
        i=0;

```

```

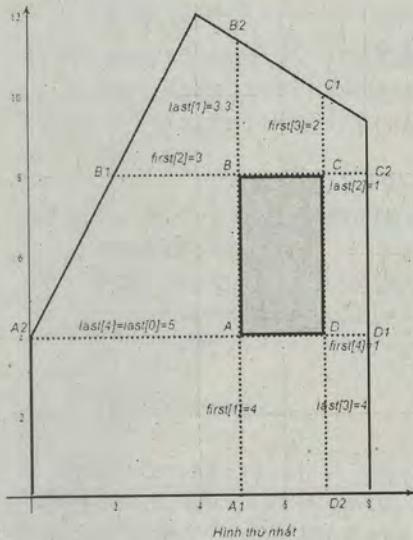
        while (chuacat(i)) i++;
        Giao(i, y1);
        j=i+1;
        while (chuacat(j)) j++;
        Giao(j, y2);
        double SL=d[i]+(y[i]+y[i])*(x0-x[i])/2+
            (y[j+1]+y2)*(x[j+1]-x0)/2 +
            d[N]-d[j+1];
        if (SL<0) SL=-SL;
        return SL;

void Solve() {
    keodaimang();
    S=0;
    for (int i=0; i!=N; i++)
        S += (y[i+1]+y[i])*(x[i+1]-x[i]);
    S=S/2;
    if (S>0) doihiuong();
    if (S<0) S=-S;
    danhsolai();
    d[0]=0;
    for (int i=1;i<=N; i++) //Xây dựng mảng d
        d[i] = d[i-1] + (y[i]+y[i-1])*(x[i]-x[i-1])/2;
    //Tim kiém nhì phân giá trị x0
    do {
        x0=(xmin+xmax)/2;
        double SL=S_Left();
        if (SL<S/2) xmin=x0;
        else xmax=x0;
    } while ((xmax-xmin)>=e);
    x0=(xmax+xmin)/2;
    fout << x0 << endl;
    fout.close();

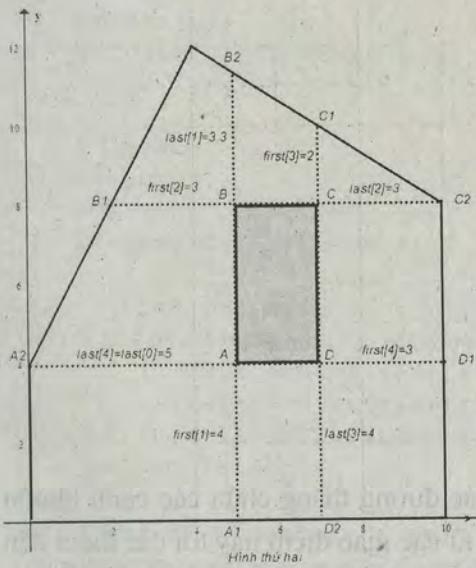
int main() {
    Read_input();
    Solve();
    return 0;
}

```

- 3.17.** Trước hết cần tìm các giao điểm của các đường thẳng chứa các cạnh khuôn với các cạnh của tấm thép và tính khoảng cách từ các giao điểm này tới các điểm đầu cạnh tương ứng của khuôn. Trong hình vẽ bên, cạnh thứ nhất của khuôn là AB. Đường thẳng AB cắt các cạnh tấm thép tại A₁ và B₂. Ta gán first[1] = AA₁ và last[1] = BB₂. Tương tự đường thẳng chứa cạnh thứ hai của khuôn là BC cắt các cạnh tấm thép tại B₁ và C₂. Ta gán first[2] = BB₁ và last[2] = CC₂. Tương tự tại các đỉnh C và D... Cuối cùng để thuận tiện lập trình, thêm first[0] = first[4], last[0] = last[4].



là 1 hoặc 2 thì các nhát cắt được chọn tới các đỉnh khuôn chỉ theo một loại là A_1A , B_1B , C_1C và D_1D hoặc chỉ theo loại kia là A_2A , B_2B , C_2C và D_2D . Nếu giá trị cờ báo là 3 thì các nhát cắt được chọn tới đỉnh có lỗn cả hai loại. Ta xét từng trường hợp:



này bằng $B_1C_2 + BA_1 + AD_1 + DC = (AA_1 + BB_1 + CC_2 + DD_1) + (AB + BC + CD + DA)$ = tổng 4 nhát cắt được chọn tới 4 đỉnh + chu vi khuôn.

Khi flag = 1 hoặc flag = 2: (Xem hình vẽ thứ hai, flag = 1). Trong những trường hợp này, không có hai nhát cắt nào trong các nhát cắt đã chọn tới đỉnh là thẳ

Khi cắt tấm thép để tạo khuôn, tất nhiên phải có các nhát cắt dọc trên các cạnh của khuôn (vì các cạnh này không trùng cạnh tấm thép). Ngoài ra còn các nhát cắt tới các đỉnh của khuôn. Tới đỉnh A của khuôn ta chỉ cắt theo một trong hai đường A_1A hoặc A_2A . Nếu $A_2A < A_1A$ thì cắt theo A_2A và ghi nhận cờ báo sự việc này là flag := flag or 2, ngược lại nếu $A_2A \geq A_1A$ thì cắt theo đường A_1A thì ghi nhận flag := flag or 1. Tương tự tại các đỉnh B, C và D cũng tiến hành như vậy. Do đặc điểm của phép toán or ($1 \text{ or } 1 = 1$, $2 \text{ or } 2 = 2$; $1 \text{ or } 2 = 3$, $3 \text{ or } 1 = 3$, $3 \text{ or } 2 = 3$) nên ta thấy: nếu cuối cùng giá trị của cờ báo flag

Khi flag = 3 (xem hình vẽ thứ nhất): Ta chọn nhát cắt đầu tiên là đường thẳng chéo một cạnh của khuôn và hai nhát cắt đã chọn tới hai điểm đầu cạnh này, rồi sau đó theo các nhát cắt đã chọn tới các đỉnh còn lại mà cắt dọc các cạnh còn lại của khuôn thép. Vậy trong trường hợp này, tổng các nhát cắt đã được chọn tới đỉnh cộng với chu vi khuôn thép chính là tổng độ dài nhỏ nhất phải cắt. Ví dụ: Trong hình vẽ thứ nhất: Do flag = 3 nên đầu tiên cắt theo đường B_1C_2 qua hai đỉnh B và C, sau đó cắt BA_1 , rồi AD_1 , cuối cùng là DC . Tổng

hàng nên để tạo nhát cắt đầu tiên ta cần để ý tới đỉnh khuôn nào có hiệu $\text{abs}(\text{first}[i]-\text{last}[i-1])$ nhỏ nhất và dành thay đổi một chút so với dự kiến đã chọn: đó là để cắt tới đỉnh này sẽ chọn nhát cắt dài hơn thay vì chọn nhát cắt ngắn hơn. Với ví dụ như hình vẽ thứ hai, ta chọn đỉnh B. Thay cho nhát cắt BB₁ ta chọn nhát cắt BB₂ để tạo nhát cắt đầu tiên là theo B₂B tới A₁, sau đó là AD₁ và DC₁, cuối cùng là cạnh CD của khuôn. Lưu ý do khi gán cờ flag, ta đã cộng dự định cắt theo B₁B (vì B₁B < BB₂) vào kết quả tổng độ dài đường cắt, nhưng bây giờ lại cắt theo BB₂ để tổng thể lợi hơn (vì tại các đỉnh khác của khuôn ta toàn chọn cắt theo đường ngắn hơn) nên phải điều chỉnh lại kết quả dự định ban đầu bằng cách bù thêm vào kết quả tổng độ dài đường cắt một lượng là BB₂-BB₁.

Chương trình:

```
#include <iostream>
#include <fstream>
#include <math.h>
#define maxN 2002
#define e 1.0E-6
#define VC 1.0E+15
using namespace std;
struct polygon {
    double x[maxN], y[maxN];
} a, b; //Đa giác tấm thép và đa giác khuôn thép
int n, m; //Số đỉnh tấm thép, số đỉnh khuôn thép
double first[maxN], last[maxN]; //Xem ý nghĩa trong phần hướng dẫn
double sum; //Tổng độ dài của nhát cắt tối ưu
int coincide[maxN]; //Xác nhận cạnh của khuôn thép có thuộc cạnh tấm thép không
ifstream fin("bai17_c8.inp");
ofstream fout ("bai17_c8.out");
void Read_input() { //Đọc dữ liệu vào (số cạnh, tọa độ các đỉnh của tấm thép và khuôn
    fin >> n;
    for (int i=1; i<=n; i++)
        fin >> a.x[i] >> a.y[i];
    a.x[n+1] = a.x[1]; a.y[n+1]=a.y[1];
    a.x[0]=a.x[n]; a.y[0]=a.y[n];
    fin >> m;
    for (int i=1; i<=m; i++)
        fin >> b.x[i] >> b.y[i];
    b.x[m+1] = b.x[1]; b.y[m+1]=b.y[1];
    b.x[0]=b.x[m]; b.y[0]=b.y[m];
    fin.close();
}
double abs(double x) {
    if (x<0) x=-x;
}
```

```

        return x;
    }

    int ccw(int u, int v) {
        // Xét chiều quay giữa vecto nối đỉnh u của tam thép với hai đỉnh v và v+1 của khuôn
        double dx1, dy1, dx2, dy2, tichcheo;
        dx1=b.x[v]-a.x[u]; dy1=b.y[v]-a.y[u];
        dx2=b.x[v+1]-a.x[u]; dy2=b.y[v+1]-a.y[u];
        tichcheo=dx1*dy2-dx2*dy1;
        if (abs(tichcheo)<e) return 0;
        else
            if (tichcheo<0) return 1;
            else return -1;
    }

    void line (double x1, double y1, double x2, double y2,
    double &aa, double &bb, double &cc) {
        // Xác định 3 hệ số a, b, c của phương trình tổng quát ax + by + c của đường thẳng qua (x1; y1) và (x2; y2)
        if (x1==x2) {
            aa=1; bb=0; cc=x1;
            return;
        }
        if (y1==y2) {
            aa=0; bb=1; cc=y1;
            return;
        }
        aa=y1-y2; bb=x2-x1;
        cc=y1*(x2-x1)-x1*(y2-y1);
    }

    double dist(double x1, double y1, double x2, double y2) {
        // Tính khoảng cách hai điểm
        return sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
    }

    void intersection_point(int u, int v) {
        // Xác định giao điểm cạnh nối đỉnh u và u + 1 của khuôn với cạnh nối đỉnh v và v + 1 của tam thép
        double xx, yy, dd, dx, dy, a1, b1, c1, a2, b2, c2;
        line(b.x[u], b.y[u], b.x[u+1], b.y[u+1], a1, b1, c1);
        line(a.x[v], a.y[v], a.x[v+1], a.y[v+1], a2, b2, c2);
        dd=a1*b2-a2*b1;
        dx=c1*b2-c2*b1;
        dy=a1*c2-a2*c1;
        xx=dx/dd; // Hoành độ giao điểm M
        yy=dy/dd; // Tung độ giao điểm M
        double i1=dist(xx,yy, b.x[u],b.y[u]); // Kc từ M đến đỉnh u của khuôn
        double i2=dist(xx,yy,b.x[u+1],b.y[u+1]); // Kc M đến đỉnh u + 1 của khuôn
        double i3=dist(b.x[u],b.y[u],b.x[u+1],b.y[u+1]); // Kc đỉnh u và u + 1
        if (abs(i1+i3-i2)<e) first[u]=i1; // Xác nhận M gần đỉnh u hơn, tạo first[]
        else last[u]=i2; // Xác nhận M gần đỉnh u+1 hơn, xây dựng mảng last[]
    }
}

```

```

void Solve() {
    for (int i=1; i<=m; i++) {
        for (int j=1; j<=n; j++) {
            int c1=ccw(j,i);
            int c2=ccw(j+1,i);
            if (c1*c2<=0) {
                Đường thẳng nối đỉnh i và i+1 của khuôn cắt đường thẳng nối đỉnh j và j+1 của tấm thép
                if ((c1==0) && (c2==0)) { //cạnh khuôn trên cạnh tấm thép
                    coincide[i]=1;
                    first[i]=0;
                    last[i]=0;
                    break;
                }
                else
                    intersection_point(i,j); //Tìm giao điểm hai đường thẳng nêu trên
            }
        }
    }
    first[m+1]=first[1]; last[m+1]=last[1];
    first[0]=first[m]; last[0]=last[m];
    //Thực hiện thuật toán trong hướng dẫn
    int flag=0;
    sum=0;
    double min=VC;
    for (int i=1; i<=m; i++) {
        if (abs(first[i]-last[i-1])<min)
            min=abs(first[i]-last[i-1]);
        if (first[i]<=last[i-1]) { //Cắt theo đoạn first[i]
            flag|= 1;
            sum += first[i];
        }
        else { //Cắt theo đoạn last[i-1]
            flag|= 2;
            sum += last[i-1];
        }
    }
    if (flag<3)
        sum += min; //Điều chỉnh lại dự định ban đầu
    //Cộng thêm chu vi khuôn
    for (int i=1; i<=m; i++)
        if (coincide[i]==0)
            sum += dist(b.x[i], b.y[i], b.x[i+1], b.y[i+1]);
    fout << sum << endl;
}

int main() {
    Read_input();
    Solve();
    system("pause");
    return 0;
}

```

8.18. Sắp xếp các tia tăng theo góc tạo bởi tia đó với chiều dương trục hoành. Có thể chứng minh giao điểm cao nhất của các cặp tia là giao điểm của hai tia kế nhau sau khi sắp xếp như trên.

Chú ý 1: tính góc tạo bởi một tia với trục hoành bằng hàm theta().

Chú ý 2: Xem lại bài toán giải tam giác ABC khi biết BC = a, cotB = c1, cotC = c2 thì khoảng cách từ A đến BC là $|a/(c1 - c2)|$.

Chương trình:

```
#include <fstream>
#include <vector>
#define maxN 10001
using namespace std;

ifstream fin ("BAI18_C8.INP");
ofstream fout ("BAI18_C8.OUT");
int N;
vector < double > x, z, t;
vector < pair <double,int > > g;
double best_h;
int t1, t2;
double angle(int i) { //Tính góc tạo bởi tia thứ i (i=0,...N-1) tạo với trục hoành
    double goc, dx, dy, ax, ay;
    dx = z[i]-x[i]; dy=t[i];
    ax = dx; if (ax<0) ax = -ax;
    ay = dy; if (ay<0) ay = -ay;
    if ((dx==0) && (dy==0)) goc = 0;
    else
        goc = dy/ (ax+ay);
    if (dx<0) goc = 2 - goc;
    else if (dy<0) goc= 4 + goc;
    return goc*90.0;
}
void Read_input() {
    double xi, zi, ti;
    fin >> N; // Số lượng tia
    x.reserve(N); z.reserve(N); t.reserve(N); g.reserve(N);
    for (int i=0; i!=N; i++) {
        fin >> xi >> zi >> ti; // Thông tin về tia i (i=0,1,...N-1)
        x.push_back(xi); z.push_back(zi); t.push_back(ti);
        double goc=angle(i);
        g.push_back(make_pair(goc, i)); // Góc và số hiệu của tia i
    }
    close();
}
```

```

n. Có
ia kẽ
tC =
nh

        Read_input();
        sort(g.begin(), g.end()); // Sắp tăng các tia theo góc tạo bởi tia và trục hoành
        best_h = -1;
        for (int i=0; i!=N-1; i++) {
            int tial = g[i].second; // số hiệu của tia thứ i sau khi xếp tăng
            int tia2 = g[i+1].second; // số hiệu của tia i+1 sau khi sắp tăng
            double b1=x[tial];
            double a1=(z[tial]-b1)/t[tial]; //cô tang của góc tạo bởi tia i và trục hoành
            double b2=x[tia2];
            double a2=(z[tia2]-b2)/t[tia2]; //cô tang của góc tạo bởi tia i+1 và trục hoành
            double H =(b2-b1)/ (a1-a2); //Độ cao của giao điểm giữa hai tia i và i+1
            if (H>best_h) { //Lưu lại độ cao, тоa độ giao điểm cao nhất
                best_h = H;
                t1 = tial;
                t2 = tia2;
            }
        }
        cout << best_h << endl; //Độ cao của giao điểm cao nhất
        if (best_h != -1)
        cout << t1+1 << " " << t2+1 << endl; // Số hiệu hai tia (tính từ 1 đến N
        fout.close();
        return 0;

```

8.19. Nếu đa giác A bao đa giác B thì hoành độ đỉnh trái nhất của A sẽ nhỏ hơn hoành độ đỉnh trái nhất của B. Với điều kiện của các đa giác nêu trong đề bài thì hoành độ đỉnh trái nhất của A nhỏ hơn hoành độ đỉnh trái nhất của B cũng là điều kiện đủ để A bao B.

Vậy có thuật toán:

- Tìm các hoành độ đỉnh trái nhất của mỗi đa giác.
- Sắp tăng mảng các hoành độ này.
- Khi đó đa giác ở vị trí thứ i ($i = 0, 1, \dots, N - 1$) của mảng hoành độ đã sắp tăng thì sẽ có i đa giác bao nó.

Chương trình:

```

#include <iostream>
#include <vector>
#define VC 32767
#define maxN 10000
using namespace std;
int N, ni;
vector<pair<long, int>> x;
ifstream fin ("BAI19_C8.INP");

```

```

ofstream fout ("BAI19_C8.OUT");
void Read_input() {
    int u, v;
    fin >> N;
    for (int i=0; i!=N; i++) {
        int xleft = VC;
        fin >> ni;
        for (int j=0; j!=ni; j++) {
            fin >> u >> v;
            if (u<xleft) xleft=u;
        }
        x.push_back(make_pair(xleft,i));
    }
}
int main() {
    Read_input();
    sort(x.begin(), x.end());
    int res[maxN];
    for (int i=0; i!=N; i++)
        res[x[i].second] = i;
    for (int i=0; i!=N; i++)
        fout << res[i] << endl;
    return 0;
}

```

8.20. Đầu tiên xác định khoảng cách hai tâm của hai vòng: Nếu chúng không giao nhau thì khoảng cách này tạm cho là vô cùng (vì có thể kéo chúng xa nhau mãi). Nếu giao nhau thì khoảng cách này bằng tổng hai bán kính (vì có thể kéo chúng đến vị trí tiếp xúc ngoài nhau).

Khi chọn hai vòng A và B bất kì kéo chúng theo hai hướng ngược nhau thì có thể liên quan tới vòng thứ ba là C nối với cả hai vòng A và B, vì vậy khoảng cách giữa hai vòng A và B có thể bị thay đổi như hai trường hợp:

- Khi A và B không giao nhau (khoảng cách là vô cùng) thì có thể kéo ba vòng thành hàng thẳng nên khoảng cách hai tâm A và B bằng khoảng cách hai tâm A và C cộng khoảng cách hai tâm C và B là $(RA + RC) + (RC + RB)$.
- Khi A và B cũng giao nhau thì khoảng cách này đã được xác định là khoảng cách hữu hạn, nay do bị ràng buộc bởi C nên phải xác định lại khoảng cách này: nếu khoảng cách này lớn hơn tổng khoảng cách hai tâm A và C với khoảng cách hai tâm C và B thì khoảng cách A và B phải bằng khoảng cách này.

Vậy trong cả hai trường hợp luôn thay khoảng cách hai tâm A và B bởi tổng khoảng cách hai tâm A và C với khoảng cách hai tâm C và B.

Cuối cùng duyệt lại các cặp hai tâm để tìm ra hai tâm cách xa nhau nhất. Kết quả cần cộng thêm hai bán kính của hai vòng có tâm kéo xa nhau nhất.

Chương trình:

```
#include <iostream>
#include <fstream>
#include <math.h>
#define maxN 502
#define VC 1.0E+30
using namespace std;
int N;
double x[maxN], y[maxN], r[maxN];
double dist[maxN][maxN];
double Res=0;
bool giao;
ifstream fin ("BAI20_C8.INP");
ofstream fout ("BAI20_C8.OUT");
double kc(int i, int j) {
    return sqrt((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j]));
}
void Read_input() {
    fin >> N;
    for (int i=0; i!=N; i++)
        fin >> x[i] >> y[i] >> r[i];
    fin.close();
}
void Solve() {
    for (int i=0; i!=N; i++) //Khởi tạo khoảng cách giữa hai vòng i và j
        for (int j=0; j!=N; j++)
            dist[i][j] = VC;
    for (int i=0; i!=N; i++) //Xác định khoảng cách giữa hai vòng i và j giao nhau
        for (int j=0; j!=N; j++) {
            if (i!=j)
                giao = kc(i,j) < r[i] + r[j]; //Điều kiện giao nhau
            if (giao)
                dist[i][j] = r[i] + r[j]; //Khoảng cách hai vòng giao nhau
        }
    //Điều chỉnh lại khoảng cách hai vòng i và j do vòng k giao với i và k giao với j
    for (int k=0; k!=N; k++) //Vòng thứ ba
        for (int i=0; i!=N; i++) { //Vòng thứ nhất
            if (dist[i][k]!=VC) //Vòng thứ nhất và thứ ba giao nhau
                for (int j=0; j!=N; j++) //Vòng thứ hai
                    if (dist[k][j]!=VC) //Vòng thứ hai và thứ ba giao nhau
                        if (dist[i][j] > dist[i][k] + dist[k][j]) //Điều kiện điều chỉnh
                            dist[i][j] = dist[i][k]+ dist[k][j] ; //Khoảng cách mới giữa i và j
```

```

        }
    // Tìm khoảng cách hai vòng xa nhất lưu vào biến Res
    for (int i=0; i!=N; i++)
        for (int j=0; j!=N; j++)
            if(i!=j)
                if ( (dist[i][j]!=VC) && (Res<dist[i][j]+r[i]+r[j]))
        )
            Res = dist[i][j]+r[i]+r[j];
    fout << Res << endl;
    cout << Res << endl;
    fout.close();
}
int main() {
    Read_input();
    Solve();
    return 0;
}

```

CHUYÊN ĐỀ 9

9.1. Xem ví dụ 2 (Tài liệu chuyên Tin học, Quyển 3, trang 49). Do đề bài 9.1 yêu cầu người *thua* là người lấy được quân cờ cuối cùng nên vị trí kết thúc của trò chơi này là 1 (số quân cờ trên bàn còn là C = 1). Vậy vị trí 1 thuộc tập P. Suy diễn tương tự ví dụ 2, tìm được các vị trí thuộc tập P là các vị trí có C quân mà (C – 1) chia hết cho (m + 1). Các vị trí còn lại thuộc tập N.

Chương trình:

```

#include <iostream>
using namespace std;
int c, m, m1, move;
bool ok; //cờ báo thắng, thua
void nhap() {
    cout << "Nhập số quân cờ ban đầu: ";
    cin >> c;
    cout << "Nhập số quân tối đa của một phép chuyển: ";
    cin >> m;
    m1=m;
    m1++;
}
void thuatthang() {
    move = (c-1)% m1; //Số quân lấy theo thuật thắng
    c -=move; //c là số quân cờ còn lại ứng với vị trí thuộc tập P: c-1 chia hết cho m+1
    cout<<"May chuyen so quan: "<<move
         <<". So quan con la : "<<c<<"\n";
    ok = !ok;
}

```

```

    }
void keodai() {
    c-=1; // c là số quân cờ còn lại sau khi lấy 1 quân (để kéo dài trò chơi)
    cout<<"May chuyen so quan: 1. So quan con la : "<<c<<"\n";
    ok = !ok;
}

void nguoidi() {
    cout<<"Ban chuyen bao nhieu quan? ";
    cin>>move; // Người chơi lấy move quân cờ (nhập từ bàn phím)
    c-=move;
    cout<<"So quan con lai la: "<<c<<"\n";
    ok = !ok;
}

int main() {
    nhap();
    ok=true;
    while (c>1) {
        nguoidi();
        if (c==1) break;
        if ((c-1)%m1!=0) thuathang();
        else keodai();
    }
    if (ok==true) cout<<"May thang\n";
    else cout<<"Ban thang\n";
    system("pause");
    return 0;
}

```

9.2. a) Gọi tập các vị trí là $X = \{0, 1, 2, \dots\}$ và tập trừ là $S = \{1, 3, 5, 7\}$. Dựa vào định nghĩa tập các vị trí có lợi P và tập các vị trí không có lợi N có thể chứng minh tập $P = \{x \in X | x \bmod 2 = 0\}$ và $N = X \setminus P = \{x \in X | x \bmod 2 = 1\}$.

b) Gọi tập các vị trí là $X = \{0, 1, 2, \dots\}$, tập trừ là $S = \{1, 3, 6\}$. Dựa vào định nghĩa tập các vị trí có lợi P và tập các vị trí không có lợi N có thể chứng minh $P = \{x \in X | x \text{ chia } 9 \text{ có dư là } 0, 2, 4\}$ và $N = \{x \in X | x \text{ chia } 9 \text{ có dư là } 1, 3, 5, 6, 7, 8\}$. Thực vậy: Trước hết rõ ràng thấy vị trí kết thúc là $0 \in P$. Giả sử $x \in N$, nếu x chia 9 dư là 1, 3, 5 thì chọn $s = 1$; nếu x chia 9 dư 7 thì chọn $s = 3$; nếu x chia 9 dư 6, 8 thì chọn $s = 6$; khi đó $x - s \in P$. Vậy từ một vị trí bất kì $x \in N$, luôn có cách trừ để đi tới một vị trí của P.

Ngược lại, với $x \in P$ và với mọi số tùy ý $s \in S$, ta sẽ chứng minh $(x - s) \in N$. Thực vậy: Nếu $s = 1$ thì $x - s$ chia 9 có dư 8 (hay là -1), 1, 3; nếu $s = 3$ thì $x - s$

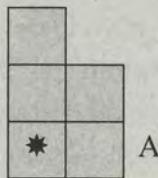
chia cho 9 có dư 6 (hay là -3), 8 (hay là -1), 1; nếu $s = 6$ thì $x - s$ chia 9 có dư là 3 (hay là -6), 5 (hay là -4), 7 (hay là -2).

c) Gọi tập các vị trí là $X = \{0, 1, 2, \dots\}$, tập trừ là $S = \{1, 2, 4, 8, 16, \dots\}$ là tập các luỹ thừa của 2. Sẽ chứng minh $P = \{x \in X | x \text{ chia hết cho } 3\}$. Thực vậy: Đầu tiên $0 \in P$. Giả sử $x \in N$, thì x chia cho 3 có dư là 1 hoặc 2. Nếu x chia 3 dư 1 thì chọn $s = 1$; nếu x chia 3 dư 2 thì chọn $s = 2$. Trong cả hai trường hợp $x - s$ đều chia hết cho 3 nên $x - s \in P$. Do đó với mọi $x \in N$ thì luôn có thể chọn một số s sao cho $x - s \in P$.

Ngược lại, với $x \in P$, với mọi số tuỳ ý $s \in S$, ta sẽ chứng minh $(x - s) \in N$. Thực vậy nếu $x - s \in P$ thì $s = x - (x - s)$ cũng chia hết cho 3. Điều này vô lí vì $s = 2^k$ (k nguyên không âm) không thể chia hết cho 3.

Chương trình: *Học sinh tự lập trình.*

9.3. a) Xét trường hợp A chiếm ô $(3; 1)$. Các ô còn lại của bảng là:



Các phép chuyển tiếp theo là:

+ Nếu đối thủ chiếm ô $(1; 2)$ thì chúng ta chiếm ô $(2; 1)$. Ngược lại nếu đối thủ chiếm ô $(2; 1)$ thì chúng ta chiếm ô $(1; 2)$. Cả hai tình huống đều buộc đối thủ đi tiếp chiếm ô $(1; 1)$ và thua.

+ Nếu đối thủ chiếm ô $(1; 3)$ thì chúng ta chiếm ô $(2; 2)$, hoặc đối thủ chiếm ô $(2; 2)$ thì chúng ta chiếm ô $(1; 3)$. Cả hai trường hợp đều dẫn đến đối thủ nhận hình còn lại gồm 3 ô là ô $(1; 1)$ và hai ô kè bên phải và kè bên trên. Đối thủ lấy một trong hai ô kè, chúng ta chiếm nốt ô kè kia. Cuối cùng đối thủ buộc phải nhận ô còn sót lại là ô $(1; 1)$ và thua.

Do đó phép chiếm $(3; 1)$ là phép chuyển giành chiến thắng cho A (người đi đầu), vậy đây là một vị trí thuộc tập vị trí N.

Vị trí này là duy nhất. Thực vậy:

+ Nếu A đi trước và chiếm một ô bên trên hoặc bên phải ô $(3; 1)$ thì không thể chiến thắng vì đối thủ đi tiếp sẽ chiếm ngay ô $(3; 1)$ và sẽ chiến thắng.

- Nếu A chiếm ô $(1; 2)$ hoặc $(2; 1)$ thì tương ứng đối thủ sẽ chiếm ô $(2; 1)$ hoặc $(1; 2)$, buộc A đi tiếp phải nhận ô $(1; 1)$.
 - Nếu A chiếm ô $(2; 2)$, đối thủ sẽ chiếm ô $(4; 1)$, đưa A về hình còn ô $(1; 1)$ cùng với 2 ô bên trên và 2 ô bên phải. A chiếm thêm một phần trên hoặc một phần bên phải thì đối thủ cũng chiếm như thế trên phần kia, cho đến khi buộc A phải nhận ô $(1; 1)$.
 - Nếu A chiếm $(1; 3)$, đối thủ sẽ chiếm ô $(7; 1)$, đưa A về hình còn hai băng nằm ngang (băng dưới ngắn hơn băng trên đúng 1 ô). Do đó bắt kể A di chuyển như thế nào, đối thủ luôn tìm được cách di chuyển đưa A về một trạng thái mới mà trạng thái cuối cùng là hai băng: băng trên có độ dài bằng 0; băng dưới có độ dài bằng 1 - nói cách khác: chỉ còn ô $(1; 1)$ dành cho A.
 - nếu A chiếm ô $(2; 3)$; đối thủ sẽ chiếm ô $(2; 2)$; đưa A về hình còn một băng đứng phía trên ô $(1; 1)$ và một băng nằm bên phải ô $(1; 1)$. Tổng số ô trên hai băng không kể ô $(1; 1)$ là $2 + 7 = 9$. Do A phải đi trước, nên đối thủ sẽ lựa được cách di thích hợp:
 - Nếu A ăn hết một băng thì đấu thủ ăn hết băng còn lại.
 - Nếu A ăn xong vẫn còn hai băng thì đấu thủ đi sao cho vẫn còn hai băng và tổng số ô bị loại của phép di chuyển của A và của đối thủ luôn lẻ, nên số ô còn lại của hai băng luôn chẵn. Cuối cùng dẫn tới tình trạng A phải nhận hình gồm 3 ô: $(1; 1)$; $(1; 2)$ và $(2; 1)$ nên sẽ thua.
- b) Trò chơi trong bảng tổng quát $m \times n$ ô vuông, người A (người chơi trước) sẽ chiến thắng nếu luôn tạo ra các nước đi đúng. Ta có thể chứng minh điều này, tuy nhiên chỉ ra cụ thể từng bước đi như thế nào thì còn là bài toán mở chờ đợi khám phá thuật toán hiệu quả trong tương lai. Sau đây là một cách chứng minh A sẽ thắng bằng phản chứng:
- Giả sử A không thể chiến thắng với mọi vị trí bắt đầu của hình chữ nhật X kích thước $m \times n$, nghĩa là mọi vị trí trên hình chữ nhật X đều là vị trí P. Khi đó mọi hình tạo ra sau bất kì phép đi ban đầu nào đều là vị trí N. Do đó có thể tạo ra một hình chữ nhật Y kích thước $m \times n_1$ nhỏ hơn có số dòng vẫn là m nhưng số cột là $n_1 < n$, là một vị trí thuộc tập N. Đối thủ nhận được vị trí thuộc tập N trên Y, nên tồn tại ít nhất một phép đi để đối thủ có thể chuyển tới vị trí thuộc tập P và tiếp tục chơi theo thuật thắng (luôn đi tới vị trí thuộc P) để cuối cùng chiến thắng. Nhưng

nếu như vậy thì A cũng có thể dùng phép đi này của đối thủ làm phép chuyển đầu tiên ngay trong hình chữ nhật X (kích thước $m \times n$) và chọn các phép đi tiếp theo của đối thủ trên Y làm các bước đi của mình trên X để dành chiến thắng vì phép đi này tương ứng trên Y và trên X sẽ tạo ra các cặp hình sau đó có thể chỉ khác nhau một số ô thuộc cột lớn hơn n_1 . Nhưng đến một thời điểm nhất định nào đó sẽ bằng nhau vì những ô ở cột lớn hơn n_1 ngày càng bị loại dần). Do đó chúng ta giả sử người chơi thứ nhất không thể chiến thắng với mọi cách chiếm vị trí ban đầu của hình chữ nhật là sai. Nói cách khác, luôn tồn tại một ô vuông mà người đi đầu nếu chiếm thì sẽ chiến thắng.

Chương trình: Học sinh tự lập trình.

9.4. a) Biểu diễn số thẻ có trên cọc thành dạng nhị phân. Người A (người chơi đầu tiên) cần loại số 1 bên phải nhất (bit 1 thấp nhất) của biểu diễn, khi đó xảy ra hai trường hợp:

- Hoặc không còn số 1 nào trong biểu diễn, nghĩa là A đã thắng;
- Hoặc còn một số số 1 bên trái số 1 vừa bị loại, nhưng đối thủ không thể loại bỏ bất kì số 1 nào trong những số 1 này vì số quân loại đi của đối thủ không vượt quá số quân A vừa loại, nghĩa là đối thủ chưa thể dành chiến thắng.
- Tiếp tục A lại loại bỏ số 1 bên phải nhất ...
- Quá trình hữu hạn vì n giảm dần, A sẽ phá được số 1 cuối cùng và chiến thắng.

Vậy trò chơi này người đi đầu luôn chiến thắng.

b) Rõ ràng người chơi sau chỉ thắng khi $N = 2$. Khi đó A lấy 1, rồi B lấy 1.

Chương trình: Học sinh tự lập trình.

9.5. Dãy số Fibonacci là $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots$ ($F_1 = 1; F_2 = 1; F_{n+1} = F_n + F_{n-1}$ với $n \geq 2$). Lời giải trò chơi Fibonacci Nim dựa trên định lí Zeckendorf: “Mỗi số nguyên dương luôn có thể viết duy nhất thành tổng một vài số Fibonacci phân biệt không kề nhau”.

- Khi số thẻ η không là số Fibonacci, người đi trước muốn thắng mỗi lần cần chọn số hạng nhỏ nhất trong phân tích số thẻ hiện tại thành tổng các số Fibonacci phân biệt không kề nhau.

- Khi n là số Fibonacci (giả sử là F_i), trong biểu diễn chỉ có một số duy nhất nhưng người đi đầu không được quyền loại tất cả các thẻ nên người đi đầu sẽ tạo ra một số thẻ nhỏ hơn. Số mới này có thể thuộc một trong hai loại sau:
 - + Số mới là số Fibonacci (giả sử là F_k), thì đối thủ thắng vì bây giờ đối thủ sẽ loại hết quân còn lại (*Điều này thực hiện được vì $2(F_i - F_k) \geq F_k \Leftrightarrow 2F_i - 3F_k \geq 0$. Bất đẳng thức này đúng. Thật vậy: Thử thấy đúng khi $i < 4$, còn khi $i > 3$ thì: $2F_i - 3F_k \geq 2F_i - 3F_{i-1} = 2(F_{i-1} + F_{i-2}) - 3F_{i-1} = 2F_{i-2} - (F_{i-2} + F_{i-3}) = F_{i-2} - F_{i-3} \geq 0$, Đúng.*
 - + Số mới không là số Fibonacci, khi đó nếu đối thủ biết thuật thăng (diệt số nhỏ nhất trong biểu diễn Fibonacci) thì đối thủ thăng. Nếu đối thủ không biết thuật thăng (đưa cho ta một số mới cũng không là số Fibonacci) thì nếu có thể thực hiện được thuật chiến thăng thì ta thăng. Trường hợp ngược lại, không thể thực hiện được chiến thăng, ta loại cầm chừng một số quân nào đó chờ dịp thuận lợi khác giành lại thẻ chiến thăng.

Chương trình:

```
#include <iostream>
using namespace std;
int f[21], bit[21];
int n, player, cur, turn;
void phantich(int x) { // Phân tích x thành tổng các số Fibonacci, đánh dấu chúng bởi bit
    memset(bit, 0, sizeof(bit));
    for (int i=20; i>=0; i--)
        if (x>=f[i]) {
            bit[i]=1;
            x-=f[i];
        }
}
int main() {
    cout << "Nhập số quân ban đầu: ";
    cin >>n ; // Nhập từ bàn phím số quân ban đầu
    f[0]=1; // Tạo dãy Fibonacci
    f[1]=1;
    for (int i=2; i<=20; i++)
        f[i]=f[i-1]+f[i-2];
    player=0; // Số hiệu của máy là 0
    int may;
    while (true){
        turn++; // Lượt chơi
        if (!player) { // Nếu là máy chơi
            int sum=0;
            for (int i=0; i<21; i++)
                if (bit[i])
                    sum+=f[i];
            if (sum>n)
                cout << "Máy thua" << endl;
            else
                cout << "Máy thắng" << endl;
        }
        if (turn>20)
            break;
    }
}
```

```

cout << "COMPUTER's TURN\n";
phantich(n); //Phân tích n thành tổng các số Fibonacci
bool co=false; //Cờ báo
for (int i=0;i<=20;i++) //Chọn số Fibonacci nhỏ nhất trong phân tích n
    if (!(turn==1 && f[i]==n) && bit[i]) {
/* Nếu (không là lượt đầu tiên hoặc n không là số Fibonacci) và phân tích được thành tổng các
số Fibonacci mà số hạng bé nhất là F[i] thì: */
    co=true; //cờ báo còn tiếp tục thuật thằng
    cout << "It moved: " << f[i] << " card(s)\n";
    may = f[i]; //số quân máy lấy
    n-=f[i];
    break;
}
if (!co) { //Nếu không thì kéo dài trò chơi
    cout << "It moved " << 1 << " card(s)\n";
    may = 1; n--;
}
if (!n) { //Hết quân, máy thắng
    cout << "COMPUTER WON!!\n";
    break;
}
}
else { //Nếu đến lượt người chơi
    cout << "YOUR TURN\n";
    cout << "Please enter number of card(s): ";
do {
    cin >> cur; //Nhập từ bàn phím số quân người chơi lấy
    if (cur<=2*may) break; //số quân lấy hợp lệ
    else cout << " number invalid. Need to re-enter...";
}while (cur> 2*may);
cout << "You moved " << cur << " card(s)\n";
n-=cur;
if (!n) { //Hết quân. Người thằng
    cout << "YOU WON!!\n";
    break;
}
}
player^=1; //Đổi lượt chơi (sử dụng phép XOR
cout << '\n';
cout << " card(s) left! " << n << '\n'; //Số quân còn lại sau mỗi lượt
cout << '\n';
}
system("pause");
return 0;
}

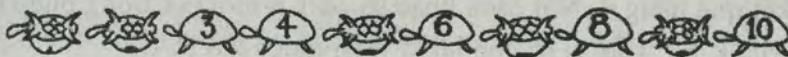
```

9.6. Thuật toán của trò chơi Porker Nim như trò Nim chuẩn ba cọc, chỉ thêm bước sau: Nếu B di chuyển bằng cách tăng thêm x quân vào một cọc nào đó thì A lại rút đúng x quân ra khỏi cọc này. Cụ thể: Do tổng Nim của 3, 4 và 6 bằng 1 (khác 0) nên người A áp dụng chiến thuật thắng của trò Nim chuẩn là chuyển về vị trí (2, 4, 5) có tổng Nim bằng 0. Sau đó nếu người B di chuyển giảm kích thước một cọc nào đó thì bị rơi vào vị trí có tổng Nim khác 0, người A vẫn chuyển được về vị trí có tổng bằng 0. Nếu người B di chuyển bằng cách tăng thêm x quân vào một cọc nào đó thì A lại rút đúng x quân đó ra khỏi cọc này. B lại tăng thì A lại rút ra...

Do trò chơi không cho phép chơi kéo dài mãi và mỗi lượt đi A đều lấy ra khỏi bàn nhất một quân nên sau hữu hạn bước đi (số bước đi tối đa là $2^*(x_1 + \dots + x_n)$) trò chơi sẽ kết thúc và A thắng.

Chương trình: Học sinh tự lập trình.

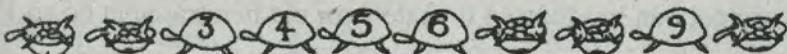
9.7. a) Có thể coi trò chơi này là một hình thức của trò chơi Nim: Rùa bò ở ô n như một cọc có n quân. Một phép di chuyển rùa: lật ngửa rùa đang bò tại ô x, kèm theo có thể lật ngược lại một rùa ở bên trái ô x là ô y ($y < x$). Nếu rùa tại ô y đang bò mà bị lật ngửa lại thì tương đương với việc di chuyển hết quân tại hai cọc x và cọc y. Nếu rùa tại ô y đang nằm ngửa lại được lật thành bò thì tương đương với việc xoá cọc x và thêm cọc y, hay có thể coi như di chuyển $x - y$ quân khỏi cọc x để còn y quân. Ví dụ với $n = 10$ như hình vẽ dưới:



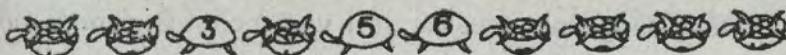
Trạng thái ban đầu là (3, 4, 6, 8, 10) có tổng Nim bằng 3 (khác 0) nên người đi đầu sẽ thắng. Do tổng Nim của 3, 4, 6, 8 bằng 9, nên cần lật rùa 10 và 9 thì chiếm được vị trí mới có tổng Nim bằng 0 ($9 \oplus 9 = 0$).

Các rùa đang bò bây giờ là 3, 4, 6, 8, 9 tạo thành một vị trí P (tổng Nim bằng 0).

Sau khi người thứ hai đi tiếp bằng cách lật rùa 8 và 5, các rùa đang bò lại là 3, 4, 5, 6, 9:



Tổng Nim bây giờ là 13, nên chỉ có một phép chuyển tốt cho vị trí này là lật úp 9 và 4 (nghĩa là bỏ đi 13 quân). Trạng thái mới là (3, 5, 6) có tổng Nim bằng 0:

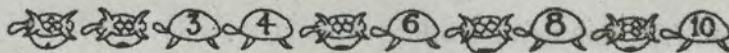


Quá trình tiếp diễn tương tự...

Kết luận: Một bước chuyển trong Nim thành một trong ba bước lật rùa như sau:

- Giảm kích thước một cọc bằng cách lật ngửa một rùa và lật úp một rùa như phép đầu tiên (ngửa 10, úp 9).
- Loại hai cọc với các kích thước hiện đang có bằng cách lật ngửa hai rùa (ví dụ bước đi lần thứ hai của người thứ nhất đã úp 9 và 4).
- Loại một cọc bằng cách lật ngửa một rùa thích hợp ví dụ do 4, 6, 8, 10 có tổng Nim bằng 0 (là một vị trí P) nên bước đầu tiên người thứ nhất cũng có thể chỉ cần lật ngửa rùa 3.

Do trạng thái:



có tổng Nim dương nên người đi đầu sẽ thắng khi áp dụng thuật giành thắng là: sử dụng một trong ba bước lật rùa nêu ở câu a) (giảm kích thước một cọc, loại hai cọc, loại một cọc) để luôn luôn trao cho đối thủ vị trí có tổng Nim bằng 0.

Chương trình: *Học sinh tự lập trình.*

9.8. Theo định lí Moore, vị trí (x_1, x_2, \dots, x_n) là vị trí thuộc tập P trong Nim_k khi và chỉ khi x_1, x_2, \dots, x_n viết dạng cơ số 2 trong phép cộng không nhớ theo modulo $(k + 1)$ cho kết quả bằng 0. Nói cách khác số số 1 ở mỗi cột trong phép cộng nào đều chia hết cho $k + 1$. Các vị trí còn lại là vị trí thuộc tập N.

Sau đây là một cách chứng minh định lí Moore, nó cũng cung cấp một thuật toán giải bài toán:

Trước hết chú ý rằng vị trí kết thúc là vị trí P. Cần kiểm tra hai nội dung sau:

- Mọi phép chuyển từ vị trí P đều đến vị trí N. Thật vậy, xét một phép chuyển từ vị trí P (tổng các cột đều chia hết cho $k + 1$) trong phép cộng theo cơ số $k + 1$ không nhớ của các số x_1, x_2, \dots, x_n (viết dạng nhị phân) bỏ qua các cột bên trái toàn số 0 và để ý tới cột bên trái nhất có số 1 (gọi cột l). Mỗi phép di chuyển chúng ta sẽ làm thay đổi k số trong n số x_1, x_2, \dots, x_n . Nhưng ở cột l, tại mỗi dòng (ứng với một số) chúng ta chỉ thay số thành 0 hoặc giữ nguyên 1 (vì các cọc ứng với số được chọn là bị giàn). Do tổng đang tồn tại của các số 1 và 0 ở cột l là giảm đi một lượt không vượt quá k vậy tổng này không còn chia hết cho $k + 1$ và chúng

chi lấy quân ở nhiều nhất là k cọc nên *tổng các số ở cột l* sau phép chuyển sẽ không chia hết cho $k + 1$. Suy ra tổng Nim (cộng theo cơ số $k + 1$, không nhớ) của n cọc sau phép chuyển từ vị trí P là số khác 0, nói cách khác từ vị trí P luôn chuyển tới vị trí N.

- Ngược lại, từ một vị trí N luôn có phép chuyển tới vị trí P. Thật vậy: Trước hết trong phép cộng không nhớ theo cơ số $k + 1$ của các số x_1, x_2, \dots, x_n tìm cột trái nhất có tổng các số 1 không chia hết cho $k + 1$. Ta cần chọn ra không quá k hàng trong n hàng x_1, x_2, \dots, x_n để biến đổi các bít 0 và 1 trên cột này của các hàng đã chọn sao cho tổng các số 1 của cột này chia hết cho $k + 1$. Quá trình biến đổi tiếp theo lại diễn ra với cột bên phải cột này. Giả sử quá trình biến đổi đến cột j và trước đó đã biến đổi các bít thuộc các cột bên trái cột j trên r hàng (bây giờ r hàng này còn có x số 1 và $r - x$ số 0 thuộc cột j). Chúng ta còn có thể chọn thêm $k - r$ hàng nữa (nếu cần). Tổng các số 1 trên cột j khi chia cho $k + 1$ còn dư là s . Xảy ra hai trường hợp sau:

a) Nếu $r + s \leq k$ thì chọn thêm s hàng và xoá s số 1 của cột này thuộc s hàng mới chọn.

b) Nếu $r + s > k$ thì:

+ Nếu $s \leq x$ thì ta xoá s số 1 của cột này nằm trong r hàng đã chọn, không chọn thêm hàng mới.

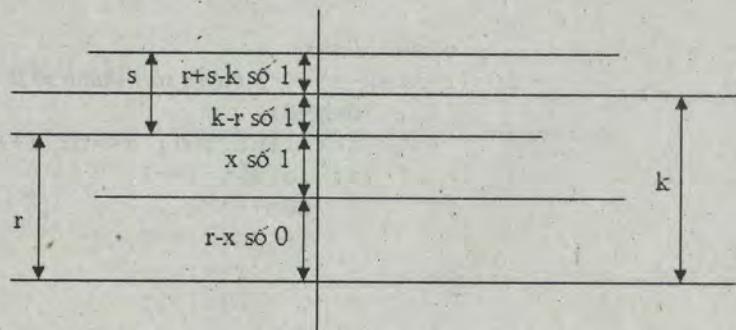
+ Nếu $s > x$ ta lại chia làm hai trường hợp:

- Nếu $r - x > k - s$ (số số 0 trong r hàng lớn hơn hoặc bằng $k - s + 1$ số 1 cần thêm cho cột này) thì ta

thêm $k - s + 1$ số 1 vào r hàng đã chọn của cột này (đổi 0 thành 1).

- Nếu $r - x \leq k - s$ (số số 0 trong r hàng không đủ biến thành

$k + 1 - s$ số 1) trường hợp này có thể viết lại bất đẳng thức $r - x \leq k - s$ thành $r + s - k \leq x$. Do đó có thể xoá $r + s - k$ số 1 của cột này trong x số 1 và $k - r$ số 1 thuộc cột này trên $k - r$ hàng mới chọn thêm (đổi 1 thành 0).



Chương trình:

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
const int maxn= 30, maxbit=10, lt2[9]={256,128,64,32,16,8,4,2,1};
int a[maxn][maxbit];
int n, k, d[maxn];
bool la_N;
int s11(int j) { //Đếm số lượng số 1 thuộc cột j của các hàng đã đánh dấu chọn
    int x=0;
    for (int i=1; i<=n; i++)
        if ((d[i]==1)&&(a[i][j]==1)) x++;
    return x;
}
void di(int n, int k) { //Đi từ vị trí thuộc tập N tới vị trí thuộc tập P
    int s, r, x, dem, sum;
    r = 0;
    for (int i=0; i<maxn; i++) d[i]=0; //Đánh dấu các hàng đều chưa chọn
    for (int j=1; j<9; j++) {
        s = a[n+1][j];
        if (s>0) { //s là số dư khi chia tổng các số 1 ở cột j cho (k+1)
            x = s11(j); //Số lượng số 1 ở cột j
            if (s<=x) { //số số 1 lớn hơn số dư s thì chuyển s số 1 thành 0
                dem = 0;
                for (int i=1; i<=n; i++)
                    if ((d[i]==1)&&(a[i][j]==1)) {
                        dem++;
                        a[i][j]=0;
                        if (dem==s) break;
                    }
            }
            else //s>x
                if (r-x<=k-s) { //biến s số 1 thành số 0
                    dem=0;
                    for (int i=1; i<=n; i++)
                        if (a[i][j]==1) {
                            dem++;
                            if (d[i]==0) {
                                r++;
                                d[i]=1;
                            }
                            a[i][j]=0;
                            if (dem==s) break;
                        }
                }
        }
    }
}
```

```

        else { // r-x>k-s, biến k-s+l số 1 thành 0
            dem=0;
            for (int i=1; i<=n; i++)
                if ((d[i]==1) && (a[i][j]==0) && (dem<=k-s)) {
                    a[i][j]=1;
                    dem++;
                }
            }
        }
    }
}

// chuyển số quân mỗi cọc từ nhị phân thành thập phân lưu vào cột 9
sum=0;
for (int j=1; j<=8; j++)
    if (a[i][j]==1) sum+=lt2[j];
a[i][9] = sum;
}

void tao_a(int n, int k) {
    // Mỗi cọc là một hàng của a, cột 0 là số quân, các cột từ 8 đến 1 biểu diễn nhị phân số quân
    int x, sum;
    for (int i=1; i<=n; i++) {
        x = a[i][0];
        for (int j=8; j>0; j--) { // Biểu diễn nhị phân số quân ở cọc i
            a[i][j] = x%2;
            x = x/2;
        }
        for (int j=1; j<=8; j++) {
            // Tính tổng các số 1 của từng cột theo mod (k+1), lưu vào hàng n+1
            sum=0;
            for (int i=1; i<=n; i++)
                sum +=a[i][j];
            a[n+1][j]= sum % (k+1);
        }
        la_N = false; // Vị trí thuộc tập P nếu mọi cột đều có tổng chia hết cho k+1
        for (int j=1; j<=8; j++)
            if (a[n+1][j]>0) {
                la_N = true;
                break;
            }
    }
}

int main() {
    int x;
    char c;
    ofstream fo("BAI08_C9.OUT");
}

```

```

ifstream fi("BAI08_C9.INP");
for (;;) { // Vòng lặp vô hạn
    string s;
    getline(fi,s); // đọc một dòng của tệp input (ứng với một test)
    stringstream ss(s); // đổ dòng vào luồng ss
    if (!(ss >> k))
        break; // phân tách dòng thành số k, nếu không được thì thoát test
    n=0;
    memset(a,0,sizeof(a));
    while (ss >> x)
        a[++n][0]=x; // tiếp tục phân tách dòng lấy số quân các cột
    tao_a(n,k); // Tạo vị trí ban đầu của trò chơi
    if (la_N==true) { // nếu vị trí ban đầu thuộc tập N thì
        fo << 1 << " "; // xác nhận A thắng
        di(n,k); // tìm nước đi đầu tiên
        for (int i=1;i<=n;i++)
            fo << a[i][9] << " "; // Hiện số quân còn lại sau nước đi
    }
    else fo << 0 ; // Vị trí ban đầu thuộc tập P thì A thua
    fo << '\n';
    if (fi.eof()) break; // Hết tệp input thì thoát vòng lặp vô hạn for
}
return 0;
}

```

9.9. Xây dựng giá trị hàm Sprague-Grundy cho từng đỉnh như sau: mỗi đỉnh x của đồ thị được gắn một số $g(x)$ là số nguyên không âm nhỏ nhất không nằm trong tập các giá trị $g(y)$ của các đỉnh y tối đa được từ x . Chú ý đỉnh kết thúc là đỉnh i thì $g(i) = 0$.

Kết quả:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
G(x)	0	0	0	0	4	1	2	1	3	0	1	0	2	2	1

Chương trình:

```

#include <stdio.h>
#include <fstream>
using namespace std;
const long MaxN = 100;
int n;
int g[MaxN];
int c[MaxN][MaxN];
int SG(int x) {
    if (g[x] > -1) return g[x];
    for (int i=0;i<n;i++)
        if (c[x][i]>0)
            g[x] = max(g[x],SG(i));
    g[x]++;
}

```

```

int v[MaxN];
for (int i=0; i<MaxN; ++i) v[i] = 0; // Tạo dãy "chuồng" rỗng
for (int y=0; y<n; ++y)
    if (c[x][y]) v[SG(y)] = 1; // "Lùa" các giá trị SG[y] vào "chuồng"
int k = 0;
while (v[k]==1) k++; // "Duyệt" dãy "chuồng", tìm "chuồng" rỗng đầu tiên
g[x] = k;
return g[x]; // Giá trị hàm Sprague_Grundy của đỉnh x

void docf() {
    int x, nx, y;
    ifstream fi ("BAI09_C9.inp");
    fi>>n;
    while (!fi.eof()) {
        fi>>x>>nx;
        for (int i=0; i<nx; i++) {
            fi>>y;
            c[x][y]=1;
        }
    }
}

int main() {
    docf();
    for (long x = 0; x<n; ++x) g[x] = -1; // Khởi tạo g
    for (long x = 0; x<n; ++x) SG(x); // Tính các giá trị g[x]
    ofstream fo ("BAI09_C9.out"); // Xuất kết quả
    for (long x = 0; x<n; ++x) fo << x << " " << g[x] << '\n';
    return 0;
}

```

9.10. Các câu trong bài tương tự nhau. Mỗi số là một đỉnh của đồ thị. Dựa vào từng trò chơi cho trong mỗi câu mà biết đỉnh kề của mỗi đỉnh. Từ đó dựa vào định nghĩa hàm Sprague-Grundy trên đồ thị, viết chương trình thích hợp.

a) Tập trù là $S = \{1, 3, 4\}$. Hiển nhiên $g(0) = 0$. Do từ vị trí 1 đi đến được vị trí 0 nên $g(1) = 1$. Từ vị trí 2 đi đến được vị trí 1 nên $g(2) = 0$. Từ vị trí 3 đi đến được các vị trí 0, 2 nên $g(3) = 1$. Từ vị trí 4 đi đến được các vị trí 0, 1, 3 nên $g(4) = 2$...
Đoạn chương trình xây dựng hàm Sprague-Grundy như sau:

```

g[0] = 0;
for (long i = 0; i != n; ++i) {
    int v[n];
    for (long k = 0; k != n; ++k) v[k] = 0;
    if (i >= 1) v[g[i-1]] = 1;
    if (i >= 3) v[g[i-3]] = 1;
}

```

```

    if (i >= 4) v[g[i-4]] = 1;
    long k = 0;
    while (v[k] == 1) k++;
    g[i] = k;
}

```

Sau khi tìm được các giá trị của hàm Sprague-Grundy, nên so sánh kết quả với bảng vị trí N và P của trò chơi này đã nêu ở trang 52, Tài liệu chuyên Tin học Quyển 3. Từ đó nên rút ra chiến thuật giành thắng của mỗi trò chơi (nếu có).

Chương trình:

a) Tập trừ là $\{S_1, S_2, \dots, S_m\}$

```

#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ifstream fi("bt10a.in");
    ofstream fo("bt10a.out");
    int N, m, i, j;
    int S[1000];
    int G[100000];
    fi>>N;
    int v[N];
    m=-1;
    while (fi>>S[++m]); //Đọc tập trừ  $S[0], S[1], \dots, S[m]$ 
    for (int i=0; i<=N; i++) G[i]=-1; //Khởi trị hàm Sprague-Grundy
    for (i=0; i<=N; i++) { //Xây dựng giá trị hàm Sprague-Grundy cho các vị trí i
        if (G[i]==-1)
            for (int k=0; k<N; ++k) v[k] = 0; //Khởi trị mảng « chuồng bò»
        rỗng
        for (j=0; j<m; j++) //Xây dựng  $G[i]$  dựa vào  $G[i-S[j]]$ 
            if (i>=S[j]) v[G[i-S[j]]]=1;
        int k = 0;
        while (v[k] == 1) k++; //Tìm chuồng rỗng đầu tiên
        G[i]=k;
    }
    for (long i=0; i<=N; i++) //Xuất giá trị hàm Sprague-Grundy
        fo<<G[i]<<" ";
    fo<<"\n";
    return 0;
}

```

e) Tập trừ là tập các số Lucas

```

#include <fstream>
using namespace std;
const long N = 100;
int g[N], L[N];

```

```

void creat_Lucas() { //Xây dựng các giá trị Lucas không vượt quá N
    for (int i=1; i<N; i++) L[i]=0;
    int U1=1, U2=3, U=4;
    L[U1]=1; L[U2]=1;
    while (U<=N) {
        L[U]=1;
        U1=U2; U2=U; U=U1+U2;
    }
}

int SG(int x) { //Hàm trả về giá trị Sprague-Grundy của vị trí x
if (g[x] > -1) return g[x];
    int v[N];
    for (int i=0; i<N; ++i) v[i] = 0;
    for (int y=1; y<x; ++y)
        if (L[y]==1) v[SG(x-y)] = 1;
    int k = 0;
    while (v[k] == 1) k++;
    g[x] = k;
    return g[x];
}

int main() {
    creat_Lucas();
    for (long x = 0; x<=N; ++x) g[x] = -1;
    for (long x = 0; x<=N; ++x) SG(x);
    ofstream fo ("BAI10_C9.OUT");
    for (long x = 1; x<=N; ++x)
        fo << x << " " << g[x] << '\n';
    return 0;
}

```

9.11. Các vị trí kết thúc là 0 hoặc 1 (do điều kiện nhiều nhất là nửa số quân trên cọc). Từ vị trí i ($1 \leq i \leq N$) có thể tới mọi trạng thái $i - k$ (với k từ 1 đến $i/2$).

Chương trình:

```

#include <iostream>
using namespace std;
const long maxn=10001;
long n, g[maxn];
void creat_g(long n) {
    long v[maxn];
    for (long i=0; i<maxn; i++) g[i]=0;
    for (long i=2; i<maxn; i++) {
        for (long j=0; j<maxn; j++) v[j]=0;
        for (long k=1; k<=(i/2); k++) v[g[i-k]]=1;
        long k=0;
        while (v[k]==1) k++;

```

```

        g[i]=k;
    }
}
int main() {
    creat_g(maxn);
    ifstream fi ("BAI11_C9.INP");
    ofstream fo ("BAI11_C9.out");
    while (fi >> n)
        fo << g[n] << endl;
    return 0;
}

```

9.12. Chỉ có một vị trí kết thúc là 0. Từ vị trí i ($1 \leq i \leq N$) có thể tới mọi trạng thái $i - k$ (với k là một ước của i , kể cả 1 và i).

Chương trình:

```

#include <iostream>
using namespace std;
const long n = 21;
int main() {
    int g[n];
    g[0] = 0;
    for (long i = 1; i != n; ++i) {
        int v[n];
        for (int k = 0; k != n; ++k) v[k] = 0;
        for (int k = 1; k <= i; ++k)
            if (i % k == 0) v[g[i - k]] = 1;
        int k = 0;
        while (v[k] == 1) k++;
        g[i] = k;
    }
    ofstream fo ("BAI12_C9.OUT");
    for (int i = 0; i != n; ++i)
        fo << g[i] << " ";
    return 0;
}

```

9.13. Trò chơi có hai vị trí kết thúc là 0 và 2. Thật vậy: Rõ ràng $g(0) = 0$ vì 0 là vị trí kết thúc; $g(1) = 1$ vì từ 1 chỉ đi tới 0; $g(2) = 0$ vì 2 là vị trí không thể tới vị trí khác (là vị trí kết thúc). Vậy $g(2k) = k - 1$ và $g(2k - 1) = k$ đã đúng khi $k = 1$.

Giả sử mệnh đề đã đúng với $1, 3, \dots, 2k - 3$, nghĩa là đã có $g(1) = 1, g(3) = 2, \dots, g(2k - 3) = k - 1$. Ta sẽ chứng minh mệnh đề đúng với $2k - 1$, nghĩa là sẽ chứng minh $g(2k - 1) = k$. Thật vậy, từ $2k - 1$ có thể tới các vị trí $2k - 3, 2k - 5, \dots, 3, 1$

(khi bót một số chẵn quân) và tới 0 (khi bót đi toàn bộ quân). Do đó số nguyên không âm nhỏ nhất chưa có trong tập $\{g(2k-3), g(2k-5), \dots, g(3), g(1), g(0)\} = \{k-1, k-2, \dots, 3, 2, 1, 0\}$ là k . Vậy $g(2k-1) = k$.

Tương tự nếu mệnh đề đúng với $2, 4, \dots, 2k-4, 2k-2$, ta sẽ chứng minh đúng với $2k$. Thật vậy, từ $2k$ có thể đi tới các vị trí: $2k-2, 2k-4, \dots, 4, 2$ (khi bót một số chẵn quân). Do đó số nguyên không âm nhỏ nhất chưa có trong tập $\{g(2k-2), g(2k-4), \dots, g(4), g(2)\} = \{k-2, k-3, \dots, 2, 1, 0\}$ là $k-1$. Vậy $g(2k) = k-1$.

Kết quả cụ thể với $n = 20$:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
g(i)	0	1	0	2	1	3	2	4	3	5	4	6	5	7	6	8	7	9	8	10	9

Chương trình:

```
#include <fstream>
using namespace std;
const long n = 21;
int main() {
    int g[n];
    g[0] = 0;
    for (long i = 1; i != n; ++i) {
        int v[n];
        for (int k = 0; k != n; ++k) v[k] = 0;
        for (int k = 2; k < i; ++k)
            if (k%2==0) v[g[i-k]] = 1;
        if (i%2==1) v[g[0]]=1;
        int k = 0;
        while (v[k]==1) k++;
        g[i] = k;
    }
    ofstream fo ("BAI13_C9.OUT");
    for (int i = 0; i!=n; ++i)
        fo << g[i] << " ";
    return 0;
}
```

9.14. Chứng minh quy nạp công thức $g(x) = \min\{k | 2^k > x\}$ với mọi $x > 0$. (*)

Vị trí kết thúc là $x = 0$.

Khi $x = 1$, lấy hết quân tới vị trí 0. Vậy $g(1) = 1$ suy ra (*) đúng với $x = 1$.

Giả sử (*) đúng với i ($i \geq 1$), nghĩa là $g(i) = j = \min\{k | 2^k > i\}$.

Cần chứng minh $g(i+1) = \min\{k | 2^k > i+1\}$ (nên chia thành hai trường hợp: i chẵn và i lẻ).

Chương trình:

Gọi ý. Như chương trình bài 9.11, chỉ thay dòng lệnh:

```
for (long k=1; k<=(i/2); k++) v[g[i-k]]=1;
```

bởi:

```
for (long k=i; 2*k>=i; k--) v[g[i-k]]=1;
```

và lưu ý $g(1) = 1$ vì trong trò chơi này chỉ có một vị trí kết thúc là vị trí 0.

Kết quả:

x =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
g(x)	0	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5

9.15. Mỗi vị trí là tọa độ một ô trên bàn cờ. Tọa độ góc trái dưới là $(0; 0)$ vị trí kết thúc nên có $g(0; 0) = 0$.

Từ vị trí $(0; 1)$ hoặc $(1; 0)$ chỉ đi đến được vị trí $(0; 0)$ nên $g(0; 1) = g(1; 0) = 1$.

Từ vị trí $(i; j)$ có thể đi tới các vị trí $(i; k)$ với $0 \leq k < j$ hoặc tới các vị trí $(k; j)$ với $0 \leq k < i$ hoặc tới các vị trí $(i - k; j - k)$ với $1 \leq k \leq i$ mà $k < j$. Vậy $g(i; j)$ là số nguyên nhỏ nhất chưa có mặt trong tập sau:

$$\{g(i; k) \mid 0 \leq k < j\} \cup \{g(k; j) \mid 0 \leq k < i\} \cup \{g(i - k; j - k) \mid 1 \leq k \leq i \text{ và } k < j\}$$

Kết quả :

BAI15_C9.OUT								
7	8	6	9	0	1	4	5	
6	7	8	1	9	10	3	4	
5	3	4	0	6	8	10	1	
4	5	3	2	7	6	9	0	
3	4	5	6	2	0	1	9	
2	0	1	5	3	4	8	6	
1	2	0	4	5	3	7	8	
0	1	2	3	4	5	6	7	

Từ bảng kết quả hàm Sprague-Grundy của trò chơi, có thể viết chương trình thực hiện thuật thăng khi chơi: Nếu quân hậu ban đầu ở ô $(x; y)$ có $g(x; y) > 0$ thì người đi trước luôn thăng bằng cách luôn cho hậu đi tới ô $(x'; y')$ có $g(x'; y') = 0$.

Chương trình:

```
#include <fstream>
using namespace std;
```

```

const long n = 8;
long g[n][n];
int main(){
    g[0][0] = 0;
    for (long i=0; i<n; ++i){
        for (long j=0; j<n; ++j){
            int v[n*n];
            for (long k=0; k<n*n; ++k) v[k] = 0;
            for (long k=0; k<i; ++k) v[g[k][j]] = 1;
            for (long k=0; k<j; ++k) v[g[i][k]] = 1;
            for (long k=1; k<= i; ++k)
                if (k<=j) v[g[i-k][j-k]] = 1;
            for (long k=0; k<n*n; ++k)
                if (v[k]==0) {g[i][j] = k; break;}
        }
    }
    ofstream fo ("BAI15_C9.out");
    for (long i=n-1; i>=0; --i){
        for (long j=0; j<n; ++j) fo<<g[i][j]<<' ';
        fo<<'\n';
    }
    return 0;
}

```

9.16. Ta đã biết hàm Sprague-Grundy của từng trò chơi:

- g_1 : Trò chơi chẵn-lẻ;
- g_2 : Trò chơi At-Least-Haft;
- g_3 : Trò chơi Nim chuẩn.

$$g_1(x) = \begin{cases} k - 1 & x = 2k \\ k & x = 2k - 1 \end{cases} \quad k \geq 1$$

$$g_2(x) = \min \{k \mid 2^k > x\}$$

$$g_3(x) = x.$$

Gọi hàm Sprague-Grundy của tổng ba trò chơi này là $g(x)$. Áp dụng định lí Sprague-Grundy tính hàm Sprague-Grundy của trò chơi tổng:

$g(18, 17, 7) = g_1(18) \oplus g_2(17) \oplus g_3(7) = 8 \oplus 5 \oplus 7 = 10 > 0$ nên $(18, 17, 7)$ là một vị trí N. Phép chuyển tối ưu là tới vị trí $(6, 17, 7)$ vì:

$$g(6, 17, 7) = g_1(6) \oplus g_2(17) \oplus g_3(7) = 2 \oplus 5 \oplus 7 = 0.$$

19 20

5 5

0) vị trí

= 1.

$\langle; j \rangle$ với

j) là số

j}

nh thực
i người

Chú ý: $g_1(x) = 2$ khi $x = 3$ hoặc $x = 6$, nhưng từ 18 quân trong trò Chẵn-Lẻ chỉ có thể tới vị trí 6.

Chương trình: Học sinh tự lập trình.

9.17. a) Trò chơi Grundy với một cọc có hai vị trí kết thúc là 1 quân và 2 quân. Từ vị trí một cọc m quân, có thể tới các vị trí $(i; j)$: một cọc i quân và một cọc j quân mà $i + j = m$ ($0 < i < j < m$). Giá trị Sprague-Grundy của vị trí $(i; j)$ là $g(i; j) = g(i) \oplus g(j)$. Từ đó suy ra giá trị Sprague-Grundy $g(m)$ là số nguyên không âm nhỏ nhất chưa có trong tập các giá trị $\{g(i; j) : i + j = m, 0 < i < j < m\}$. Khi cọc có số quân nhỏ, chương trình tính $g(m)$ không nhiều thời gian, nhưng khi số quân lớn thì chương trình chạy rất mất thời gian. Dãy giá trị $g(m)$ có tuần hoàn không? Điều này chưa rõ mặc dù người ta đã tìm ra có hơn 20 tỉ giá trị giống giá trị $g(2002)$.

101 giá trị đầu của hàm Sprague-Grundy của trò chơi Grundy từ $g(0)$ đến $g(100)$ là:

```
0 0 0 1 0 2 1 0 2 1 0 2 1 3 2 1 3 2 4 3  
0 4 3 0 4 3 0 4 1 2 3 1 2 4 1 2 4 1 2 4  
1 5 4 1 5 4 1 5 4 1 0 2 1 0 2 1 5 2 1 3  
2 1 3 2 4 3 2 4 3 2 4 3 2 4 3 2 4 3 2 4  
5 2 4 5 2 4 3 7 4 3 7 4 3 7 4 3 5 2 3 5 2
```

b) Trò chơi Grundy trên ba cọc coi như trò chơi tổng của ba trò chơi Grundy trên một cọc. Áp dụng định lí Sprague-Grundy tính hàm Sprague-Grundy của trò chơi tổng có: Vị trí $(5, 8, 13)$ có $g(5, 8, 13) = g(5) \oplus g(8) \oplus g(13) = 2 \oplus 2 \oplus 3 = 3$. Để chiến thắng cần tới vị trí $(5, 8, 5, 8)$ chẳng hạn vì $g(5, 8, 5, 8) = g(5) \oplus g(8) \oplus g(5) \oplus g(8) = 0$. Nghĩa là phép di chuyển này chia cọc thứ ba có 13 quân thành hai cọc mới có số quân là 5 và 8.

Chương trình:

a)

```
#include <iostream>  
using namespace std;  
const int n= 100;  
int g[1000], c[1000];  
void tinh_g(int m) {  
    for (int i=0; i<=n; i++) c[i]=0;  
    int i=1;  
    int j=m-1;
```

```

        while (i < j) {
            c[g[i] ^ g[j]] = 1;
            i++; j--;
        }
        i = 0;
        while (c[i] == 1) i++;
        g[m] = i;
    }
int main() {
    g[0] = 0; g[1] = 0; g[2] = 0;
    for (int k = 3; k <= n; k++) tinh_g(k);
    ofstream fo ("BAI17_C9.out");
    for (int k = 0; k <= n; k++) fo << g[k] << ' ';
    return 0;
}

```

9.18. a) Trò chơi này thực chất là trò chơi lấy và phân chia. Khi đánh đồ một hoặc hai ki ở các đầu hàng được coi là phép lấy đi khỏi hàng một hoặc hai quân. Khi đánh đồ một hoặc hai quân (ở giữa hàng) đó là phép phân chia hàng.

Trò chơi này chỉ có một vị trí kết thúc là cọc rỗng, vậy $g(0) = 0$. Gọi x là vị trí hiện có, y là các vị trí sinh ra sau một phép lăn bóng, có bảng sau với $x = 1, 2, 3, 4, 5$:

Định x	Các đỉnh y kế tiếp x	Các giá trị g(y)	g(x)
0			0
1	0	0	1
2	1, 0	1, 0	2
3	2, 1, (1, 1)	2, 1, 1 \oplus 1 = 0	3
4	3, 2, (1, 2), (1, 1)	3, 2, 1 \oplus 2 = 3, 1 \oplus 1 = 0	1
5	4, 3, (3, 1), (2, 2), (1, 2)	1, 3, 3 \oplus 1 = 2, 2 \oplus 2 = 0, 1 \oplus 2 = 3	4

Bảng $g(m)$ ($m = 0, 1, \dots, 83$) là:

0	1	2	3	1	4	3	2	1	4	2	6
4	1	2	7	1	4	3	2	1	4	6	7
4	1	2	8	5	4	7	2	1	8	6	7
4	1	2	3	1	4	7	2	1	8	2	7
4	1	2	8	1	4	7	2	1	4	2	7
4	1	2	8	1	4	7	2	1	8	6	7

Khi $x \geq 72$, các giá trị $g(x)$ bắt đầu tuần hoàn với chu kỳ 12, dãy giá trị $g(x)$ sau được lặp mãi:

x	...	72	73	74	75	76	77	78	79	80	81	82	83	...
g(x)	...	4	1	2	8	1	4	7	2	1	8	2	7	...

b) Từ kết quả câu a), để tiện sử dụng trong câu b), chúng ta trích lấy 14 giá trị đầu của hàm Sprague-Grundy của trò chơi Kayler có bảng sau đây:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
g(x)	0	1	2	3	1	4	3	2	1	4	2	6	4	1

Vị trí hiện tại (1, 11) có $g(1, 11) = g(1) \oplus g(11) = 1 \oplus 6 = 7 > 0$ là vị trí N.

Nếu đánh đỗ một quân đứng riêng lẻ thì còn cọc 11 quân có $g(11) = 6 > 0$, là vị trí N.

Nếu đánh đỗ một hoặc hai quân ở hai đầu cọc 11 quân (không chia đôi cọc 11) để còn cọc 10 hoặc 9 quân thì đến các vị trí (1, 10) hoặc (1, 9) đều là vị trí N vì:

$g(1, 10) = 1 \oplus 2 > 0$ hoặc $g(1, 9) = 1 \oplus 4 > 0$.

Vậy cần phân chia cọc 11 thành hai cọc mới:

- Nếu đánh đỗ một quân, chia 10 quân còn lại thành hai cọc sẽ đến các vị trí mới là (1, 1, 9), (1, 2, 8), (1, 3, 7), (1, 4, 6), (1, 5, 5).
- Nếu đánh đỗ hai quân, chia 9 quân còn lại thành hai cọc sẽ đến các vị trí mới là (1, 1, 8), (1, 2, 7), (1, 3, 6), (1, 4, 5).

Nhận thấy $g(1, 3, 7) = 1 \oplus 3 \oplus 2 = 0$ vậy có thể chọn phép chuyển chiến thắng là đánh đỗ một quân chia cọc 11 quân thành hai cọc mới có ba quân và bảy quân. Có thể kiểm tra thấy đây cũng là phép chuyển duy nhất để chiến thắng. Quân ki phải hạ đỗ là 6 hoặc 10.

Chương trình: a)

```
#include <fstream>
using namespace std;
const int maxn=83;
int g[maxn];
void tinh(int m) {
    int i, j, p;
    int c[maxn];
```

```

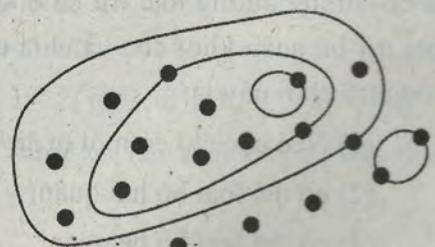
for (int i=0; i<maxn; i++) c[i]=0;
c[g[m-1]]=1; // đánh đố 1 quân ở một đầu
c[g[m-2]]=1; // đánh đố 2 quân ở một đầu
i=1; j=m-2; // các cách đánh đố 1 quân ở giữa
while (i<=j) {
    c[g[i]^g[j]]=1;
    i++; j--;
}
i=1; j=m-3; // các cách đánh đố 2 quân ở giữa
while (i<=j) {
    c[g[i]^g[j]]=1;
    i++; j--;
}
i=0;
while (c[i]==1) i++;
g[m] = i;
}

int main() {
    g[0]=0; g[1]=1; g[2]=2;
    for (int k=3; k<=maxn; k++) tinh(k);
    ofstream fo ("BAI18_C9.out");
    for (int k=0; k<=maxn; k++) {
        fo<<g[k]<<' ';
        if (k%12==11) fo<<'\n';
    }
    return 0;
}

```

9.19. a) Coi mỗi miền trong trò chơi Rims như một cọc trong trò chơi Nim nhiều cọc. Mỗi phép di chuyển trong trò chơi Rims sẽ chia trang giấy thành các miền và loại bỏ ít nhất một chấm tròn. Điều này giống như phép di chuyển trong trò chơi Nim, ngoại trừ một điều là phép di chuyển trong Rims còn có thể chia một vùng thành hai vùng không rỗng, nghĩa là thêm cọc mới.

Nhưng do $a \oplus b \leq a + b$ nên khi một cọc $(x + a + b)$ chia thành hai cọc a và b và mất đi x quân thì coi như cọc $(x + a + b)$ mất đi một số quân là $x + a \oplus b$. Vậy chơi trò Rims ta có thể coi như chơi theo Nim nhiều cọc. Từ một vị trí x_1 dẫn đến x_2 trong Nim, ta luôn tìm được một phép chuyển thích hợp trong Rims để từ vị trí tương ứng y_1 dẫn đến y_2 và ngược lại (kí hiệu \oplus là

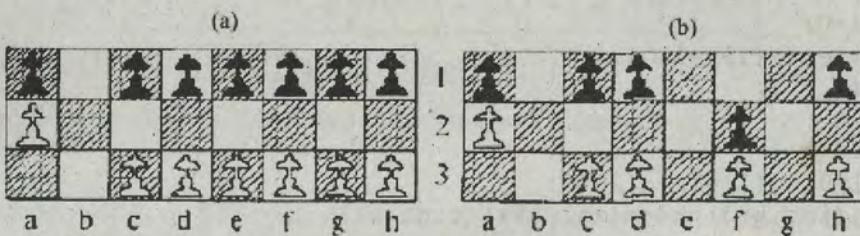


phép XOR).

b) Trong hình bên là vị trí $(3, 4, 5)$ có giá trị Nim là $3 \oplus 4 \oplus 5 = 2$ là vị trí N, cần chuyển đến vị trí P $(1, 4, 5)$ vì $1 \oplus 4 \oplus 5 = 0$. Điều này có thể thực hiện bằng cách vẽ vòng kín qua hai trong ba chấm ngoài cùng.

Chương trình: *Học sinh tự lập trình.*

9.20. a) Ví dụ nếu Trắng đi $a3 - a2$, Đen đi $b1 - a2$ bắt quân Trắng ở $a2$, Trắng đi $b3 - a2$ bắt quân đen tại $a2$. Có hình ảnh như hình (a): một cặp quân tốt bị loại và đổi lượt đi.



Bây giờ đến lượt Đen đi. Giả sử Đen đi $f1 - f2$, Trắng $e3 - f2$, Đen $e1 - f2$, Trắng $g3 - f2$, Đen $g1 - f2$. Có kết quả như hình (b): hai cặp quân tốt bị loại và đổi lượt đi.

Do vậy trò chơi này có thể coi là trò chơi trên một hàng có n ô vuông để trống. Hai người chơi lần lượt ghi X vào một ô trống không kề với ô đã có X. Người ghi chữ X cuối cùng thắng.

Trò chơi này có thể mô tả như trò chơi trên một cọc có thể bỏ bớt các quân trên cọc hoặc chia cọc thành hai cọc. Nếu $n = 1$ có đúng một phép chuyển đổi $n = 0$. Với $n > 1$, một phép ghi X vào ô ở các đầu cuối hàng là tương ứng loại trừ hai ô ở đầu đó của hàng, nghĩa là loại trừ hai quân trên cọc. Đặt X tại ô cách ô đầu và ô cuối một ô tương đương loại trừ ba ô khỏi hàng, nghĩa là loại trừ ba quân khỏi cọc. Đặt X tại ô không phải đầu cuối như hai loại trên (và không phải tại ô kề ô X đã có) tương đương loại trừ ba ô khỏi hàng và chia hàng thành hai hàng, nghĩa là loại trừ ba quân khỏi cọc và chia cọc thành hai cọc mới. Vậy quy luật di chuyển trong trò chơi này là:

- (1) Nếu cọc chỉ có một quân thì loại bỏ quân này;
- (2) có thể loại bỏ hai quân;
- (3) có thể loại bỏ ba quân;

(4) chia cọc đó thành hai cọc có tổng số quân giảm đi ba.

Do đó, để thuận tiện khi lập trình tìm các giá trị hàm Sprague-Grundy của trò chơi Dawson, chúng ta phân tích thêm một vài quy ước sau:

Giả sử đã tính được giá trị hàm Sprague-Grundy của các cọc 0, 1, 2, ..., 9. Bây giờ cần tính $g(11)$. Với quy luật trên, từ cọc 11 sẽ đi tới cọc 9, cọc 8, hai cọc (1, 7), hai cọc (2, 6), hai cọc (3, 5) và hai cọc (4, 4). Nhưng cọc 9 cũng có thể coi như hai cọc (-1, 9), cọc 8 coi như hai cọc (0, 8) và quy ước thêm $g(-1) = 0$ thì ta có thể dễ viết công thức tính $g(11)$ là số nguyên không âm nhỏ nhất chưa có trong các số $g(a) \oplus g(b)$ mà $a + b = 8 (=11 - 3)$.

Các giá trị hàm Sprague-Grundy tuần hoàn với chu kì là 34, có 7 trường hợp ngoại lệ khi n thuộc $[0; 51]$ (các số in nghiêng) như sau:

0 1 1 2 0 3 1 1 0 3 3 2 2 4 0 5 2 2 3 3 0 1 1 3 0 2 1 1 0 4 5 2 7
4

0 1 1 2 0 3 1 1 0 3 3 2 2 4 4 5 5 2 3 3 0 1 1 3 0 2 1 1 0 4 5 3 7
4

8 1 1 2 0 3 1 1 0 3 3 2 2 4 4 5 5 9 3 3 0 1 1 3 0 2 1 1 0 4 5 3 7
4

8 1 1 2 0 3 1 1 0 3 3 2 2 4 4 5 5 9 3 3 0 1 1 3 0 2 1 1 0 4 5 3 7
4

...

b) Kết quả thực hiện cho bảng giá trị hàm Sprague-Grundy của trò chơi trong trường hợp $n = 18$ là:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
g(x)	0	1	1	2	0	3	1	1	0	3	3	2	2	4	0	5	2	2	3

Tìm bước chuyển chiến thắng khi $n = 18$ như sau:

Từ vị trí 18 có thể đi tới các vị trí là 16, 15, (1, 14), (2, 13), (3, 12), (4, 11), 5, 10), (6, 9), (7, 8), có giá trị Sprague-Grundy tương ứng là: 2, 5, $g(1) \oplus g(14) = 1 \oplus 0 = 1$, $g(2) \oplus g(13) = 1 \oplus 4 = 5$, $g(3) \oplus g(12) = 2 \oplus 2 = 0$, $g(4) \oplus g(11) = 0 \oplus 2 = 2$, $g(5) \oplus g(10) = 3 \oplus 3 = 0$, $g(6) \oplus g(9) = 1 \oplus 3 = 2$, $g(7) \oplus g(8) = 1 \oplus 0 = 1$.

Vậy từ vị trí 18 ($g(18) = 3 > 0$) nếu Trắng đi trước có hai cách đi ban đầu theo thuật thằng là phải đi đến được (3, 12) hoặc (5, 10).

Nếu muốn dẫn tới (3, 12) cần đi như sau:

Trắng đi e₃ – e₂, khi đó Đen buộc phải đi d₁ – e₂ (hoặc f₁ – e₂) và Trắng buộc phải đi d₃ – e₂ (hoặc f₃ – e₂). Tiếp theo Đen buộc phải đi f₁ – e₂ (hoặc d₁ – e₂) và Trắng buộc phải đi f₃ – e₂ (hoặc d₃ – e₂). Đến đây bàn cờ chia thành hai bàn cờ với kích thước 3 và 12 (là vị trí có giá trị Sprague-Grundy bằng 0, và trao cho Đen đi):

Đ	Đ	Đ		Đ		Đ	Đ	Đ	Đ	Đ	Đ	Đ	Đ	Đ	Đ	Đ	Đ	Đ	Đ
				T															
T	T	T				T	T	T	T	T	T	T	T	T	T	T	T	T	T

a b c d e f ...

Tương tự Trắng cũng có thể đi g₃ – g₂ ..., chia bàn cờ thành hai bàn cờ với kích thước 5 và 10.

Chương trình:

```
#include <fstream>
using namespace std;
const int maxn=18;
int g[maxn];
void tinh(int m) {
    int i, j, p;
    int c[maxn];
    for (int i=0; i<=maxn; i++) c[i]=0;
    i=-1; j=m-i-3;
    while (i<=j) {
        c[g[i]^g[j]]=1;
        i++; j--;
    }
    i=0;
    while (c[i]==1) i++;
    g[m] = i;
}
int main() {
    g[-1]=0; g[0]=0; g[1]=1; g[2]=1;
    for (int k=3; k<=maxn; k++) tinh(k);
    ofstream fo ("BAI20_C9.out");
    for (int k=0; k<=maxn; k++)
        fo<<g[k]<<' ';
    return 0;
}
```

9.2
với:
Kết
y=

Chu

#inc
#inc
#inc
#inc
cons
usir
long
int

long

E TLC

9.21. Dựa vào định nghĩa và các tính chất của tích Nim, liên quan của tích Nim với số Ferma 2^{2^n} (xem trang 66 Tài liệu chuyên Tin học, Quyển 3).

Kết quả tích Nim của hai số x và y (là $x \otimes y$) cho trong bảng sau:

		$x =$																
		$y =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Đ	Đ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		2	0	2	3	1	8	10	11	9	12	14	15	13	4	6	7	5
		3	0	3	1	2	12	15	13	14	4	7	5	6	8	11	9	10
		4	0	4	8	12	6	2	14	10	11	15	3	7	13	9	5	1
		5	0	5	10	15	2	7	8	13	3	6	9	12	1	4	11	14
		6	0	6	11	13	14	8	5	3	7	1	12	10	9	15	2	4
đối	kích	7	0	7	9	14	10	13	3	4	15	8	6	1	5	2	12	11
		8	0	8	12	4	11	3	7	15	13	5	1	9	6	14	10	2
		9	0	9	14	7	15	6	1	8	5	12	11	2	10	3	4	13
		10	0	10	15	5	3	9	12	6	1	11	14	4	2	8	13	7
		11	0	11	13	6	7	12	10	1	9	2	4	15	14	5	3	8
		12	0	12	4	8	13	1	9	5	6	10	2	14	11	7	15	3
		13	0	13	6	11	9	4	15	2	14	3	8	5	7	10	1	12
		14	0	14	7	9	5	11	2	12	10	4	13	3	15	1	8	6
		15	0	15	5	10	1	14	4	11	2	13	7	8	3	12	6	9

Chương trình:

```
#include <iomanip>
#include <iostream>
#include <fstream>
#include "math.h"
const long nfn = 5;
using namespace std;
long fecma[] = {2, 4, 16, 256, 65536};
int Fec(long x) { // Kiểm tra số x có dạng Ferma không?
    for (long i=0; i<nfn; ++i)
        if (fecma[i]==x) return 1;
    return 0;
}
long tichnim(long y, long x) { // Tính tích Nim y ⊗ x
    if (x<y) return tichnim(x,y); // Chuyển về x ⊗ y với x < y
    if (x*y==0) return 0; // Tính chất x ⊗ 0 = 0 ⊗ x = 0
    if ((x==1)&&(y==1)) return 1; // Tính chất x ⊗ 1 = 1 ⊗ x = x
    // x >= y
    if ~(Fec(x)) {
        if (x==y) return (x*3/2); // Tính chất  $2^x \otimes 2^x = \frac{3}{2} \times 2^x$ 
        // x > y
        return x*y; // Tính chất  $2^x \otimes y = 2^x \times y$  với mọi
    }
    // Phân tích x thành tổng Nim hoặc tích Nim của x1 và x2 mà x1 có dạng luỹ thừa của 2
}
```

```

long x1 = 1;
while (x1 * 2 <= x) x1 *= 2;
long x2 = x ^ x1; // suy ra x=x1 ⊕ x2
if (x2 == 0) { // x=(2^n) ⊕ 0=2^n;
    long i = 0;
    while (fecma[i+1] < x) i++; // Phân tích x=x1 * x2 (mà x1 là Fermat)
    x1 = fecma[i];
    x2 = x / x1;
    long t;
    // y ⊗ (x1 * x2) = y ⊗ x1 ⊕ y ⊗ x2 hoặc y ⊗ x2 ⊕ y ⊗ x1
    // Dùng kĩ thuật lấy kết quả ngẫu nhiên của một trong hai trường hợp để tránh bê tặc
    if (rand() % 2 == 1) {
        long t1 = tichnim(y, x1);
        t = tichnim(t1, x2);
    }
    else {
        long t1 = tichnim(y, x2);
        t = tichnim(t1, x1);
    }
    return t;
}
else { // x không có dạng lũy thừa của 2 thì: x=x1 ⊕ x2 --> y ⊗ x = (y ⊗ x1) ⊕ (y ⊗ x2)
    long t1 = tichnim(y, x1);
    long t2 = tichnim(y, x2);
    long t = t1 ^ t2;
    return (t);
}
}

int main() {
    ofstream fo ("BAI21_C9.out");
    for (long i = 0; i <= 15; ++i) {
        for (long j = 0; j <= 15; ++j)
            fo << setfill(' ') << setw(3) << setprecision(2) << tichnim(i, j);
        fo << '\n';
    }
    return 0;
}

```

9.22. Thêm vị trí -1 (các đồng xu đều đã bị lật sấp) là vị trí cần đi tới của đồng xu ngửa xu tại vị trí 0 . Vậy $g(-1) = 0$ và $g(0) = 1$. Từ vị trí đồng xu ngửa tại n có thể đi tới các vị trí $n - 1, n - 2, \dots, 0$, nên $g(n) = n + 1$.

Lưu ý. Nếu các đồng xu đánh số từ 1 đến n thì $g(n) = n$ như trò Nim chuẩn một cọc.

Chương trình: *Học sinh tự lập trình.*

9.23. Mỗi đồng xu ngửa tại vị trí x và mọi đồng xu bên trái nó đều xấp được coi là trạng thái bắt đầu của trò chơi lật xu Ruler với x đồng xu đánh số từ 1 đến x . Giả sử đồng xu tại vị trí n ngửa và các đồng xu khác bên trái nó đều sấp thì trạng thái tiếp theo sau lượt đi là:

- Chỉ có đồng xu $n - 1$ ngửa, các đồng xu khác bên trái đều sấp;
- Có hai đồng xu ngửa là $n - 1$ và $n - 2$, các đồng xu khác bên trái đều sấp;
- Có ba đồng xu ngửa là $n - 1, n - 2, n - 3$, các đồng xu khác bên trái đều sấp;
- ...
- $n - 1$ đồng xu ở các vị trí $n - 1, n - 2, \dots, 3, 2, 1$ đều ngửa.

Do đó trạng thái tiếp theo là tổng của các trò chơi lật xu Ruler với số đồng xu là $n - 1, n - 2, \dots, 2, 1$. Suy ra giá trị hàm Sprague-Grundy của một vị trí n trong trò chơi là số nguyên không âm nhỏ nhất chưa có trong tập các trạng thái tiếp theo của n :

$$g(n) = \text{mex} \{0, g(n - 1), g(n - 1) \oplus g(n - 2), \dots, g(n - 1) \oplus g(n - 2) \dots \oplus g(1)\}$$

Lưu ý: $G(n)$ không thể bằng 0 nên trong công thức trên phải đưa thêm vào số 0. Từ đó có thể tính được hàm g như sau:

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
g(x)	1	2	1	4	1	2	1	8	1	2	1	4	1	2	...

Chương trình:

```
#include <fstream>
#include <iomanip>
const long n = 500, MaxN = n*n;
using namespace std;
int main() {
    int g[MaxN];
    g[0] = 0;
    for (long i = 1; i<=n; ++i) { //xét các trạng thái i
        int v[MaxN];
        for (long k = 0; k<MaxN; ++k) v[k] = 0; //Tạo dãy "chuồng"
        for (long k = 1; k<=i; ++k) { //Vị trí bên trái nhất của dãy xu liên tiếp bị lật
            long t = 0;
            for (long j=k; j<i; ++j) //lật các xu từ k tới i-1
                t=t^g[j]; //Mỗi số hạng có dạng là  $g(k) \oplus g(k+1) \oplus \dots \oplus g(i-1)$ 
            v[t] = 1; //Lùa t vào "chuồng"
        }
        g[i] = v[0];
    }
}
```

```

long k = 0;
while (v[k] == 1) k++; //Duyệt "chuồng"
g[i] = k;
}
ofstream fo ("BAI23_C9.out");
for (long i = 1; i <=n; ++i) {
    fo<<setfill(' ')<<setw(4)<<setprecision(3)<<g[i];
    if (i % 20==0) fo<<'\n';
}
return 0;
}

```

9.24. Nếu đánh số đồng xu từ 0, thì các đồng xu ngửa tại vị trí 0, 1, và 2 là các trạng thái kết thúc (vì không đủ bốn đồng xu cho một phép lật) và do đó $g(0) = g(1) = g(2) = 0$. Vị trí $n \geq 3$ (ứng với đồng xu tại n ngửa, các đồng xu bên trái nó đều sấp) có thể biến thành trạng thái $0, x, n-x$, với $x < n/2$. Do $g(0) = 0$ nên $g(n) = \text{mex}\{g(x) \oplus g(n-x) | x < n/2\}$, vậy trò chơi Grunt tương đương với trò chơi tổng của các trò chơi chia đôi đồng sỏi thành hai phần với số lượng khác nhau.

Các giá trị hàm Sprague-Grundy của trò chơi này là:

0	0	0	0	0	1	0	2	1	3	0	4	2	1	3	0	4	5	1	3
0	2	5	4	6	8	2	1	3	0	5	7	1	3	2	5	0	4	6	1
3	8	2	0	5	7	4	2	8	3	1	4	6	7	10	8	2	1	7	6
4	10	8	3	0	2	7	5	4	8	10	1	3	2	8	5	9	10	0	12
3	9	6	8	4	7	12	2	1	11	9	10	7	3	11	4	14	7	3	1
2	6	9	11	12	10	6	3	11	0	7	1	4	6	5	7	15	14	8	11
16	5	6	17	3	9	2	6	5	7	17	2	9	5	7	17	10	16	11	15
14	19	4	6	8	12	16	13	0	2	6	8	13	22	4	15	11	10	2	16...

Chương trình:

```

#include <fstream>
#include <iomanip>
const long n = 160, MaxN = n*n;
using namespace std;
int main() {
    int g[MaxN];
    g[0] = 0;
    for (long i = 1; i<=n; ++i) {
        int v[MaxN];
        for (long k=0; k<= MaxN; ++k) v[k] = 0;
        for (long k=1; k<(i-1)/2; ++k)
            v[g[k]^g[i-k]] = 1;
    }
    ofstream fo ("BAI23_C9.out");
    for (long i = 1; i <=n; ++i) {
        fo<<setfill(' ')<<setw(4)<<setprecision(3)<<g[i];
        if (i % 20==0) fo<<'\n';
    }
    return 0;
}

```

```

long k = 0;
while (v[k] == 1) k++;
g[i] = k;
}
ofstream fo ("BAI24_C9.out");
for (long i = 0; i<=n; ++i) {
    fo<<setfill(' ')<<setw(3)<<setprecision(2)<<g[i];
    if (i % 20==19) fo<<'\n';
}
return 0;
}

```

9.25. Một lần đi là lật bốn đồng xu tại bốn góc của một hình chữ nhật $\{(a, b), (x, y)\}$ thỏa mãn $0 \leq a < x, 0 \leq b < y$. Đồng xu tại vị trí (x, y) đang ngửa được lật thành sấp, ba đồng xu kia đang sấp được lật thành ngửa.

Hàm Sprague-Grundy được xác định như sau:

$$g(x, 0) = g(0, y) = 0, g(1, 1) = 1,$$

$$g(x, 1) = \text{mex} \{g(a, 1) : 0 \leq a < x\} = x, g(1, y) = \text{mex} \{g(1, b) : 0 \leq b < y\} = y,$$

$$g(x, y) = \text{mex} \{g(x, b) \oplus g(y, a) \oplus g(a, b) : 0 \leq a < x, 0 \leq b < y\}$$

Từ đó tìm được bảng các giá trị của hàm Sprague-Grundy của các vị trí (x, y) . Nhận thấy bảng này cũng chính là bảng giá trị tích Nim của x và y . Vậy với định nghĩa hàm Sprague-Grundy của trò chơi này ta có một cách khác tính tích Nim đã đề cập ở bài 9.21. Ngược lại, có thể sử dụng bài 9.21 để tìm hàm Sprague-Grundy của trò chơi này.

Bảng sau cho giá trị hàm Sprague-Grundy của trò chơi Turning Corners với kích thước $n = 4$:

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	3	1	8
3	0	3	1	2	12
4	0	4	8	12	6

Dựa vào bảng giá trị này, chúng ta có thể thực hiện thuật toán giành thắng từ vị trí ban đầu. Ví dụ:

	0	1	2	3	4
0	T	T	T	T	T
1	T	T	T	H	T
2	T	T	T	T	T
3	T	T	H	T	T
4	T	T	T	T	H

Vị trí ban đầu

	0	1	2	3	4
0	T	T	T	T	T
1	T	T	T	H	T
2	T	T	T	T	T
3	T	T	H	H	H
4	T	T	T	H	T

Vị trí tiếp theo

Ta có giá trị hàm Sprague-Grundy tại vị trí ban đầu của bảng là:

$g(1, 3) \oplus g(3, 2) \oplus g(4, 4) = 3 \oplus 1 \oplus 6 = 4$ là vị trí N, và có thể di chuyển đến vị trí P bằng cách lật các đồng xu tại bốn góc của hình chữ nhật $\{(3, 3), (4, 4)\}$. Khi đó giá trị hàm Sprague-Grundy của vị trí mới là:

$$g(1, 3) \oplus g(3, 2) \oplus g(3, 3) \oplus g(3, 4) \oplus g(4, 3) = 3 \oplus 1 \oplus 2 \oplus 12 \oplus 12 = 0.$$

Chương trình:

a) Tính giá trị hàm Sprague-Grundy của trò chơi

```
#include <fstream>
#include <iomanip>
const int maxn=25;
using namespace std;
int g[maxn][maxn], v[maxn*maxn];
int n=15;
int sg(int x, int y) {
    for (int i=0; i<maxn*maxn; i++) v[i]=0;
    for (int a=0; a<x; a++)
        for (int b=0; b<y; b++)
            v[(g[x][b])^(g[a][y])^(g[a][b])] = 1;
    int i=0;
    while (v[i]==1) i++;
    return i;
}
int main() {
    g[1][1]=1;
    for (int y=2; y<=n; y++) g[1][y]=y;
    for (int x=2; x<=n; x++) g[x][1]=x;
    for (int y=2; y<=n; y++)
        for (int x=2; x<=n; x++) g[x][y]=sg(x,y);
    ofstream fo ("bt25a.out");
    for (long i = 0; i <=n; ++i) {
```

```

        for (long j = 0; j<=n; ++j)
            fo<<setfill(' ')<<setw(4)<<setprecision(3)<<(g[i][j]);
        fo<<'\n';
    }
}

```

b) Tìm bước đi đầu tiên trong Turning Corners

```

#include <iostream>
const int maxn=21;
using namespace std;
struct pt {int x; int y;};
int a[maxn][maxn], g[maxn][maxn];
int n, sg, lc, ld;
pt q[maxn*maxn];
bool ok;
int tichnim(int x, int y) { //Tính giá trị hàm Sprague-Grundy của vị trí (x, y)
    int v[maxn*maxn];
    for (int i=0; i<maxn*maxn; i++) v[i]=0;
    for (int a=0; a<x; a++)
        for (int b=0; b<y; b++)
            v[(g[x][b])^(g[a][y])^(g[a][b])] = 1;
    int i=0;
    while (v[i]==1) i++;
    return i;
}
bool duyet(int i) { // ô phải-dưới (q[i].x, q[i].y) có quân đang ngửa
// Tìm góc trái-trên (ld,lc) nếu lật 4 quân tại 4 góc hình chữ nhật (ld, lc, q[i].x, q[i].y)
int x1, x2, d, c;
ok=false;
    x1 = sg ^ (g[q[i].x][q[i].y]); // trạng thái khi lật góc phải-dưới
    for (int d=q[i].x-1; d>=0; d--)
        for (int c=q[i].y-1; c>=0; c--) {
// Tìm góc trái-trên (d,c) sao cho 3 góc còn lại đều có quân sáp
            if ((a[d][c]==0) && (a[q[i].x][c]==0) && (a[d][q[i].y]==0))
// sau khi lật 3 quân này, trạng thái trò chơi nếu bằng 0 thì lưu lại góc trái-trên, thoát
                if ((x1^(g[d][c])^(g[q[i].x][c])^(g[d][q[i].y])) == 0) {
                    lc = c; ld = d; ok=true;
                    return ok;
                }
        }
    return ok;
}
int main() {
// Xây dựng các giá trị hàm Sprague-Grundy của các vị trí trò chơi thành phần
    for (int i=0; i<maxn; i++)
        for (int j=0; j<maxn; j++)
            g[i][j] = tichnim(i,j);
}

```

```

// đọc trạng thái ban đầu của trò chơi
ifstream fi ("BAI25b_C9.inp");
    fi>>n;
int dem = 0;
for (int i=0; i<=n; i++)
for (int j=0; j<=n; j++) {
    fi>>a[i][j];
    if (a[i][j]==1) {
        dem++; // số lượng các ô có quân ngựa
        q[dem].x = i;
        q[dem].y = j;
    };
}
ofstream fo ("BAI25b_C9.out");
if (dem==0) { // nếu không có quân nào ngựa thì người đi đầu thua
    fo<<0;
    return 0;
}
// Tính giá trị hàm Sprague-Grundy của trạng thái ban đầu (của trò chơi tổng)
sg = g[q[1].x][q[1].y];
for (int i= 2; i<=dem; i++) sg ^= g[q[i].x][q[i].y];
if (sg==0) // Trạng thái ban đầu thuộc tập P(SG bằng 0), thì người đi đầu thua
    fo<<0;
else { // Trạng thái ban đầu thuộc tập N thì tìm nước đi đầu tiên giành thắng
    for (int i=dem; i>=1; i--) { // Lựa góc phải-dưới
        bool ok=duyet(i); // xem có phép di chuyển tới trạng thái 0?
        if (ok) { // có thể tới trạng thái 0 thì thắng, ghi bước đi giành thắng
            fo << 1 << '\n';
            fo<<ld<< ' '<<lc<< ' '<<q[i].x<< ' '<<q[i].y;
            break;
        }
    }
    if (!ok) fo << 0; // không có phép di chuyển tới trạng thái 0, thì thua
}
return 0;
}

```

9.26. Giả sử A xuống Hải Dương với xác suất p và xuống Hải Phòng với xác suất $1 - p$. Dựa vào ma trận lượng giá có phương trình: $100p = -50p + (1 - p)100$, suy ra: $p = 2/5$.

		Tội phạm B xuống ga	
		Hải Dương	Hải Phòng
Công an A xuống ga	Hải Dương	100	-50
	Hải Phòng	0	100

Vậy chiến thuật hỗn hợp tối ưu cho A là $(2/5; 3/5)$.

Tương tự: Nếu B xuống Hải Dương với xác suất p và xuống Hải Phòng với xác suất $1 - p$ thì: $100p - 50(1 - p) = (1 - p)100$, suy ra: $p = 3/5$.

Vậy chiến thuật hỗn hợp tối ưu cho B là $(3/5; 2/5)$.

Giá trị trò chơi (tính cho A nhận được) là $v = 100(2/5) = 40$.

Chương trình: *Học sinh tự lập trình.*

9.27. b) Trò chơi ma trận 2×2 : Xem phần hướng dẫn giải bài 9.27 trong Tài liệu chuyên Tin học, Quyển 3, trang 155.

Chương trình:

a) Chương trình tìm điểm yên ngựa của ma trận $m \times n$ (m dòng, n cột)

```
#include <iostream>
#include <iomanip>
using namespace std;
const int maxn = 10;
float a[maxn][maxn];
float value;
int n, m, dong, cot;
float mindong[maxn], maxcot[maxn];
bool ok;
void nhap() {
    ifstream fi ("BAI27a_C9.inp");
    fi>>m>>n;
    for (int i=1; i<=m; i++)
        for (int j=1; j<=n; j++) fi>>a[i][j];
}
void timmindong() {
    for (int i=1; i<=m; i++) {
        mindong[i] = 1.0E+30;
        for (int j=1; j<=n; j++)
            if (mindong[i]>a[i][j]) mindong[i] = a[i][j];
    }
}
void timmaxcot() {
    for (int j=1; j<=n; j++) {
        maxcot[j] = -1.0E+30;
        for (int i=1; i<=m; i++)
            if (maxcot[j]<a[i][j]) maxcot[j] = a[i][j];
    }
}
bool timyenngua() {
    ok = false;
    for (int i=1; i<=m; i++)
```

```

for (int j=1; j<=n; j++) {
    if (mindong[i]==maxcot[j]) {
        dong = i;
        cot = j;
        value = a[i][j];
        ok = true;
        return ok;
    }
}
return ok;
}
int main() {
    nhap();
    timmindong();
    timmaxcot();
    ofstream fo ("BAI27a_C9.out");
    if (!timyenngua()) {
        fo<<"Khong co diem yen ngua ";
        value = (a[1][1]*a[2][2]-a[1][2]*a[2][1]);
        value /= (a[1][1]-a[1][2]+a[2][2]-a[2][1]);
        fo << " Gia tri la : " << value;
    }
    else {
        fo<<"Diem yen ngua tai dong: "<< dong<< ", cot: "<< cot
            << " gia tri la : ";
        fo<< setfill(' ')<<setw(4)<<setprecision(3)<<value;
    }
    return 0;
}

```

9.28. Chọn các bộ ba hàng i_1, i_2, i_3 và số thực p tăng dần từ 0 đến 1 (p tăng dần từng 0.1 chẳng hạn) để kiểm tra $p*(\text{hàng } i_1) + (1 - p)*(\text{hàng } i_2) \geq \text{hàng } i_3$ (1). Nếu có giá trị p thỏa mãn (1) thì có thể loại bỏ hàng i_3 .

Tương tự, nếu có p để $p*(\text{cột } j_1) + (1 - p)*(\text{cột } j_2) \leq \text{cột } j_3$ (2) thì có thể loại bỏ cột j_3 .

Chương trình:

```

#include <fstream>
#include <iostream>
#include <iomanip>
using namespace std;
ifstream fi ("BAI28a_C9.inp");
ofstream fo ("BAI28a_c9.out");
const int maxn = 20;
int b[maxn][maxn];
float a[maxn][maxn];

```

```

int m, n;
bool ok, cot, hang;
bool ddc[maxn], ddh[maxn]; // đánh dấu cột, hàng đã loại
void read_input() { // đọc tệp input
    fi>>m>>n;
    for (int i=1; i<=m; i++)
        for (int j=1; j<=n; j++) fi>>a[i][j];
}
void ghi() { // ghi ma trận sau mỗi lần giản ước
    for (int i=1; i<=m; i++) {
        for (int j=1; j<=n; j++)
            if (b[i][j]==0) // chưa loại ô(i,j)
                fo<<setfill(' ')<<setw(4)<<setprecision(3)<<a[i][j];
        fo<<'\'n';
    }
    fo<<'\'n';
}
void xuli_hang() { // Kiểm tra xem có thể loại bỏ hàng nào thì loại bỏ
float p;
int j, t;
hang = false;
// Kiểm tra xem hai hàng i1 và i2 có chia phôi hàng i3 không?
for (int i1=1; i1<m; i1++)
for (int i2=i1+1; i2<=m; i2++)
for (int i3=1; i3<=m; i3++)

if ((ddh[i1]==true) && (ddh[i2]==true) && (ddh[i3]==true)) // chưa loại
if ((i3!=i1) && (i3!=i2)) { // và hàng i3 khác i1 và i2
    p = 0.000;
    ok = false;
    while ((!ok) && (p<=1)) {
        ok = true;
        for (int j=1; j<=n; j++)
            if ((b[i1][j]==0) && (b[i2][j]==0) && (b[i3][j]==0)) // chưa loại cột j
                if (p*a[i1][j]+(1-p)*a[i2][j]<=a[i3][j]) {
                    ok = false;
                    break;
                }
        if (!ok) p += 0.1; // điều chỉnh tăng thêm p
    else { // p*a[i1][j]+(1-p)*a[i2][j]>=a[i3][j] thì loại bỏ hàng i3
        ok=true;
        hang = true;
        for (int t=1; t<=n; t++) b[i3][t]=1; // loại bỏ các phần tử hàng i3
        ddh[i3]= false; // loại bỏ hàng i3
        if (p>0) { // hai hàng i1 và i2 chia phôi hàng 3
            fo<<"hang : ";

```

```

        fo<<setfill('
')<<setw(4)<<setprecision(3)<<p<<"*"><<i1;
        fo<<setfill(' ')<<setw(4)<<setprecision(3)<<(1-p);
        fo<<" * "<<i2<<" chi phoi "<<i3<<'\n';
    }
    else //p=0, nên chỉ có hàng i2 chỉ phôi hàng i3
        fo<<"hang : "<<i2<<" chỉ phoi "<<i3<<'\n';
    ghi(); //ghi ma trận sau khi loại bỏ hàng i3
    break;
}
} //hết vòng while
}

void xuli_cot() { //tương tự xử lý hàng
    float p, q;
    int t;
    cot = false;
    for (int j1=1; j1<n; j1++)
    for (int j2=j1+1; j2<=n; j2++)
    for (int j3=1; j3<=n; j3++)
    if ((ddc[j1]==true) && (ddc[j2]==true) && (ddc[j3]==true))
    if ((j3!=j1)&&(j3!=j2)) {
        q = 0.000;
        ok = false;
        while ((!ok)&&(q<=1)){
            ok = true;
            for (int i=1; i<=m; i++)
            if ((b[i][j1]==0)&&(b[i][j2]==0)&&(b[i][j3]==0))
            if (q*a[i][j1]+(1-q)*a[i][j2]>a[i][j3]) {
                ok = false;
                break;
            } //q*a[i][j1]+(1-q)*a[i][j2]<=a[i][j3] thì loại bỏ cột j3
            if (!ok) q += 0.1;
            else {
                cot = true;
                for (int t=1; t<=m; t++) b[t][j3] = 1;
                ddc[j3] = false;
                if (q>0) {
                    fo<<"cot: "<<setfill(' ')<<setw(4)<<setprecision(3)<<q;
                    fo<<"*"><<j1<<' '<<setfill(' ')<<setw(4)<<setprecision(3)<<(1-q);
                    fo<<"*"><<j2<<" chỉ phoi "<<j3<<'\n';
                }
                else
                    fo<<"cot : "<<j2<<" chỉ phoi "<<j3<<'\n';
                ghi();
                break;
            }
        }
    }
}

```

```

        }
    }

int main() {
    read_input();
    for (int i=1; i<=m; i++) ddh[i]=true;
    for (int j=1; j<=n; j++) ddc[j]=true;
    cot = true;
    hang = true;
    while ((cot==true) || (hang==true)) {
        xuli_cot();
        xuli_hang();
    }
    return 0 ;
}

```

9.29. Dựa vào định lí Minimax: Trong trò chơi hai, người có điểm tổng bằng 0 và hạn chế thì:

- (1) Tồn tại một số V hữu hạn (gọi là giá trị của trò chơi).
- (2) Có một chiến thuật hỗn hợp cho người thứ nhất mà giá trị trung bình đạt được của người thứ nhất ít nhất là V, bất kể người thứ hai làm gì.
- (3) Có một chiến thuật hỗn hợp cho người thứ hai mà giá trị trung bình người thứ hai phải trả nhiều nhất là V, bất kể người thứ nhất làm gì.

Chiến thuật hỗn hợp tối ưu của người thứ nhất là $\{p[1], p[2], p[3] \dots p[m]\}$ tương ứng với $\text{Max} \left\{ \text{Min} \left\{ \sum_{j=1}^n p[i] * a[i, j] \quad \forall i = 1, 2, \dots, m \right\} \mid \forall p \right\}$.

Chiến thuật hỗn hợp tối ưu của người thứ hai là $\{q[1], q[2], q[3], \dots, q[n]\}$ tương ứng với $\text{Min} \left\{ \text{Max} \left\{ \sum_{i=1}^m q[j] * a[i, j] \quad \forall j = 1, 2, \dots, n \right\} \mid \forall q \right\}$.

Chương trình:

```

3) << q;
(1-q);
'';
#include <iostream>
#include <iomanip>
using namespace std;
const int mn = 10;
    long maxlong=2000000000;
long a[mn][mn];
int m, n;
long minV, maxmin, maxV, minmax;
long x, y;

```

```

long lp[mn], p[mn], lq[mn], q[mn];
void read_input(ifstream &fi) {
    fi>>m>>n;
    for (int i=1; i<=m; i++)
        for (int j=1; j<=n; j++) {
            fi >> x;
            a[i][j]=x;
        }
}
void tinhmaxmin(long p[]){ //Max{Min{ $\sum_{j=1}^n p[i]*a[i,j] \quad \forall i=1,2,...m$ } |  $\forall p$ }
    minV = maxlong;
    for (int j=1; j<=n; ++j) {
        y = 0;
        for (int i=1; i<=m; i++) y += p[i]*a[i][j];
        if (y<minV) minV = y;
    }
    if (maxmin<minV) {
        maxmin= minV;
        for (int i=1; i<=m; ++i) lp[i]=p[i]; //lưu chiến thuật người thứ nhất
    }
}
void tinhminmax(long q[]){ //Min{Max{ $\sum_{i=1}^m q[j]*a[i,j] \quad \forall j=1,2,...,n$ } |  $\forall q$ }
    maxV = -maxlong;
    for (int i=1; i<=m; ++i) {
        x = 0;
        for (int j=1; j<=n; ++j) x += q[j]*a[i][j];
        if (x>maxV) maxV = x;
    }
    if (minmax>maxV) {
        minmax = maxV;
        for (int j=1; j<=n; ++j) lq[j]=q[j]; //lưu chiến thuật người thứ hai
    }
}
int main() {
    ifstream fi ("BAI29_C9.in");
    ofstream fo ("BAI29_C9.out");
    read_input(fi);
    maxmin = -maxlong;
    minmax = maxlong;
    p[1]=0;
    while (p[1]<=37) { //chọn bộ chiến thuật tối ưu cho người thứ nhất
        p[2] = 0;
        while (p[2]<=37) {
}

```

```

    p[3] = 0;
    while (p[3]<=37) {
        p[4] = 37-(p[1]+p[2]+p[3]);
        if (p[4]>=0)
            tinhmaxmin(p);
        p[3]++;
    }
    p[2]++;
}
p[1]++;
}

// ghi chiến thuật tối ưu cho người thứ nhất vào tệp output
fo<<"A dat: ";
fo<<setfill(' ')<<setw(15)<<setprecision(10)<<float(maxmin/37.0);
fo<<". Chien thuat: ";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lp[1]<<"/37";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lp[2]<<"/37";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lp[3]<<"/37";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lp[4]<<"/37"<<'\n';
q[1]=0;
while (q[1]<=37) { // chọn bộ chiến thuật tối ưu cho người thứ hai
    q[2] = 0;
    while (q[2]<=37) {
        q[3] = 0;
        while (q[3]<=37) {
            q[4] = 0;
            while (q[4]<=37) {
                q[5] = 37-(q[1]+q[2]+q[3]+q[4]);
                if (q[5]>=0) tinhminmax(q);
                q[4]++;
            }
            q[3]++;
        }
        q[2]++;
    }
    q[1]++;
}
// ghi chiến thuật tối ưu cho người thứ hai vào tệp output
fo<<"B dat: ";
fo<<setfill(' ')<<setw(15)<<setprecision(10)<<float(minmax/37.0);
fo<<". Chien thuat: ";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lq[1]<<"/37";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lq[2]<<"/37";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lq[3]<<"/37";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lq[4]<<"/37";
fo<<setfill(' ')<<setw(5)<<setprecision(4)<<lq[5]<<"/37"<<'\n';
fo<<"Gia tri tro choi la: ";
fo<< setprecision(8)<<(float) maxmin/37<<endl;
return 0;
}

```

9.30. Chạy chương trình 9.28 (giản ước ma trận). Từ ma trận:

$$A = \begin{pmatrix} 0 & -1 & 2 & 2 & 2 \\ 1 & 0 & -1 & 2 & 2 \\ -2 & 1 & 0 & -1 & 2 \\ -2 & -2 & 1 & 0 & -1 \\ -2 & -2 & -2 & 1 & 0 \end{pmatrix} \text{ đưa về } B = \begin{pmatrix} 0 & -1 & 2 \\ 1 & 0 & -1 \\ -2 & 1 & 0 \end{pmatrix} \text{ sau khi bỏ dòng 4, 5, cột 4, 5}$$

của A. Sau đó dựa vào định lí Minimax lập trình tương tự bài 9.29 sẽ tìm được chiến thuật hỗn hợp tối ưu cho từng người.

Cách khác là sử dụng định lí cân bằng (trang 89, Tài liệu chuyên Tin học, Quyển 3) và lưu ý rằng ma trận B là ma trận đối xứng bù nên giá trị trò chơi là $V = 0$. Vậy để tìm chiến thuật hỗn hợp tối ưu cho người thứ nhất hãy giải hệ:

$$\begin{cases} p_2 - 2p_3 = 0 \\ -p_1 + p_3 = 0 \\ 2p_1 - p_2 = 0 \\ p_1 + p_2 + p_3 = 1 \end{cases}$$

Sẽ có $p = q = (1/4; 1/2; 1/4; 0; 0)$ là chiến thuật tối ưu cho cả hai người.

Chương trình: Học sinh tự lập trình.

9.31. Xem trang 123, 124, Tài liệu chuyên Tin học, Quyển 3.

Có thể tạo ma trận nghịch đảo A^{-1} của ma trận khả đảo A bằng cách dùng các phép biến đổi sơ cấp để biến đổi đồng thời ma trận A và ma trận đơn vị E. Khi A thành ma trận đơn vị thì ma trận E thành ma trận A^{-1} . Các phép biến đổi sơ cấp trên ma trận là: nhân các phần tử của một hàng với số k khác 0 (định thức của ma trận tăng k lần), đổi chỗ hai hàng cho nhau (định thức của ma trận đổi dấu), cộng k lần hàng i1 vào hàng i2 (định thức của ma trận không đổi).

Chương trình:

```
#include <iostream>
#include <iomanip>
using namespace std ;
const int maxn = 10;
float a[maxn][maxn], e[maxn][maxn];
bool b[maxn];
int n;
void read_input(ifstream &fi) {
```

```

// đọc ma trận A
fi>>n;
for (int i=1; i<=n; i++)
for (int j=1; j<=n; j++) fi>>a[i][j];
// Tạo ma trận đơn vị E
for (int i=1; i<=n; i++)
for (int j=1; j<=n; j++) e[i][j]=0;
for (int i=1; i<=n; i++) e[i][i]=1;
}

void ghi(float x[][maxn], ofstream &fo) { //Ghi ma trận X vào tệp output
for (int i=1; i<=n; i++) {
    for (int j=1; j<=n; j++) {
        fo<<setfill(' ')<<setw(9)<<setprecision(4)<<x[i][j];
    }
    fo<<'\\n';
}
fo<<'\\n';
}

int xuli(ofstream &fo) {
    float x;
    for (int j=1; j<=n; ++j) {
        if (a[j][j]==0)
            for (int i=j+1; i<=n; ++i)
                if (a[i][j]!=0) {
                    for (int k=1; k<=n; ++k) {
                        x = a[j][k]; a[j][k] = a[i][k]; a[i][k] = x;
                        x = e[j][k]; e[j][k] = e[i][k]; e[i][k] = x;
                    }
                }
        else {
            fo<<"ma tran suy bien ";
            return 0;
        }
    }
    //chia hàng j cho a[j][j]
    x = 1.0/a[j][j];
    for (int k=1; k<=n; ++k) {
        a[j][k] = a[j][k]*x;
        e[j][k] = e[j][k]*x;
    }
    // Biến đổi các hàng i khác j
    for (int i=1; i<=n; ++i)
        if (i!=j) {
            x = -a[i][j];
            for (int k=1; k<=n; ++k) {
                a[i][k] = a[i][k]*a[j][j] + a[j][k]*x;
                e[i][k] = e[i][k]*a[j][j] + e[j][k]*x;
            }
        }
}

```

```

    }
}
return 0;
}
bool chua() { // Kiểm tra A đã thành ma trận đơn vị hay chưa
    for (int i=1; i<=n; i++)
        for (int j=1; j<=n; j++)
            if ((a[i][i]!=1) || ((i !=j) && (a[i][j]!=0))) return true;
    return false;
}
int main() {
    ifstream fi ("BAI31_C9.inp");
    ofstream fo ("BAI31_C9.out");
    read_input(fi);
    // ghi(a, fo); // Ghi ma trận ban đầu vào tệp output
    while (chua()) xuli(fo);
    // ghi(a, fo); // Ghi ma trận A sau biến đổi (đã thành E) vào tệp output
    ghi(e, fo); // Ghi ma trận nghịch đảo của A vào tệp output
    return 0;
}

```

- 9.32.** a) Chạy chương trình 9.27a thấy ma trận có điểm yên ngựa là $a[2, 3] = 1$.
 b) Lập chương trình tính định thức của A, chạy chương trình thấy định thức của A là $\det(A) = 5 \neq 0$ nên A có ma trận nghịch đảo.

Chạy chương trình 9.31 tìm được ma trận nghịch đảo của A là:

$$A^{-1} = \begin{pmatrix} 0.2 & -0.4 & 0.6 \\ 0.4 & 0.2 & 0.2 \\ -0.6 & 1.2 & -0.8 \end{pmatrix}$$

Từ đó suy ra nếu tồn tại chiến thuật p và q mà $p \geq 0$ và $q \geq 0$ thì có thể kết luận giá trị trò chơi là $V = 1/(1^T \cdot A^{-1} \cdot 1)$

- (ở đây kí hiệu $1^T \cdot A^{-1} \cdot 1$ coi là tổng các phần tử của A^{-1}) nên $V = 1$.
- c) Nếu dựa vào đẳng thức $q = (A^{-1} \cdot 1) / (1^T \cdot A^{-1} \cdot 1) = V \cdot (A^{-1} \cdot 1)$ thì tìm được $q = (0.4, 0.8, -0.2)$ không là một chiến thuật $q \geq 0$ vì $q[3] = -0.2 < 0$, nên không thể dựa vào chiến thuật q này để tìm chiến thuật tối ưu của người thứ hai.
- d) Trong trường hợp này có thể phát hiện A có điểm yên ngựa là $a[2, 3] = 1$, nên giá trị đúng của giá trị trò chơi là $V = 1$ và thấy ngay chiến thuật tối ưu cho người thứ hai là $(0; 0; 1)$ và chiến thuật tối ưu của người thứ nhất là $(0; 1; 0)$.

Chương trình:

b) Chương trình tính định thức của ma trận

```
#include <iostream>
#include <iomanip>
using namespace std;
const int maxn = 10;
float a[maxn][maxn];
bool b[maxn];
int n;
void read_input(ifstream &fi) {
    fi >> n;
    for (int i=1; i<=n; i++)
        for (int j=1; j<=n; j++) fi >> a[i][j];
}
float tinhDT (int x, int y, int m) {
    int i, j, x1;
    float tong, k;
    if (m==1) return a[y][x];
    else {
        tong = 0;
        x1 = 1;
        while (b[x1]==true) x1++;
        k = 1;
        for (i=1; i<=m; ++i) {
            b[x1] = true;
            j = x1;
            do
                j = j%n+1;
            while (b[j]);
            tong = tong + a[y][x1]*k*tinhDT(j,y+1,m-1);
            k = -k;
            b[x1] = false;
            x1 = j;
        }
    }
    return (tong);
}
int main() {
    ifstream fi ("BAI32b_C9.inp");
    read_input(fi);
    ofstream fo ("BAI32b_C9.out");
    fo << setfill(' ') << setw(6) << setprecision(4) << tinhDT(1,1,n);
    return 0;
}
```

9.33. Trước hết lập ma trận trò chơi rồi thu gọn về dạng:

$$A = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 16 \end{pmatrix}$$

có ma trận nghịch đảo là:

$$A^{-1} = \begin{pmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.125 & 0 \\ 0 & 0 & 0 & 0.0625 \end{pmatrix}$$

Nếu tồn tại các chiến thuật tối ưu $p \geq 0$ và $q \geq 0$ thì giá trị trò chơi là:

$$V = \frac{1}{1^T A^{-1} 1} = \frac{1}{0.5 + 0.25 + 0.125 + 0.0625} = 1.06667.$$

Chiến thuật tối ưu cho người chơi thứ hai là:

$$q = V(A^{-1} \cdot 1) = (0.53333; 0.26667; 0.13333; 0.06667).$$

Chiến thuật tối ưu cho người chơi thứ nhất là:

$$p^T = V \cdot 1^T \cdot A^{-1} = (0.53333; 0.26667; 0.13333; 0.06667; 0; 0; \dots),$$

$$\text{ở đây } 1 = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \text{ và } 1^T = (1 \ 1 \ \dots \ 1).$$

Cũng có thể giả sử tồn tại $q \geq 0$, rồi tìm p bằng cách giải hệ:

$$\begin{cases} 2p_1 = V \\ 4p_2 = V \\ 8p_3 = V \\ 16p_4 = V \\ p_1 + p_2 + p_3 + p_4 = 1 \end{cases}$$

và lập luận do ma trận đối xứng nên $q = p$.

Chương trình: Học sinh tự lập trình.

9.34. Xem phần hướng dẫn trang 158, Tài liệu chuyên Tin học, Quyển 3.

Chiến thuật Baye của người thứ nhất đáp ứng lại p là $p = (p_1, p_2, p_3)$ sao cho $p^T A q$ đạt lớn nhất:

$$p^T A q = (p_1 \ p_2 \ p_3) \cdot \begin{pmatrix} 0 & 7 & 2 & 4 \\ 1 & 4 & 8 & 2 \\ 9 & 3 & -1 & 6 \end{pmatrix} \cdot \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.4 \end{pmatrix} = (p_1 \ p_2 \ p_3) \begin{pmatrix} 3.4 \\ 3.4 \\ 4.6 \end{pmatrix}$$

$= 3.4p_1 + 3.4p_2 + 4.6p_3$. Với $p_1 + p_2 + p_3 = 1$.

Tương tự, sau khi tìm được $p = (0; 0; 1)$ có chiến thuật Baye của người thứ hai đáp ứng lại p là $q = (q_1, q_2, q_3, q_4)$ sao cho $p^T A q$ đạt giá trị bé nhất:

$$p^T A q = (0 \ 0 \ 1) \begin{pmatrix} 0 & 7 & 2 & 4 \\ 1 & 4 & 8 & 2 \\ 9 & 3 & -1 & 6 \end{pmatrix} \cdot \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = 9q_1 + 3q_2 - q_3 + 6q_4$$

với $q_1 + q_2 + q_3 + q_4 = 1$. Kết quả có $q = (0; 0; 1; 0)$.

Chương trình: *Học sinh tự lập trình.*

9.35. Xem trang 102 – 104, Tài liệu chuyên Tin học, Quyển 3.

Chương trình:

```
#include <iostream>
#define max 20
using namespace std;
double a[max][max];
int m, n, p, q;
bool stop;
double x, v;
double c11[max], c12[max];
void read_input() {
    ifstream fi ("bt35.inp");
    fi >> m >> n;
    for (int i=1; i<=m; ++i)
        for (int j=1; j<=n; ++j) fi >> a[i][j];
}
void write_out(ofstream &fo) {
    for (int i=0; i<=m+1; ++i) {
        for (int j=0; j<=n+1; ++j) fo << a[i][j] << ' ';
```

```

        fo<<'\n';
    }
    fo<<'\n';
}
void step1_2(ofstream &fo) {
    int i, j;
    fo<<"Buoc 2"<<'\n';
    x = 1.0E+30;
    for (i=1; i<=m; ++i)
        for (j=1; j<=n; ++j)
            if (a[i][j]<x) x = a[i][j];
    if (x<0)
        for (i=1; i<=m; ++i)
            for (j=1; j<=n; ++j) a[i][j] -= x;
    for (i=1; i<=m; ++i) a[i][n+1] = 1;
    for (j=1; j<=n; ++j) a[m+1][j] = -1;
    for (i=1; i<=m; ++i) a[i][0] = i+100;
    for (j=1; j<=n; ++j) a[0][j] = j;
}
void step3(ofstream &fo) {
    double temp;
    q=1;
    while ( ((a[m+1][q]>0) || (a[m+1][q]==0)) && (q<n+1)) ++q;
    if (q<n+1) {
        temp = 1.0E+30;
        p = 0;
        for (int i=1; i<=m; ++i)
            if (a[i][q]>0) {
                if ((a[i][n+1]/a[i][q])<temp) {
                    temp = a[i][n+1]/a[i][q];
                    p = i;
                }
            }
        if (p>0) fo<<"chot o dong "<<p<< " cot "<<q<<'\n';
        else {
            fo<<"Bai toan suy bien : Cac o cua mot cot = 0 ";
            abort ();
        }
    }
    else stop = true;
}
void step4_5(ofstream &fo) {
    int i, j;
    double temp;
    if (p>0) {
        temp=1/a[p][q];
        for (i=1; i<=m+1; ++i)

```

```

        for (j=1; j<=n+1; ++j)
        if ((i!=p) && (j!=q))
            a[i][j] -= a[i][q]*a[p][j]*temp;

        for (j=1; j<=n+1; ++j)
            if (j!=q) a[p][j] *= temp;

        for (i=1; i<=m+1; ++i)
            if (i!=p) a[i][q] *= (-temp);

        a[p][q] = temp;
    //step 5
        temp = a[0][q];
        a[0][q] = a[p][0];
        a[p][0] = temp;
        write_out(fo);
    }
}

void step7 (ofstream &fo) {
    int k, i, j;
    v = 1/a[m+1][n+1];
    if (x<0) v = v+x;
    fo<<"Gia tri tro choi la "<< v <<'\n';
    fo<<"Chien thuat toi uu cua nguoi I : ";
    for (k=101; k<=100+m; ++k)
        for (i=1; i<=m; ++i)
            if (k==a[i][0]) cl1[k-100] = 0;
    for (k=101; k<=100+m; ++k)
        for (j=1; j<=n; ++j)
            if (k==a[0][j]) cl1[k-100]=a[m+1][j]/a[m+1][n+1];
    for (i=1; i<=m; ++i) fo<<cl1[i]<<' ';
    fo<<'\n';
    fo<<"Chien thuat toi uu cua nguoi II : ";
    for (k=1; k<=n; ++k)
        for (j=1; j<=n; ++j)
            if (a[0][j]==k) cl2[k] = 0;
    for (k=1; k<=n; ++k)
        for (i=1; i<=m; ++i)
            if (a[i][0]==k) cl2[k] = a[i][n+1]/a[m+1][n+1];
    for (int j=1; j<=n; ++j) fo<<cl2[j]<<' ';
}

void step6(ofstream &fo) {
    stop = false;
    while (stop==false) {
        step3(fo);
        write_out(fo);
        if (stop==false) step4_5(fo);
    }
}

```

```

    }
    if (stop==true) step7(fo);
}
int main() {
    ofstream fo ("BAI35_C9.out");
    read_input();
    write_out(fo);
    step1_2(fo);
    step6(fo);
}

```

9.36. Người thứ hai có hai tập thông tin, mỗi tập có hai đỉnh đi ra theo các nhãn D1, D2. Do đó tập các chiến thuật nguyên thuỷ của người thứ hai là:

D1D1: Lần thứ nhất giấu ở phòng I, lần thứ hai giấu ở phòng I.

D1D2: Lần thứ nhất giấu ở phòng I, lần thứ hai giấu ở phòng II.

D2D1: Lần thứ nhất giấu ở phòng II, lần thứ hai giấu ở phòng I.

D2D2: Lần thứ nhất giấu ở phòng II, lần thứ hai giấu ở phòng II.

Người thứ nhất có hai tập thông tin, mỗi tập có hai đỉnh đi ra theo các nhãn là T1, T2. Do đó các chiến thuật nguyên thuỷ của người thứ nhất là:

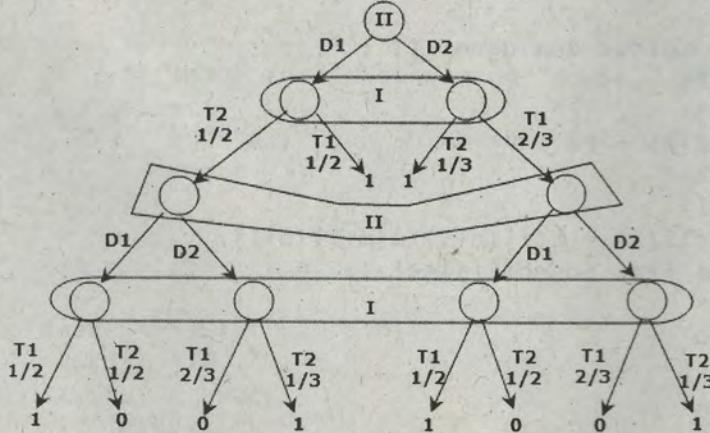
T1T1: Lần thứ nhất tìm ở phòng I, lần thứ hai tìm ở phòng I.

T1T2: Lần thứ nhất tìm ở phòng I, lần thứ hai tìm ở phòng II.

T2T1: Lần thứ nhất tìm ở phòng II, lần thứ hai tìm ở phòng I.

T2T2: Lần thứ nhất tìm ở phòng II, lần thứ hai tìm ở phòng II.

Cây Kuhn của trò chơi là :



Từ đó xây dựng được hàm lượng giá:

	D1D1	D1D2	D2D1	D2D2
T1T1	1.(1/2)=1/2	1.(1/2)=1/2	1.(2/3)(1/2)=1/3	0
T1T2	1.(1/2)=1/2	1.(1/2)=1/2	0	1.(2/3)(1/3)=2/9
T2T1	1.(1/2)(1/2)=1/4	0	1.(1/3)=1/3	1.(1/3)=1/3
T2T2	0	1.(1/2)(1/3)=1/6	1.(1/3)=1/3	1.(1/3)=1/3

c nhẫn
Ma trận trò chơi là:

$$\begin{pmatrix} 1/2 & 1/2 & 1/3 & 0 \\ 1/2 & 1/2 & 0 & 2/9 \\ 1/4 & 0 & 1/3 & 1/3 \\ 0 & 1/6 & 1/3 & 1/3 \end{pmatrix}$$

Giải bằng phương pháp đơn hình có kết quả: giá trị trò chơi là 0.2556.

Chiến thuật tối ưu cho người thứ nhất: (0.1555; 0.2333; 0.2445; 0.3667).

Chiến thuật tối ưu cho người thứ hai: (0.1333; 0.2000; 0.2667; 0.4000).

+ Tuy nhiên, tệp input thường chỉ ghi giá trị gần đúng của ma trận trò chơi, ví dụ như:

$$\begin{pmatrix} 0.5 & 0.5 & 0.3333 & 0 \\ 0.5 & 0.5 & 0 & 0.2222 \\ 0.25 & 0 & 0.3333 & 0.3333 \\ 0 & 1.6666 & 0.3333 & 0.3333 \end{pmatrix}$$

Giá trị trò chơi là 0.238078

Chiến thuật tối ưu của người I: (0.190463 0.285694 0 0.523844)

Chiến thuật tối ưu của người II: (0.285694 0 0.285722 0.428584)

+ Nếu tệp input cho giá trị gần đúng với độ chính xác khá cao như sau:

$$\begin{pmatrix} 0.5000000000 & 0.5000000000 & 0.3333333333 & 0 \\ 0.5000000000 & 0.5000000000 & 0 & 0.2222222222 \\ 0.2500000000 & 0 & 0.3333333333 & 0.3333666666 \\ 0 & 1.6666666666 & 0.3333333333 & 0.3333333333 \end{pmatrix}$$

thì kết quả output là :

Giá trị trò chơi là 0.255556

Chiến thuật tối ưu của người I: 0.155556 0.233333 0.244444 0.366667

Chiến thuật tối ưu của người II: 0.133333 0.2 0.266667 0.4

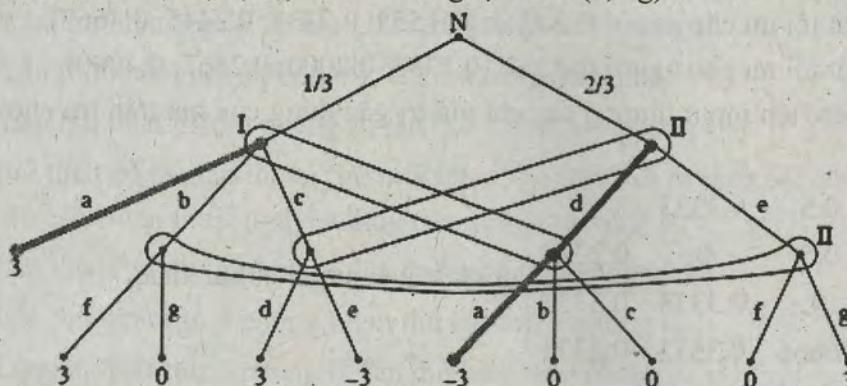
Chương trình: Học sinh tự lập trình.

9.37. Xem trang 162, Tài liệu chuyên Tin học, Quyển 3.

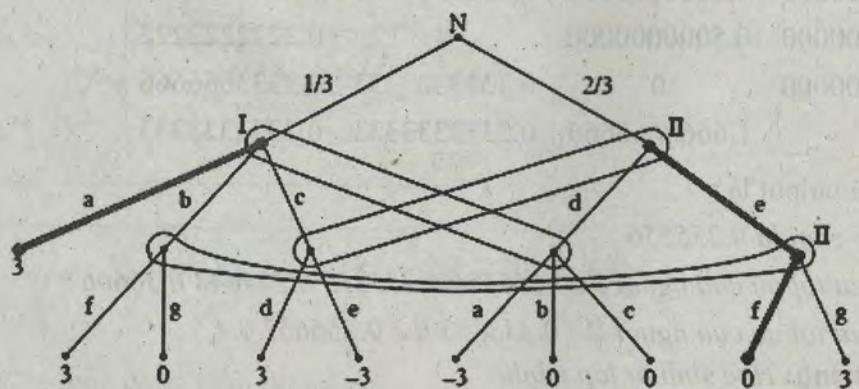
Người thứ nhất có một tập thông tin gồm hai đỉnh, nhẫn đi ra từ hai đỉnh này là a, b và c, nên tập chiến thuật nguyên thủy của thứ nhất là {a, b, c}.

Người thứ hai có hai tập thông tin. Tập thứ nhất có hai đỉnh, nhẫn đi ra từ hai đỉnh này là d và e. Tập thứ hai có hai đỉnh, nhẫn đi ra từ hai đỉnh này là f và g. Do đó tập chiến thuật nguyên thủy của người thứ hai là {df, dg, ef, eg}.

Xét ô (a, df): Người thứ nhất dùng chiến thuật a, đạt ngay ba điểm với xác suất $1/3$ không cần biết người thứ hai dùng chiến thuật gì. Đồng thời nếu người thứ hai dùng d (người thứ nhất dùng a) thì đạt -3 điểm với xác xuất $2/3$ (không cần chờ người thứ hai dùng f). Vậy giá trị trung bình trả về cho người thứ nhất ứng với ô (a, df) là $(1/3) \times 3 + (2/3) \times (-3) = -1$. Tương tự với ô (a, dg).

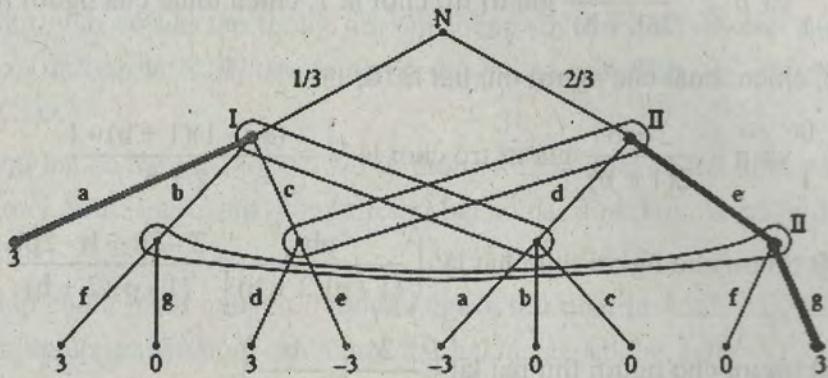


Xét ô (a, ef): Người thứ nhất dùng a, đạt ngay 3 điểm với xác xuất $1/3$, không cần biết người thứ hai dùng chiến thuật gì. Đồng thời nếu người thứ hai dùng e rồi f thì đạt ngay 0 điểm với xác xuất $2/3$ (không cần quan tâm chiến thuật của người thứ nhất). Vậy giá trị trung bình trả về cho người thứ nhất ứng với ô (a, ef) là $(1/3) \times 3 + (2/3) \times (0) = 1$.

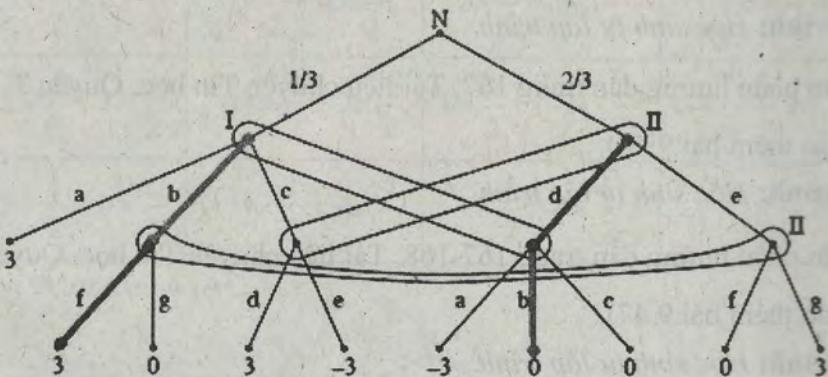


Tương tự với ô (a, eg):

này là a,
hai đỉnh
t. Do đó
xác suất
thứ hai
cần chờ
ng với ô



Xét ô (b, df): Người người thứ nhất dùng b, người người thứ hai dùng d, có ngay 0 điểm với xác xuất $2/3$ (không cần biết người thứ hai dùng tiếp chiến thuật gì). Đồng thời, nếu người người thứ hai dùng tiếp là f (khi người người thứ nhất dùng b) thì đạt 3 điểm với xác xuất $1/3$. Vậy giá trị trung bình trả về cho người thứ nhất là $(1/3) \times 3 + (2/3) \times (0) = 1$.



Các ô còn lại học sinh tự tìm tiếp.

(Tham khảo thêm bài 9.44)

Chương trình: *Học sinh tự lập trình.*

9.38. Xem trang 162-164, Tài liệu chuyên Tin học, Quyển 3.

Khi $p = 1$, giá trị trò chơi là p , chiến thuật của người thứ nhất: $(x; 1-x; 0; 0)$ với $0 \leq x \leq 1$, chiến thuật người thứ hai là $(0; 1)$.

Khi $p = 0$, giá trị trò chơi là -1 , chiến thuật của người thứ nhất: $(0; 1; 0; 0)$, chiến thuật của người thứ hai là $(1; 0)$.

Khi $\begin{cases} p \neq 0 \\ p \neq 1 \end{cases}$ và $p \geq \frac{2+b}{2(1+b)}$ giá trị trò chơi là 1, chiến thuật của người thứ nhất:

(1; 0; 0; 0), chiến thuật của người thứ hai là (0; 1).

Khi $\begin{cases} p \neq 0 \\ p \neq 1 \end{cases}$ và $p < \frac{2+b}{2(1+b)}$ giá trị trò chơi là $V = \frac{(4p-1)(1+b)-1}{2+b}$

Chiến thuật tối ưu cho người thứ nhất là: $\left(\frac{pb}{(1-p)(2+b)}; \frac{2-2p+b-2pb}{(1-p)(2+b)}; 0; 0 \right)$.

Chiến thuật tối ưu cho người thứ hai là: $\left(\frac{2}{2+b}; \frac{b}{2+b} \right)$.

Dễ dàng lập trình để xuất ra những kết quả nêu trên (phụ thuộc tham số p).

Chương trình: *Học sinh tự lập trình.*

9.39. Xem phần hướng dẫn trang 165-166, Tài liệu chuyên Tin học, Quyển 3.

(Tham khảo thêm bài 9.45).

Chương trình: *Học sinh tự lập trình.*

9.40. Xem phần hướng dẫn trang 167, Tài liệu chuyên Tin học, Quyển 3.

(Tham khảo thêm bài 9.46).

Chương trình: *Học sinh tự lập trình.*

9.41. Xem phần hướng dẫn trang 167-168, Tài liệu chuyên Tin học, Quyển 3.

(Tham khảo thêm bài 9.47).

Chương trình: *Học sinh tự lập trình.*

9.42. Xem phần hướng dẫn trang 168, Tài liệu chuyên Tin học, Quyển 3.

(Tham khảo thêm bài 9.48).

Chương trình: *Học sinh tự lập trình.*

9.43. Xem phần hướng dẫn trang 168-170, Tài liệu chuyên Tin học, Quyển 3.

(Tham khảo thêm bài 9.49).

Chương trình: *Học sinh tự lập trình.*

9.44. Hướng dẫn lập bảng giá trị của hàm trả về cho người thứ nhất:

Người thứ nhất có hai tập thông tin, tập thông tin thứ nhất có một đỉnh với các nhãn đi ra từ đỉnh là A, B, tập thông tin thứ hai có một đỉnh với các nhãn đi ra từ đỉnh là C, D.

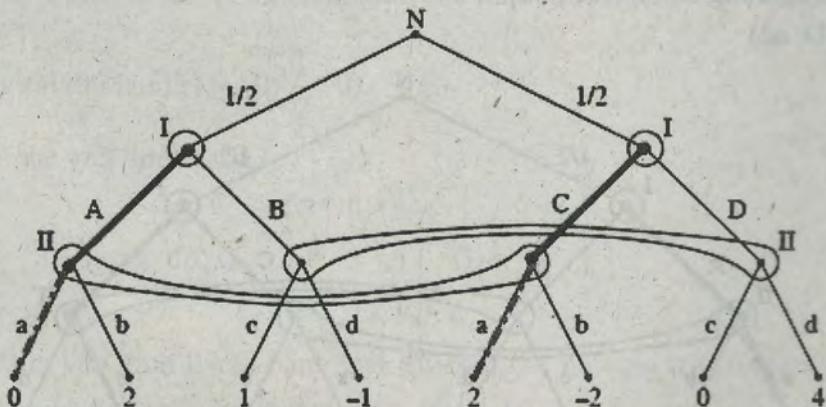
Người thứ hai có hai tập thông tin. Tập thông tin thứ nhất có hai đỉnh với các nhãn đi ra từ mỗi đỉnh là a, b, tập thông tin thứ hai có hai đỉnh với các nhãn đi ra từ mỗi đỉnh là c, d.

Do đó, tập chiến thuật nguyên thuỷ của người thứ nhất là {AC, AD, BC, BD} và tập chiến thuật nguyên thuỷ của người thứ hai là {ac, ad, bc, bd}.

Dựa vào cây trò chơi, lập được bảng giá trị trả về cho người thứ nhất như sau:

	ac	ad	bc	bd
AC	$0 \times \frac{1}{2} + 2 \times \frac{1}{2}$	$0 \times \frac{1}{2} + 2 \times \frac{1}{2}$	$2 \times \frac{1}{2} + (-2) \times \frac{1}{2}$	$2 \times \frac{1}{2} + (-2) \times \frac{1}{2}$
AD	$0 \times \frac{1}{2} + 0 \times \frac{1}{2}$	$0 \times \frac{1}{2} + 4 \times \frac{1}{2}$	$2 \times \frac{1}{2} + 0 \times \frac{1}{2}$	$2 \times \frac{1}{2} + 4 \times \frac{1}{2}$
BC	$1 \times \frac{1}{2} + 2 \times \frac{1}{2}$	$(-1) \times \frac{1}{2} + 2 \times \frac{1}{2}$	$1 \times \frac{1}{2} + (-2) \times \frac{1}{2}$	$(-1) \times \frac{1}{2} + (-2) \times \frac{1}{2}$
BD	$1 \times \frac{1}{2} + 0 \times \frac{1}{2}$	$(-1) \times \frac{1}{2} + 4 \times \frac{1}{2}$	$1 \times \frac{1}{2} + 0 \times \frac{1}{2}$	$(-1) \times \frac{1}{2} + 4 \times \frac{1}{2}$

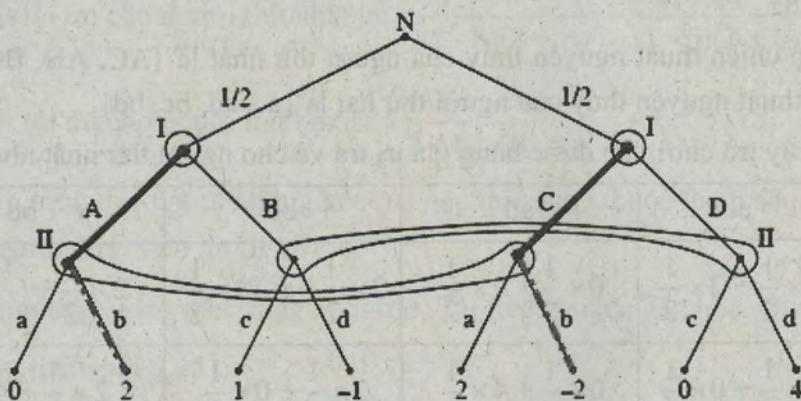
Giải thích thêm: Xét ô (AC, ac):



Người thứ nhất dùng chiến thuật AC, người hai dùng chiến thuật ac. Khi người thứ hai mới thực hiện a thì đã tới lá (không phải thực hiện c nữa) nên kết quả trả về cho người thứ nhất là 0 (ứng với $A - a$) và 2 (ứng với $C - a$) suy ra tổng kết quả trả về cho thứ nhất ở ô (AC, ac) là $0 \times \frac{1}{2} + 2 \times \frac{1}{2}$.

Tương tự áp dụng cho ô (AC, ad).

Xét ô (AC, bc):

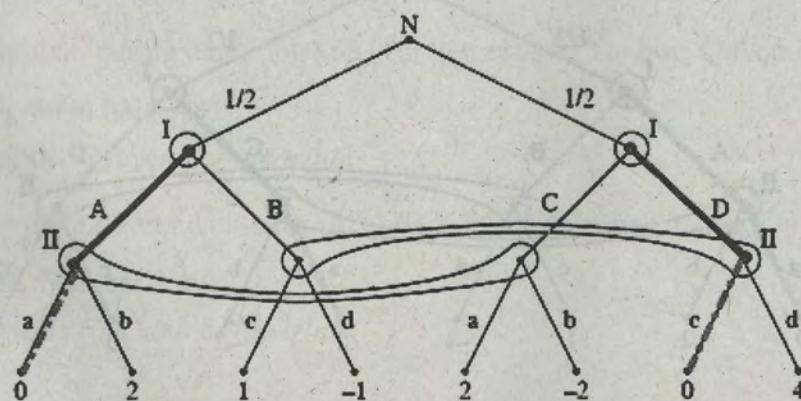


Khi người thứ hai mới thực hiện b thì đã tới lá (không phải thực hiện c nữa) nên kết quả trả về cho người thứ nhất là $\frac{1}{2} \times 2$ (ứng với $A - b$) và $\frac{1}{2} \times (-2)$ (ứng với $C - b$)

suy ra tổng kết quả trả về cho người thứ nhất ở ô (AC, bc) là $2 \times \frac{1}{2} + (-2) \times \frac{1}{2}$.

Tương tự áp dụng cho ô (AC, bd).

Xét ô (AD, ac):



người thứ nhất chọn A thì người thứ hai chọn a, khi người thứ nhất chọn D thi người thứ hai chọn c. Do đó kết quả trả về cho người thứ nhất trong trường hợp này là $\frac{1}{2} \times 0$ (ứng với A - a) và $\frac{1}{2} \times 0$ (ứng với D - c), suy ra kết quả trả về cho thứ nhất ở ô (AD, ac) là $0 \times \frac{1}{2} + 0 \times \frac{1}{2}$.

Các ô còn lại được tìm tương tự. Ma trận trò chơi là:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 3 \\ 1.5 & 0.5 & -0.5 & -1.5 \\ 0.5 & 1.5 & 0.5 & 1.5 \end{pmatrix}$$

Giải trò chơi: Cột 2 bị chi phối bởi cột 3 nên loại cột 2 còn ma trận:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 3 \\ 1.5 & -0.5 & -1.5 \\ 0.5 & 0.5 & 1.5 \end{pmatrix}$$

Tổ hợp hàng 1 và 2 chi phối hàng 4 nên loại hàng 4 còn ma trận:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 3 \\ 1.5 & -0.5 & -1.5 \end{pmatrix}$$

Ma trận không có điểm yên ngựa. Chạy thuật toán đơn hình, tìm được giá trị trò chơi là 0.5. Chiến thuật tối ưu cho người thứ nhất là $(0; \frac{2}{3}; \frac{1}{3}; 0)$, chiến thuật tối

ưu cho người thứ hai là $(\frac{1}{2}; 0; \frac{1}{2}; 0)$.

9.45. Có thể viết dưới dạng:

$$G = \begin{pmatrix} G_1 & 2 \\ 1 & G_2 \end{pmatrix} \text{trong đó: } G_1 = \begin{pmatrix} 3 & 0 & 5 \\ 1 & 3 & 5 \\ 4 & 4 & 1 \end{pmatrix}, G_2 = \begin{pmatrix} 6 & 3 \\ 4 & 7 \end{pmatrix}.$$

Xét G_1 : Dựa vào định lí cân bằng giải trò chơi G_1 : Gọi giá trị trò chơi là v_1 . Gọi chiến thuật của người thứ nhất là $p_1 = (x_1; x_2; x_3)$.

Giải hệ $\begin{cases} 3x_1 + x_2 + 4x_3 = v_1 \\ 3x_2 + 4x_3 = v_1 \\ 5x_1 + 5x_2 + x_3 = v_1 \\ x_1 + x_2 + x_3 = 1 \end{cases}$ tìm được: $v_1 = \frac{91}{31}$ và chiến thuật tối ưu của người

thứ nhất là: $p_1 = \left(\frac{6}{31}; \frac{9}{31}; \frac{16}{31} \right)$.

Tương tự, chiến thuật tối ưu của người thứ hai là $q_1 = (y_1; y_2; y_3)$ tìm được từ hệ:

$$\begin{cases} 3y_1 + 5y_3 = v_1 \\ y_1 + 3y_2 + 5y_3 = v_1 \\ 4y_1 + 4y_2 + y_3 = v_1 \\ y_1 + y_2 + y_3 = 1 \end{cases}$$

suy ra $q_1 = \left(\frac{12}{31}; \frac{8}{31}; \frac{11}{31} \right)$.

Xét G_2 : Dựa vào các công thức đã xây dựng với trò chơi ma trận 2×2 , tìm thấy G_2 có giá trị trò chơi là $v_2 = 5$. Chiến thuật của người thứ nhất là $p_2 = \left(\frac{1}{2}; \frac{1}{2} \right)$ và

của người thứ hai là $q_2 = \left(\frac{2}{3}; \frac{1}{3} \right)$.

Xét G : Vậy G thành G' là $\begin{pmatrix} 91 & 2 \\ 31 & 5 \\ 1 & 5 \end{pmatrix}$. Giá trị trò chơi G' là: $\frac{131}{51}$. Chiến thuật tối

ưu của người thứ nhất là: $p' = \left(\frac{124}{153}; \frac{29}{153} \right)$.

Chiến thuật tối ưu của người thứ hai là $q' = \left(\frac{31}{51}; \frac{20}{51} \right)$.

Kết luận: Trò chơi G có giá trị là $\frac{131}{51}$.

Chiến thuật tối ưu của người thứ nhất là:

$$\left(\frac{124}{153} \times \frac{6}{31}; \frac{124}{153} \times \frac{9}{31}; \frac{124}{153} \times \frac{16}{31}; \frac{29}{153} \times \frac{1}{2}; \frac{29}{153} \times \frac{1}{2} \right) = \left(\frac{24}{153}; \frac{36}{153}; \frac{64}{153}; \frac{29}{306}; \frac{29}{306} \right).$$

Chiến thuật tối ưu của người thứ hai là:

$$\left(\frac{31}{51} \times \frac{12}{31}; \frac{31}{51} \times \frac{8}{31}; \frac{31}{51} \times \frac{11}{31}; \frac{20}{51} \times \frac{2}{3}; \frac{20}{51} \times \frac{1}{3} \right) = \left(\frac{12}{51}; \frac{8}{51}; \frac{11}{51}; \frac{40}{153}; \frac{20}{153} \right).$$

Giải thích thêm:

$$+ \text{Giá trị trò chơi } G = \begin{pmatrix} G_1 & 2 \\ 1 & G_2 \end{pmatrix} \text{ bằng giá trị trò chơi } G' = \begin{pmatrix} 91 & 2 \\ 31 & 5 \\ 1 & 5 \end{pmatrix}.$$

+ Giả sử G_1 có chiến thuật tối ưu cho người thứ nhất là $p^1 = (p_{11}, p_{12}, p_{13})$; G_2 có chiến thuật tối ưu cho người thứ nhất là $p^2 = (p_{21}, p_{22})$; G' có chiến thuật tối ưu cho người thứ nhất là $p' = (p'_1, p'_2)$ thì trong trò chơi G có chiến thuật tối ưu cho người thứ nhất là:

$$p = (p_1 \times p_{11}, p_1 \times p_{12}, p_1 \times p_{13}, p_1 \times p_{21}, p_1 \times p_{22}).$$

Tương tự: Giả sử chiến thuật tối ưu cho người thứ hai trong các trò chơi G_1, G_2, G' tương ứng là: $q^1 = (q_{11}, q_{12}, q_{13}), q^2 = (q_{21}, q_{22}), q' = (q'_1, q'_2)$ thì chiến thuật tối ưu cho người thứ hai trong G là:

$$q = (q_1 \times q_{11}, q_1 \times q_{12}, q_1 \times q_{13}, q_1 \times q_{21}, q_1 \times q_{22}).$$

9.46. a) Quy ước Q là giá trị trả về cho người thứ nhất nếu trò chơi được kéo dài

vô hạn. Gọi giá trị trò chơi G khi được kết thúc là v và kí hiệu $A = \begin{pmatrix} v & 1 & 0 \\ 1 & 0 & v \\ 0 & v & 1 \end{pmatrix}$.

$$v = \text{Val}(A), \text{ khi đó } \det(A) = -v^3 - 1.$$

+ Nếu $v = -1$ thì A không có điểm yên ngựa, dùng thuật toán đơn hình tìm được $v = 0$ (mâu thuẫn với $v = -1$) vậy không xảy ra $v = -1$.

+ Nếu $v \neq -1$ thì $\det(A) \neq 0$, có thể sử dụng định lí cân bằng tìm giá trị của A như sau: (giả sử chiến thuật của người thứ nhất là $p = (p_1; p_2; p_3)$). Theo nguyên lí

$$\text{cân bằng có } \begin{cases} v = v \cdot p_1 + p_2 \\ v = p_1 + v \cdot p_3 \text{ suy ra } v = 1/2; p_1 = p_2 = p_3 = 1/3. \\ v = v \cdot p_2 + p_3 \end{cases}$$

Kết luận: Giá trị trò chơi G luôn là $\frac{1}{2}$, không phụ thuộc giá trị của Q . Chiến thuật $(\frac{1}{3}; \frac{1}{3}; \frac{1}{3})$ là chiến thuật tối ưu cho cả hai người chơi.

b) Gọi giá trị trò chơi G_0 khi được kết thúc là v và kí hiệu $A = \begin{pmatrix} v & 5 \\ 1 & 0 \end{pmatrix}$.

Giá trị trò chơi G_0 phụ thuộc vào Q . Thật vậy:

- Với $Q \geq 1$ thì chiến thuật cho người thứ nhất là chọn mäng dòng đầu. Khi đó:
 - + Nếu $1 \leq Q \leq 5$ thì A có điểm yên ngựa tại $(1; 1)$ nên giá trị trò chơi là Q .
 - + Nếu $Q \geq 5$ thì A có điểm yên ngựa tại $(1, 2)$ nên giá trị trò chơi là 5 .
- Với $Q < 1$, không có điểm yên ngựa, dùng định lí cân bằng giải trò chơi thì giá trị trò chơi là v xác xỉ $\frac{5}{6-v}$, giải ra được v xác xỉ 1 .

Kết luận: Giá trị trò chơi G_0 (là v khi kết thúc được) phụ thuộc vào Q (giá trị trò chơi G_0 khi kéo dài mãi). Nhưng v chỉ thuộc đoạn $[1; 5]$ và lân cận 1.

9.47. Gọi giá trị của các trò chơi G_1, G_2, G_3 khi được kết thúc tương ứng là v_1, v_2 và v_3 . Người chơi thứ nhất có thể bảo đảm $v_1 > 0, v_2 > 0$ và $v_3 > 0$, ví dụ như bằng cách luôn luôn dùng chiến thuật $(0.5; 0.5)$. Hơn nữa có thể thấy $v_2 \leq 1$ và $v_3 \leq 2$, suy ra $v_1 < 1$, nên không có trò chơi nào có điểm yên ngựa. Áp dụng kết quả của trò chơi ma trận 2×2 ta có:

$$v_1 = \frac{v_2 \cdot v_3}{v_2 + v_3}; \quad v_2 = \frac{1}{2 - v_1}; \quad v_3 = \frac{4}{4 - v_1}.$$

$$\text{Suy ra } v_1 \cdot v_2 + v_1 \cdot v_3 = v_2 \cdot v_3 \Rightarrow \frac{v_1}{2 - v_1} + \frac{4v_1}{4 - v_1} = \frac{4}{(2 - v_1) \cdot (4 - v_1)}$$

$$\Leftrightarrow 5v_1^2 - 12v_1 + 4 = 0.$$

Tìm được $v_1 = \frac{2}{5}$ (do $0 < v_1 < 1$). Do đó có bảng kết quả sau đây với mọi giá trị của Q :

Trò chơi	Giá trị trò chơi	Chiến thuật tối ưu của người thứ nhất và người thứ hai (như nhau)
G_1	$\frac{2}{5}$	$(\frac{16}{25}; \frac{9}{25})$
G_2	$\frac{5}{8}$	$(\frac{5}{8}; \frac{3}{8})$
G_3	$\frac{10}{9}$	$(\frac{5}{9}; \frac{4}{9})$

9.48. a) Gọi v là giá trị trò chơi G khi kết thúc.

$$\text{Có } v = \text{Val} \begin{pmatrix} 3 & 1 + \frac{2v}{3} \\ 0 & 2 + \frac{v}{3} \end{pmatrix} = \frac{6+v}{4-\frac{v}{3}}.$$

Dẫn tới phương trình: $v^2 - 9v + 18 = 0$. Vậy $v = 3$ hoặc $v = 6$.

Thử lại :

+ Khi $v = 6$ có $G = \begin{pmatrix} 3 & 5 \\ 0 & 4 \end{pmatrix}$ có điểm yên ngựa $(1; 1)$, vậy giá trị trò chơi khi kết thúc là $v = 3$ (mâu thuẫn). Vậy loại trường hợp này.

+ Khi $v = 3$ có $G = \begin{pmatrix} 3 & 3 \\ 0 & 3 \end{pmatrix}$ có điểm yên ngựa $(1, 1)$ nên giá trị trò chơi khi kết thúc $v = 3$ là phù hợp. Chiến thuật cho cả hai người chơi là $(1; 0)$.

b) Gọi v_1 và v_2 tương ứng là giá trị trò chơi G_1 và G_2 khi kết thúc.

$$G_1 = \begin{pmatrix} 4 & 1-0.25G_2 \\ 0 & 3+0.75G_2 \end{pmatrix}; \quad G_2 = \begin{pmatrix} -5 & 0 \\ -1-0.2G_1 & 0.8G_1 \end{pmatrix}.$$

Do giả thiết các trò chơi không có điểm yên ngựa nên ta áp dụng kết quả của trò chơi ma trận 2×2 có:

$$v_1 = \text{Val} \begin{pmatrix} 4 & 1-0.25v_2 \\ 0 & 3+0.75v_2 \end{pmatrix} = \frac{12+3v_2}{6+v_2} \text{ suy ra } v_2 = \frac{6v_1-12}{3-v_1}.$$

Chiến thuật tối ưu người thứ nhất là $(p, 1-p)$ với $p = \frac{3+0.75v_2}{6+v_2}$. (a1)

Chiến thuật tối ưu của người thứ hai là $(q, 1 - q)$ với $q = \frac{2 + v_2}{6 + v_2}$. (a2)

$$v_2 = \text{Val} \begin{pmatrix} -5 & 0 \\ -1 - 0.2v_1 & 0.8v_1 \end{pmatrix} = \frac{4v_1}{4 - v_1}.$$

Chiến thuật tối ưu của người thứ nhất là $(p; 1 - p)$ với $p = \frac{-1 - v_1}{4 - v_1}$. (b1)

Chiến thuật tối ưu của người thứ hai là $(q; 1 - q)$ với $q = \frac{-0.8v_1}{4 - v_1}$. (b2)

Giải hệ: $\begin{cases} v_1 = \frac{12 + 3v_2}{6 + v_2} \\ v_2 = \frac{4v_1}{4 - v_1} \end{cases}$ có $v_2 = \frac{4v_1}{4 - v_1} = \frac{6v_1 - 12}{3 - v_1}$ được $v_1 = 6 \pm 2\sqrt{3}$,

Nếu $v_1 = 6 + 2\sqrt{3}$ thì $v_2 = -4\sqrt{3}$. Nếu $v_1 = 6 - 2\sqrt{3}$ thì $v_2 = 4\sqrt{3}$.

Thay giá trị của v_1 và v_2 vào các công thức tương ứng (a1), (a2), (b1), (b2) sẽ có chiến thuật tối ưu của hai người chơi trong G_1 và G_2 .

9.49. Trong bài tập này, nếu áp dụng định lí Shapley tìm chiến thuật tối ưu bằng giải hệ các phương trình:

$$v_1 = \text{Val} \begin{pmatrix} 4 + 0.3v_1 & 0 + 0.4v_2 \\ 1 + 0.4v_2 & 3 + 0.5v_1 \end{pmatrix}$$

$$\Leftrightarrow v_1 = [(4 + 0.3v_1)(3 + 0.5v_1) - (1 + 0.4v_2)0.4v_2]/(6 + 0.8v_1 - 0.8v_2) \quad (1)$$

$$v_2 = \text{Val} \begin{pmatrix} 0 + 0.5v_1 & -5 \\ -4 & 1 + 0.5v_2 \end{pmatrix} \Leftrightarrow v_2 = [0.5v_1(1 + 0.5v_2) - 20]/(10 + 0.5v_1 + 0.5v_2) \quad (2)$$

sẽ gặp nhiều trở ngại vì đây là hệ hai phương trình bậc hai hai ẩn dạng không đặc biệt nên khó giải. Vì vậy trong trường hợp này chúng ta dùng máy tính lập trình theo phương pháp lặp Shapley để tìm gần đúng chiến thuật tối ưu (xem phần hướng dẫn trong Tài liệu chuyên Tin học, Quyển 3).

Kết quả chạy chương trình với $k = 20$ là:

Giai đoạn 1 :

G1 có $v1 = 2$ $p = (0.333333, 0.666667)$ $q = (0.5, 0.5)$

G2 có $v2 = -2$ $p = (0.5, 0.5)$ $q = (0.6, 0.4)$

Giai doan 2 :
 G1 co v1= : 2.01739 p=(0.413043 , 0.586957) q=(0.521739 , 0.478261)
 G2 co v2= : -2 p=(0.4 , 0.6) q=(0.5 , 0.5)
 Giai doan 3 :
 G1 co v1= : 2.02096 p=(0.413364 , 0.586636) q=(0.521895 , 0.478105)
 G2 co v2= : -1.99826 p=(0.399652 , 0.600348) q=(0.499566 , 0.500434)
 Giai doan 4 :
 G1 co v1= : 2.02204 p=(0.413416 , 0.586584) q=(0.52193 , 0.47807)
 G2 co v2= : -1.99765 p=(0.399633 , 0.600367) q=(0.49952 , 0.50048)
 Giai doan 5 :
 G1 co v1= : 2.02238 p=(0.413431 , 0.586569) q=(0.521941 , 0.478059)
 G2 co v2= : -1.99744 p=(0.39963 , 0.60037) q=(0.499509 , 0.500491)
 Giai doan 6 :
 G1 co v1= : 2.02249 p=(0.413436 , 0.586564) q=(0.521945 , 0.478055)
 G2 co v2= : -1.99738 p=(0.399629 , 0.600371) q=(0.499505 , 0.500495)
 Giai doan 7 :
 G1 co v1= : 2.02253 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99736 p=(0.399629 , 0.600371) q=(0.499504 , 0.500496)
 Giai doan 8 :
 G1 co v1= : 2.02254 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 9 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 10 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 11 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 12 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 13 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 14 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 15 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
 Giai doan 16 :
 G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
 G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)

```

Giai doan 17 :
G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
Giai doan 18 :
G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
Giai doan 19 :
G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)
Giai doan 20 :
G1 co v1= : 2.02255 p=(0.413438 , 0.586562) q=(0.521946 , 0.478054)
G2 co v2= : -1.99735 p=(0.399629 , 0.600371) q=(0.499503 , 0.500497)

```

Chương trình:

```

#include <fstream>
const int max=3;
float v1, v2, p, q ;
float g1[max][max], g2[max][max];
int k, i;
void tinhv(ofstream &fo) {
    float mauso;
    v1 = g1[1][1]*g1[2][2]-g1[2][1]*g1[1][2];
    mauso = g1[1][1]+g1[2][2]-g1[2][1]-g1[1][2];
    v1 /= mauso;
    p = g1[2][2]-g1[2][1];
    p /= mauso;
    q = g1[2][2]-g1[1][2];
    q /= mauso;
    fo<<"G1 co v1= : "<< v1;
    fo<<" p="(<<p<<" , "<<(1-p)<<") ";
    fo<<" q="(<<q<<" , "<<(1-q)<<") \n";
    v2 = g2[1][1]*g2[2][2]-g2[2][1]*g2[1][2];
    mauso = g2[1][1]+g2[2][2]-g2[2][1]-g2[1][2];
    v2 /= mauso; p = g2[2][2]-g2[2][1];
    p /= mauso; q = g2[2][2]-g2[1][2];
    q /= mauso;
    fo<<"G2 co v2= : "<< v2;
    fo<<" p="(<<p<<" , "<<(1-p)<<") ";
    fo<<" q="(<<q<<" , "<<(1-q)<<") \n";
}
int main() {
    ofstream fo ("bai43_C9.out");
    ifstream fi ("bai43_C9.inp");
    fi>>k;
    v1 = 0;
    v2 = 0;
    for (int i=1; i<=k; ++i) {
        g1[1][1] = 4+0.3*v1; g1[1][2] = 0.4*v2;
        g1[2][1] = 1+0.4*v2; g1[2][2] = 3+0.5*v1;
        g2[1][1] = 0.5*v1; g2[1][2] = -5;
    }
}

```

```

g2[2][1] = -4;           g2[2][2] = 1+0.5*v2;
fo<<"Giai doan "<<i<< " : \n";
tinhv(fo);
fo<<"\n";
}
}

```

9.50. Trò chơi này tương tự trò chơi lấy 31 quân cờ khỏi bàn nhưng có điểm khác biệt đó là ràng buộc là mỗi số nguyên (từ 1 đến 6) không được chọn quá bốn lần.

a) Theo thuật toán của trò chơi lấy bớt quân cờ, diễn biến trò chơi như sau: đầu tiên bạn chọn 3 (để tổng còn là $31 - 3 = 28$ chia hết cho $6 + 1$), nhưng sau đó đối thủ sẽ chọn 4, tổng còn 24. Lần thứ hai bạn lại chọn 3 (để tổng còn lại là 21 chia hết cho 7), nhưng đối thủ lại chọn 4. Hai lần sau cũng diễn ra tương tự và tổng đã là $4 \times 7 = 28$. Đến lần thứ năm đến lượt bạn đi (đã hết bốn quân 3 và bốn quân 4) chỉ có thể chọn 1, 2, 5, hoặc 6. Nếu bạn chọn 5 hoặc 6 thì bạn tạo ra tổng lớn hơn 31 nên bạn thua. Nếu bạn chọn 1, đối thủ chọn 2 tạo ra tổng là 31, đến bạn sẽ tạo ra tổng lớn hơn 31 (bạn thua). Nếu bạn chọn 2, đối thủ chọn 1 tạo ra tổng là 31, đến bạn sẽ tạo ra tổng lớn hơn 31 (bạn thua).

Vậy trong bài toán này, do có ràng buộc mỗi số nguyên không được chọn quá bốn lần nên việc áp dụng chiến thuật thắng của trò chơi “lấy bớt quân cờ” không đạt mong muốn (nghĩa là sẽ thua).

b) Nhận thấy nếu người nào sau mỗi lần đi tạo được tổng của hai người thuộc dãy số 10, 17, 24, 31 thì cuối cùng người đi tiếp theo sẽ thua (vì tạo ra tổng của hai người vượt trên 31). Gọi người đi đầu là A, đối thủ là B. Một chiến thuật giành thắng cho người A là: Đầu tiên A chọn 5. Có ba trường hợp:

- *B chọn 6*, thì A chọn 6 để sau khi đi A sẽ tới vị trí 17, tiếp theo B chọn x thì A chọn $7 - x$ để luôn luôn chiếm được 24 rồi 31.
- *B chọn 1 hoặc 2 hoặc 3 hoặc 4*, thì A chọn tương ứng là 4 hoặc 3 hoặc 2 hoặc 1 để A chiếm được số 10, tiếp theo B chọn x thì A chọn $7 - x$ để chiếm được 17 rồi 24 và cuối cùng là 31.
- *B chọn 5*, thì $A = 2, B = 5, A = 2, B = 5$ (*hết bốn quân 5*), $A = 2$, lúc này tổng hai người đã là 26, B đi tiếp không thể chiếm được 31 vì hết quân 5 nên B phải đi 1 hoặc 2 hoặc 3 hoặc 4 (nếu B đi quân 6 thì B thua ngay), đến lượt A đi tiếp và dễ dàng chiếm được 31 bằng cách chọn tương ứng là 4 hoặc 3 hoặc 2 hoặc 1. Trong quá trình này nếu B không chọn đúng như trên (để B chiếm tổng là 10, 17, 24 là các vị trí chiến thắng) thì sau khi B

đi lệch khỏi các vị trí chiến thắng này A phải kịp thời chọn ngay số quân phù hợp để chiến thắng.

Chương trình:

```
#include <iostream>
using namespace std;
int sum=0, player, cur, turn, lay[7];
void chienthuat(int cur,int &may) {
    if (turn==2) { //Lượt thứ hai của máy
        if (cur==6) may=6;
        else
            if (cur==5) may=2;
            else      may = 5-cur;
    }
    else { //Những lượt sau : thứ ba, thứ tư, ...
        if (sum%7!=3) {
            for (int m=1; m<7; m++)
                if ((sum+m)%7==3) {
                    may=m; break;
                }
        }
        else may = 7 - cur;
    }
}
int main() {
    turn=1;
    cout <<"COMPUTER's TURN\n";
    cout << "It moved: " << 5 << " card(s)\n"; //Lượt đầu của máy
    sum += 5; //Tổng số của hai người
    lay[5]++; //Đếm số lần đã lấy số 5
    cout << "Sum " << sum << '\n';
    player=1; //Báo hiệu đến lượt người chơi
    int may; //Số quân máy lấy
    while (true) { //Vòng lặp thể hiện trò chơi sau khi máy lấy lần thứ nhất (5 quân)
        if (!player) { //Nếu đến lượt máy chơi
            cout <<"COMPUTER's TURN\n";
            chienthuat(cur, may); //Chiến thuật giành thắng của máy
            lay[may]++; //Đếm số lần lấy quân có số hiệu là may
            sum += may; //Thêm vào tổng hai người
            cout << "It moved: " << may << " card(s)\n";
            if (sum>31){ //Thông báo Người chơi thắng vì máy đã tạo ra tổng >31
                cout << "You WON!!\n";
                break;
            }
        }
        else { // Đến lượt người đi
            turn++; //Số lượt đã chơi
            cout << "YOUR TURN\n";
            cout << "Please enter number of card(s) 1..6: ";
            bool OK;
```

```

    do {
        cin >> cur; //Nhập từ bàn phím số quân người chơi lấy
        OK = (lay[cur]<4) && (cur>0) && (cur<7); //Hợp lệ
        if (OK) break; //Hợp lệ thì thoát vòng lặp
        else cout << "Number invalid. Need to re-enter...";
    } while (!OK);
    cout << "You moved " << cur << " card(s)\n";
    sum += cur; //Thêm vào tổng hai người
    lay[cur]++; //Đếm số lần lấy quân có số hiệu là cur
    if (sum>31) { //Máy thắng vì người chơi vừa tạo ra tổng>31
        cout << "Computer WON!!\n";
        break;
    }
}
player ^= 1; //Đổi người
cout << "Sum = " << sum << '\n'; //Thông báo tổng của hai người
cout << '\n';
}
system("pause");
return 0;
}

```

9.51. Gọi người đi đầu là A, đối thủ là B. Chúng ta sử dụng phương pháp quy nạp để chứng minh A chiến thắng:

- Nếu $N = 1$, rõ ràng A chiếm $(1; 2)$ buộc B nhận $(1; 1)$ nên B thua.
- Nếu $N = 2$, A vẫn thắng: A chiếm ô $(2; 2)$, B phải chiếm một trong hai ô $(2; 1)$ hoặc $(1; 2)$, tiếp theo A chiếm nốt ô còn lại, buộc B nhận ô $(1; 1)$ nên B thua (A thắng).
- Giả sử A dành chiến thắng trong trò chơi Thin Chomp $2 \times (N - 1)$. Xét trò chơi Thin Chomp $2 \times N$: A bắt đầu chiếm ngay ô góc trên-phải là ô $(N; 2)$.
 - Nếu B chiếm một ô của hàng dưới thì B đã chuyển cho A trò chơi Thin Chomp với kích thước ngắn hơn nên A thắng (theo giả thiết quy nạp). Do đó có thể giả sử B không chiếm ô của hàng dưới.
 - Nếu B chiếm $(1; 2)$ (ở trên ô $(1; 1)$) thì sau đó chỉ còn hàng dưới, A chiếm $(2; 1)$ chỉ còn lại ô $(1; 1)$ buộc B phải nhận nên B thua.
 - Nếu B không chiếm $(1; 2)$ thì sau khi chiếm một ô khác của hàng trên, hàng trên sẽ ít hơn hàng dưới ít nhất là hai ô bên phải. Sau bước chuyển của B, đến lượt A sẽ chiếm một ô hàng dưới sao cho số ô còn lại của hàng dưới bằng số ô của hàng trên (nói cách khác lại dẫn tới tình trạng cũ: thanh sô-cô-la mất ô ở góc trên-phải, nhưng số cột nhỏ dần đi). Song số ô của hàng trên là hữu hạn nên đến một lúc nào đó sau lượt A đi, hàng trên chỉ còn ô $(1; 2)$ và hàng dưới chỉ còn ô $(2; 1)$, B chiếm

một trong hai ô này rồi A chiếm ô còn lại khác ô (1; 1), buộc B phải nhận ô (1; 1) nên B thua.

9.52. a) Khi $N = 4$, nếu A đặt S vào ô thứ nhất thì B đặt S vào ô thứ tư. Bằng lối bây giờ đối xứng và bản thân từ SOS cũng đối xứng nên tình huống A đặt tiếp vào ô thứ hai cũng giống như đặt tiếp vào ô thứ ba, vậy không mất tính tổng quát ta chỉ xét trường hợp A đặt vào ô thứ hai: Nếu A đặt O vào ô thứ hai thì B đặt tiếp S vào ô thứ ba tạo ra từ SOS trong ba ô vuông liên tiếp từ ô thứ nhất đến ô thứ ba vậy B thắng. Ngược lại nếu A đặt S vào ô thứ hai thì B đặt O vào ô thứ ba tạo ra từ SOS trong ba ô vuông liên tiếp từ ô thứ hai đến ô thứ tư, vậy B vẫn thắng.

b) Khi $N = 7$, để thuận lợi trình bày, ta gọi bốn ô vuông liên tiếp: S, rỗng, rỗng, S là một thế trận thất bại cho ai đi vào trước. Khi $N = 7$, A đặt S vào ô chính giữa (ô 4). Không mất tính tổng quát B chọn một ô bên phải (là ô 5 hoặc 6 hoặc 7):

Nếu B chọn ô 7:

+ Giả sử B đặt S vào ô 7, khi đó A cũng đặt S vào ô 1 tạo ra hai thế trận thất bại (bên trái và bên phải) chờ B đi vào, vậy A thắng.

+ Giả sử B đặt O vào ô 7, A vẫn đặt S vào ô 1, tạo ra một thế trận thất bại ở bên trái chờ đợi B đi vào. Nhưng B không đi vào thế trận này, B sẽ đặt tiếp quân vào ô 5 hoặc ô 6, để khỏi thua ngay, nếu đặt ô 5 thì B đặt S, nếu đặt ô 6 thì B đặt O, A đặt quân tùy ý vào ô còn lại bên phải cốt dồn B trong bước tiếp theo phải vào thế thất bại bày sẵn ở bên trái. Do hết ô bên phải nên buộc B phải vào thế trận thất bại bên trái.

Nếu B chọn ô 6: B không thể đặt S vào ô 6 (vì nếu thế, A sẽ đặt O vào ô 5, giành thắng) vậy B đặt O vào ô 6, tiếp theo A đặt S vào ô 1 tạo thế trận thất bại bên trái chờ B phải đi vào, B không vào bên trái thì B đặt O vào ô 5 hoặc ô 7, A đi nốt ô còn lại bên phải buộc B phải vào thế trận thất bại chờ B ở bên trái.

Nếu B chọn ô 5: B phải đặt S vào ô 5, tiếp theo A đặt S vào ô 1, B buộc phải chọn ô 6 hoặc ô 7: nếu chọn ô 7 thì đặt O, nếu chọn ô 6 thì đặt S. Tiếp theo A đặt nốt quân tùy ý vào ô còn lại của bên phải, buộc B vào thế trận thất bại bên trái.

Vậy trong mọi trường hợp, B thua và A thắng.

b) Gọi phần bảng đã điền là ổn định nếu không có từ “SOS” và không có một phép chuyển đơn độc nào tạo ra được “SOS”; ngược lại gọi là không ổn định (nghĩa là bảng một phép di chuyển có thể tạo ra “SOS”. Vì vậy khi giành được bảng không ổn định là giành được chiến thắng. Ví dụ một số bảng sau là ổn định:

S		S
---	--	---

O		S
---	--	---

O		O
---	--	---

S		O
---	--	---

Một số bảng sau không ổn định:

(1; 1)

S	S	S	S	O	S	S	O	S	S
---	---	---	---	---	---	---	---	---	---

S	S	O	S	S	S	S	O	O	O
---	---	---	---	---	---	---	---	---	---

Với bảng ổn định, ta gọi một ô rỗng là ô *xấu* nếu đặt hoặc S hoặc O vào nó thì sinh ra bảng không ổn định. Ví dụ: Bảng sau có hai ô *xấu* là ô 2 và ô 3:

S		S
---	--	---

Bổ đề: Một ô rỗng là xấu khi và chỉ khi nó trong khối bốn ô vuông liên tiếp có dạng: "S, ô rỗng, ô rỗng, S".

S		S
---	--	---

Chứng minh. Nếu ô vuông thứ hai là ô *xấu* thì sau khi đặt O vào nó sẽ làm khối thành không ổn định. Do đó cần phải có ô S ở ô thứ nhất và một ô rỗng ở ô thứ ba. Tương tự đặt S vào nó cũng tạo ra khối không ổn định nên cần phải có ô thứ ba là rỗng và ô thứ tư là S. Tương tự, nếu ô *xấu* là ô thứ ba thì cũng suy ra ô thứ nhất và ô thứ tư phải là S.

Từ bổ đề suy ra luôn luôn có một số chẵn ô *xấu*. Do đó người chơi thứ hai (B) có thuật giành thắng như sau:

- Nếu bảng là không ổn định thì B chơi phép di chuyển chiến thắng (tạo ra SOS), trái lại thì tiếp tục như sau:
- Bước di chuyển thứ nhất: B đặt S cách xa một đầu bảng và cách xa ô của A vừa di chuyển (lần đầu) ít nhất bốn ô vuông (Bảng dài nên đủ để thực hiện yêu cầu này)
- Bước di chuyển thứ hai: B đặt S cách ô đặt thứ nhất của B ba ô. (Bước di chuyển thứ hai của A không cần trả lời vì điều này có thể thực hiện ít nhất ở một phía của ô S thứ nhất mà B đã đặt). Sau bước di chuyển thứ hai, B đã tạo ra hai ô *xấu*, ai đi tiếp vào một trong hai ô này sẽ thua vì sẽ tạo ra bảng không ổn định, người đi tiếp tạo được SOS. Do đó trò chơi này không có hòa.
- Trên bất kì phép di chuyển tiếp theo nào, B sẽ chơi ô không *xấu*. Ô vuông như thế luôn tồn tại vì: nếu bảng là ổn định sẽ có số lẻ ô rỗng (do $N = 2000$ là chẵn, A đi trước, nên đến lượt B luôn nhận số lẻ ô rỗng) và số chẵn ô *xấu*.

Do tồn tại ô *xấu* sau bước di chuyển thứ hai của B nên trò chơi không hoà và do B luôn luôn có thể làm bảng ổn định (bằng cách đi vào ô không *xấu*) nên A không thể thắng. Vậy B thắng.

Chương trình:

```
#include <iostream>
#define maxN 17
using namespace std;
int N= maxN-1, o;
char quan, A[maxN];
```

```

bool Stop=false;
int turn=0;
int vitriA1, vitriA2, vitriA, vitriB1, vitriB2, vitriB;
void Init(){
    for (int i=0; i!=maxN; i++) A[i]='-';
}
void HienA() {
    cout << " | ";
    for (int i=1; i!=10; i++) cout << i << " | ";
    for (int i=10; i!=maxN; i++) cout << i << " | ";
    cout << endl;
    cout << " | ";
    for (int i=1; i!=10; i++) cout << " | ";
    for (int i=10; i!=maxN; i++) cout << " | ";
    cout << endl;
    cout << " | ";
    for (int i=1; i!=10; i++) cout << A[i] << " | ";
    for (int i=10; i!=maxN; i++) cout << A[i] << " | ";
    cout << endl;
}
bool chienthangS(int i) { //Đặt S vào A[i] sẽ thắng
    if (A[i]!='-') return false;
    bool OK= ((i>2)&&(A[i-1]=='O')&&(A[i-2]=='S')) ;
    OK = (OK || ((i<maxN-2)&&(A[i+1]=='O')&&(A[i+2]=='S')));
    return OK;
}
bool chienthangO(int i) { //Đặt O vào A[i] sẽ thắng
    if (A[i]!='-') return false;
    bool OK= ((i>1)&&(A[i-1]=='S')&&(A[i+1]=='S')&&(i<maxN-1)) ;
    return OK;
}
bool diduoc(int i, char &q) { //Chọn quân q đặt vào A[i] không làm mất ổn định
    if (A[i]!='-') return false;
    bool OK, OK1, OK2;
    OK1 = ((A[i-1]=='S')&&(A[i+1]=='-')) || ((A[i-1]=='-')&&(A[i+1]=='S')) ;
    OK1 = ((!OK1) &&(i>1)&& (i<maxN-1));
    if (OK1) {
        q='O';
        return OK1;
    }
    else {
        OK2 = ((A[i-2]=='S')&&(A[i-1]=='-')) || ((A[i+1]=='-')&&(A[i+2]=='S')) ;
        OK2 = ((!OK2) &&(i>1)&& (i<maxN-1));
        if (OK2) {
            q='S';
            return OK2;
        }
    }
    return false;
}

```

```

    }
void maydi() { //Chiến thuật máy đi giành thắng (máy là người đi sau: B)
    if (turn==1) { //Lần đi thứ nhất của máy
        for (int i=1; i!=maxN; i++) {
            if (A[i]!='-') {
                vitriA1 = i;
                break;
            }
        }
        if (vitriA1<maxN/2) vitriB1= maxN-4;
        else vitriB1= 4;
        A[vitriB1]='S';
        cout << "MAY DI "<< endl;
        cout << "May dat S tai vitri: " << vitriB1 << endl;
    }
    else if (turn==2) { //Lần đi thứ hai của máy
        int i=vitriB1-1;
        int dem=0;
        while ((A[i]=='-') && (i>=1) ) { dem++; i--; }
        if (dem>2) vitriB2=vitriB1-3;
        else vitriB2=vitriB1+3;
        A[vitriB2]='S';
        cout << "May dat S tai vitri: " << vitriB2 << endl;
    }
    else { //Những lần đi sau lần thứ hai
        int i;
        for (i=1; i!=maxN; i++) {
            if (chienthangS(i)) {
                vitriB = i;
                A[i] = 'S';
                cout << "May dat S tai vitri: " << vitriB << endl;
                Stop=true;
                return;
            }
            else if (chienthangO(i)) {
                vitriB = i;
                A[i] = 'O';
                cout << "May dat O tai vitri: " << vitriB << endl;
                Stop=true;
                return;
            }
            else if (diduoc(i, quan)) {
                A[i]=quan;
                cout << "May dat " << quan << " tai vitri: " << i << endl;
                return;
            }
        }
    }
}

```

```

        }
    else if (A[1]=='-') {
        A[1]='O';
        cout << "May dat O tai vitri: " << i << endl;
        return;
    }
}
}

void nguoidi() {
    bool OK;
    char quan;
    turn++; //Đếm lượt đi
    cout << endl;
    cout << "NGUOI DI luot " << turn << ":" << endl;
    cout << "Chon vitri nao? 1.." << maxN-1 << " : ";
    do { //Nhập vị trí ô
        cin >> o;
        OK = (A[o]=='-') && (o>0) && (o<maxN);
        if (OK) break;
        else
            cout << "Nhập chưa hợp lệ. Nhập lại ";
    } while (!OK);
    cout << "Dat S hay O ? ";
    do { //Nhập chọn quân là S hay O
        cin >> quan;
        OK = (quan=='S') || (quan=='O');
        if (OK) break;
        else
            cout << "Nhập chưa đúng S, O . Đề nghị nhập lại...";
    } while (!OK);
    A[o] = quan;
}

int main() {
    Init();
    HienA();
    Stop=false;
    while (!Stop) {
        nguoidi();
        HienA();
        maydi();
        HienA();
    }
    cout << "MAY THANG! ";
    system("pause");
    return 0;
}

```

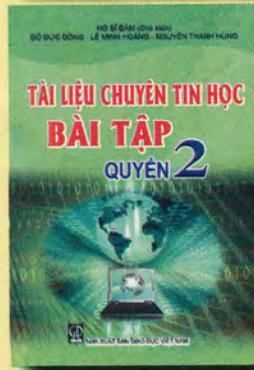
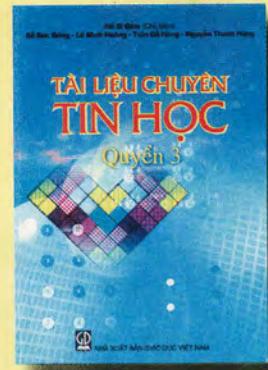
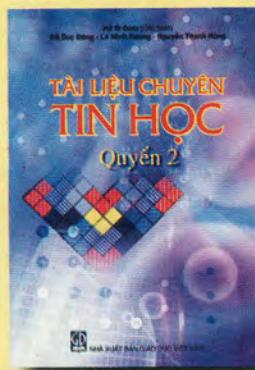
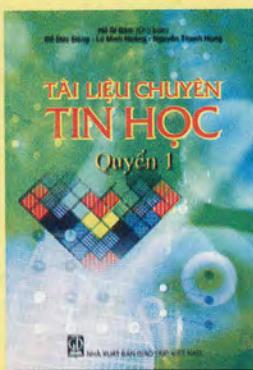
MỤC LỤC

LỜI NÓI ĐẦU	3
PHẦN 1. BÀI TẬP	5
CHUYÊN ĐỀ 8. HÌNH HỌC TÍNH TOÀN	5
BÀI TẬP BỔ SUNG	11
CHUYÊN ĐỀ 9. LÝ THUYẾT TRÒ CHƠI	17
BÀI TẬP BỔ SUNG	31
PHẦN 2. HƯỚNG DẪN GIẢI BÀI TẬP	34
CHUYÊN ĐỀ 8	34
CHUYÊN ĐỀ 9	86



VƯƠNG MIỆN KIM CƯƠNG
CHẤT LƯỢNG QUỐC TẾ

**GIỚI THIỆU BỘ SÁCH
TÀI LIỆU CHUYÊN TIN HỌC DÀNH CHO HỌC SINH PHỔ THÔNG**



*Bạn đọc có thể mua sách tại các Công ty Sách - Thiết bị trường học ở các địa phương
hoặc các cửa hàng sách của Nhà xuất bản Giáo dục Việt Nam :*

- **Tại TP. Hà Nội :** 45 Phố Vọng ; 187, 187C Giảng Võ ; 232 Tây Sơn ; 25 Hàn Thuyên;
51 Lò Ðúc ; 45 Hàng Chuối ; Ngõ 385 Hoàng Quốc Việt;
17 T2 - 17T3 Trung Hòa - Nhân Chính ; Tòa nhà HESCO Văn Quán - Hà Đông
- **Tại TP. Đà Nẵng :** 78 Pasteur ; 145 Lê Lợi ; 223 Lê Đình Lý.
- **Tại TP. Hồ Chí Minh :** 261C Lê Quang Định, Quận Bình Thạnh ; 231 Nguyễn Văn Cừ, Quận 5 ;
23 Đinh Tiên Hoàng, Phường Đa Kao, Quận 1, TP. Hồ Chí Minh
- **Tại TP. Cần Thơ :** 162D Đường 3 tháng 2, Phường Xuân Khánh, Quận Ninh Kiều.
- **Tại Website bán hàng trực tuyến :** www.sach24.vn

Website : www.nxbgd.vn

ISBN 978-604-0-08217-6

9 786040 082176



Giá : 28.000 đ