

Phan Công Minh

Hồ Sỹ Việt Anh , Ngô Văn Hoàng , Bùi Minh Trí , Nguyễn Cảnh Toàn.

MỘT SỐ VẤN ĐỀ ĐÁNG CHÚ Ý TRONG MÔN TIN HỌC



Vinh 9-2009

LỜI NÓI ĐẦU

Với mong muốn tổng hợp những thuật toán hay, các cách giải Độc Đáo trong quá trình học tập ngôn ngữ lập trình Pascal. Nhóm tác giả

Phan Công Minh : Học sinh chuyên Tin A2K35

Hồ Sỹ Việt Anh , Ngô Văn Hoàng , Bùi Minh Trí và Nguyễn Cảnh Toàn – Học sinh chuyên Tin khóa 36

Đã viết cuốn tài liệu **“Một số vấn Đề Đáng chú ý trong môn tin học”**.

Trong phạm vi nội dung của tài liệu này không Đề cập Đến các kiến thức, thuật toán cơ bản mà tập trung vào các kĩ thuật, thuật toán mở rộng cũng như cách kết hợp, ứng dụng chúng Để giải quyết các bài toán tin, Đặc biệt là các dạng bài thường gặp trong các kì thi.

Hi vọng cuốn tài liệu này có thể giúp ích cho bạn Đọc, nhất là các bạn học sinh trong Đội tuyển trong việc học tập , bồi dưỡng Tin học. Rất mong nhận Được sự Đóng góp của các bạn Để cuốn tài liệu hoàn thiện hơn.

Thay mặt nhóm tác giả:

Phan Công Minh

Email: congminh91@yahoo.com

MỤC LỤC

Lời nói Đầu	2
Duyệt nhánh cận	3
Duyệt ưu tiên	8
Tìm kiếm chuỗi	12
Xử lý bit	15
Quy hoạch Động trạng thái	21
Quy hoạch Động vị trí cấu hình	26
Quy hoạch Động trên cây	35
Sắp xếp topo và ứng dụng	44
Phát hiện chu trình	49
Chu trình Euler	53
Chu trình âm và ứng dụng	56
Tô màu Đồ thị	71
Thuật toán Ford Bellman kết hợp Queue vòng	81
Thuật toán Dijkstra với các Đỉnh ảo	84
Một số ứng dụng thuật toán Dijkstra	92
Luồng Mincost	99
Các công thức hình học	104
Một dạng bài Quy hoạch động	107
Trie Tree và ứng dụng	117
Duyệt bằng cách chia đôi tập hợp	123
Một số bài toán về cây khung	127
Tìm kiếm nhị phân và ứng dụng	133
Xếp lịch công việc	137
Xử lý số nguyên lớn và Hàm Mod	167
Heap và ứng dụng	173
Interval Tree	181
Binary Index Tree	190
Các lỗi trong Pascal	
Kinh nghiệm thi cử	

VẤN ĐỀ: DUYỆT NHÁNH CẬN.

Duyệt nhánh cận là phương pháp phổ biến sử dụng Để giải quyết một số lượng lớn các bài toán tin, Đặc biệt là các bài toán thực tế. Cấu trúc của chương trình duyệt nhánh cận Được mô tả sơ lược như sau:

Chọn một nghiệm của bài toán làm cận (c)

Thực hiện bước duyệt thứ i

- Nếu i là bước sau bước cuối cùng thì kiểm tra nghiệm mới tìm Được và cập nhật cận (c), ghi nhận một nghiệm tốt hơn của bài toán.

- Sau bước duyệt thứ i, kiểm tra nghiệm sẽ tìm Được có khả năng tốt hơn cận (c) hay không - gọi là Điều kiện nhánh cận. Nếu có gọi thực hiện bước thứ i+1. còn không thì quay lại bước i-1.

Hai yếu tố quan trọng của chương trình là cận (c) khởi tạo ban Đầu và Điều kiện nhánh cận. Việc chọn một cận (c) và Điều kiện kiểm tra nhánh cận tốt sẽ giúp bước duyệt tránh Đi vào những hướng mà chắc chắn không tìm ra Được kết quả tốt hơn. Điều có ảnh hưởng trực tiếp Đến Độ hiệu quả của chương trình.

Dưới Đây ta xét một số cách kiểm tra nhánh cận hay dùng thông qua các bài toán cụ thể.

Bài toán:

Chu trình Hamilton nhỏ nhất.

Cho Đồ thị vô hướng G. Mỗi cạnh Được gán một trọng số nhất Định (lớn hơn 0), tìm một chu trình Đi qua tất cả các Đỉnh, mỗi Đỉnh một lần và có tổng trọng số nhỏ nhất.

Gợi ý: Duyệt nhánh cận là phương pháp duy nhất Để giải quyết bài toán nêu trên.

Chu trình Đi qua N Đỉnh sẽ bao gồm N cạnh. Giả sử tại bước thứ i, Đã Đi qua k cạnh và còn n-k cạnh nữa cần phải Đi qua thì Điều kiện Để Đi tiếp bước thứ i+1 là

$$s(k) + (n-k) \cdot \text{minc} < s_{\text{min}}$$

Trong Đó:

- $s(k)$ là tổng chi phí của k cạnh Đã Đi qua
- minc là trọng số của cạnh nhỏ nhất trong số các cạnh còn lại chưa Được Đi qua của Đồ thị.
- s_{min} là tổng trọng số của một chu trình Đã tìm Được nhưng chưa tối ưu. Có thể khởi tạo s_{min} ban Đầu là một số vô cùng lớn.

Bài toán:

Xếp valy.

Một va ly có thể chứa tối Đa W Đơn vị trọng lượng. Có N loại Đồ vật, số lượng mỗi loại không hạn chế. Loại Đồ vật thứ i có trọng lượng A_i và có giá trị C_i . Hỏi nên chọn những loại Đồ vật nào và số lượng bao nhiêu Để xếp vào va ly sao cho:

- tổng trọng lượng của các vật không vượt quá giới hạn W của valy
- tổng giá trị của các vật là lớn nhất

Gợi ý: Với mỗi loại Đồ vật i, gọi T_i là giá trị riêng của nó: $T_i = C_i / A_i$. Sau Đó sắp các loại Đồ vật theo thứ tự giảm dần của T_i .

Phương pháp duyệt nhánh cận tại mỗi bước duyệt loại Đồ vật i, ta lấy k Đồ vật loại này, khi Đó Điều kiện Để duyệt tiếp loại Đồ vật i+1 (Điều kiện nhánh cận) là:

- $k \cdot A_i \leq w$
- $s(i) + k \cdot C_i + (w - k \cdot A_i) \cdot T_{i+1} > s_{\max}$

Trong Đó:

- w là trọng lượng mà valy có thể chứa thêm sau bước duyệt loại Đồ vật $i-1$
- $s(i)$ là tổng giá trị hiện Đang có của valy sau bước duyệt loại Đồ vật $i-1$
- s_{\max} là tổng giá trị valy của một các xếp Đã tìm Được nhưng chưa tối ưu. Có thể khởi tạo s_{\max} ban Đầu bằng 0.

Bài toán:

Xâu ABC.

Tìm một xâu chỉ gồm các ký tự A,B,C sao cho

- Có Độ dài N cho trước ($N \leq 100$)
- Hai Đoạn con bất kì liên nhau Luôn khác nhau (Đoạn con là một dãy các ký tự liên tiếp của xâu).
- Có ít ký tự C nhất

Gợi ý: Ta có nhận xét trong 4 ký tự liên tiếp luôn có ít nhất một ký tự C.

Do Đó Điều kiện nhánh cận tại bước duyệt ký tự i của xâu là

- xâu có i ký tự Đã tạo Được không có 2 Đoạn con liên tiếp trùng nhau.
- $s(i) + (n-i) \div 4 < s_{\min}$

Trong Đó

- $s(i)$ là số ký tự C Đã dùng Để tạo xâu i ký tự
- s_{\min} là số ký tự C của một xâu khác thỏa mãn 2 yêu cầu Đầu của Đề bài nhưng chưa tối ưu. Có thể khởi tạo s_{\min} ban Đầu là một số Đủ lớn.

Luyện tập:

Bài toán:

Tour du lịch rẻ nhất (TOUR)

Một khu thắng cảnh gồm n Điểm Đánh số từ 1 tới n ($n \leq 100$) và m Đường Đi hai chiều giữa các cặp Địa Điểm Đó, chi phí Đi trên các Đường Đi là biết trước (≤ 10000).

Một Tour du lịch là một hành trình xuất phát từ một Địa Điểm Đi thăm ≥ 2 Địa Điểm khác và quay trở về Điểm xuất phát, ngoại trừ Địa Điểm xuất phát, không Địa Điểm nào bị thăm tới hai lần. Chi phí của một Tour du lịch là tổng chi phí các quãng Đường Đi qua.

Yêu cầu: Hãy tìm Tour du lịch có chi phí rẻ nhất.

Dữ liệu: TOUR.INP

- ✂ Dòng 1: Ghi hai số nguyên dương n, m
- ✂ m dòng tiếp theo mỗi dòng có dạng $x \ y \ c$. Cho biết có Đường Đi trực tiếp nối Địa Điểm x với Địa Điểm y và chi phí Đi quãng Đường Đó là c .

Kết quả: TOUR.OUT

- ✂ Dòng 1: Ghi số 1 nếu như tồn tại hành trình theo yêu cầu, ghi số 0 nếu không tồn tại hành trình.

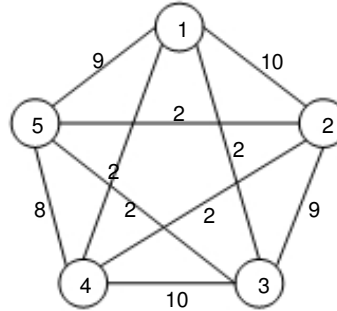
Nếu dòng Đầu tiên ghi số 1:

- ✂ Dòng thứ 2 ghi chi phí của tour tìm Được
- ✂ Dòng thứ 3 ghi số k là số Địa Điểm tới thăm

✧ Dòng thứ 4 gồm k số, số thứ i là Địa Điểm tới thăm thứ i trong tour, quy ước Địa Điểm thăm Đầu tiên là Địa Điểm xuất phát, Địa Điểm thăm thứ k (Địa Điểm cuối cùng) là Địa Điểm mà từ Đó quay trở lại Điểm xuất phát Để kết thúc hành trình. Các số trên một dòng của Input/Output File Được ghi cách nhau ít nhất một dấu cách.

Ví dụ:

TOUR.INP	TOUR.OUT
5 10	1
1 3 2	10
2 4 2	5
3 5 2	3 5 2 4 1
4 1 2	
5 2 2	
1 2 10	
2 3 9	
3 4 10	
4 5 8	
5 1 9	



Bài toán:

Robot cứu hỏa - HSG QG 2007 (QBROBOT)

Trên một mạng lưới giao thông có n nút, các nút Được Đánh số từ 1 Đến n và giữa hai nút bất kỳ có không quá một Đường nối trực tiếp (Đường nối trực tiếp là một Đường hai chiều). Ta gọi Đường Đi từ nút s Đến nút t là một dãy các nút và các Đường nối trực tiếp có dạng:

$$s = u_1, e_1, u_2, \dots, u_i, e_i, u_{i+1}, \dots, u_{k-1}, e_{k-1}, u_k = t,$$

trong Đó u_1, u_2, \dots, u_k là các nút trong mạng lưới giao thông, e_i là Đường nối trực tiếp giữa nút u_i và u_{i+1} (không có nút u_j nào xuất hiện nhiều hơn một lần trong dãy trên, $j = 1, 2, \dots, k$).

Biết rằng mạng lưới giao thông Được xét luôn có ít nhất một Đường Đi từ nút 1 Đến nút n.

Một robot chứa Đầy bình với w Đơn vị năng lượng, cần Đi từ trạm cứu hỏa Đặt tại nút 1 Đến nơi xảy ra hỏa hoạn ở nút n, trong thời gian ít nhất có thể. Thời gian và chi phí năng lượng Để robot Đi trên Đường nối trực tiếp từ nút i Đến nút j tương ứng là t_{ij} và c_{ij} ($1 \leq i, j \leq n$). Robot chỉ có thể Đi Được trên Đường nối trực tiếp từ nút i Đến nút j nếu năng lượng còn lại trong bình chứa không ít hơn c_{ij} ($1 \leq i, j \leq n$). Nếu robot Đi Đến một nút có trạm tiếp năng lượng (một nút có thể có hoặc không có trạm tiếp năng lượng) thì nó tự Động Được nạp Đầy năng lượng vào bình chứa với thời gian nạp coi như không Đáng kể.

Yêu cầu: Hãy xác Định giá trị w nhỏ nhất Để robot Đi Được trên một Đường Đi từ nút 1 Đến nút n trong thời gian ít nhất.

Dữ liệu: QBROBOT.INP

✧ Dòng Đầu tiên chứa một số nguyên dương n ($2 \leq n \leq 500$)

✧ Dòng thứ hai chứa n số, trong Đó số thứ j bằng 1 hoặc 0 tương ứng ở nút j có hoặc không có trạm tiếp năng lượng ($j = 1, 2, \dots, n$)

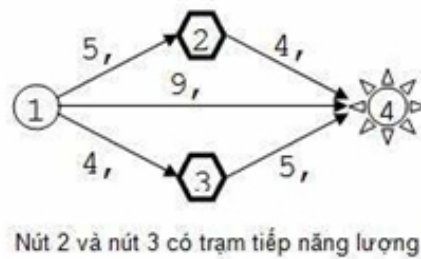
- ✧ Dòng thứ ba chứa số nguyên dương m ($m \leq 30000$) là số Đường nối trực tiếp có trong mạng lưới giao thông
 - ✧ Dòng thứ k trong số m dòng tiếp theo chứa 4 số nguyên dương i, j, t_{ij}, c_{ij} ($t_{ij}, c_{ij} \leq 10000$) mô tả Đường nối trực tiếp từ nút i Đến nút j , thời gian và chi phí năng lượng tương ứng.
- Hai số liên tiếp trên một dòng trong file dữ liệu cách nhau ít nhất một dấu cách.

Kết quả: QBROBOT.OUT

- ✧ Ghi ra số nguyên dương w tìm Được.

Ví dụ:

QBROBOT.INP	QBROBOT.OUT
4	3
0 1 1 0	
5	
1 2 5 4	
1 3 4 3	
1 4 9 4	
2 4 4 1	
3 4 5 2	



Gợi ý: Điều kiện ưu tiên là Đường Đi có thời gian ít nhất, do Đó, cho w có giá trị là một số vô cùng lớn, sau Đó DIJTRA tìm Đường Đi ngắn nhất (có thời gian ít nhất). Gọi $D[i]$ là Độ dài Đường Đi ngắn nhất từ i Đến n .

Chặt nhị phân giá trị w , với mỗi w , tìm Đường Đi Đến n Đảm bảo robot luôn có năng lượng trên Đường Đi. Đồng thời Đường Đi phải là ngắn nhất.

Giả sử Đang ở Đỉnh u , năng lượng Đang còn là w' , thực hiện bước duyệt thử Đi qua Đỉnh v nếu

- (u,v) là một cạnh
- Hoặc u là trạm tiếp năng lượng, hoặc năng lượng $w' \geq c_{u,v}$
- $s(u) + t_{u,v} + d[v] = d[1]$ trong Đó $s(u)$ là tổng thời gian Đã Đi từ 1 Đến u

Bài toán:

Tìm Đường Đi (ROADS)

Có N thành phố $1..N$ nối bởi các con Đường một chiều. Mỗi con Đường có hai giá trị: Độ dài và chi phí phải trả Để Đi qua. Bob ở thành phố 1. Bạn hãy giúp Bob tìm Đường Đi ngắn nhất Đến thành phố N , biết rằng Bob chỉ có số tiền có hạn là K mà thôi.

Dữ liệu: ROADS.INP

- ✧ Dòng Đầu tiên ghi t là số test.

Với mỗi test:

- ✧ dòng Đầu ghi K ($0 \leq K \leq 10000$).
- ✧ Dòng 2 ghi N , $2 \leq N \leq 100$.
- ✧ Dòng 3 ghi R , $1 \leq R \leq 10000$ là số Đường nối.

- ✂ Mỗi dòng trong N dòng sau ghi 4 số nguyên S, D, L, T mô tả một con Đường nối giữa S và D với Độ dài L ($1 \leq L \leq 100$) và chi phí T ($0 \leq T \leq 100$). Lưu ý có thể có nhiều con Đường nối giữa hai thành phố.

Kết quả: ROADS.OUT

- ✂ Với mỗi test, in ra Độ dài Đường Đi ngắn nhất từ 1 Đến N mà tổng chi phí không quá K. Nếu không tồn tại, in ra -1.

Ví dụ:

ROADS.INP	ROADS.OUT
2	11
5	-1
6	
7	
1 2 2 3	
2 4 3 3	
3 4 2 4	
1 3 4 1	
4 6 2 1	
3 5 2 0	
5 4 3 2	
0	
4	
4	
1 4 5 2	
1 2 1 0	
2 3 1 1	
3 4 1 0	

Bài toán:

Romeo and Juliet

Romeo và Juliet bị giam trong một mê cung hình chữ nhật kích thước $M \times N$ ô vuông. Mê cung có hai loại ô: tự do và cấm. Trong quá trình Đi trong mê cung, tại mỗi bước ta chỉ có thể di chuyển Đến ô tự do kề cạnh theo một trong bốn hướng: Đông (E), Tây (W), Nam (S), Bắc (N).

Ban Đầu, Romeo ở tại ô $[1,1]$ và Juliet ở tại ô $[M,N]$. Để thoát khỏi mê cung, Romeo cần di chuyển Đến ô $[X,Y]$ và Juliet cần di chuyển tới ô $[Z,T]$.

Để cứu cặp tính nhân, Shakespeare Đưa cho bạn một cấm mang là xâu D chỉ gồm các kí tự E, W, N, S thể hiện một loạt các bước di chuyển và cho biết có thể chọn cho mỗi người một số bước di chuyển theo thứ tự cho trong dãy, mỗi bước di chuyển chỉ Được một người chọn và mọi bước di chuyển trong dãy Đều phải Được chọn thì mỗi người mới có thể Đi Đến Đích của mình. Yêu cầu bạn hãy chọn cho mỗi người dãy bước di chuyển thích hợp. Nếu có nhiều lời giải chỉ cần Đưa ra một lời giải.

0	0	1	0	0
0	0	0	1	1
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0

Ví dụ với mê cung như bảng bên kí hiệu 1 là ô cấm, 0 là ô tự do, Romeo cần Đến ô [2,2], Juliet cần Đến ô [2,2], Juliet cần Đến ô [4,4] và xâu D là SNEEWW thì việc chọn các bước di chuyển của hai người có thể viết dưới dạng các xâu chỉ gồm 2 kí tự R và J: RJRRJR; có nghĩa là Romeo thực hiện 4 bước di chuyển Nam (S), Đông (E), Đông (E), Tây (W), còn Juliet thực hiện 2 bước di chuyển Bắc (N), Tây (W). Một lời giải khác cho bởi xâu RJRRRJ.

Dữ liệu: ROJU.INP

- ✂ Dòng thứ nhất ghi hai số M, N ($3 \leq M \leq 20$, $3 \leq N \leq 60$).
- ✂ Tiếp theo là M dòng, dòng thứ i trong M dòng này ghi N số 0 hoặc số 1 thể hiện tình trạng của các ô dòng thứ i.
- ✂ Tiếp theo là một dòng ghi 4 số X, Y, Z, T
- ✂ Dòng tiếp theo ghi số nguyên dương K ($3 \leq K \leq 200$) là Độ dài xâu D. ✂ Dòng cuối cùng ghi từ Đầu dòng xâu D.

Kết quả: ROJU.OUT

- ✂ Ghi ra từ Đầu dòng một xâu C Độ dài K chỉ gồm các kí tự R và J thể hiện sự lựa chọn các bước di chuyển của hai người. Nói rõ hơn, nếu kí tự thứ U của xâu C là R/J có nghĩa là Romeo/Juliet chọn bước di chuyển thứ U trong xâu D.

VẤN ĐỀ: DUỆT ƯU TIÊN

Duyệt là một phương pháp khá mạnh Để giải quyết phần lớn các bài toán tin học. Một số bài có kích thước dữ liệu lớn Đòi hỏi phải có thứ tự ưu tiên các phép duyệt Để nhanh chóng tìm ra kết quả. Dưới Đây ta xét một số bài toán Điển hình như vậy.

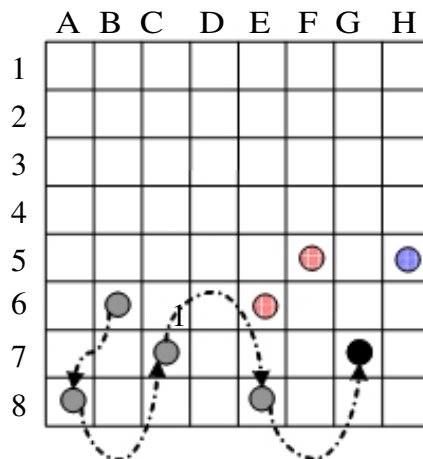
Bài toán:

Mã Đi tuần

Cho bàn cờ tổng quát $n \times n$ và một quân mã ($n \leq 100$). Hãy chỉ ra một hành trình của Mã xuất phát từ ô (x,y) , Đi qua tất cả các ô còn lại của bàn cờ, mỗi ô Đúng một lần (luật Đi của mã như luật cờ vua).

Gợi ý: Nếu xem mỗi ô của bàn cờ là một Đỉnh của Đồ thị thì bài toán này tương Đương với bài toán tìm Đường Đi Haminton (Đường Đi qua tất cả các Đỉnh và qua mỗi Đỉnh Đúng một lần). Thuật toán duy nhất cho bài toán này là duyệt Đệ quy quay lui.

Tại mỗi bước duyệt, Giả sử ta Đang ở Đỉnh u , từ Đỉnh u ta ưu tiên Đi sang Đỉnh v có cạnh nối với u mà bậc của v là nhỏ nhất (Định nghĩa bậc của một Đỉnh v là số Đỉnh nối với Đỉnh v Đó mà chưa Được thăm).



Xét ví dụ trên hình, con mã Đang Đã Đi Được 4 bước và Đang ở ô $(7,G)$. Từ ô này con mã có thể nhảy Đến 1 trong 3 ô tiếp theo là $(6,E)$ - có bậc là 6; ô $(5,F)$ - có bậc là 7; hoặc ô $(5,H)$ - có bậc là 3. Con mã sẽ ưu tiên nhảy ô có bậc nhỏ hơn trước - là ô $(5,H)$.

Sự ưu tiên trong phương pháp duyệt trên giúp chương trình chạy nhanh hơn rất nhiều so với khi duyệt Đơn thuần. Tuy nhiên với trường hợp bài toán vô nghiệm (không có Đường Đi Haminton) thì chương trình cũng chạy rất lâu vì phải duyệt qua hết toàn bộ không gian dữ liệu. Do Đó cần dùng thêm biến chặn thời gian Để xử lý trường hợp này.

Bài toán:

Cặp Điểm gần nhất

Trên mặt phẳng tạo Độ cho N Điểm phân biệt ($N \leq 20000$). Tìm 2 Điểm mà khoảng cách giữa chúng là ngắn nhất.

Gợi ý: Nếu duyệt thông thường, thuật toán mất $O(N^2)$ và chạy rất lâu với N lớn. Sự ưu tiên trong phép duyệt sau Đây giúp thuật toán chạy trong thời gian cho phép và cho kết quả chính xác với phần lớn các test.

Ta sắp xếp lại các Điểm theo một tiêu chí nào Đó có liên quan Đến khoảng cách. Chẳng hạn lấy thêm một Điểm bất kì nằm ngoài tập Điểm và sắp lại N Điểm Đã cho theo thứ tự tăng (giảm) dần khoảng cách Đến Điểm này. Hay sắp tăng (giảm) N Điểm theo hoành Độ x. Hay sắp tăng (giảm) N Điểm theo tung Độ y...

Sau khi sắp xếp như vậy, với mỗi Điểm i , ta ưu tiên xét các Điểm j trong không gian lân cận với nó trước - tức là những Điểm gần Điểm i trong danh sách Đã sắp xếp. Độ rộng của không gian lân cận (có thể gọi là không gian tìm kiếm) do ta tự quy Định, không gian xét càng rộng thì Độ chính xác của chương trình càng cao nhưng thời gian chạy càng lâu.

Ví dụ với $N=20000$. sau khi sắp xếp lại ta Được dãy các Điểm $i_1, i_2, \dots, i_{20000}$. với mỗi Điểm i_k , nếu ta quy Định không gian tìm kiếm là 100 thì ta chỉ xét khoảng cách của i_k với các Điểm $i_{k+1}, i_{k+2}, \dots, i_{k+100}$.

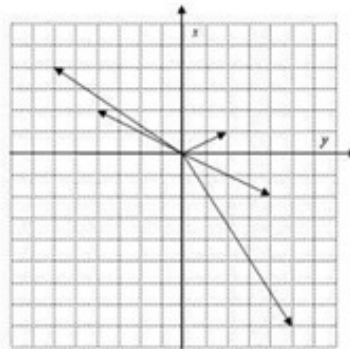
Việc sắp xếp các Điểm và chọn không gian tìm kiếm hợp lý sẽ giúp chương trình có Độ tin cậy cao và chạy trong thời gian cho phép.

Chú ý: Bài toán trên còn có thuật giải Độ phức tạp $N \log N$ tuy nhiên khá phức tạp và không Được nói ở Đây.

Bài toán:

Tổng vector

Vector và các phép toán với vector Được sử dụng Để giải quyết nhiều bài toán trong hình học, vật lý, ... Ta có thể biểu diễn vector trong mặt phẳng bởi cặp hai số (x, y) . Các số x, y Được gọi là các toạ Độ của vector (xem hình vẽ minh hoạ).



Hình vẽ minh họa. Vectơ trong mặt phẳng

Tổng của hai (hay nhiều hơn hai) vector là vector mà các toạ Độ của nó là tổng của các toạ Độ tương ứng của tất cả các vector trong tổng, nghĩa là tổng của n ($n \leq 2$) vector

$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ là vector (x, y) với $x = \sum_{i=1}^n x_i, y = \sum_{i=1}^n y_i$

Ta gọi trọng lượng của vector (x, y) là số $w(x, y) = x^2 + y^2$.

Ví dụ: Cho 3 vector $(-1, 3)$, $(2, -5)$ và $(4, -2)$, khi Đó ta có tổng của chúng là vector $(-1, 3) + (2, -5) + (4, -2) = (-1+2+4, 3-5-2) = (5, -4)$ có trong lượng là 41.

Yêu cầu: Cho n vector trong mặt phẳng: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$. Cần tìm tập con (gồm ít ra là hai phần tử) của tập các vector đã cho với trọng lượng của tổng của tất cả các vector trong nó là lớn nhất.

Dữ liệu: SUMVEC.INP

- ✂ Dòng Đầu tiên chứa số nguyên n ($2 \leq n \leq 15000$)
- ✂ n dòng tiếp theo mô tả n vector Đã cho: mỗi dòng chứa một cặp tọa Độ của một vector. Giả thiết rằng: Các tọa Độ là các số nguyên có trị tuyệt Đối không quá 30000. Trong số các vector Đã cho không có vector nào là $(0, 0)$. Các số trên cùng dòng Được ghi cách nhau bởi dấu cách.

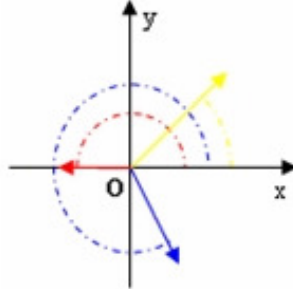
Kết quả: SUMVEC.OUT

- ✂ Trọng lượng của vector tổng của các vector trong tập con tìm Được.

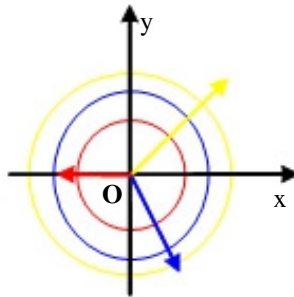
Ví dụ:

SUMVEC .INP	SUMVEC.OUT
5	202
5 -8	
-4 2	
4 -2	
2 1	
-6 4	

Gợi ý: Nếu duyệt tổ hợp các vector thông thường thì sẽ có Độ phức tạp là $O(2^n)$ - không thể chạy với n lớn. Thuật toán duyệt ưu tiên sau Đây chỉ mất $O(n^2)$ bằng cách sắp xếp các vector theo góc hợp với tia Ox (chú ý góc hợp này có giá trị trong nửa khoảng $[0;360)$).



Trong dãy các vector Đã sắp xếp, ta ưu tiên xét tổng các vector liên tiếp nhau. Do Đó chỉ cần duyệt 2 biến chạy i và j ; sau Đó kiểm tra tổng các vector Đoạn từ vector i Đến vector j trong dãy Đã sắp xếp.



Luyện tập:

Bài toán:

Phòng cháy (FIRE)

Để Đối phó với tình hình biến Động của giá xăng dầu, nước X quyết Định xây dựng một kho dự trữ dầu với quy mô cực lớn. Kho chứa dầu sẽ bao gồm N bể chứa dầu hình trụ tròn mà ta sẽ biểu diễn trên bản Đồ bằng N hình tròn, hình tròn thứ i có tọa Độ là (X_i, Y_i) và bán kính R_i , các hình tròn không có Điểm chung trong với nhau (nhưng có thể tiếp xúc).

Để Đảm bảo an toàn phòng cháy chữa cháy, người ta cần xác Định 2 bể chứa dầu gần nhau nhất Để tăng cường cách ly khi xảy ra hỏa hoạn.

Biết rằng khoảng cách giữa 2 bể chứa dầu thứ i và thứ j chính bằng khoảng cách giữa 2 Đường tròn tương ứng và bằng $D_{ij} - R_i - R_j$, trong Đó D_{ij} là khoảng cách Euclide giữa 2 Điểm (X_i, Y_i) và (X_j, Y_j) .

Bạn hãy giúp những người quản lý tìm ra 2 bể chứa dầu này.

Dữ liệu: FIRE.INP

- ✂ Dòng thứ nhất ghi số nguyên dương N là số bể chứa dầu.
- ✂ Dòng thứ i trong N dòng tiếp theo ghi 3 số nguyên X_i, Y_i, R_i là tọa Độ và bán kính bể chứa dầu thứ i.

Kết quả: FIRE.OUT

- ✂ Gồm 1 dòng duy nhất là khoảng cách của 2 bể chứa dầu bé nhất tìm Được.

Giới hạn:

- ✂ $2 \leq N \leq 10000$.
- ✂ $|X_i|, |Y_i| \leq 10^6$.
- ✂ $0 < R_i \leq 10^6$.
- ✂ Kết quả ghi chính xác Đến 4 chữ số sau dấu phẩy.

Ví dụ:

FIRE.INP	FIRE.OUT
3	0.0990
0 0 1	
4 0 2	
5 5 3	

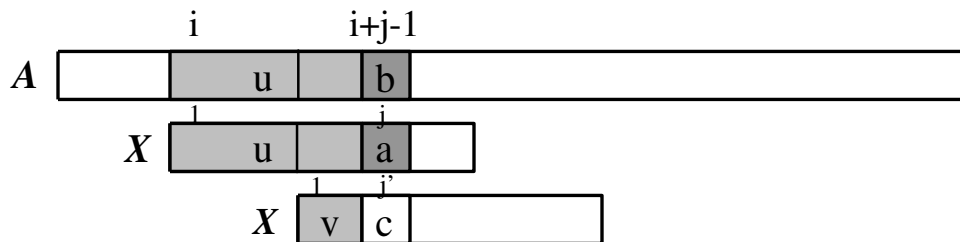
VẤN ĐỀ: TÌM KIẾM CHUỖI.

Bài toán: Cho chuỗi A có Độ dài m, một chuỗi X có Độ dài n. Kiểm tra xem chuỗi X xuất hiện trong chuỗi A tại các vị trí nào.

Hướng giải quyết:

Cách thứ nhất: Duyệt mọi vị trí i của chuỗi A, với mỗi vị trí kiểm tra chuỗi X có xuất hiện ở vị trí này hay không. Thuật giải có Độ phức tạp $O(m*n)$. Một thuật giải tốt hơn chỉ mất $O(m+n)$:

Cách thứ hai: Thuật toán Knuth-Morris-Pratt (KMP).



Thuật toán dựa vào một cách kiểm tra khá trực quan như sau: Ta cố Định chuỗi A và di chuyển chuỗi X trượt theo chuỗi A. Giả sử chuỗi X Đang ở vị trí i so với chuỗi A. xét lần lượt các kí tự X_1, X_2, \dots, X_n với các kí tự tương ứng của chuỗi A: $A_i, A_{i+1}, \dots, A_{i+n-1}$. Giả sử sự khác nhau Đầu tiên xuất hiện ở vị trí $X_j \neq A_{i+j-1}$. Gọi $X_{1..j-1} = u = A_{i..i+j-2}$. Lúc này ta sẽ dịch chuyển chuỗi X một Đoạn ngắn nhất thỏa mãn:

- Phần Đầu của chuỗi X trùng với phần cuối của $A_{i..i+j-2}$. Sau khi dịch chuyển, vị trí so sánh tương ứng với A_{i+j-1} là $X_{j'}$. Ta có $X_{1..j'-1}$ trùng với phần cuối của $A_{i..i+j-2}$ mà $A_{i..i+j-2} = X_{1..j-1}$ cho nên $X_{1..j'-1}$ trùng với phần cuối của $X_{1..j-1}$.
- $X_j \neq X_{j'}$

Do Đó nếu xét Đến vị trí j mà $X_j \neq A_{i+j-1}$ thì ta cần quan tâm Đến vị trí j' thỏa mãn $X_{1..j'-1}$ dài nhất trùng với phần cuối của $X_{1..j-1}$ và $X_j \neq X_{j'}$.

Sử dụng $Next[j]$ có ý nghĩa $Next[j] = j'$ nếu:

- $X_{1..j'-1}$ dài nhất trùng với phần cuối của $X_{1..j-1}$
- $X_j \neq X_{j'}$

Mảng $Next[j]$ có thể khởi tạo bằng Quy hoạch Động.

Mô tả thuật toán:

```
procedure Init;
var
  j, jj: integer;
begin
  j := 1;
  jj := 0;
  Next[1] := 0;
  while (j <= n) do
  begin
    while (jj > 0) and (X[j] <> X[jj]) do jj := Next[jj];
    Inc(j);
    Inc(jj);
    if X[j] = X[jj] then Next[j] := Next[jj] else Next[j] := jj;
  end;
end;

procedure Process;
var
  i, j: integer;
Begin
  Init
  i := 1;
  j := 1;
  repeat
    while (j > 0) and (X[j] <> A[i]) do j := Next[j];
    Inc(i);
    Inc(j);
    if j > n then
    begin
      Writeln(' xuất hiện tại ', i - j + 1);
      j := Next[j];
    end;
  until i > m;
End;
```

Luyện tập:

Bài toán:

Xâu con (SUBSTR)

Cho chuỗi A và chuỗi B chỉ gồm các chữ cái thường. Chuỗi B Được gọi là xuất hiện tại vị trí i của chuỗi A nếu: $A[i] = B[1]$, $A[i+1] = B[2]$, ..., $A[i+\text{length}(B)-1] = B[\text{length}(B)]$.

Yêu cầu: Hãy tìm tất cả các vị trí mà B xuất hiện trong A.

Dữ liệu: SUBSTR.INP

- ✂ Dòng 1: chuỗi A
- ✂ Dòng 2: chuỗi B
- ✂ Độ dài A, B không quá 1000000.

Kết quả: SUBSTR.OUT

- ✂ Ghi ra các vị trí tìm Được trên 1 dòng (thứ tự tăng dần). Nếu B không xuất hiện trong A thì bỏ trắng.

Ví dụ:

SUBSTR.INP	SUBSTR.OUT
aaaaa aa	1 2 3 4

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

VẤN ĐỀ: XỬ LÝ BIT

Như ta Đã biết, mọi số tự nhiên Đều có thể biểu diễn Được dưới dạng nhị phân. Ví dụ 13 : 1101

42: 101010

Đối với máy tính, các giá trị số Được lưu bởi các biến dưới dạng chuỗi nhị phân có Độ dài tương ứng với kiểu biến Đó.

Trong Pascal, một số kiểu biến phổ biến như :

- Byte: có Độ dài 8 bit.
- Integer, Word: có Độ dài $2*8 = 16$ bit
- Longint: có Độ dài $4*8 = 32$ bit

Do Đó nếu biến x có kiểu byte và lưu giá trị 13 thì dãy bit của biến x là : 00001101

Các bit của biến Được Đánh số từ phải sang trái bắt Đầu từ 1. Ở ví dụ trên bit thứ nhất của x là 1 và bit thứ 2 của x là 0 ...

Các phép toán xử lý bit Đối với 2 dãy bit x1,x2 sẽ xử lý lần lượt bit thứ nhất của x1 với bit thứ nhất của x2, bit thứ 2 của x1 với bit thứ 2 của x2,..., bit thứ k của x1 với bit thứ k của x2, cứ như vậy cho Đến hết.

Có 4 phép xử lý bit cơ bản sau Đây:

Phép cộng bit (**OR**)

Kết quả bằng tổng giá trị 2 bit trừ trường hợp cả 2 bit Đều có giá trị là 1 thì kết quả phép OR là 1.

Vd: x1 = 13 (00001101)
 x2 = 42 (00101010)
 x1 **OR** x2 = 47 (00101111)

Phép Đảo bit (**NOT**)

Đảo giá trị bit 0 thành 1 và 1 thành 0

Vd: x = 13 (00001101)
 NOT(x) = 242 (11110010)

Phép nhân bit (**AND**)

Kết quả bằng tích của 2 giá trị bit

Vd: x1 = 13 (00001101)
 x2 = 42 (00101010)
 x1 **AND** x2 = 8 (00001000)

Phép loại trừ bit (**XOR**)

Kết quả là 0 nếu 2 bit Được **XOR** có giá trị giống nhau
Kết quả là 1 nếu 2 bit Được **XOR** có giá trị khác nhau

Vd: x1 = 13 (00001101)
 x2 = 42 (00101010)
 x1 **XOR** x2 = 39 (00100111)

Đối với dãy bit còn có 2 phép dịch chuyển cơ bản sau:

Phép dịch phải (**Shr k**)

Dịch chuyển dãy sang phải k bit

Vd: x = 13 (00001101)
 x **Shr** 2 = 3 (00000011)

Phép dịch trái (**Shl** k) Dịch chuyển dãy sang trái k bit
Vd: $x = 13$ (00001101)
 x **Shl** 2 = 3 (00110100)

Bây giờ, ta xét các thao tác xử lí بیت dựa trên các phép toán cơ bản trên.

1/Lấy giá trị bit : Cho biết bit thứ k của biến x nhận giá trị 0 hay 1.

```
Function GetBit(k,x:Word):Byte;  
Begin  
    GetBit := (x Shr (k-1)) and 1;  
End;
```

2/Gán giá trị bit : gán giá trị c cho bit thứ k của biến x.

```

Procedure SetBit(c:byte; k:Word; var x: Word);
Begin
    If c = 1 then x := x or (1 shl(k-1))
    Else x := x and (not (1 shl (k-1)));
End;

```

Ưu Điểm của xử lý bit:

Về bộ nhớ: Đối với ngôn ngữ lập trình có bộ nhớ hạn chế như Turbo Pascal, xử lý bit có thể Được sử dụng Để tạo mảng Đánh dấu. Vd 1 biến boolean chỉ Đánh dấu Được 1 phần tử có giá trị True hay False trong khi nếu xử lý bit thì 1 biến boolean tương ứng với 8 bit giá trị 0,1 do Đó có thể Đánh dấu cho 8 phần tử.

Về tốc Độ: Các phép xử lý trên bit có tốc Độ nhanh hơn nhiều lần so với các phép xử lý khác.

Ví dụ:

- Hai phép (x **div** 2) và (x **shr** 1) là tương đương nhau nhưng phép (x **shr** 1) có tốc Độ nhanh hơn nhiều lần.
- Dùng dãy bit Để Đánh dấu True - False có tốc Độ xử lý nhanh hơn nhiều so với dùng mảng Để Đánh dấu. Do Đó trong các bài toán Đòi hỏi việc thay Đổi trạng thái Đánh dấu nhiều lần thì người ta vẫn hay dùng xử lý bit Để cải thiện tốc Độ chương trình.

Ứng dụng của xử lý بیت qua các bài toán cụ thể sau:

Bài toán:

Liệt kê tập hợp con

Cho tập hợp con N phần tử khác nhau, liệt kê tất cả các tập hợp con của tập N phần tử đã cho (kể cả tập rỗng).

Gợi ý: Xét dãy gồm N bit. Mỗi tập con của tập N phần tử Đều có thể biểu diễn bằng dãy N bit này và ngược lại mỗi dãy bit có Độ dài N Đều biểu diễn 1 tập con của tập N phần tử. Do Đó ta có thể duyệt tất cả các dãy bit Độ dài N Để xuất ra tập con tương ứng.

Mặt khác ta có nhận xét tất cả các dãy bit Độ dài N tương ứng với giá trị 0 cho Đến $2^n - 1$

$$\begin{aligned} 00\dots000 &= 0 \\ 00\dots001 &= 1 \\ 00\dots010 &= 2 \\ &\dots\dots\dots \\ 11\dots110 &= 2^n - 2 \\ 11\dots111 &= 2^n - 1 \end{aligned}$$

Cho nên ta có một thuật toán cụ thể sau: Duyệt biến x chạy từ 0 Đến $2^n - 1$. Với mỗi giá trị x, xuất ra tập con dựa vào dãy bit của nó.

Bài toán:

Số Đặc biệt(SNUM)

Cho dãy N số tự nhiên có giá trị trong Đoạn $[0,65535]$ ($N \leq 1000000$). Trong Đó chỉ có một số xuất hiện 1 lần - gọi là số Đặc biệt, các số còn lại có số lần xuất hiện là một số chẵn. Yêu cầu tìm ra số Đặc biệt này.

Dữ liệu: SNUM.INP

- ✂ Dòng Đầu tiên ghi giá trị N
- ✂ N dòng tiếp theo, mỗi dòng ghi một số trong dãy N số

Kết quả: SNUM.OUT

- ✂ Số Đặc biệt tìm Được

Ví dụ:

SNUM.INP	SNUM.OUT
5	145
12345	
9876	
12345	
145	
9876	

Gợi ý:

Cách thứ nhất: Xếp N số tự nhiên này lên N và dồn các số này về phía bên phải, theo vd trên ta Được bảng số:

```

1  2  3  4  5
   9  8  7  6
1  2  3  4  5
   1  4  5
   9  8  7  6
    
```

Bây giờ ta xét theo từng cột của bảng số và bỏ Đi những chữ số có số lần xuất hiện là một số chẵn: Cột 1 bỏ chữ số 1, Cột 2 bỏ chữ số 2 và 9, ...Sau khi bỏ, các cột còn lại chữ số có số lần xuất hiện là một số lẻ.

```

1  2  3  4  5
   9  8  7  6
1  2  3  4  5
   1  4  5
   9  8  7  6
    
```

Ghép các chữ số này lại ta Được số Đặc biệt : 145. (các bạn tự chứng minh tính Đúng Đắn của cách làm).

Ta có một thuật toán cụ thể như sau: Dùng mảng 6x10 phần tử Để lưu thông tin xuất hiện của các chữ số từ 0 Đến 9 ở các cột từ 1 Đến 6 (do các số có giá trị trong Đoạn [0,65535] nên có nhiều nhất 6 cột). Phần tử (k,x) lưu số lần xuất hiện của chữ số x tại cột thứ k.

Đọc từ file INP lần lượt N số, với mỗi số, tăng số lần xuất hiện các chữ số ở cột tương ứng bằng cách thay Đổi mảng 6x10. Sau Đó dựa vào mảng lưu số lần xuất hiện này Để tìm ra số Đặc biệt.

Cách làm trên khá hay nhưng với dữ liệu lớn thì chương trình chạy rất lâu. Việc kết hợp xử lý bit vào thuật toán giúp chương trình chạy nhanh hơn nhiều.

Cách thứ hai: Dựa vào tư tưởng của cách làm trên, ta có thể chuyển mỗi số về dạng nhị phân và cũng sắp vào bảng như trên, Thực hiện phép XOR của số thứ 1 với số thứ 2, rồi lấy kết quả thực hiện tiếp với số thứ 3 ... cứ làm như vậy cho Đến khi thực hiện hết phép XOR với số thứ N. Vì phép XOR tương đương với việc loại trừ 2 bit giống nhau nên sau khi thực hiện xong, kết quả cho ta những bit có số lần xuất hiện là số lẻ và Đó cũng là dãy bit của số Đặc biệt cần tìm.

```
X:=0;  
For i:=1 to N do  
  Begin  
    Readln(fi,y);  
    x := x xor y;  
  End;  
Writeln(x);
```

Bài toán:

Bàn cờ thể (CHESSCBG)

Một bàn cờ thể là một bảng gồm 4 dòng, 4 cột. Mỗi thể cờ là một cách sắp xếp 8 quân cờ, hai quân khác nhau ở hai ô khác nhau. Bài toán Đặt ra là cho hai thể cờ 1 và 2, hãy tìm một số ít nhất bước di chuyển quân Để chuyển từ thể 1 sang thể 2; một bước di chuyển quân là một lần chuyển quân cờ sang ô trống kề cạnh với ô quân cờ Đang Đứng.

Dữ liệu: CHESSCBG.INP

- ✂ Gồm 8 dòng, mỗi dòng là một xâu nhị phân Độ dài 4 mà số 1/0 tương ứng với vị trí có hoặc không có quân cờ.
- ✂ Bốn dòng Đầu là thể cờ 1
- ✂ Bốn dòng sau là thể cờ 2.

Kết quả: CHESSCBG.OUT

- ✂ Gồm 1 dòng duy nhất là số bước chuyển quân ít nhất

Ví dụ:

CHESSCBG.INP	CHESSCBG.OUT
1111	4
0000	
1110	
0010	
1010	
0101	
1010	
0101	

Gợi ý: Mã hóa mỗi trạng thái bàn cờ bằng 1 dãy nhị phân 16 bit tương ứng. Sử dụng phương pháp loang Để tìm Đường Đi ngắn nhất từ trạng thái Đầu về trạng thái cuối.

Luyện tập:

Bài toán:

Tìm dòng Đặc biệt(SLINE)

Cho một tệp văn bản có n dòng ($n \leq 100000$) mỗi dòng có Độ dài không quá 255 ký tự. Trong Đó có một dòng chỉ xuất hiện một lần duy nhất, còn các dòng còn lại có số lần xuất hiện là một số chẵn.

Yêu cầu: hãy viết chương trình Để tìm dòng Đặc biệt Đó.

Dữ liệu: SLINE.INP

✂ Ghi các dòng thông tin và kết thúc tệp bởi dòng gồm 3 ký tự ###

Kết quả: SLINE.OUT

✂ Ghi ra dòng Đặc biệt tìm Được.

Ví dụ:

SLINE.INP	SLINE.OUT
-How are you?	-My name is PCM
-I'm fine	
-How are you?	
-My name is PCM	
-I'm fine	
###	

Bài toán:

Mê cung (MECUNG)

Một mê cung hình chữ nhật gồm M dòng và N cột ô vuông, $M, N \leq 100$. Các dòng (cột) Đánh số từ 1 Đến M (từ 1 Đến N) từ trên xuống (từ trái qua phải). Một robot Đứng ở ô (X,Y). Từ một ô bất kì, robot có thể di chuyển Đến 1 trong 4 ô kề cạnh. Mỗi ô của mê cung ứng với 1 số trong phạm vi 0..15 với ý nghĩa quy Định những hướng robot có thể di chuyển Đến. Mỗi hướng Được quy Định bởi 1 số hiệu: Trên: 1; Dưới: 2; Phải: 4; Trái: 8. Giá trị của ô bằng tổng các số hiệu các hướng mà robot có thể di chuyển Đến. VD ô (2,3)

có giá trị $13 = 1 + 4 + 8$ có nghĩa là từ ô (2,3) có thể di chuyển theo hướng trên, phải hay trái Để sang ô kề cạnh tiếp theo.

Cho thông tin về mê cung $M \times N$, vị trí ô xuất phát (X,Y).

Yêu cầu: Tìm Đường Đi ngắn nhất Để robot có thể thoát ra Được mê cung.

Dữ liệu: MECUNG.INP

✂ Dòng Đầu tiên ghi 4 số M,N,X,Y

✂ Tiếp theo ma trận số M dòng, mỗi dòng N cột biểu diễn thông tin về mê cung.

Kết quả: MECUNG.OUT

✂ Số bước di chuyển ít nhất Để robot thoát ra khỏi mê cung.

Bài toán:

Chuyến du lịch (TRIP)

Trong kì nghỉ hè năm nay sherry Được bố thưởng cho 1 tour du lịch quanh N Đất nước tươi Đẹp với nhiều thắng cảnh nổi tiếng (vì sherry rất ngoan). Tất nhiên sherry sẽ Đi bằng máy bay.

Giá vé máy bay từ Đất nước i Đến Đất nước j là C_{ij} (dĩ nhiên C_{ij} có thể khác C_{ji}). Tuy Được bố thưởng cho nhiều tiền Để Đi du lịch nhưng sherry cũng muốn tìm cho mình 1 hành trình với chi phí rẻ nhất có thể Để dành tiền mua quà về tặng mọi người (Các chuyến bay của sherry Luôn Được Đảm bảo an toàn tuyệt Đối).

Bạn hãy giúp sherry tìm 1 hành trình Đi qua tất cả các nước, mỗi nước Đúng 1 lần sao cho chi phí là bé nhất nhé.

Dữ liệu: TRIP.INP

✂ Dòng 1: N ($5 < N < 16$)

✂ Dòng thứ i trong N dòng tiếp theo: Gồm N số nguyên, số thứ j là C_{ij} ($0 < C_{ij} < 10001$)

Kết quả: TRIP.OUT

✂ Gồm 1 dòng duy nhất ghi chi phí bé nhất tìm Được

Ví dụ:

TRIP.INP	TRIP.OUT
6	8
0 1 2 1 3 4	
5 0 3 2 3 4	
4 1 0 2 1 2	
4 2 5 0 4 3	
2 5 3 5 0 2	
5 4 3 3 1 0	

Gợi ý: Gọi $F[i,x]$ là chi phí ngắn nhất khi Đang ở Đất nước i và các Đất nước Đã Đi qua Được Đánh dấu bằng dãy bit của biến x. Sử dụng tư tưởng Quy hoạch Động giải quyết.

VẤN ĐỀ: QUY HOẠCH ĐỘNG TRẠNG THÁI

Với một bài toán cụ thể, Để tìm ra Được lời giải cần Đi từ bước 1 Đến bước thứ k. Trong Đó tại bước thứ i nảy sinh nhiều trạng thái hướng Đi khác nhau, Để tìm ra hướng Đi Đúng cho bước thứ i lại dựa vào trạng thái của bước Đi thứ i-1 và có thể xác Định bằng một công thức cụ thể nếu tập các trạng thái là hữu hạn.

Bài toán:

Chọn ô - HSG QG 2006 (SELECT)

Cho một bảng hình chữ nhật kích thước $4 \times n$ ô vuông. Các dòng Được Đánh số từ 1 Đến 4, từ trên xuống dưới, các cột Được Đánh số từ 1 Đến n từ trái qua phải.

Ô nằm trên giao của dòng i và cột j Được gọi là ô (i,j). Trên mỗi ô (i,j) có ghi một số nguyên a_{ij} , $i = 1, 2, 3, 4$; $j = 1, 2, \dots, n$. Một cách chọn ô là việc xác Định một tập con khác rỗng S của tập tất cả các ô của bảng sao cho không có hai ô nào trong S có chung cạnh. Các ô trong tập S Được gọi là ô Được chọn, tổng các số trong các ô Được chọn Được gọi là trọng lượng của cách chọn. Tìm cách chọn sao cho trọng lượng là lớn nhất.

Ví dụ: Xét bảng với $n=3$ trong hình vẽ dưới Đây:

	1	2	3
1	-1	9	3
2	-4	5	-6
3	7	8	9
4	9	7	2

Cách chọn cần tìm là tập các ô $S = \{(3,1), (1,2), (4,2), (3,3)\}$ với trọng lượng 32. **Dữ**

liệu: SELECT.INP

- ✂ Dòng Đầu tiên chứa số nguyên dương n là số cột của bảng.
- ✂ Cột thứ i trong số 4 dòng tiếp theo chứa n số nguyên là n số trên hàng i của bảng.

Kết quả: SELECT.OUT

- ✂ Gồm 1 dòng duy nhất là trọng lượng của cách chọn tìm Được.

Hạn chế:

- ✂ Trong tất cả các test: $n \leq 10000$.
- ✂ $|a_{ij}| \leq 30000$.

Ví dụ:

SELECT.INP	SELECT.OUT
3 -1 9 3 -4 5 -6 7 8 9 9 7 2	32

Gợi ý: Mỗi cột có 4 dòng nên có thể dùng biến x có 4 bit nhị phân Để mô tả trạng thái chọn của cột i : bit thứ k bằng 1 (0) tương ứng với ô dòng thứ k của cột Được (không Được) chọn.

Gọi $F[i, x]$ là trọng lượng lớn nhất nếu xét từ cột 1 Đến cột i và trạng thái chọn của cột i Được biểu diễn bằng biến x . Công thức Quy hoạch Động là:

$$F[i, x] = \max (F[i-1, x'] + \text{sum}(i, x))$$

Trong Đó

- x và x' là hai trạng thái chọn của 2 cột liên tiếp nhau (i và $i-1$) Do Đó 2 trạng thái phải thỏa mãn Điều kiện không có 2 ô nào Được chọn kề nhau.
- $\text{sum}(i, x)$ là trọng lượng ứng với trạng thái chọn x của cột i .

Bài toán:

Cúm gia cầm (H5NX)

Đại dịch cúm gia cầm H5Nx Đã tấn công Đất nước Peace. Các cơ quan y tế Đã nhanh chóng cách ly các gia cầm bệnh. Theo các nghiên cứu khoa học cho thấy nếu con vật nào có chứa Đoạn mã gen HNH hoặc HHH sẽ có nguy cơ nhiễm bệnh rất cao cần phải cách ly Để theo dõi. Các chủng gia cầm trong nước Luôn có cấu tạo từ 2 loại gen H và N, vì vậy bộ y tế cần biết số lượng gia cầm khỏe mạnh (không có gen bệnh) Để tiêm phòng bệnh.

Tất cả các gia cầm Luôn có Độ dài gen bằng nhau là M , hãy viết chương trình giúp bộ y tế tính ra số lượng gia cầm khỏe mạnh.

Dữ liệu: H5NX.INP

✂ Gồm không quá 1000 dòng, mỗi dòng là một số nguyên M .

Kết quả: H5NX.OUT

✂ Với mỗi giá trị M xuất ra một số duy nhất Q là kết quả mà bạn tìm Được. Vì kết quả có thể rất lớn nên chỉ cần xuất $Q \bmod 2005$.

Giới hạn:

- ✂ $1 \leq M \leq 1.000.000$
- ✂ Thời gian: 0.5 s/test

Ví dụ:

H5NX.INP	H5NX.OUT
3	6

Gợi ý: Gọi $F[i, x]$ là số lượng các Đoạn gen :

- có Độ dài i .
- 2 gen cuối cùng có trạng thái bit là x . Hay x gồm 2 bit nhị phân thể hiện 2 gen cuối cùng (bit 1 tương ứng với gen H và bit 0 tương ứng với gen N).
- Không chứa các Đoạn HNH, HHH.

Bài toán:

Chuyến du lịch (TRIP)

Trong kì nghỉ hè năm nay sherry Được bố thưởng cho 1 tour du lịch quanh N Đất nước tươi Đẹp với nhiều thắng cảnh nổi tiếng (vì sherry rất ngoan). Tất nhiên sherry sẽ Đi bằng máy bay.

Giá vé máy bay từ Đất nước i Đến Đất nước j là C_{ij} (dĩ nhiên C_{ij} có thể khác C_{ji}). Tuy Được bố thưởng cho nhiều tiền Để Đi du lịch nhưng sherry cũng muốn tìm cho mình 1 hành trình với chi phí rẻ nhất có thể Để dành tiền mua quà về tặng mọi người (Các chuyến bay của sherry Luôn Được Đảm bảo an toàn tuyệt Đối).

Bạn hãy giúp sherry tìm 1 hành trình Đi qua tất cả các nước, mỗi nước Đúng 1 lần sao cho chi phí là bé nhất nhé.

Dữ liệu: TRIP.INP

- ✂ Dòng 1: N ($5 < N < 16$)
- ✂ Dòng thứ i trong N dòng tiếp theo: Gồm N số nguyên, số thứ j là C_{ij} ($0 < C_{ij} < 10001$)

Kết quả: TRIP.OUT

- ✂ Gồm 1 dòng duy nhất ghi chi phí bé nhất tìm Được

Ví dụ:

TRIP.INP	TRIP.OUT
6	8
0 1 2 1 3 4	
5 0 3 2 3 4	
4 1 0 2 1 2	
4 2 5 0 4 3	
2 5 3 5 0 2	
5 4 3 3 1 0	

Gợi ý: Gọi $F[i,x]$ là tổng chi phí nếu sherry Đang ở Đất nước i và trạng thái các Đất nước Đã Đi qua Được lưu vào biến x : gồm N bit nhị phân 0,1 (bit thứ k có giá trị bằng 1/0 có ý nghĩa Đã Đi qua Đất k hay chưa). Công thức Quy hoạch Động:

$$F[i,x] = \min(F[i',x'] + C[i',i])$$

Trong Đó

- i' là Đất nước Đã Được Đánh dấu Đi qua trong x.
- x' là trạng thái giống như trạng thái x nhưng Đất nước i chưa Được Đánh dấu.

Luyện tập:

Bài toán:

Đàn bò hỗn loạn(MIXUP)

Mỗi trong N cô bò ($4 \leq N \leq 16$) của bác John có một số seri phân biệt S_i ($1 \leq S_i \leq 25,000$). Các cô bò tự hào Đến nỗi mỗi cô Luôn Đeo một chiếc vòng vàng có khắc số seri của mình trên cổ theo kiểu các băng Đảng giang hồ.

Các cô bò giang hồ này thích nổi loạn nên Luôn xếp hàng chờ vắt sữa theo một thứ tự gọi Được gọi là 'hỗn loạn'.

Một thứ tự bò là 'hỗn loạn' nếu trong dãy số seri tạo bởi hàng bò, hai số liên tiếp khác biệt nhau nhiều hơn K ($1 \leq K \leq 3400$). Ví dụ, nếu $N = 6$ và $K = 1$ thì 1, 3, 5, 2, 6, 4 là một thứ tự 'hỗn loạn' nhưng 1, 3, 6, 5, 2, 4 thì không (vì hai số liên tiếp 5 và 6 chỉ chênh lệch 1).

Yêu cầu: Cho biết có bao nhiêu cách khác nhau Để N cô bò sắp thành thứ tự 'hỗn loạn'? **Dữ**

liệu: MIXUP.INP

- ✂ Dòng 1: Hai số N và K cách nhau bởi khoảng trắng
- ✂ Dòng 2..N+1: Dòng i+1 chứa một số nguyên duy nhất là số seri của cô bò thứ i: S_i

Kết quả: MIXUP.OUT

- ✂ Dòng 1: Một số nguyên duy nhất là số cách Để N cô bò sắp thành thứ tự 'hỗn loạn'. Kết quả Đảm bảo nằm trong phạm vi kiểu số nguyên 64-bit.

Ví dụ:

MIXUP.INP	MIXUP.OUT
4 1	2
3	
4	
2	
1	

Gợi ý: Gọi $F[i,x]$ là số cách sắp các cô bò Được Đánh dấu trong biến x (là biến gồm N bit nhị phân) mà cô bò thứ i Được xếp sau cùng.

Bài toán: Cô gái chăn bò (COWGIRL)

Trên một thảo nguyên nhỏ bé có 1 gia Đình gồm 3 anh em: 2 người anh trai là Nvutri và Andorea còn người em gái là Lola. Cuộc sống gia Đình khá giả nhưng gia Đình có truyền thống chăn nuôi và muốn Để các con tự lập nên cha mẹ 3 người quyết Định Để các con hằng ngày sẽ Đi chăn 1 số bò nào Đó (tùy ý 3 người con).

Thảo nguyên là 1 cánh Đồng chia làm $M*N$ ô vuông, mỗi con bò chỉ Đứng trong 1 ô và mỗi ô chỉ chứa 1 con bò. Chỉ có 1 quy tắc duy nhất là không bao giờ Được Để 4 con bò tạo thành 1 hình vuông $2*2$ hoặc Để trống 1 khu Đất $2*2$.

Hai người anh mãi chơi nên Đã hồi lộ kem Để Lola chăn bò 1 mình. Lola muốn biết tất cả có bao nhiêu cách xếp bò thỏa mãn quy tắc trên Để Đề phòng mọi trường hợp. Vì con số này rất lớn nên hãy giúp Lola tính toán con số này.

Dữ liệu: COWGIRL.INP

- ✂ Dòng Đầu gồm 1 số T duy nhất là số test ($T \leq 111$)
- ✂ T dòng tiếp theo gồm 2 số M, N cho biết kích thước của thảo nguyên ($M*N \leq 30$)

Kết quả: COWGIRL.OUT

- ✂ Gồm T dòng, mỗi dòng ứng với 1 test là số cách xếp bò của test Đó.

Ví dụ:

COWGIRL.INP	COWGIRL.OUT
1	2
1 1	

Gợi ý: cho $M*N \leq 30$ nên tồn tại một số M hay N không lớn hơn 5. giả sử là M là số dòng không quá 5. Ta có thể sử dụng số có không quá 5 bit nhị phân Để mô tả trạng thái chọn ở mỗi cột.

.....

VẤN ĐỀ: QUY HOẠCH ĐỘNG VỊ TRÍ CẤU HÌNH

Cho một dãy các cấu hình thỏa mãn Điều kiện nào Đó Được sắp theo thứ tự từ Điển (D). Bài toán thường có 2 yêu cầu:

- Cho cấu hình (c), hỏi (c) nằm ở vị trí nào trong từ Điển.
- Cho vị trí k, yêu cầu tìm cấu hình (c) ứng với vị trí k trong từ Điển.

Tư tưởng chính Để giải quyết yêu cầu 1 là tìm số lượng cấu hình có vị trí nằm trước (c) trong từ Điển. Từ Đó suy ra vị trí của (c).

Tư tưởng chính Để giải quyết yêu cầu 2 là thu hẹp dần phạm vi từ Điển (lần lượt bỏ Đi những cấu hình (c) có vị trí bé hơn k, tạo thành một từ Điển mới Đồng thời giảm chỉ số k cho tương ứng với từ Điển mới).

Bài toán:

Dãy nhị phân (1).

Cho từ Điển bao gồm tất cả các dãy N bit nhị phân. VD với N=3 ta có từ Điển dãy bit nhị phân là: 000,001,010,011,100,101,110,111.

- Cho một dãy (c) gồm N bit nhị phân, hỏi (c) nằm ở vị trí nào trong từ Điển.
- Cho một số k, hỏi dãy bit nhị phân nào ở vị trí k trong từ Điển.

Gợi ý: Thực ra vị trí của (c) trong từ Điển cũng chính là dạng thập phân của (c) cộng thêm

1. Nhưng Để làm quen với tư tưởng Quy hoạch Động vị trí cấu hình, ta tạm thời quên Đi cách giải Đơn giản này và sử dụng một tư tưởng khác như sau:

Gọi $F[i]$ là số lượng các dãy i bit nhị phân.

Để thấy $F[i] = 2 * F[i-1]$ (hay $F[i] = 2^i$). $F[0] = 1$ Giả

sử (c): $a_1 a_2 \dots a_n$

Ta chỉ quan tâm Đến các bit 1 của (c), giả sử bit 1 xuất hiện ở những vị trí t

(c): $a_1 a_2 \dots a_{k-1} 1 a_{k+1} \dots a_n$

Khi Đó (c) sẽ có vị trí lớn hơn vị trí của những cấu hình dạng $x_1 x_2 \dots x_{t-1} 0 a_{t+1} \dots a_n$ (trong Đó x_i nhận giá trị 0 hoặc 1). Hay nói cách khác sẽ có $F[t-1]$ cấu hình dạng $x_1 x_2 \dots x_{t-1} 0 a_{t+1} \dots a_n$ có vị trí bé hơn t.

Gọi s là số lượng các cấu hình có vị trí bé hơn t

$$s = \sum_{i=1}^{st} F[t_i - 1], \text{ trong Đó } k_i \text{ là các vị trí xuất hiện bit 1 trong (c).}$$

Suy ra vị trí của (c) trong từ Điển là s+1.

```
S:=0;  
For t:=1 to n do  
  If c[t]=1 then s:=s + F[t-1]  
Writeln('vị trí ',s+1);
```

Để tìm cấu hình (c) ở vị trí k, ta duyệt lần lượt các bit của (c) từ bit thứ 1 Đến N. Nếu bit thứ t nhận giá trị 1 hay (c): $a_1 a_2 \dots a_{t-1} 1 \dots$ thì rõ ràng (c) có vị trí lớn hơn vị trí của các cấu hình dạng $a_1 a_2 \dots a_{t-1} 0 x_{t+1} \dots x_n$ (trong Đó x_i nhận giá trị 0 hoặc 1). Hay nói cách khác k sẽ lớn hơn $F[n-t]$. Vậy

- nếu $k \leq F[n-t]$ thì bit thứ t của (c) nhận giá trị 0
- nếu $k > F[n-t]$ thì bit thứ t của (c) nhận giá trị 1, Đồng thời gán lại $k = k - F[n-t]$

```
for t:=1 to n do
  if k<=f[n-t] then c[t]:=0
  else
    begin
      c[t]:=1;
      k:=k - F[n-t];
    end;
For i:=1 to n do write(c[i])
```

Bài toán: **Dãy nhị phân (NP2)**

Một tập hợp S gồm các dãy N bit 0, 1 trong Đó không có hai bit 1 nào kề nhau. Ví dụ với N = 5 thì S gồm các dãy 00000, 00001, 000101,... Tập S Được sắp xếp theo chiều tăng dần của số nguyên tương ứng mà dãy bit biểu diễn. Cho số N và một số nguyên M hãy cho biết dãy bit thứ M trong S.

Dữ liệu: NP2.INP

✂ Dòng duy nhất ghi hai giá trị N, M ($N \leq 40$, M Đảm bảo có nghiệm)

Kết quả: NP2.OUT

✂ Dãy N số 0, 1 ghi liền nhau mô tả dãy nhị phân tìm Được.

Ví dụ:

NP2.INP	NP2.OUT
5 3	000101

Gợi ý:

Gọi $F[i]$ là số phần tử của tập S ứng với $N = i$.

- $F[i] = F[i-1] + F[i-2]$
- $F[0] = F[1] = 1$

Cách xây dựng dãy bit thứ M tương tự như bài trên.

Bài toán: **Số hiệu hoán vị (SHHV)**

Xét tất cả các hoán vị của dãy số tự nhiên $(1, 2, \dots, n)$ ($1 \leq n \leq 12$) Giả sử rằng các hoán vị Được sắp xếp theo thứ tự từ Điển.

Yêu cầu:

- ✂ Cho trước 1 hoán vị. Tìm số hiệu của hoán vị Đó trong dãy Đã sắp xếp
- ✂ Cho trước số hiệu của 1 hoán vị trong dãy hoán vị Đã sắp xếp. Tìm hoán vị Đó

Dữ liệu: SHHV.INP

- ✂ Dòng 1: Chứa n số a_1, a_2, \dots, a_n (dãy hoán vị n phần tử)
- ✂ Dòng 2: Chứa số p (số hiệu của hoán vị trong dãy hoán vị n phần tử)

Kết quả: SHHV.OUT

- ✂ Dòng 1: Ghi số q (số hiệu của dãy hoán vị a_i)
- ✂ Dòng 2: Ghi n số b_1, b_2, \dots, b_n (dãy hoán vị có số hiệu p)

Ví dụ:

SHHV.INP	SHHV.OUT
2 1 3 4	3 2 3 1

Gợi ý: Gọi $F[i]$ là số lượng các hoán vị có i chữ số

- $F[i] = F[i-1] * i$ (hay $F[i] = i!$).
- $F[1] = 1$

$Fin(i)$ là hàm cho biết số lượng các số trong Đoạn $a_{i+1} \dots a_n$ và có giá trị bé hơn $a[i]$ Yêu cầu 1:

```
S:=0;  
For i:=1 to n do  
    s:= s + fin(i)*f[n-1]  
writeln(s+1)
```

Yêu cầu 2:

```
fillchar(fr,sizeof(fr),true);  
for i:=1 to n do  
    begin  
        for j:=1 to n do  
            if fr[j] then  
                if (k>f[n-i]) then dec(k,f[n-i]) else break;  
                a[i]:=j; fr[j]:=false;  
        end;  
    for i:=1 to n do write(fo,a[i],' ');
```

Luyện tập:

Bài toán:

Liên lạc vũ trụ (IMPULSE)

Để liên lạc với tàu thăm dò tự Động ngồi ta chuẩn bị danh sách các thông báo, Đánh số từ 1 trở Đi và cài vào trong bộ nhớ của máy tính trên trạm thăm dò. Số lượng thông báo là 1000000. Trạm Điều khiển mặt Đất hoặc tàu thăm dò chỉ cần phát Đi số tự thay vì cho chuyển bằng hệ thống phát xung Laser Định hướng. Nhưng việc phát xung (tức là các tín hiệu 1). Vì vậy các nhà khoa học quyết Định phát mỗi số ứng với một dãy bit có không quá 3 số 1. Các dãy bit có cùng Độ dài là 200, Được sắp xếp lại theo thứ tự tăng dần của giá trị nhị phân tương ứng. Số thứ tự của dãy bit trong danh sách sẽ chính là số nguyên cần gửi.

Yêu cầu: Hãy lập trình cài vào máy phát chuyển Đổi từ giá trị số sang xâu bit cần phát. **Dữ**

liệu: IMPULSE.INP

- ✂ Dòng Đầu tiên là số lượng thông báo cần phát R ($R \leq 10000$).
- ✂ Các dòng sau chứa các số nguyên dương (số thứ tự thông báo), các số nếu ở trên 1 dòng - cách nhau ít nhất 1 dấu cách.

Kết quả: IMPULSE.OUT

✂ Xâu bit ứng với số cần phát . Bỏ qua các số 0 trước 1 Đầu tiên trong xâu , trừ trường hợp số cần phát là 1 thì kết quả ra Được ghi là 0.

Ví dụ:

IMPULSE.INP	IMPULSE.OUT
6	0
1 2 3 4 5 100	1
	10
	11
	100
	100000110

Bài toán:

Đánh số các tập con (SUBSET)

Giả sử A là tập N số nguyên dương Đầu tiên, $N \leq 30$. Với mỗi tập con B của A, ta luôn viết các phần tử của B theo giá trị tăng dần: $B = \{b_1, b_2, \dots, b_K\}$ với $b_1 < b_2 < \dots < b_K$

Ta xếp thứ tự các tập con của A theo nguyên tắc sau: giả sử B và C là hai tập con của A: $B = \{b_1, b_2, \dots, b_K\}$, $C = \{c_1, c_2, \dots, c_M\}$, $B < C$ nếu có $i \leq \min(K, M)$ sao cho $b_i = c_1, \dots, b_{i-1} = c_{i-1}$, và $b_i < c_i$ hoặc $b_i = c_i$ và $K = \min(K, M)$. Trên cơ sở xếp thứ tự các tập con, ta Đánh số các tập con của A từ 1 Đến 2^N , tập rỗng Được Đánh số 1.

Ví dụ với $n = 3$, thứ tự các tập hợp là:

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

Yêu cầu:

- ✂ Cho số N và một số hiệu $S > 1$, tìm tập B có số hiệu S.
- ✂ Cho số N và tập C không trống, tìm số hiệu của C.

Dữ liệu: SUBSET.INP

- ✂ Dòng Đầu tiên ghi 2 số nguyên dương N, S
- ✂ Dòng tiếp theo ghi N, tiếp theo là dãy K số c_1, c_2, \dots, c_K

Kết quả: SUBSET.OUT

- ✂ Dòng Đầu là chỉ số các phần tử trong tập B ứng với số hiệu S. ✂
- Dòng thứ hai ghi số hiệu ứng với tập C.

Ví dụ:

SUBSET.INP	SUBSET.OUT
3 8	3
3 2 3	7

Bài toán:

Chuỗi hạt (NLACE)

Khi tiến hành khai quật khảo cổ ở một vương quốc xa xưa nọ, các nhà khoa học khai quật Được rất nhiều chuỗi hạt lạ. Sau khi quan sát, các nhà khoa học thấy rằng các chuỗi hạt có một số Đặc Điểm chung.

Mỗi chuỗi hạt là một sợi dây Được Đính các hạt ngọc làm bằng một chất liệu cổ xưa. Các chuỗi hạt đều có số lượng hạt ngọc bằng nhau. Hơn nữa, mỗi hạt ngọc là một hình cầu có Đường kính là một số nguyên dương, và nếu lần từ trái sang phải trên chuỗi hạt, người ta thấy các hạt ngọc có Đường kính tăng dần. Nếu Đánh số vị trí các hạt ngọc bắt Đầu từ 1, theo thứ tự từ trái sang phải, người ta nhận thấy rằng hạt ngọc thứ i có Đường kính không vượt quá $2i$. Các nhà khoa học cho rằng, dân tộc cổ xưa này hẳn Đã làm ra tất cả các chuỗi hạt có cùng những Đặc Điểm này, dù chúng hiện còn Đang rải rác ở Đâu Đó trên trái Đất.

Sau Đó không lâu, các nhà khoa học tìm ra một mảnh da, trên Đó có ghi một con số theo loại chữ số cổ xưa. Họ cho rằng mảnh da này có liên quan Đến các chuỗi hạt kỳ lạ nọ. Sau nhiều cố gắng, các nhà khoa học Đã Đưa Được con số trên mảnh da về hệ chữ số thập phân, và ký hiệu là X .

Manh mỗi Đến Đây thì dừng lại, vì các nhà khoa học không tìm thấy Được vết tích nào khác nữa, và cũng không tìm ra Được mối quan hệ giữa X và các chuỗi hạt. Đến Đây, một nhà khoa học người Việt Đề nghị, hãy thử xác Định chuỗi hạt có thứ tự từ Đầu là X , biết Đầu Đây sẽ là manh mối?

Yêu cầu: Bạn hãy viết chương trình giúp nhà khoa học xác Định chuỗi hạt có thứ tự từ Đầu là X .

Dữ liệu: NLACE.INP

- ✂ Dòng 1: chứa số nguyên dương N , là số hạt ngọc trong mỗi chuỗi hạt.
- ✂ Dòng 2: chứa số nguyên dương X .

Kết quả: NLACE.OUT

- ✂ Gồm 1 dòng duy nhất, chứa N số nguyên, cách nhau một khoảng trắng, xác Định chuỗi hạt có thứ tự từ Đầu là X (Để biểu diễn một chuỗi hạt, cần in ra N số nguyên tương ứng là Đường kính của các hạt ngọc trong chuỗi hạt, theo thứ tự từ trái sang phải).

Giới hạn:

- ✂ N là số nguyên dương trong phạm vi $[1, 250]$.
- ✂ X là số nguyên dương trong phạm vi từ 1 Đến số lượng tối Đa các chuỗi hạt.

Ví dụ:

NLACE.INP	NLACE.OUT
2	2 3
4	

Giải thích: các chuỗi hạt sắp theo thứ tự từ Đầu lần lượt là 1 2, 1 3, 1 4, 2 3, 2 4. Chuỗi hạt thứ 4 là 2 3.

Bài toán:

Thứ tự xâu (STRING)

Các chữ cái trong bộ chữ cái tiếng Anh a..z Được Đánh số từ 1 Đến 26.

Một xâu $W = c_1 \dots c_{i-1}c_i$ chỉ gồm các chữ cái thường tiếng Anh khác nhau Được gọi là một từ Đúng nếu trong xâu Đó, ký tự c_K Đứng sau ký tự c_{K-1} trong bộ chữ cái với mọi $1 < K \leq i$.

Với thứ tự này ta có thể Đánh số các từ theo nguyên tắc sau:

- Xem mỗi chữ cái là một từ Độ dài 1 với số hiệu như trên
- Chia mọi từ thành các nhóm $W(1), W(2), \dots, W(K)$ trong Đó $W(i)$ là nhóm gồm mọi từ có Độ dài i
- Trong mỗi nhóm, ta xếp các từ theo thứ tự từ Điển
- Lần lượt Đánh số liên tiếp từ 1, nhóm có Độ dài bé Đánh số trước, trong mỗi nhóm, thứ tự Đánh số là thứ tự từ Điển

Ví dụ về số hiệu một số từ: ac - 28; bc - 52; vwxyz - 83681

Cho một xâu S chỉ gồm các chữ cái tiếng Anh thường. Nói chung S không là từ nhưng ta có thể cắt S một cách duy nhất thành một dãy từ sao cho không thể ghép hai từ kề nhau thành một từ. Khi Đó ta có thể mã hóa S thành một dãy số nguyên dương $C(S)$ mà các số hạng của dãy này tương ứng là các số hiệu của các từ của dãy từ nhận Được. Ví dụ: nếu $S = abazvwxyz$ thì $C(S) = 27\ 51\ 83681$

Yêu cầu:

- ✂ Cho một xâu S Độ dài không quá 1000, tìm dãy $C(S)$.
- ✂ Cho một dãy $C(S')$ là mã hóa của một xâu S' thỏa mãn Điều kiện như yêu cầu 1, hãy khôi phục lại từ S' .

Dữ liệu: STRING.INP

- ✂ Dòng thứ nhất ghi từ Đầu dòng xâu S .
- ✂ Dòng thứ hai ghi dãy $C(S')$, hai số liên tiếp cách nhau ít nhất một dấu trống.

Kết quả: STRING.OUT

- ✂ Dòng thứ nhất ghi dãy $C(S)$.
- ✂ Dòng thứ hai ghi xâu S' .

Ví dụ:

STRING.INP	STRING.OUT
abazvwxyz	27 51 83681
27 51 83681	abazvwxyz

Bài toán:

Mã vạch (BARCODE)

Một mã vạch (thường dùng Để ghi giá trị hàng hoá) là một dãy các vạch Đen trắng xen kẽ nhau bắt Đầu bằng vạch Đen bên trái , mỗi vạch có một Độ rộng tính bằng số nguyên dương . Hình sau cho ví dụ về một mã 4 vạch trái trên $1+2+3+1 = 7$ Đơn vị rộng:



Tổng quát , một tập mã vạch $Mv(n,k,m)$ là tập mọi mã vạch gồm k vạch trái trên Đúng n Đơn vị rộng và mỗi vạch không rộng quá m Đơn vị .

Mã vạch trong ví dụ trên thuộc tập $Mv(7,4,3)$ nhưng không thuộc tập $Mv(7,5,2)$. ta có thể biểu diễn mỗi mã vạch bởi một dãy nhị phân bằng cách dùng số 1 biểu thị một Đơn vị rộng màu Đen và số 0 biểu thị một Đơn vị rộng màu trắng . Ví dụ mã vạch trong hình trên sẽ Được biểu thị bởi dãy 1001110

Với biểu diễn nhị phân như vậy mỗi tập $Mv(n,k,m)$ với các giá trị cụ thể của m, k, n là một tập các xâu Độ dài n chỉ gồm các ký tự 0 hay 1 . Ta có thể liệt kê tập Đó theo thứ tự từ Điển với quy ớc ký tự 0 trước ký tự 1 và theo thứ tự liệt kê Đó ta cho mỗi mã vạch một số hiệu .

Ví dụ tập $Mv(7,4,3)$ gồm 16 mã vạch sẽ Được biểu diễn bởi 16 xâu nhị phân Độ dài 7 và theo cách sắp xếp từ Điển của các biểu diễn nhị phân ta có các mã vạch với các số hiệu từ 1 Đến 16 là :

01: 1000100	09: 1100100
02: 1000110	10: 1100110
03: 1001000	11: 1101000
04: 1001100	12: 1101100
05: 1001110	13: 1101110
06: 1011000	14: 1110010
07: 1011100	15: 1110100
08: 1100010	16: 1110110

Dữ liệu: BARCODE.INP

✂ Dòng thứ nhất ghi 3 số n, k, m ($1 \leq n, k, m \leq 33$) là ba tham số của tập mã vạch

✂ Dòng thứ hai ghi số s ($s \leq 50$)

✂ Trong s dòng tiếp theo ghi mỗi dòng một xâu gồm n ký tự 0 hay 1 là biểu nhị phân của một mã vạch thuộc tập $Mv(n,k,m)$.

Kết quả: BARCODE.OUT

✂ S dòng , mỗi dòng ghi số hiệu của mã vạch Đã cho

Ví dụ :

BARCODE.INP	BARCODE.OUT
7 4 3 16	5
5 5	16
1001110	4
1110110	5
1001100	1
1001110	
1000100	

Gợi ý: Xem mỗi vạch như một số nguyên dương có giá trị bằng Độ rộng của vạch Đó. Gọi $F[n,k,m]$ là số dãy mà:

- Mỗi dãy có k số
- Tổng các số trong mỗi dãy là n
- Mỗi số trong dãy có giá trị không lớn hơn m

Chuyển một mã vạch nhị phân sang dãy số tương ứng. Khi Đó số ở vị trí lẻ tương ứng với vạch 1 và số ở vị trí chẵn tương ứng với vạch 0. Dựa vào dãy số và mảng F Để xuất ra số hiệu của mã vạch.

Bài toán:

Dãy số Catalan (CATALAN)

Cho số nguyên dương N, dãy Catalan cấp n là dãy $C(1), C(2) \dots C(2n+1)$ gồm các số nguyên không âm thỏa mãn : $C(1) = C(2n+1) = 0$ với i bất kì $1 \leq i \leq 2n$ thì $C(i), C(i+1)$ hơn kém nhau 1 Đơn vị.

Với mỗi n ta sắp xếp các dãy Catalan theo thứ tự từ Điển, Đánh số từ 1 trở Đi .

Yêu cầu :

- ✂ Cho một dãy Catalan, hãy tìm thứ tự của dãy.
- ✂ Cho số nguyên dương k hãy tìm dãy có thứ tự k.

Dữ liệu: CATALAN.INP

- ✂ Dòng Đầu ghi n. ($n \leq 15$)
- ✂ Dòng hai ghi một dãy Catalan cấp n
- ✂ Dòng ba ghi một số nguyên dương k (k có thể rất lớn nhưng Đảm bảo luôn có nghiệm)

Kết quả: CATALAN.OUT

- ✂ Dòng 1 ghi số thứ tự dãy ở dòng hai của file dữ liệu ✂
- Dòng 2 ghi dãy ứng với số thứ tự k

Ví dụ:

CATALAN.INP	CATALAN.OUT
4	12
0 1 2 3 2 1 2 1 0	0 1 2 3 2 1 2 1 0
12	

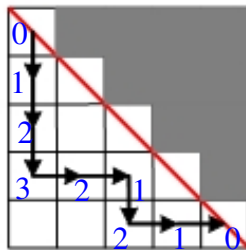
Gợi ý: Xét lưới vuông $(n+1) \times (n+1)$. Một quy tắc Đi thỏa mãn:

- xuất phát từ ô (1,1) Đến ô (n+1,n+1)
- Mỗi bước Đi chỉ Được di chuyển Đến ô kề cạnh bên phải hoặc bên dưới.
- Không Được di chuyển qua ranh giới là Đường chéo nối Đỉnh trái trên và phải dưới của lưới.

Nếu quy Định ô(1,1) tương ứng với số 0 và

- mỗi bước di chuyển sang phải, ta tạo số mới bằng số liền trước trừ Đi 1.
- mỗi bước di chuyển xuống dưới, ta tạo số mới bằng số liền trước cộng thêm 1.

Dễ thấy mỗi cách Đi từ ô(1,1) Đến (n+1,n+1) tương Đương với một dãy catalan tương ứng.



Cách Đi tương ứng với dãy $0\ 1\ 2\ 3\ 2\ 1\ 2\ 1\ 0$ ($n = 4$)

Gọi $F[i,j]$ là số Đường Đi từ ô (i,j) Đến ô $(n+1,n+1)$ - $F[i,j]$ Được tính bằng công thức Quy hoạch Động.

Giả sử tại ô (u,v) ta di chuyển xuống ô phía dưới thì cách Đi này sẽ có số thứ tự lớn hơn cách Đi từ ô (u,v) di chuyển sang ô bên phải (với $u > v$). Do Đó ta chỉ quan tâm Đến những ô (u,v) mà tại Đó thực hiện di chuyển xuống ô phía dưới, ta cộng vào s thêm $F[u,v+1]$. Kết quả số thứ tự của dãy catalan tương ứng với cách Đi là $s+1$.

VẤN ĐỀ: QUY HOẠCH ĐỘNG TRÊN CÂY

Cây là Đồ thị vô hướng Đặc biệt có những tính chất sau Đây:

- Liên thông và không có chu trình.
- Giữa 2 Đỉnh bất kỳ có duy nhất một Đường Đi Đơn.
- Nếu bỏ Đi 1 cạnh bất kì thì cây phân thành 2 cây con.

Thuật toán quy hoạch Động trên cây chủ yếu dựa vào các tính chất Đặc biệt trên của cây. Thông thường với Đồ thị cây, người ta cần xác Định một cây quan hệ cha - con Để thực hiện công việc quy hoạch Động. Việc xác Định này có thể sử dụng phép duyệt DFS, khi Đó những Đỉnh v Được thăm tiếp theo trong quá trình duyệt Đỉnh u sẽ nhận Đỉnh u làm cha.

Tư tưởng Quy hoạch Động là tìm thông tin của nút cha dựa vào thông tin của những nút con Đã tìm Được trước Đó. Các bạn sẽ hiểu thêm qua các bài toán cụ thể sau Đây:

Bài toán: Cho 1 cây , hãy tìm đường đi dài nhất trong cây đó .

<problem PT07Z(spoj.pl/problems/PT07Z)>

Input :

Dòng đầu tiên là n , số nút của cây ($n \leq 100000$)

n-1 dòng sau , mỗi dòng là 2 số u,v mô tả 1 cạnh của cây.

Output

Độ dài đường đi dài nhất trong cây đó .

Input

4

1 2

2 3

3 4

Output

3

Thuật Toán : Đầu tiên , coi nút 1 là nút gốc chẳng hạn ,Thực hiện duyệt từ lá về gốc ,gọi $f[i,1]$ là độ dài đường đi dài nhất mà tập đỉnh của nó là các nút con của i (không chứa đỉnh i), $f[i,2]$ là độ dài đường đi dài nhất mà trong tập đỉnh này chứa điểm i.

Công thức Qhd:

$$F[I,1]=\max(f[i1,1] , f[i1,2] , f[i2,1] , f[i2,2] , \dots, f[ik,1] , f[ik,2])$$

$$F[I,2]=\max(f[iu,2] + f[iv,2])$$

Trong đó :+ i1,i2,..ik là các nút con kề với nút i

+ Iu,iv là 2 trong số các nút con kề với nút I (iu <> iv)

Kết quả là $\max(f[1,1] , f[1,2])$;

Trong chương trình mẫu sau sẽ trình bày 1 thuật giải khác, đó là

+đầu tiên thực hiện DFS từ một đỉnh bất kỳ nào đó <đỉnh 1 chẳng hạn> giả sử đỉnh có khoảng cách lớn nhất đến đỉnh 1 là đỉnh k. Thực hiện DFS lần thứ 2 từ đỉnh k, thì kết quả là khoảng cách từ đỉnh có khoảng cách xa k nhất đến k.

Bài toán : *Cho 1 cây có trọng số , hãy tìm đường đi dài nhất trong cây đó*

Input :

Dòng đầu tiên là n , số nút của cây ($n \leq 100000$)

n-1 dòng sau , mỗi dòng là 2 số u,v,c mô tả 1 cạnh của cây.($c \leq 10000$)

Output

Độ dài đường đi dài nhất trong cây đó .

Input

4

1 2 1

2 3 1

3 4 1

Output

3

Thuật toán tương tự như bài trên , chú ý cần khai báo kết quả kiểu int64 .

Bài 5 Nhãn của cây <vn.spoj.pl/problems/ITREE>

Cho đồ thị cây có trọng số gồm N đỉnh , các đỉnh được đánh số từ 1 -> N . Gốc của cây là đỉnh 1 . Cha của đỉnh u là 1 đỉnh có số hiệu nhỏ hơn u . Mỗi đỉnh có một nhãn là 1 số thực $A[i]$. Trong đó nhãn của đỉnh 1 bằng 1 và nhãn của đỉnh lá bằng 0 . Biết rằng $A[v] \leq A[u]$ nếu v là con của u . Giá trị của 1 cây = Tổng (($A[u] - A[v]$) * Trọng số cạnh (u,v) , với u là cha của v) Bây giờ người ta cho biết các cạnh của đồ thị và trọng số của các cạnh này nhưng không cho biết các $A[i]$. Hãy tính xem giá trị của cây thấp nhất là bao nhiêu.

Input

Dòng 1 là số nguyên T là số bộ test . ($1 \leq T \leq 50$) . T nhóm dòng tiếp theo mô tả từng bộ test .
Mỗi bộ Test sẽ có cấu trúc như sau :

Dòng 1 : số nguyên dương N ($1 \leq N \leq 1000$) .

Từ dòng 2 -> dòng N : dòng thứ i gồm 2 số nguyên dương u và c ($1 \leq u < i$, $0 \leq c \leq 1000$) cho biết cha của nút i là nút u và cạnh nối (u,i) có trọng số là c .

Output

Với mỗi test ghi ra giá trị thấp nhất có thể đạt được của cây trên 1 dòng với độ chính xác là 2 chữ số sau dấu chấm.

Example

Input:

```
1
4
1 1
1 2
2 1
```

Output:

```
3.00
```

Thuật toán : tồn tại 1 cách gán nhãn trên cây thỏa mãn điều kiện bài toán và nhãn các đỉnh là 0 và 1

Từ đó suy ra công thức qhd sau

+ gọi $f[I,0]$ và $f[I,1]$ là giá trị nhỏ nhất của cây gốc I , nếu tất cả các nút con của I đã được gán nhãn hết và giá trị gán cho nút I tương ứng là 0,1

+công thức QHd

$F[I,0]$ = tổng giá trị các nút con của I và chúng đều được gán nhãn là 0 (do giá trị nút con phải bé hơn giá trị nút cha)

$$F[I,1] = \max(f[i1,0], f[i1,1]) + \dots + \max(f[ik,0], f[ik,1]);$$

Trong đó $i1, \dots, ik$ là các nút con kề với nút i

Bài toán: **Bữa tiệc vui vẻ (GUEST)**

Công ty trách nhiệm hữu hạn “ Vui vẻ “ có n cán bộ Đánh số từ 1 Đến N . Cán bộ i có Đánh giá Độ vui tính là H_i ($i = 1, 2 \dots N$) . Ngoài trừ giám đốc công ty , mỗi cán bộ có 1 thủ trưởng trực tiếp của mình .

Bạn cần giúp công ty mời một nhóm cán bộ Đến dự dạ tiệc “ Vui vẻ “ sao cho trong số những người Được mời không Đồng thời có mặt nhân viên và thủ trưởng trực tiếp và Đồng thời tổng Đánh giá vui tính của những người dự tiệc là lớn nhất .

Giả thiết rằng mỗi một thủ trưởng không có quá 20 cán bộ trực tiếp dưới quyền.

Dữ liệu: GUEST.INP

✂ Dòng Đầu tiên ghi số cán bộ của công ty ($1 < n < 1001$)

✂ Dòng thứ i trong số N Dòng tiếp theo ghi hai số nguyên dương T_i , V_i trong Đó T_i

là số hiệu của thủ trưởng trực tiếp và V_i là Độ vui tính của cán bộ i . Quy ước $T_i = 0$

nếu như số hiệu của giám Đốc công ti

Kết quả: GUEST.OUT

✂ Dòng Đầu tiên ghi hai số M và V . Trong Đó M là tổng số cán bộ Được mời còn V là tổng Độ vui về của các cán bộ Được mời

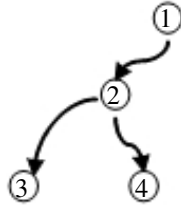
Dòng thứ K trong số M dòng tiếp theo , mỗi dòng ghi số hiệu các cán bộ Được mời

Ví dụ :

GUEST.INP GUEST.OUT

3	2 7
0 3	1
1 6	3
2 4	

Gợi ý: Có thể thấy sơ Đồ quan hệ trong công ty có dạng cây cha-con.



Gọi $F1[i]$ là tổng Độ vui tính của cách chọn nếu xét trong những người dưới quyền quản lý của i và i là người Được chọn.

Gọi $F0[i]$ là tổng Độ vui tính của cách chọn nếu xét trong những người dưới quyền quản lý của i và i là người không Được chọn.

Công thức Quy hoạch Động:

- $F1[i] = V_i$ nếu i không là xếp của ai
- $F0[i] = 0$ nếu i không là xếp của ai

Trường hợp còn lại

- $F1[i] = \sum F0[i'] + V_i$
- $F0[i] = \sum \max(F0[i'], F1[i'])$

Trong Đó i' là những nút nhận i làm cha trực tiếp.

Kết quả tổng Độ vui về lớn nhất là $\max(F1[j], F0[j])$ trong Đó j là số hiệu giám Đốc của công ty. Đưa những người Được chọn dựa vào mảng $F1$ và $F0$.

Bài toán: Tô màu nhỏ nhất (CTREE)

Cho một cây gồm N nút, hãy tìm cách gán mỗi Đỉnh một nhãn nguyên dương sao cho:

- + Hai nút có cạnh nối Được gán bởi hai nhãn khác nhau. +
- Tổng giá trị các nhãn là nhỏ nhất.

Dữ liệu:

- ✂ Dòng Đầu tiên ghi N ($1 \leq N \leq 10000$).
- ✂ $N-1$ dòng tiếp theo, mỗi dòng ghi hai nút là hai Đầu mút của một cạnh thuộc cây.

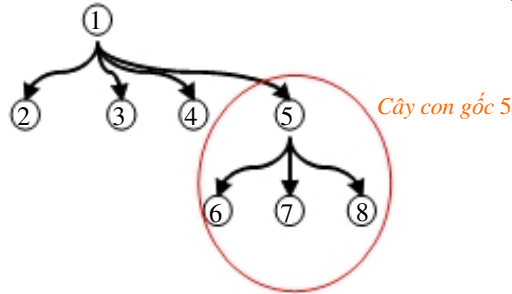
Kết quả:

- ✂ Dòng Đầu tiên ghi S là tổng giá trị nhãn tìm Được.
- ✂ N dòng tiếp theo, dòng thứ i ghi nhãn gán cho Đỉnh i trong phép gán tối ưu tìm Được.

Ví dụ:

CTREE.INP	CTREE.OUT
8	11
1 2	3
1 3	1
1 4	1
1 5	1
5 6	2
5 7	1
5 8	1
	1

Gợi ý: Cây Được cho dưới dạng danh sách cạnh, chưa có một xác Định cha-con cụ thể, do Đó có thể duyệt Đồ thị bằng DFS Để tạo một thứ tự cha-con bất kì cho cây.



Một thứ tự cha-con của cây với 1 là gốc

Để dàng chứng minh Được có thể dùng không quá 3 màu Để tô màu cây. Đánh số 3 màu Đó là 1,2,3. Gọi $F[i,x]$ là tổng giá trị khi tô cây con gốc i trong Đó nút i Được tô màu x ($1 \leq x \leq 3$).

Công thức Quy hoạch Động:

- $F[i,x] = x$ nếu i là nút lá (không có nút con nào).
- $F[i,x] = \sum_{i'=1}^k \max(F[C[i'],x'])$ nếu i không phải là nút lá.

Trong Đó

- nút i có k nút con trực tiếp $C[1], C[2], \dots, C[k]$
- $\max(F[C[i'],x'])$ là giá trị lớn nhất trong các $F[C[i'],x']$ với $1 \leq x' \leq 3$ và $x' \neq x$

Tư tưởng Được tiến hành bằng kĩ thuật xác Định cây quan hệ cha con kết hợp với Quy hoạch Động trong phép duyệt DFS Đồ thị:

Phép duyệt DFS Đỉnh i

Đánh dấu Đã thăm Đỉnh i

Nếu i không nối với Đỉnh nào chưa thăm thì $F[i,x] = x$ với $x:1 \rightarrow 3$

Ngược lại nếu i là Đỉnh nối với i' và chưa Được thăm

Gọi phép duyệt Đỉnh i - i' Đánh dấu là con của i

Tính $F[i,x]$ dựa vào các $F[i',x']$

Luyện tập:

Bài toán:

Cây P Đỉnh (PTREE)

Cho một cây gồm N Đỉnh , mỗi Đỉnh có 1 nhãn C[i] gọi là trọng số của Đỉnh i . Hãy tìm 1 cây con gồm P Đỉnh sao cho tổng trọng số của cây con này là lớn nhất . Hiểu 1 cách Đơn giản là tìm P Đỉnh sao cho P Đỉnh này liên thông và tổng trọng số là lớn nhất .

Dữ liệu: PTREE.INP

- ✂ Dòng 1 : 2 số nguyên dương N và P . ($1 \leq P \leq N \leq 200$) .
- ✂ Dòng 2 : N số nguyên dương C[1] , ... C[N] . ($-1000 \leq C[i] \leq 1000$) .
- ✂ N - 1 dòng tiếp theo , mỗi dòng gồm 2 số nguyên dương u , v mô tả 1 cạnh của Đồ thị .

Kết quả: PTREE.OUT

- ✂ Gồm 1 dòng ghi ra P số nguyên là chỉ số của P Đỉnh Được chọn .

Ví dụ:

PTREE.INP	PTREE.OUT
3 2	2 3
1 2 3	
1 2	
2 3	

Gợi ý: Quy hoạch Động trên cây 2 lần lồng nhau:

Gọi F[i,p] là tổng trọng số lớn nhất nếu xét trong cây con gốc i chọn p Đỉnh liên thông và i cũng là Đỉnh Được chọn. Để thấy cần chọn p Đỉnh mà Đỉnh i Đã Được chọn nên cần chọn thêm p-1 Đỉnh nữa ở các nhánh con của gốc i. Giả sử i có k nút con trực tiếp C₁,C₂...C_k. Bài toán tính F[i,p] lại Được chuyển thành bài toán tính tổng giá trị lớn nhất khi chọn p-1 Đỉnh ở k nhánh con. Gọi FF[ii,pp] là tổng trọng số lớn nhất khi chọn pp Đỉnh ở ii nhánh C₁,C₂...C_{ii} ($1 \leq ii \leq k, 1 \leq pp \leq p-1$). Quy hoạch Động Để tính FF[ii,pp]. Và F[i,p] = C[i] + FF[k,p-1].

Bài toán:

Xây cầu (BRIDGES)

Đất nước Delta là quốc Đảo lớn trên thế giới. Đất nước gồm N Đảo nhỏ Được Đánh số từ 1 Đến N. Việc Đi lại giữa các Đảo là rất khó khăn. Vì kinh tế còn rất kém phát triển, nhà nước phải khó khăn lắm mới mở Được N - 1 tuyến phà biển Để người dân người dân có thể Đi lại Được giữa hai Đảo bất kì. Cách Đây không lâu, Đất nước mới nhận Được sự Đầu tư lớn của các nước tư bản. Nhà vua quyết Định xây mới K cây cầu Để thay cho K tuyến phà. Các cây cầu mới Được xây dựng sẽ nối liền hai Đảo mà trước Đây có tuyến phà nối trực tiếp. Nhà vua muốn tính toán Để chọn K tuyến phà nào Để xây thành cầu sau cho

tổng thời gian Để Đi lại giữa mọi cặp Đỉnh là nhỏ nhất. Tức là:
$$\sum_{A=1}^{N-1} \sum_{B=A+1}^N T_{A,B}$$
 Đạt giá trị

nhỏ nhất. Trong Đó T_{A,B} là thời gian Đi từ Đảo A Đến Đảo B. Bạn hãy giúp nhà Vua tính toán chọn ra K trong số N - 1 tuyến phà Để thay thế bằng cầu.

Dữ liệu: BRIDGES.INP

- ✂ Dòng thứ nhất ghi 4 số nguyên N, K, VP, VC trong Đó VP là vận tốc nếu Đi bằng phà và VC là vận tốc nếu Đi bằng cầu. VP và VC có Đơn vị là m/s
- ✂ N - 1 dòng tiếp theo, mỗi dòng ghi 3 số U V L thể hiện giữa Đảo U và Đảo V Đã có một tuyến phà, và khoảng cách giữa U và V là L mét.

Kết quả: BRIDGES.OUT

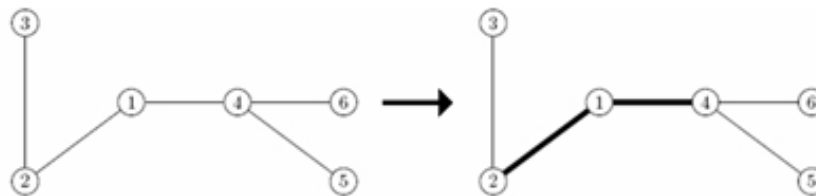
- ✂ In ra K số là số hiệu của tuyến phà cần Được thay thế bằng cầu.

Giới hạn:

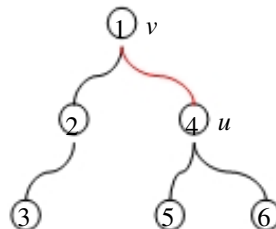
- ✂ $1 \leq K < N \leq 10\,000$
- ✂ $1 \leq VP, VC \leq 100\,000$
- ✂ $1 \leq L_{U,V} \leq 10^6$

Ví dụ:

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



Gợi ý: Mạng lưới các tuyến phà có dạng một Đồ thị cây.



$$F[4] = 3$$

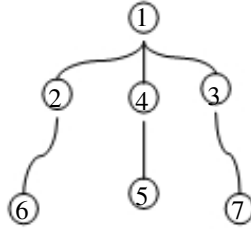
$$T(3,1) = 3 * (6-3) = 9$$

Với mỗi tuyến phà hay cạnh (u,v) trên cây, Định nghĩa $T(u,v)$ = số Đường Đi giữa 2 Đỉnh A và B có Đi qua cạnh (u,v) . Nếu bỏ cạnh (u,v) thì cây sẽ phân thành 2 cây con, một cây chứa u có F_u Đỉnh và cây còn lại chứa $(N - F_u)$ Đỉnh. Do Đó $T(u,v) = F_u * (N - F_u)$. Có thể tính $F[u]$ là số nút con của cây gốc u (các nút con của u bao gồm các u' là con của u và u'' là con của u' ...) bằng phương pháp Quy hoạch Động trên cây. Sau Đó tính Được $T(u,v)$ với (u,v) là một cạnh của cây Đồng thời v là cha trực tiếp của u. Cần tìm ra K cạnh (u,v) có $T(u,v)$ lớn nhất chính là các tuyến phà cần Được thay thế.

Bài toán:

Cây cân bằng (BALANCE)

Cho một cây T với N nút ($1 \leq N \leq 20000$) Được Đánh số từ 1 Đến N. Hai nút hoặc là nối với nhau bởi một cạnh duy nhất hoặc không nối với nhau. Xóa bất kì nút nào trong cây sẽ sinh ra một rừng: rừng là tập hợp một hoặc nhiều cây. Định nghĩa cây cân bằng của một nút là kích cỡ của cây lớn nhất trong rừng T Được tạo bởi bằng cách xóa nút T. VD cho một cây



Xóa nút 4 tạo ra hai cây với các nút của chúng là {5} và {1,2,3,6,7}. Cây lớn hơn trong hai cây có năm nút, do Đó cây cân bằng của nút 4 là năm...

Dữ liệu: BALANCE.INP

- ✂ Dòng Đầu tiên ghi số nguyên dương N.
- ✂ Mỗi dòng trong N dòng tiếp theo ghi hai số tương ứng với một cạnh trên cây.

Kết quả: BALANCE.OUT

- ✂ Dòng Đầu là số thứ tự nút có cân bằng nhỏ nhất.
- ✂ Dòng tiếp theo là cân bằng của nút Đó.

Bài toán:

Rải sỏi (STONE)

Xét trò chơi rải sỏi với một người chơi như sau: Cho cây T và một Đống sỏi gồm K viên. Ở mỗi bước người ta lấy 1 viên sỏi từ Đống sỏi và Đặt vào một nút lá tùy chọn. Nếu nút p có r nút lá và tất cả và tất cả các nút lá Đều có sỏi thì người ta gom tất cả các viên sỏi ở lá lại, Đặt 1 viên ở nút p, xóa các nút lá của nó và hoàn trả r - 1 viên sỏi còn lại vào Đống sỏi.

Trò chơi kết thúc khi Đã Đặt Được 1 viên sỏi vào nút gốc

Nhiệm vụ Đặt ra là theo cấu trúc của cây T, xác Định số viên sỏi tối thiểu ban Đầu Để trò chơi có thể kết thúc bình thường. Cây có n nút ($N \leq 400$), nút gốc Được Đánh số là 1.

Dữ liệu: STONE.INP

- ✂ Dòng Đầu: số n
- ✂ Dòng thứ i trong số n dòng tiếp theo có dạng: i m i_1 i_2 ... i_m . Trong Đó m là số nút con của nút i; i_1, i_2, \dots, i_m : Các nút con của nút i.

Kết quả: STONE.OUT

- ✂ Số lượng viên sỏi tối thiểu cần thiết

Ví dụ:

STONE.INP
7
1 2 2 3
2 2 5 4
3 2 6 7

STONE.OUT
3

Bài toán: Đoạn Đường cực trị (LUBENICA)

Mạng lưới giao thông ở 1 nước bao gồm N thành phố (Đánh số từ 1 Đến N) và N-1 Đường nối các thành phố với nhau. Có một Đường Đi duy nhất giữa mỗi cặp thành phố. Mỗi con Đường có một Độ dài xác Định.

Viết chương trình, với mỗi K cặp thành phố cho trước, tìm Độ dài của con Đường ngắn nhất và dài nhất trên Đường Đi giữa 2 thành phố này.

Dữ liệu: LUBENICA.INP

- ✂ Dòng Đầu tiên chứa số nguyên N, $2 \leq N \leq 100\,000$.
- ✂ Mỗi dòng trong số N-1 dòng tiếp theo chứa 3 số nguyên A, B, C cho biết có một con Đường Độ dài C giữa thành phố A và thành phố B. Độ dài của mỗi con Đường là số nguyên dương không vượt quá 1000000.
- ✂ Dòng tiếp theo chứa số nguyên K, $1 \leq K \leq 100\,000$.
- ✂ Mỗi dòng trong số K dòng tiếp theo chứa 2 số nguyên D và E - chỉ số của 2 thành phố cần truy vấn.

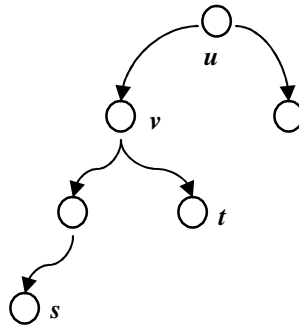
Kết quả: LUBENICA.OUT

- ✂ Mỗi dòng trong số K dòng chứa 2 số nguyên - Độ dài của con Đường ngắn nhất và dài nhất trên Đường nối giữa 2 thành phố tương ứng.

Ví dụ:

LUBENICA.INP	LUBENICA.INP	LUBENICA.INP
5	7	9
2 3 100	3 6 4	1 2 2
4 3 200	1 7 1	2 3 1
1 5 150	1 3 2	3 4 5
1 3 50	1 2 6	2 7 4
3	2 5 4	1 5 3
2 4	2 4 4	5 6 1
3 5	5	5 9 2
1 2	6 4	1 8 3
	7 6	5
	1 2	6 9
	1 3	7 8
	3 5	9 4
		1 2
		7 3
LUBENICA.OUT	LUBENICA.OUT	LUBENICA.OUT
100 200	2 6	1 2
50 150	1 4	2 4
50 100	6 6	1 5
	2 2	2 2
	2 6	1 4

Gợi ý: Chọn một nút u làm gốc và DFS thiết lập quan hệ cha-con trên cây.



Gọi $F[i]$ là Độ dài cạnh ngắn nhất trên Đường Đi từ i về gốc u

Gọi $SF[i]$ là tổng số cạnh có Độ dài bằng $F[i]$ trên Đường Đi từ i gốc u

Với 2 Đỉnh s, t tìm v là Đỉnh chung gần nhất trên Đường Đi từ s về u và Đường Đi từ t về u .

Có thể tính Được Độ dài Đoạn ngắn nhất c_1 trên Đường Đi từ s về v dựa vào $F[s]$, $SF[s]$ và $F[v]$, $SF[v]$.

Tương tự tính Được c_2 là Đoạn ngắn nhất trên Đường Đi từ t về s

Kết quả Đoạn ngắn nhất trên Đường Đi từ s Đến t là $\min(c_1, c_2)$

Thuật toán tương tự cho yêu cầu tìm Đoạn Đường dài nhất.

Bài toán:

Cây nhị phân (BTR)

Xét cây nhị phân, mỗi nút của cây có 2 nút con: nút trái và nút phải. Mỗi nút có một giá trị nguyên. Nếu nút cha có giá trị x , thì nút con trái có giá trị $2x$ và nút con phải có giá trị $2x+1$. Nút gốc của cây có giá trị 1.

Xuất phát từ nút, gốc Đường duyệt cây Đi thăm một nút nào Đó có thể mô tả bằng xâu chứa các ký tự từ tập $\{L,R,P\}$. Ký tự L nói lên rằng từ nút hiện tại ta sẽ Đi sang nút trái của nó trên Đường thăm, còn ký tự R - xác Định việc Đi sang nút phải. Ký tự P xác Định việc tạm dừng, tại bước này không chọn nhánh Đi tiếp. Ô cuối cùng Đi dẫn tới là ô Được thăm. Giá trị ô này Được gọi là giá trị thăm của Đường. Ví dụ Đường duyệt LR có giá trị thăm là 5, còn Đường RPP có giá trị thăm là 3.

Trong nhiều trường hợp, người ta phải duyệt cây và thăm một số nút. Để biểu diễn tập các Đường duyệt cây người ta sử dụng ký tự '*' Đại diện cho ký tự bất kỳ trong tập $\{L, R, P\}$. Ví dụ tập Đường duyệt L^*R sẽ xác Định 3 Đường duyệt: LLR, LRR và LPR với các giá trị thăm tương ứng là 9, 11 và 5. Như vậy tổng giá trị thăm của tập Đường này là $9 + 11 + 5 = 25$.

Yêu cầu: Cho xâu S Độ dài không quá 104 ký tự xác Định một Đường duyệt hoặc tập Đường duyệt. Hãy xác Định tổng giá trị thăm xác Định bởi s.

Dữ liệu: BTR.INP

✂ Gồm một dòng chứa xâu s

Kết quả: BTR.OUT

✂ Một số nguyên là tổng giá trị thăm

Ví dụ:

BTR.INP	BTR.OUT
L*R	25

Gợi ý:

```
F[0]:=1;
cc:=1;
for i:=1 to length(s) do
begin
  if s[i]='L' then F[i]:=F[i-1]*2
  else if s[i]='R' then F[i]:=F[i-1]*2 + cc
  else if s[i]='P' then begin end
  else
    begin
      F[i]:=F[i-1]*5 + cc;
      cc:=cc*3;
    end;
writeln(F[i])
```

VẤN ĐỀ: SẮP XẾP TOPO VÀ ỨNG DỤNG

Bài toán:

Sắp xếp Topo

Cho Đồ thị có hướng **không chu trình**. Tìm cách sắp xếp thứ tự các Đỉnh sao cho mọi cung (u,v) của Đồ thị thì Đỉnh u nằm trước Đỉnh v trong dãy đã sắp xếp. Hay nói cách khác là tìm cách Đánh số các Đỉnh của Đồ thị (các Đỉnh khác nhau có số khác nhau) sao cho mọi cung (u,v) đều nối Đỉnh có chỉ số bé hơn Đến Đỉnh có chỉ số lớn hơn.

Hướng giải quyết:

Cách làm thứ nhất: trong Đồ thị có hướng không chu trình thì luôn tồn tại ít nhất một Đỉnh có bậc vào bằng 0 (có thể chứng minh bằng phản chứng). Lần lượt lấy các Đỉnh bậc vào bằng 0 ra khỏi Đồ thị, cho xếp ở vị trí Đầu tiên, bỏ các cung ra từ những Đỉnh bậc 0 này thì Đồ thị sẽ xuất hiện những Đỉnh bậc 0 mới. lại lấy những Đỉnh bậc 0 này xếp vào vị trí tiếp theo. Cứ làm như vậy cho Đến khi bỏ hết các Đỉnh của Đồ thị.

Cách làm thứ hai (Đơn giản và dễ cài Đặt hơn): Thực hiện quá trình duyệt DFS Đồ thị, Đỉnh nào Được duyệt xong sớm hơn sẽ nằm ở vị trí cuối hơn hay có chỉ số lớn hơn. Mô tả cách làm này như sau:

```
Procedure DFS(u:integer);
Begin
    visited[u]:=true; {Đánh dấu Đỉnh u Đã thăm}
    for v:=1 to n do
        If a[u,v] and (not visited[v]) then {có cung (u,v) và Đỉnh v
chưa thăm}
        DFS(v);
    num[u]:=id; {gắn chỉ số cho Đỉnh u là id}
    q[id]:=u; {mảng thứ tự các Đỉnh}
    dec(id);
End;
Procedure Process;
Begin
    fillchar(visited,sizeof(visited),false);
    id:=n;
    for u:=1 to n do
        If not visited[u] then DFS(u);
End;
```

Thuật toán sắp xếp topo có nhiều ứng dụng quan trọng sẽ Được xét sau Đây.

Bài toán:

Đường Đi dài nhất.

Cho Đồ thị có hướng không chu trình, tìm Đường Đi dài nhất xuất phát từ một Đỉnh và kết thúc tại một Đỉnh (Độ dài Đường Đi Được Đỉnh nghĩa là số cung trên Đường Đi).

Hướng giải quyết: do Đồ thị có hướng không chu trình nên có thể dùng thuật toán sắp xếp topo Để Đánh số Đồ thị (dùng các số từ 1 Đến n Để Đánh số).

Goi $F[i]$ là Độ dài Đường Đi dài nhất nếu bắt Đầu từ Đỉnh có chỉ số i.

Áp dụng Quy hoạch Động: $\square F[n] = 0$

Kết quả Độ dài Đường Đi dài nhất là $\max(F[i])$. Dùng thêm mảng truy vết trong phần QHD nếu muốn xuất ra Đường Đi dài nhất.

```

Topo; {thuật toán sx topo dùng mảng q Để lưu thứ tự Đỉnh Đã sắp xếp}
{q[i]=u nghĩa là Đỉnh u ở vị trí i hay có nhãn là i}
F[n]:=0;
For i:=n-1 downto 1 do
  Begin
    k:=0;
    for j:=i+1 to n do
      if a[q[i],q[j]] then
        {nếu có cung nối Đỉnh có nhãn i với Đỉnh có nhãn j}
        if f[j]+1 > k then
          begin
            k:=f[j]+1;
            trace[i]:=j;
          end;
        F[i]:=k;
  End;

K:=0;
For i:=1 to n do k:=max(k,F[i]);
Writeln(k); {Độ dài Đường Đi dài nhất}
    
```

Chú ý: có thể tìm Độ dài Đường Đi dài nhất bằng cách khác là :

Lượt 1: bỏ Đi các Đỉnh có bậc vào bằng 0 trong Đồ thị. Bỏ Đi các cung ra từ nó. Sau khi bỏ, Đồ thị xuất hiện những Đỉnh có bậc vào bằng 0 mới. Tiếp tục thực hiện lượt bỏ thứ 2 tương tự lượt 1. Cứ thực hiện như thế cho Đến khi nào bỏ hết các Đỉnh của Đồ thị thì số lượt bỏ Được thực hiện chính là Độ dài Đường Đi dài nhất.

Bài toán:

Tổng số Đường Đi

Cho Đồ thị có hướng không chu trình và 2 Đỉnh s,t. Cho biết có bao nhiêu Đường Đi từ s Đến t. (2 Đường Đi khác nhau nếu thứ tự các Đỉnh trên chúng khác nhau)

Hướng giải quyết: Đầu tiên sử dụng thuật toán topo Để Đánh số Đồ thị. Mảng num[u] cho biết chỉ số của Đỉnh u và mảng q[i] cho biết Đỉnh có chỉ số i. (num[u]=i \forall q[i]=u)

Áp dụng tư tưởng Quy hoạch Động gọi F[i] là số con Đường xuất phát ở Đỉnh có chỉ số i và kết thúc ở Đỉnh t.

Khởi tạo $\square F[q[t]] = 1$
 $\square F[i] = 0 (\forall i \neq q[t])$
 $= \square F[i] = \sum F[j] \setminus j : i+1 \rightarrow n; (i, j) \text{ là cung}$

Công thức QHD:
 $\square i: q[t]-1 \rightarrow q[s]$

Kết quả của bài toán là F[q[s]].

```
Topo;
Fillchar(F,sizeof(F),0);
F[q[t]]:=0;
For i:=q[t]-1 downto q[s] do
  Begin
    k:=0;
    for j:=i+1 to n do
      if a[q[i],q[j]] then
        {nếu có cung nối Đỉnh có nhãn i với Đỉnh có nhãn j}
        k:=k + F[j];
      F[i]:=k;
    End;
  Writeln(F[q[s]]);
```

Luyện tập:

Bài toán: **Thực hiện dự án**

Có một dự án Được chia thành nhiều công việc nhỏ hơn. Mỗi công việc cần một khoảng thời gian nào Đó Để hoàn thành. Đồng thời một số công việc “ràng buộc nhau” và chỉ Được bắt Đầu thực hiện khi Đã hoàn thành xong một số công việc nào Đó. Tìm thời gian ít nhất Để hoàn thành dự án (cho biết có thể thực hiện nhiều công việc một lúc nếu các công việc này không “ràng buộc nhau”).

Dữ liệu:

- ✧ Dòng Đầu tiên ghi n là số công việc của dự án $n \leq 100$
- ✧ Dòng tiếp theo ghi n số , số thứ i là thời gian T_i Để thực hiện công việc i. ($T_i \leq 100.000$)
- ✧ N dòng tiếp theo, dòng thứ i liệt kê chỉ số các công việc j mà công việc j phải Được hoàn thành trước khi thực hiện công việc i. Quy ước dòng thứ i ghi số 0 nếu công việc i không bị ràng buộc bởi công việc nào (có thể thực hiện ở bất cứ thời Điểm nào).

Kết quả:

- ✧ Dòng Đầu ghi YES hay NO tương ứng với việc có thể hoàn thành dự án hay không.
- ✧ Nếu dòng Đầu là YES thì dòng sau là thời gian ít nhất Để hoàn thành dự án.

Gợi ý: Xét Đồ thị có n Đỉnh Đại diện cho n công việc. Xây dựng cung (u,v) nếu công việc v chỉ Được thực hiện sau khi hoàn thành công việc u và trọng số cung này bằng thời gian thực hiện công việc v. Xây dựng thêm Đỉnh n+1 có cung nối với tất cả các Đỉnh còn lại và trọng số bằng thời gian thực hiện công việc Đó.

Nếu Đồ thị Đã xây dựng có chu trình thì dự án sẽ không thể hoàn thành. Ngược lại thời gian ít nhất Để hoàn thành dự án là trọng số Đường Đi có trọng số lớn nhất trên Đồ thị.

Bài này sử dụng thuật toán giống bài “Đường Đi dài nhất” ở trên chỉ có Điều ở bài trên các cung có trọng số là 1 còn ở Đây các cung có trọng số là thời gian Để thực hiện công việc tương ứng.

Chú ý: Thuật toán Topo có thể sử dụng Để kiểm tra xem Đồ thị có chu trình hay không ?.

Bài toán: **Vòng Đua xe Đạp (BIC)**

Một vòng Đua xe Đạp Được tổ chức trên N thành phố, Đánh số từ 1 Đến N. Có M Đường nối (một chiều) giữa các thành phố. Vòng Đua bắt Đầu từ thành phố 1 và kết thúc tại thành phố 2.

Yêu cầu: Hỏi có bao nhiêu cách tổ chức các vòng Đua? (Biết hai vòng Đua là khác nhau nếu chúng không sử dụng các tuyến Đường như nhau)

Dữ liệu: BIC.INP

- ✂ Dòng 1: N, M
 - ✂ M dòng tiếp theo: mỗi dòng chứa hai số nguyên A, B, cho biết có một Đường nối giữa thành phố A và thành phố B
- Các thành phố có thể nối với nhau bởi nhiều hơn một con Đường

Kết quả: BIC.OUT

- ✂ Gồm 1 dòng duy nhất: số cách tổ chức các vòng Đua.
- ✂ Nếu kết quả có nhiều hơn 9 chữ số, chỉ cần in ra 9 chữ số cuối cùng. ✂ Nếu có vô số cách tổ chức các Đường Đua, in ra “inf”.

Giới hạn:

- ✂ $1 \leq N \leq 10^4$
- ✂ $1 \leq M \leq 10^5$

Ví dụ:

BIC.INP	BIC.OUT
8 14	6
6 7	
6 8	
7 5	
5 2	
5 3	
4 8	
1 6	
5 2	
7 5	
6 4	
1 4	
5 2	
7 4	
8 3	

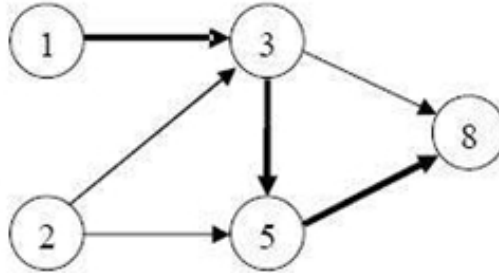
BIC.INP	BIC.OUT
2 2	inf
1 2	
2 1	

Bài toán: Lò cò - HSG QG 2008 (JUMP)

Nhảy lò cò là trò chơi dân gian của Việt Nam. Người trên hành tinh X cũng rất thích trò chơi này và họ Đã cải biên trò chơi này như sau: Trên mặt phẳng vẽ n vòng tròn Được Đánh số từ 1 Đến n. Tại vòng tròn i người ta Điền số nguyên dương ai. Hai số trên hai vòng tròn tùy ý không nhất thiết phải khác nhau. Tiếp Đến người ta vẽ các mũi tên, mỗi mũi tên hướng từ một vòng tròn Đến một vòng tròn khác. Quy tắc vẽ mũi tên là: Nếu có ba số ai, aj, ak thỏa mãn $a_k = a_i + a_j$ thì vẽ mũi tên hướng từ vòng tròn i Đến vòng tròn k

và mũi tên hướng từ vòng tròn j Đến vòng tròn k. Người chơi chỉ Được di chuyển từ một vòng tròn Đến một vòng tròn khác nếu có mũi tên xuất phát từ một trong số các vòng tròn, di chuyển theo cách mũi tên Đã vẽ Để Đi Đến các vòng tròn khác. Người thắng cuộc sẽ là người tìm Được cách di chuyển qua nhiều vòng tròn nhất.

Ví dụ: Với 5 vòng tròn và các số trong vòng tròn là 1, 2, 8, 3, 5, trò chơi Được trình bày trong hình dưới Đây:



Khi Đó có thể di chuyển Được nhiều nhất qua 4 vòng tròn (tương ứng với Đường di chuyển Được tô Đậm trên hình vẽ).

Yêu cầu: Hãy xác Định xem trong trò chơi mô tả ở trên, nhiều nhất có thể di chuyển Được qua bao nhiêu vòng tròn.

Dữ liệu: JUMP.INP

- ✂ Dòng Đầu chứa số nguyên n ($3 \leq n \leq 1000$);
 ✂ Dòng thứ hai chứa dãy số nguyên dương a_1, a_2, \dots, a_n ($a_i \leq 10^9, i=1, 2, \dots, n$). ✂
 Hai số liên tiếp trên một dòng Được ghi cách nhau bởi dấu cách.

Kết quả: JUMP.OUT

- ☯ Ghi ra số lượng vòng tròn trên Đường di chuyển tìm Được.

Ví dụ:

JUMP.INP	JUMP.OUT
5 1 2 8 3 5	4

VẤN ĐỀ: PHÁT HIỆN CHU TRÌNH

Bài toán:

Chu trình trong Đồ thị có hướng

Cho Đồ thị có hướng G , kiểm tra xem Đồ thị có chu trình không và xuất ra chu trình đó.

Hướng giải quyết:

Cách thứ nhất: Với mỗi cung (u,v) của Đồ thị G , thử bỏ cung (u,v) và kiểm tra xem có Đường Đi từ v Đến u không. Nếu có thì Đường Đi này kết hợp với cung (u,v) tạo thành một chu trình.

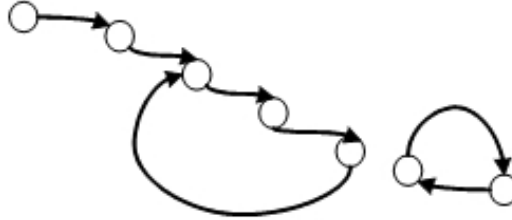
Cách thứ hai: Với mỗi Đỉnh u của Đồ thị, nếu Đỉnh này chưa Được duyệt thì thực hiện quá trình DFS từ Đỉnh Đó, trong quá trình DFS, nếu gặp Đỉnh v mà Đỉnh v lại Đang nằm trong quá trình duyệt DFS thì có chu trình xuất phát từ Đỉnh Đó. Chú ý: Đánh dấu những Đỉnh Đã duyệt trong quá trình DFS Để hạn chế những Đỉnh u .



```
Procedure DFS(u:integer);
var v:integer;
begin
  inDFS[u]:=true; {danh dau dinh u dang trong qua trinh DFS}
  visited[u]:=true; {danh dau dinh u da duoc tham}
  for v:=1 to n do
    if a[u,v] then
      if not visited[v] then
        begin
          DFS(v);
          Trace[v]:=u; {mang de truy vet duong di}
        end
      else if inDFS[v] then
        begin
          {xuat chu trinh xuat phat tai v va ket thuc tai u dua vao
          mang Trace}
        end;
      inDFS[u]:=false; {ket thuc qua trinh DFS tu dinh u}
    end;
end;

Procedure Process;
Var i:integer;
Begin
  Fillchar(inDFS,sizeof(inDFS),false);
  Fillchar(visited,sizeof(visited),false);
  For i:=1 to n do
    If not visited[i] then DFS(i);
End;
```

Mở rộng vấn Đề: Nếu Đồ thị vô hướng G là dạng Đặc biệt: mỗi Đỉnh có Đúng 1 cung ra thì việc phát hiện chu trình Được thực hiện một cách dễ dàng hơn: từ một Đỉnh có thể lần theo các cung nối tiếp nhau Để xác Định xem có chu trình xuất phát từ Đỉnh Đó không.



Ngoài ra, vì mỗi Đỉnh có Đúng 1 cung ra nên mỗi Đỉnh hoặc không thuộc chu trình nào hoặc thuộc duy nhất 1 chu trình. Do Đó ta có thể liệt kê tất cả các chu trình của Đồ thị bằng cách tìm một chu trình, bỏ chu trình Đó khỏi Đồ thị và thực hiện tìm kiếm chu trình tiếp theo cho Đến khi nào Đồ thị hết chu trình.

Ứng dụng chu trình trong dạng Đồ thị Đặc biệt này Để giải 2 bài toán sau:

Bài toán: **Tập các lá bài cực Đại (CARD)**

Cho n lá bài ($n \leq 20000$) Được Đánh số hiệu từ 1 Đến N. Trên mỗi lá bài ghi một số nguyên $F[i]$, ($1 \leq F[i] \leq n$, $i = 1..n$), có thể có nhiều lá bài cùng ghi một số. Hãy chọn ra trong n lá bài trên một tập nhiều nhất các lá bài sao cho tập hợp các số hiệu của các lá bài Được chọn giống hệt với tập hợp của các số ghi trên các lá bài.

Dữ liệu: CARD.INP

- ✂ Dòng Đầu ghi số n.
- ✂ Dòng tiếp theo gồm n số, số thứ i là số ghi trên lá bài thứ i.

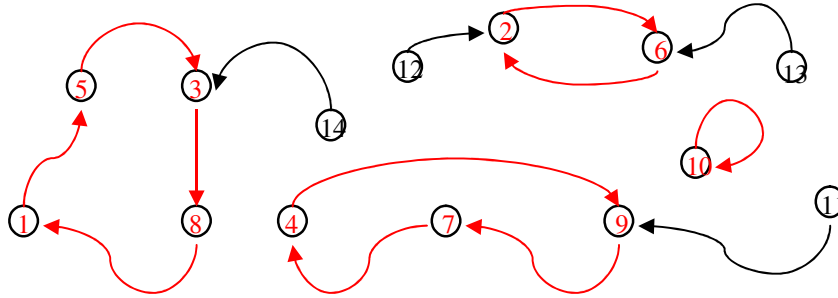
Kết quả: CARD.OUT

- ✂ Dòng Đầu ghi số k, là số lớn nhất các lá bài Được chọn.
- ✂ Dòng tiếp theo ghi k số là số hiệu của các lá bài Được chọn theo thứ tự tăng dần.

Ví dụ:

CARD.INP	CARD.OUT
14	10
5 6 8 9 3 2 4 1 7 10 9 2 6 3	1 2 3 4 5 6 7 8 9 10

Gợi ý: Xét Đồ thị n Đỉnh, mỗi Đỉnh Đại diện cho một lá bài. Lá bài i có ghi số nguyên $F[i]$ thì có cung nối Đỉnh i với Đỉnh $F[i]$ trên Đồ thị. Dễ thấy Đây là Đồ thị Đặc biệt như Đã nói ở trên.



Tìm tất cả các chu trình trên Đồ thị, các Đỉnh nằm trong các chu trình sẽ Đại diện cho các lá bài trong tập Được chọn. (các bạn tự chứng minh tính Đúng Đắn của thuật toán).

Bài toán: Hệ thống dữ liệu của Ngân Hàng (HTDL)

Một ngân hàng có N chi nhánh có tên từ 1 Đến N, mỗi chi nhánh có một hệ thống dữ liệu (HTDL), hai chi nhánh khác nhau có HTDL khác nhau. Trong một lần thay Đổi máy tính của toàn bộ N chi nhánh, do sơ xuất, người ta Đã cài Đặt không Đúng vị trí của các HTDL, chẳng hạn HTDL tại A là của chi nhánh B, tại B là của chi nhánh C, ... (có thể có chi nhánh Đã giữ Đúng HTDL của nó), *mặc dù hai chi nhánh khác nhau vẫn giữ hai HTDL khác nhau.*

Cần phải tiến hành trao Đổi các HTDL giữa các chi nhánh sao cho mỗi chi nhánh có Được HTDL của nó. Giữa hai chi nhánh có thể tiến hành trao Đổi trong một ngày, hai cặp máy khác nhau có thể làm Đồng thời công việc này trong cùng một ngày. Hãy tính xem cần ít nhất bao nhiêu ngày Để hoàn tất công việc này.

Dữ liệu: HTDL.INP

- ✂ Dòng Đầu ghi số $N \leq 10000$.
- ✂ Dòng thứ hai ghi N số, số thứ i là HTDL của chi nhánh mà chi nhánh i Đang giữ hiện thời.

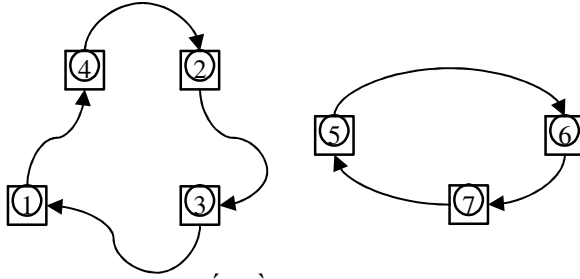
Kết quả: HTDL.OUT

- ✧ Dòng Đầu ghi số M là số ngày ít nhất cần cho việc tráo Đổi.
- ✧ M dòng tiếp theo, mỗi dòng gồm n số, số thứ i trong n số Đó là số hiệu chi nhánh tiến hành tráo Đổi với chi nhánh thứ i trong ngày Đó.

Ví dụ :

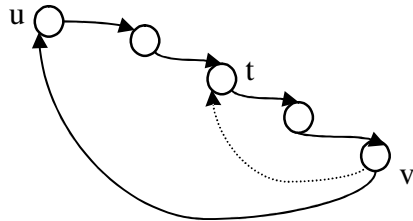
HTDL.INP	HTDL.OUT
7	2
4 3 1 2 6 7 5	3 4 1 2 7 6 5
	1 2 4 3 5 7 6

Gợi ý: Xem N chi nhánh ngân hàng là N Định của Đồ thị, chi nhánh i Đang giữ HTDL của chi nhánh j thì có cung (i,j) trên Đồ thị. Dễ thấy Đồ thị Đã xây dựng là Đồ thị Đặc biệt như Đã nói ở trên.



Không những thế Đồ thị này còn Đặc biệt hơn ở chỗ không những mỗi Đỉnh có duy nhất một cung ra mà mỗi Đỉnh còn có duy nhất một cung vào. Do Đó có thể chứng minh Được rằng mỗi Đỉnh của Đồ thị thuộc duy nhất một chu trình hay nói cách khác có thể phân Đồ thị thành các chu trình Độc lập (không có Đỉnh chung).

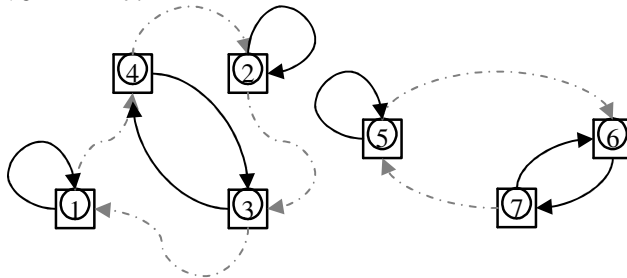
Chứng minh: từ Đỉnh u bất kì, Đi lần theo cung ra, do số Đỉnh có hạn nên Đến một lúc nào Đó, ta gặp phải một Đỉnh v mà Đỉnh v có cung ra duy nhất Đến Đỉnh t Đã Đi qua. Ta sẽ chứng minh Đỉnh t Đó là Đỉnh u . Thật vậy, giả sử t khác u thì Đỉnh t sẽ có 2 cung vào: (v, t) và một cung khác trên Đường Đi từ u Đến $t \Rightarrow$ vô lý. Vậy xuất phát từ một Đỉnh u bất kì, ta tìm Được một chu trình cũng kết thúc ở u . Mặt khác mỗi Đỉnh chỉ có một cung ra và một cung vào nên không có Đỉnh nào thuộc 2 chu trình. Hay các chu trình là Độc lập với nhau.



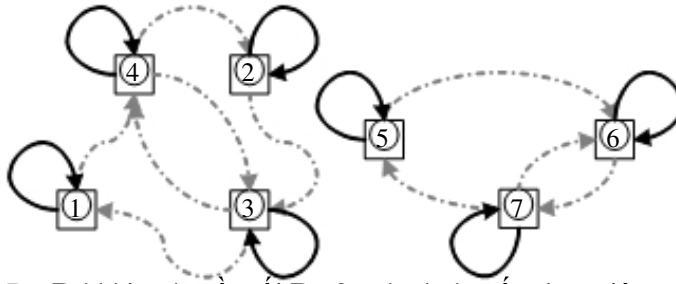
Trở lại bài toán, sau khi xây dựng xong Đồ thị, ta cần Đưa Đồ thị về dạng mà mỗi Đỉnh có cung nối với chính nó. Xét lần lượt các chu trình:

Với mỗi chu trình $x_1, x_2, \dots, x_n, x_1$. Trong ngày 1 ta tiến hành hoán Đổi HTDL của chi nhánh x_1 với chi nhánh x_n , chi nhánh x_2 với chi nhánh $x_{n-1} \dots$ chi nhánh $x_{n \div 2}$ với chi nhánh $x_{n - (n \div 2) + 1}$.

Ở ví dụ trên ta có chu trình 1,4,2,3,1 - trong ngày 1 tiến hành hoán Đổi cung ra của Đỉnh 1 với Đỉnh 3, của Đỉnh 4 với Đỉnh 2. và với chu trình 5,6,7,5 - ta hoán Đổi cung ra của Đỉnh 5 với Đỉnh 7.



Sau khi hoán Đổi hoặc Đồ thị Đã Được Đưa về trạng mong muốn, hoặc Đồ thị có Được phân thành các chu trình mà mỗi chu trình chỉ có 1 Đỉnh hoặc 2 Đỉnh (Với trường hợp chu trình có 2 Đỉnh, ta cần thêm một ngày nữa Để hoán Đổi HTDL của 2 chi nhánh này cho nhau)



Do Đó bài toán cân tối Đa 2 ngày hoàn tất công việc

Luyện tập:

Bài toán:

Kênh xung yếu - HSG QG 06 (CIRARC)

Một hệ thống n máy tính (các máy tính Được Đánh số từ 1 Đến n) Được nối lại thành một mạng bởi m kênh nối, mỗi kênh nối hai máy nào Đó và cho phép truyền tin một chiều từ máy này Đến máy kia. Ta gọi một **mạch vòng** của mạng Đã cho là một dãy các máy tính và các kênh nối chúng có dạng:

$$u_1, e_1, u_2, \dots, u_i, e_i, u_{i+1}, \dots, u_{k-1}, e_{k-1}, u_k, e_k, u_1$$

trong Đó u_1, u_2, \dots, u_k là các máy tính khác nhau trong mạng, e_i - kênh truyền tin từ máy u_i Đến máy u_{i+1} ($i = 1, 2, \dots, k-1$), e_k là kênh truyền tin từ máy u_k Đến máy u_1 . Một kênh truyền tin trong mạng Được gọi là kênh xung yếu nếu như bất cứ mạch vòng nào của mạng cũng Đều chứa nó.

Yêu cầu: Hãy xác Định tất cả các kênh xung yếu của mạng Đã cho. **Dữ**

liệu: CIRARC.INP

- ✂ Dòng Đầu tiên chứa 2 số nguyên dương n và m .
- ✂ Dòng thứ i trong số m dòng tiếp theo mô tả kênh nối thứ i bao gồm hai số nguyên dương u_i, v_i cho biết kênh nối thứ i cho phép truyền tin từ máy u_i Đến máy v_i .

Các số trên cùng một dòng Được ghi cách nhau bởi dấu cách.

Kết quả: CIRARC.OUT

- ✂ Dòng Đầu tiên ghi số nguyên k là số lượng kênh xung yếu trong mạng Đã cho. Ghi $k = -1$ nếu mạng không chứa kênh xung yếu.
- ✂ Nếu $k > 0$ thì mỗi dòng trong số k dòng tiếp theo ghi thông tin về một kênh xung yếu tìm Được theo qui cách mô tả giống như trong file dữ liệu vào.

Hạn chế: Trong tất cả các test: $n \leq 1000$, $m \leq 20000$

Ví dụ:

CIRARC.INP	CIRARC.OUT
2 2	2
1 2	1 2
2 1	2 1

CIRARC.INP	CIRARC.OUT
3 3	-1
1 2	
2 3	
1 3	

Gợi ý: Xét Đồ thị có hướng biểu diễn hệ thống n máy tính. Tìm một chu trình bất kì (có Độ dài càng nhỏ càng tốt). Với mỗi cung trong chu trình này, thử bỏ cung này Đi và kiểm tra xem Đồ thị còn chu trình không. Nếu không thì cung vừa bỏ Đi ứng với 1 kênh xung yếu.

Bài toán: **Hệ thống thông báo hoàn thiện (TBHT)**

Một trường có N học sinh với tên 1..N, $N \leq 10000$. Một hệ thống thông báo trong trường Được tổ chức và hoạt Động như sau. Mỗi học sinh chọn một học sinh duy nhất khác (Được gọi là người kế tiếp) Để truyền trực tiếp thông báo. Mỗi học sinh khi nhận Được thông báo phải truyền cho người kế tiếp mình.

Hệ thống thông báo Được gọi là hoàn thiện nếu khi một học sinh bất kỳ phát Đi một thông báo nào Đó tới người kế tiếp, người Đó lại truyền cho người kế tiếp, cứ tiếp tục như vậy, thông báo sẽ Được truyền Đến mọi người trong trường kể cả người ban Đầu Đã phát Đi thông báo.

Dữ liệu: TBHT.INP

- ✂ Dòng thứ nhất ghi số nguyên dương N.
- ✂ Trong N dòng tiếp theo, dòng thứ i ghi tên người kế tiếp của người i.

Cần xét xem hệ thống thông báo Đã cho có hoàn thiện không. Nếu không hãy thay Đổi người kế tiếp của một số ít nhất người Để nhận Được một hệ thống thông báo hoàn thiện.

Kết quả: TBHT.OUT

- ✂ Dòng thứ nhất ghi số T là số người cần thay Đổi người kế tiếp (T=0 có nghĩa là hệ thống là hoàn thiện).
- ✂ Nếu $T > 0$, trong T dòng tiếp theo mỗi dòng ghi hai số U, V có nghĩa là V là người kế tiếp mới của U.

Ví dụ:

TBHT.INP	TBHT.OUT
4	1
2	4 1
3	
4	
2	

Gợi ý: Chuyển về bài toán Đồ thị Đặc biệt, cần thay Đổi cung ra của ít nhất các Đỉnh sao cho Đồ thị là một chu trình duy nhất. Cách giải quyết Đơn giản là:

Xuất phát từ Đỉnh u có bậc vào bằng 0, Đi lần theo các cung cho Đến khi nào gặp Đỉnh v mà Đỉnh v có cung ra nối Đến một Đỉnh Đã Đi qua (Đỉnh này khác u) thì tiến hành thay Đổi cung ra của Đỉnh v nối Đến u. Lặp lại bước trên cho Đến khi không còn Đỉnh u nào có bậc vào bằng 0.

Khi Đó hoặc Đồ thị là 1 chu trình (dạng hoàn thiện) hoặc Đồ thị gồm nhiều chu trình riêng biệt - trong trường hợp này cần thực hiện thêm bước “ghép nối” các chu trình với nhau Để Được một chu trình duy nhất

Bài toán:

Chu trình cơ bản (CIRCUIT)

Một khu du lịch có n Địa Điểm Đánh số $1, 2, \dots, n$ và một số Đường Đi hai chiều nối những cặp Địa Điểm Đó. Giữa hai Địa Điểm bất kỳ có nhiều nhất là một Đường Đi nối chúng.

Một khách du lịch xuất phát từ Địa Điểm S muốn Đi thăm một số Địa Điểm khác rồi sau Đó quay trở về S . Để tránh sự nhầm lẫn, ông ta muốn tìm một hành trình không qua một con Đường hay một Địa Điểm nào quá một lần (Tất nhiên, ngoại trừ Địa Điểm S phải có mặt trong hành trình hai lần bởi Đó là nơi bắt Đầu cũng như kết thúc hành trình).

Yêu cầu: Hãy chỉ Đường cho du khách Đó.

Dữ liệu: CIRCUIT.INP

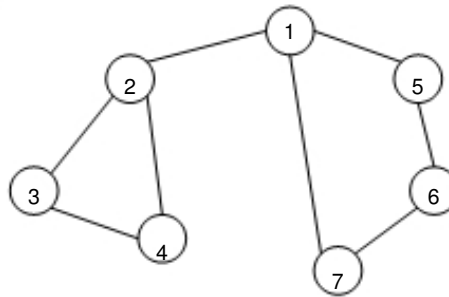
- ✂ Dòng 1: Chứa hai số n, S ($3 \leq n \leq 200$).
- ✂ Các dòng tiếp theo, mỗi dòng chứa hai số nguyên dương u, v cho ta thông tin: giữa hai Địa Điểm u và v có một Đường Đi hai chiều nối chúng.

Kết quả: CIRCUIT.OUT

- ✂ Dòng 1: Ghi từ YES hay NO tùy theo có tồn tại hành trình thoả mãn yêu cầu của du khách hay không
- ✂ Nếu dòng 1 ghi từ YES, dòng 2 ghi hành trình tìm Được: Bắt Đầu là Địa Điểm S , tiếp theo là danh sách các Địa Điểm sẽ Đi qua theo Đúng thứ tự trong hành trình, cuối cùng lại là Địa Điểm S .

Ví dụ:

CIRCUIT.INP	CIRCUIT.OUT
7 1	YES
1 2	1 7 6 5 1
1 5	
1 7	
2 3	
2 4	
3 4	
5 6	
6 7	



VẤN ĐỀ: CHU TRÌNH EULER

Bài toán: Chu trình Euler trong Đồ thị vô hướng.

Cho Đồ thị vô hướng G liên thông, tìm chu trình xuất phát từ một Đỉnh, Đi qua tất cả các cạnh, mỗi cạnh một lần và kết thúc tại Đỉnh xuất phát. Chu trình như vậy gọi là chu trình Euler.

Hướng giải quyết:

Trước hết ta chứng minh Định lý quan trọng sau Đây: Một Đồ thị vô hướng liên thông có chu trình Euler khi và chỉ khi mọi Đỉnh của nó Đều có bậc chẵn.

Chứng minh chiều thuận: Gọi u là Đỉnh xuất phát, chu trình Euler $u, u_1, u_2, \dots, u_m, u$. Nếu xem chu trình Euler $u, u_1, u_2, \dots, u_m, u$ là một Đường Đi (d) xuất phát từ u và kết thúc tại u . Với Đỉnh v bất kì, ta Định nghĩa số lần Đi qua Đỉnh v trong Đường Đi (d) là số lần Đi từ Đỉnh v Đến v rồi từ v Đến v '. Suy ra một lần Đi qua Đỉnh v tương ứng với Đi qua 2 cạnh nối với Đỉnh v . Mặt khác chu trình Euler Đi qua tất cả các cạnh một lần nên có thể dựa vào số lần Đi qua Đỉnh v Để xác Định bậc của Đỉnh v . Cụ thể như sau:

Nếu v khác u , gọi số lần Đi qua Đỉnh v là k thì số cạnh nối với Đỉnh v là $2k$ hay Đỉnh v có bậc chẵn.

Nếu v trùng với u , gọi số lần Đi qua Đỉnh v là k thì số cạnh nối với v là $2k$ cộng thêm cạnh (u, u_1) và cộng thêm cạnh (u_m, u) tức là $2k+2$, suy ra bậc của Đỉnh u cũng là một số chẵn.

Vậy với Đồ thị vô hướng G liên thông có chu trình Euler thì mọi Đỉnh của G Đều có bậc chẵn.

Chứng minh chiều nghịch: Gọi G là Đồ thị vô hướng liên thông mà mọi Đỉnh Đều có bậc chẵn. Ta sẽ chứng minh G có chu trình Euler.

Thật vậy, nếu xuất phát từ Đỉnh u bất kì trong Đồ thị, Đi theo một Đường Đi (d) bất kì qua các cạnh (mỗi cạnh chỉ qua nhiều nhất 1 lần) và Đường Đi chỉ dừng lại tại Đỉnh v nếu từ v không thể Đi tiếp Được nữa (do các cạnh nối với v Đều Đã Được Đi qua). Ta sẽ chứng minh Đỉnh v trùng với Đỉnh u .

Ta có (d): $u, u_1, u_2, \dots, u_m, v$. Giả sử v không trùng với u , gọi k là số lần Đi qua Đỉnh v thì số cạnh nối với v là $2k$ cộng thêm cạnh (u_m, v) hay bậc của Đỉnh v là $2k+1$ (là một số lẻ) - Vô lý. Vậy v trùng với u .

Như vậy mọi Đường Đi (d) bất kì khi kết thúc thì cho ta một chu trình, gọi là C_1 , bỏ chu trình này ra khỏi Đồ thị và tiếp tục một Đường Đi khác, ta lại Được chu trình C_2 , cứ làm như vậy cho Đến khi mọi cạnh của Đồ thị Đều Đã Được Đi qua, ta Được tập các chu trình C_1, C_2, \dots, C_t . Do Đồ thị liên thông nên tập các chu trình này có giao nhau tại các Đỉnh. Do Đó có thể móc nối các chu trình này Để tạo thành một chu trình duy nhất. Chu trình Đó chính là chu trình Euler

Định lý trên giúp kiểm tra Đồ thị có chu trình Euler hay không, nếu có, sử dụng thuật toán sau Đây Để tìm ra chu trình.

Mô tả thuật toán:

Cho Đỉnh u bất kì vào Stack

Lặp

Lấy từ Stack ra Đỉnh u.

Nếu tồn tại Đỉnh v mà (u,v) Đang là cạnh thì bỏ cạnh (u,v) và thêm Đỉnh v vào Stack

Ngược lại in ra Đỉnh u là Đỉnh trên Đường Đi Euler.

Cho Đến khi Stack rỗng.

Procedure **process**;

Begin

Top:=1; Q[1]:=1;

Repeat

u:=Q[top];

For v:=1 to n do

 If a[u,v] > 0 then

 Begin

 dec(a[u,v]); dec(a[v,u]);

 Inc(top); Q[top]:=v;

 Break;

 End;

 If u = Q[top] then

 begin

 writeln(u, ' -> ');

 dec(top);

 end;

until top = 0;

End;

Bài toán:

Đường Đi Euler trong Đồ thị vô hướng

Cho Đồ thị vô hướng G liên thông, tìm Đường Đi xuất phát từ 1 Đỉnh qua tất cả các cạnh, mỗi cạnh Đúng một lần và kết thúc tại một Đỉnh khác. Đường Đi như vậy gọi là Đường Đi Euler.

Hướng giải quyết: Điều kiện cần và Đủ Để Đồ thị G gồm n Đỉnh có Đường Đi Euler là n-2 Đỉnh của G Đều có bậc chẵn và 2 Đỉnh còn lại có bậc lẻ.

Thuật toán tìm Đường Đi Euler bằng cách thêm vào 1 cạnh ảo nối 2 Đỉnh có bậc lẻ, sau Đó tìm chu trình Euler rồi bỏ cạnh ảo trên chu trình vừa tìm Được.

Bài toán:

Chu trình Euler trong Đồ thị có hướng.

Cho Đồ thị có hướng G liên thông yếu (nếu xem cách cung là cách cạnh thì G trở thành Đồ thị vô hướng liên thông), tìm chu trình xuất phát từ một Đỉnh, Đi qua tất cả các cung, mỗi cạnh một lần và kết thúc tại Đỉnh xuất phát. Chu trình như vậy gọi là chu trình Euler.

Hướng giải quyết: Cũng chứng minh tương tự như Đối với Đồ thị vô hướng: G liên thông yếu có chu trình Euler khi và chỉ khi mọi Đỉnh của G Đều có bán bậc vào bằng bán bậc ra (số cung vào bằng số cung ra).

Với Định lý trên, ta cũng có một cách tìm chu trình Euler (nếu có) tương tự:

Cho Đỉnh u bất kì vào Stack

Lặp

Lấy từ Stack ra Đỉnh u .

Nếu tồn tại Đỉnh v mà (u,v) Đang là cung thì bỏ cung (u,v) và thêm Đỉnh v vào Stack

Ngược lại in ra Đỉnh u là Đỉnh trên Đường Đi Euler.

{chú ý Đường Đi in ra sẽ bị ngược chiều với các cung}

Cho Đến khi Stack rỗng.

Procedure **process**;

Begin

Top:=1; Q[1]:=1;

Repeat

u:=Q[top];

For v:=1 to n do

If a[u,v] > 0 then

Begin

dec(a[u,v]);

Inc(top); Q[top]:=v;

Break;

End;

If u = Q[top] then

begin

writeln(u, ' <- ');

dec(top);

end;

until top = 0;

End;

Bài toán:

Đường Đi Euler trong Đồ thị có hướng

Cho Đồ thị có hướng G liên thông yếu, tìm Đường Đi xuất phát từ 1 Đỉnh qua tất cả các cung, mỗi cung Đúng một lần và kết thúc tại một Đỉnh khác. Đường Đi như vậy gọi là Đường Đi Euler.

Hướng giải quyết: Điều kiện cần và Đủ Để Đồ thị G gồm n Đỉnh có Đường Đi Euler là $n-2$ Đỉnh của G Đều có bán bậc vào bằng bán bậc ra, và 2 Đỉnh còn lại, 1 Đỉnh có bán bậc vào hơn bán bậc ra là 1 (Đỉnh u), 1 Đỉnh có bán bậc ra hơn bán bậc vào là 1 (Đỉnh v).

Thuật toán tìm Đường Đi Euler bằng cách thêm vào 1 cung ảo (u,v) nối 2 Đỉnh có chênh lệch bán bậc vào - bán bậc ra là 1, sau Đó tìm chu trình Euler trên Đồ thị có hướng rồi bỏ cung ảo trên chu trình vừa tìm Được.

Một số bài toán áp dụng chu trình Euler.

Bài toán:

Nỗi từ (SECRET)

Để mở Được cánh cửa bí mật ở một lâu Đài cổ, các nhà khảo cổ học phải giải quyết vấn Đề sau: Có một lượng lớn các mảnh nam châm ở trước cánh cửa, trên mỗi mảnh ghi một từ. Các mảnh cần Được sắp xếp thành một dãy sao cho kí từ Đầu tiên của từ trên mỗi mảnh (từ mảnh thứ 2 trở Đi) phải giống kí tự cuối cùng của từ trên mảnh trước. Ví dụ mảnh có từ “acm” có thể xếp sau mảnh có từ “motorola”.

Cho biết các từ trên N mảnh nam châm, hãy giúp các nhà khảo cổ học kiểm tra xem có thể ghép các mảnh thành một dãy hay không Để mở Được cánh cửa bí mật.

Dữ liệu: SECRET.INP

- ✂ Dòng Đầu là số mảnh nam châm ($N \leq 30000$)
- ✂ N dòng tiếp theo, mỗi dòng một xâu kí tự mô tả một từ Được viết trên một mảnh nam châm (các từ gồm các kí tự từ 'a' Đến 'z', các từ khác rỗng và có Độ dài không quá 9).

Kết quả: SECRET.OUT

- ✂ Ghi trên dòng Đầu tiên, nếu có thể xếp Được thì ghi "possible", nếu không thì ghi "impossible"

Gợi ý: Xét Đồ thị G gồm cách Đỉnh Đại diện cho các kí tự từ 'a' Đến 'z'. Với mỗi từ trên mảnh nam châm, ta thêm một cung nối kí từ Đầu của từ Đến kí tự cuối của từ trên G. Bài toán trở thành kiểm tra Đồ thị có hướng G có Đường Đi Euler hay không. (chú ý kiểm tra tính liên thông yếu của G).

.

Bài toán:

Vẽ Đồ thị (GRAPH)

Cho Đơn Đồ thị vô hướng G có N Đỉnh và M cạnh. Xét bài toán vẽ Đồ thị như sau: Mỗi lần vẽ ta Đặt bút tại một Đỉnh, di chuyển bút qua các Đỉnh, mỗi lần Đi từ Đỉnh này Đến một Đỉnh khác, ta lại vẽ thêm một cạnh. Ta không Được phép vẽ một cạnh quá một lần (tính cả hai chiều). Với yêu cầu như thế, có thể với một lần Đặt bút ta không thể vẽ Được hết các cạnh mà phải dùng nhiều lần nhắc.

Yêu cầu: Hãy tìm cách vẽ Đồ thị cho trước sao cho số lần nhắc bút là nhỏ nhất. **Dữ**

liệu: GRAPH.INP

- ✂ Dòng Đầu ghi hai số nguyên dương N, M ($N \leq 100$, $M \leq 1000$)
- ✂ M dòng tiếp theo, mỗi dòng ghi hai số u,v mô tả một cạnh của Đồ thị.

Kết quả: GRAPH.OUT

- ✂ Dòng Đầu ghi K là số lần nhắc bút cần dùng.
- ✂ K dòng tiếp theo, mỗi dòng mô tả một lần vẽ, trong Đó số Đầu tiên là số cạnh Được vẽ, tiếp theo là các Đỉnh xuất hiện trên Đường vẽ tương ứng.

Ví dụ:

GRAPH.INP	GRAPH.OUT
5 5	1
1 2	5 1 2 3 4 2 5
2 3	
3 4	
2 4	
2 5	

Gợi ý: Xét từng thành phần liên thông G' của Đồ thị G .

- Nếu G' không có cạnh thì không cần dùng nét vẽ nào.
 - Nếu G' không có Đỉnh bậc lẻ thì cần dùng 1 nét vẽ : theo chu trình Euler
 - Nếu G' có k Đỉnh bậc lẻ (k luôn là số chẵn) thì cần dùng $k/2$ nét vẽ : bằng cách thêm vào G' $k/2$ cạnh ảo nối các Đỉnh bậc lẻ; sau khi thêm, các Đỉnh của G' đều có bậc chẵn. Tìm chu trình Euler trên G' rồi bỏ đi $k/2$ cạnh ảo, ta Được $k/2$ lần nhắc bút tương ứng.
- (các bạn tự chứng minh tính Đúng Đắn của cách làm)

Luyện tập:

Bài toán: Cột cây số (MSTONE)

Một mạng lưới giao thông gồm n thành phố và m tuyến Đường xa lộ hai chiều. Giữa hai thành phố bất kỳ có nhiều nhất là một xa lộ nối trực tiếp từ thành phố này tới thành phố kia. Trên mỗi xa lộ, người ta Đã dựng sẵn các cột cây số Để chỉ Đường cho hành khách.

Để Điền số km trên các cột cây số, người ta sử dụng một rô-bốt. Muốn Điền Đủ các cột cây số trên một tuyến Đường (u, v) thì rô bốt phải thực hiện một chuyến Đi từ u tới v và một chuyến Đi từ v về u , cứ sau mỗi km thì dừng lại và ghi vào một mặt của một cột cây số.

Ví dụ: Để Điền các cột cây số trên tuyến Đường Hà Nội - Hải Phòng. Đầu tiên rô bốt xuất phát từ Hà Nội, cứ Đi mỗi km thì dừng lại và Điền vào cột cây số dòng "Hà Nội ... km", tất nhiên chỉ có thể Điền vào mặt quay về hướng Hải Phòng bởi Rô bốt không biết Được từ Đó Đến Hải Phòng còn bao xa. Muốn Điền dòng chữ "Hải Phòng ... km" lên mặt còn lại của các cột cây số thì rô bốt phải thực hiện hành trình từ Hải Phòng trở về Hà Nội

Yêu cầu: Giả thiết rằng hệ thống giao thông Đảm bảo sự Đi lại giữa hai thành phố bất kỳ. Hãy tìm một hành trình của Rô bốt xuất phát từ thành phố 1, Đi viết Đầy Đủ lên các cột cây số rồi quay trở về thành phố 1, sao cho mỗi mặt của cột cây số bất kỳ nào cũng chỉ bị viết một lần.

Dữ liệu: MSTONE.INP

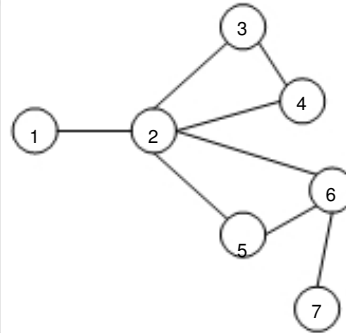
- ✂ Dòng 1: Chứa hai số n, m cách nhau một dấu cách ($2 \leq n \leq 200$)
- ✂ m dòng tiếp theo, mỗi dòng ghi hai số u, v cách nhau một dấu cách: cho biết giữa hai thành phố u và v có một tuyến xa lộ nối chúng

Kết quả: MSTONE.OUT

- ✂ Ghi các hành trình rô bốt phải Đi: Bắt Đầu từ thành phố 1, tiếp theo là các thành phố Đi qua theo Đúng thứ tự trong hành trình, kết thúc là thành phố 1. Các số hiệu thành phố phải ghi cách nhau ít nhất một dấu cách hoặc dấu xuống dòng.

Ví dụ:

MSTONE.INP	MSTONE.OUT
7 8	1 2 6 7 6
1 2	5 2 5 6 2 4 3 2 3 4 2 1
2 3	
3 4	
4 2	
2 5	
5 6	
6 7	
6 2	



Bài toán:

Người Đưa thư (POS)

Một bưu tá ở vùng quê cần chuyển thư cho người dân ở các ngôi làng cũng như ở trên các con Đường nối giữa các ngôi làng. Bạn cần giúp bưu tá tìm hành trình Đi qua mỗi ngôi làng và mỗi con Đường ít nhất một lần (dữ liệu vào Đảm bảo một hành trình như vậy tồn tại). Tuy nhiên, mỗi hành trình còn Được gắn với một chi phí. Người dân ở các ngôi làng Đều muốn bưu tá Đến làng mình càng sớm càng tốt. Vì vậy mỗi ngôi làng Đã thỏa thuận với bưu Điện, nếu làng i là làng thứ k phân biệt Được thăm trên hành trình và $k \leq w_i$, làng i sẽ trả $w_i - k$ euros cho bưu Điện. Nếu $k > w_i$, bưu Điện Đồng ý trả $k - w_i$ euros cho ngôi làng. Ngoài ra, bưu Điện còn trả bưu tá một euro khi Đi qua mỗi con Đường trên hành trình.

Có n ngôi làng, Được Đánh số từ 1 Đến n . Bưu Điện Được Đặt ở ngôi làng số một, do Đó hành trình cần bắt Đầu và kết thúc tại ngôi làng này. Mỗi ngôi làng Được Đặt ở giao Điểm của hai, bốn, hoặc tám con Đường. Có thể có nhiều Đường nối giữa hai ngôi làng. Con Đường có thể là một vòng nối một ngôi làng với chính nó.

Yêu cầu: Viết chương trình xác Định một hành trình Đi qua mỗi ngôi làng và mỗi con Đường ít nhất một lần, sao cho tổng lợi nhuận của bưu Điện là lớn nhất (hay tổng thiệt hại là bé nhất).

Dữ liệu: POS.INP

- ✂ Dòng Đầu tiên chứa 2 số nguyên n, m , cách nhau bởi khoảng trắng; n ($1 \leq n \leq 200$), là số ngôi làng và m là số con Đường.
- ✂ Mỗi dòng trong số n dòng sau chứa một số nguyên dương. Dòng thứ $i+1$ chứa số w_i , $0 \leq w_i \leq 1000$, xác Định chi phí Được trả bởi làng i .
- ✂ Mỗi dòng trong số m dòng sau chứa hai số nguyên dương cách nhau bởi khoảng trắng, mô tả một con Đường nối hai ngôi làng.

Kết quả: POS.OUT

- ✂ Dòng Đầu tiên chứa số nguyên dương k , Độ dài của hành trình.
- ✂ Dòng thứ hai theo chứa $k+1$ số cho biết các ngôi làng Được thăm theo thứ tự trên hành trình, cách nhau bởi khoảng trắng, trong Đó $v_1 = v_{k+1} = 1$.

Ví dụ:

POS.INP	POS.OUT
6 7	7
1	1 5 4 2 1 6 3 1
7	
4	
10	
20	
5	
2 4	
1 5	
2 1	
4 5	
3 6	
1 6	
1 3	

Gợi ý: Xét mọi Đường Đi Euler (d) (hoặc chu trình Euler (c)) của Đồ thị. Dễ dàng chứng minh Được thứ Đến thăm của các ngôi làng không quan trọng hay nói cách khác mọi Đường Đi (d) (hoặc chu trình (c)) Đều có cùng một tổng lợi nhuận thu Được. Do Đó chỉ cần chỉ ra một Đường Đi Euler (d) (hoặc chu trình Euler (c)) bất kì.

Bài toán:

Thám hiểm mê cung (PCYCLE)

Một mê cung gồm có N phòng và M hành lang nối các phòng, giữa hai phòng bất kì có không quá một hành lang nối chúng.

Một người muốn khám phá mê cung, anh ta sẽ xuất phát từ một phòng và Đi dọc theo tất cả các hành lang sao cho mỗi hành lang Được Đi qua Đúng một lần, rồi lại trở về vị trí xuất phát. Mỗi hành lang có một giá trị c cho biết khi Đi qua nó thì năng lượng nhà thám hiểm sẽ cộng thêm với c (c có thể âm hay dương). Nhà thám hiểm bắt Đầu xuất phát với năng lượng bằng 0, anh ta sẽ chết nếu sau khi Đi hết một hành lang nào Đó mà mức năng lượng nhỏ hơn 0.

Yêu cầu: Hãy giúp nhà thám hiểm tìm ra một hành trình an toàn thỏa mãn các yêu cầu Đã Đưa ra.

Dữ liệu: PCYCLE.INP

- ✂ Dòng 1 là 2 số nguyên N, M. ($1 \leq N \leq 200$)
- ✂ M dòng tiếp theo, dòng thứ i gồm 3 số nguyên u, v, c cho biết có 1 hành lang nối phòng u với phòng v và giá trị năng lượng là c. ($|c| \leq 10000$).

Kết quả: PCYCLE.OUT

- ✂ Nếu có không có hành trình nào an toàn thì ghi ra -1. Ngược lại ghi ra M+1 số nguyên là chỉ số phòng trên Đường Đi. Từ phòng xuất phát, qua các phòng, hành lang rồi quay trở về phòng xuất phát.

Ví dụ:

PCYCLE.INP	PCYCLE.OUT
3 3	2 1 3 2
1 2 2	
1 3 -1	
2 3 -1	

Gợi ý: Tìm một chu trình Euler bất kì (E): $u_1, u_2, \dots, u_m, u_1$

Gọi $F[i] = \sum_{k=2}^i c[u_{k-1}, u_k]$ ($F[1] = 0$)

Tìm Đỉnh s có $F[s]$ nhỏ nhất.

- Nếu $F[s] < 0$ thì không tồn tại hành trình an toàn
- Nếu $F[s] \geq 0$ thì lấy s là Đỉnh xuất phát, Đi theo chu trình (E) là một hành trình an toàn.

(bạn Đọc tự chứng minh tính Đúng Đắn của thuật toán)

... ..

VẤN ĐỀ: CHU TRÌNH ÂM VÀ ỨNG DỤNG

Bài toán:

Phát hiện chu trình âm

Cho Đồ thị vô hướng G , các cung có trọng số (có thể âm hoặc dương). Kiểm tra xem Đồ thị có tồn tại chu trình âm (tổng trọng số các cung trong chu trình bé hơn 0) hay không?

Hướng giải quyết: Sử dụng thuật toán FORD BELLMAN.

Với 2 Đỉnh không có cung nối trực tiếp, giả sử cho nó một cung ảo nối 2 Đỉnh Đó với trọng số là một số rất lớn (bằng maxlongint chẳng hạn).

Xuất phát từ 1 Đỉnh s bất kì, dùng thuật toán FORD BELLMAN Để tìm Đường Đi ngắn nhất từ s Đến các Đỉnh còn lại.

Nhắc lại về thuật toán FORD BELLMAN có cấu trúc như sau:

```
for v:=1 to n do d[v] := +∞; {d[v] là nhãn trọng số Đường Đi ngắn nhất từ s tới v}
d[s] := 0;
repeat
  stop:=true;
  for u:=1 to n do
    for v:=1 to n do
      if a[u,v] then {nếu có cung (u,v)}
      if d[v] > d[u] + c[u,v] then {Điều kiện Để tối ưu nhãn}
      begin
        d[v]:=d[u] + c[u,v];
        stop:=false;
      end;
until stop;
```

Chú ý rằng thuật toán FORD BELLMAN chỉ cần tối Đa $n-1$ (n là số Đỉnh của Đồ thị) lần lặp Để cho Đường Đi ngắn nhất từ Đỉnh s Đến Đỉnh t bất kì trong trường hợp Đồ thị không có chu trình âm (vì mọi Đường Đi ngắn nhất Đều có số cạnh tối Đa là $n-1$). Hay nói cách khác vòng lặp repeat-until thực hiện tối Đa n lần.

Nhưng với trường hợp Đồ thị có chu trình âm thì thuật toán FORD BELLMAN ở trên sẽ bị lặp vô hạn. Và ta có thể sử dụng Điều này Để kiểm tra xem Đồ thị có chu trình âm hay không nếu vòng lặp thực hiện lần thứ $n+1$.

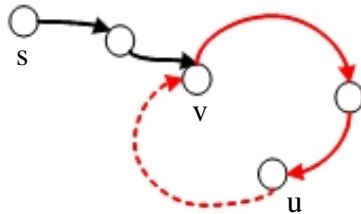
Cũng có thể nhận ra chu trình âm nếu tại một bước tối ưu nhãn $d[v]$ nào Đó, ta có $d[v]$ bé hơn tổng trọng số các cung âm trong Đồ thị.

```
for v:=1 to n do d[v] := maxlongint;
d[s] := 0;
count:=0;
repeat
  stop:=true;
  inc(count); {tăng số lần lặp}
  if count = n + 1 then
    begin
      {thông báo có chu trình âm}
      Break;
    end;
  for u:=1 to n do
    for v:=1 to n do
      if a[u,v] then
        if d[v] > d[u] + c[u,v] then
          begin
            d[v]:=d[u] + c[u,v];
            if d[v] < sum then {sum là giá trị tổng trọng số các cung âm}
              begin
                {thông báo có chu trình âm}
                Break;
              end;
          end;

          stop:=false;
        end;
until stop;
```

Tuy nhiên cách này khá mất thời gian, một cách khác hiệu quả hơn và có thể xuất ra Được chu trình âm nếu có.

Cũng dựa trên thuật toán FORD BELLMAN, tại mỗi bước sử dụng Đỉnh u Để tối ưu nhân Đỉnh v ($d[v] := d[u] + c[u,v]$) ta thử truy vết ngược Đường ngắn nhất hiện tại từ u về s xem có Đi qua Đỉnh v hay không .Nếu có thì tồn tại chu trình âm xuất phát ở v.



Thuật toán FORD BELLMAN Được viết lại như sau:

```
Procedure check(u,v:integer):boolean;
{ct kiểm tra Đường Đi ngắn nhất hiện tại từ u về s có Đi qua v hay không}
begin
  While (u<>v) and (u<>s) do
    U:=trace[u];
  If u=v then check:=true else check:=false;
End;

Procedure Print(u,v:integer);
{xuất chu trình âm xuất phát ở v và kết thúc ở u}
Begin
  Write(v,' <- ',u);
  While u<>v do
    Begin
      Write(' <- ',trace[u]);
      u:=trace[u];
    End;
End;

Procedure FordBellman;
Và u,v:integer
begin
  for v:=1 to n do d[v] := maxlongint;
  d[s] := 0;
  repeat
    stop:=true;
    for u:=1 to n do
      for v:=1 to n do
        if a[u,v] then
          if d[v] > d[u] + c[u,v] then
            begin
              d[v]:=d[u] + c[u,v];
              if check(u,v) then
                begin
                  Print;
                  Break;
                end;
              trace[v]:=u; {mảng dùng Để truy vết Đường Đi ngắn nhất}
              stop:=false;
            end;
          until stop;
        end;
  end;
end;
```

Chu trình âm Được ứng dụng Để giải quyết các bài toán thực tế sau:

Bài toán:

Buôn tiền (MONEY)

Một người làm việc ở một ngân hàng ngoại tệ theo dõi tỉ giá hối Đoái phát hiện ra là: Nếu khôn khéo, thì từ một lượng ngoại tệ ban Đầu, nhờ chuyển Đổi sang các loại ngoại tệ khác, anh ta có thể thu Được lợi nhuận Đáng kể.

Ví dụ: Nếu anh ta có 1 USD và tỉ giá hối Đoái giữa các ngoại tệ như sau:

1 USD	=	0.7 bảng Anh
1 bảng Anh	=	9.5 Franc Pháp
1 Franc Pháp	=	0.16 USD

Khi Đó với 1 USD anh ta có thể mua Được $0.7 * 9.5 * 0.16 = 1.064$ USD nhờ việc chuyển Đổi tiền qua bảng Anh, rồi từ bảng Anh sang Franc Pháp, và cuối cùng lại quay về USD. Nhờ Đó mỗi USD Đã Dem lại cho anh ta lợi nhuận là 0.064USD.

Giả sử trong nhà băng quản lý n loại ngoại tệ Đánh số 1, 2, ..., n. Biết bảng tỉ giá hối Đoái $R[i, j]$ ($1 \leq i, j \leq n$). (Tức là 1 Đơn vị ngoại hối i mua Được $R[i, j]$ Đơn vị ngoại hối j). Cần xác Định xem có cách Đổi tiền Dem lại lợi nhuận hay không ?

Dữ liệu: MONEY.INP

- ✂ Dòng Đầu tiên chứa số n ($n \leq 100$)
- ✂ Dòng thứ i trong số n dòng tiếp theo chứa n số thực dương $R[i, 1], R[i, 2], \dots, R[i, n]$.

Kết quả: MONEY.OUT

- ✂ Dòng Đầu tiên ghi YES hoặc NO tương ứng với việc có hoặc không có cách Đổi tiền sinh lợi nhuận
- ✂ Nếu dòng Đầu tiên là YES thì dòng thứ hai ghi hai số u và s. Trong Đó u là loại tiền xuất phát, còn s là lợi nhuận thu Được nhờ cách Đổi 1 Đơn vị tiền u. Dòng thứ ba ghi trình tự cần tiến hành Đổi tiền Để thu lại Được lợi nhuận bắt Đầu từ loại tiền xuất phát

Các số trên một dòng của Input/Output File Được ghi cách nhau ít nhất một dấu cách Lợi nhuận (nếu có) trong Output File có thể chỉ cần làm tròn giữ lại 6 chữ số sau dấu chấm thập phân.

Ví dụ:

MONEY.INP					
5					
1.00	1.10	0.83	0.81	0.85	
0.83	1.00	0.86	1.09	0.81	
0.89	0.84	1.00	0.83	1.02	
0.84	0.83	1.01	1.00	0.84	
1.09	0.84	0.87	0.90	1.00	

MONEY.OUT	
YES	
1	0.007160
1 2 4	

Gợi ý: Xét Đồ thị có hướng n Đỉnh, mỗi Đỉnh Đại diện cho 1 loại tiền tệ và các cung nối giữa các Đỉnh với trọng số là tỉ giá giữa 2 loại tiền tệ này. Cần tìm một chu trình có tích trọng số trên các cung lớn hơn 1.

Gán lại trọng số các cung $c[i, j] := -\ln(c[i, j])$

Tìm chu trình âm trên Đồ thị với trọng số mới. Nếu có chu trình âm thì chu trình này Đại diện cho cách Đổi tiền mang lại lợi nhuận.

Thật vậy, giả sử chu trình qua các cung trọng số e_1, e_2, \dots, e_m .

Ta có $e_1 + e_2 + \dots + e_m < 0$

- ✗ $-\ln(c_1) - \ln(c_2) - \dots - \ln(c_m) < 0$
- ✗ $\ln(c_1) + \ln(c_2) + \dots + \ln(c_m) > 0$
- ✗ $\ln(c_1 \cdot c_2 \cdot \dots \cdot c_m) > 0$
- ✗ $c_1 \cdot c_2 \cdot \dots \cdot c_m > e^0$
- ✗ $c_1 \cdot c_2 \cdot \dots \cdot c_m > 1$

Hay các cung c_1, c_2, \dots, c_m tạo thành một chu trình tương ứng với cách Đổi tiền mang lại lợi nhuận.

Bài toán:

Vận chuyển hàng (TRANS)

Công ty MCA là một công ty vận tải nổi tiếng tại Đất nước Peace, với mạng lưới hoạt Động trong khắp cả nước. Sắp Đến kỷ niệm 30 năm thành lập, giám Đốc công ty quyết Định mở một Đợt khuyến mãi lớn cho tất cả các khách hàng. Cụ thể là công ty sẽ mở một số tuyến Đường vận chuyển miễn phí cho khách hàng.

Mỗi tuyến Đường như vậy sẽ xuất phát từ một thành phố, qua một số thành phố (không Đi qua thành phố nào 2 lần) rồi quay về nơi xuất phát. Nếu tính chi phí vận chuyển trung bình trên từng tuyến Đường thì chi phí chi ra cho Đợt khuyến mãi này sẽ không nhỏ nên giám Đốc cũng muốn các tuyến Đường này thỏa Điều kiện tổng Độ dài Đường Đi chia cho tổng số con Đường trong tuyến Đường Đó (gọi là chi phí của tuyến Đường) là nhỏ nhất.

Cho một mạng lưới vận chuyển hàng của công ty MCA hãy tìm ra tuyến Đường thích hợp nhất (thỏa cả 2 Điều kiện trên) cho Đợt khuyến mãi. Xuất ra chi phí của tuyến Đường tìm Được.

Dữ liệu: TRANS.INP

- ✗ Dòng Đầu tiên ghi hai số nguyên N và M là số thành phố và số con Đường trong mạng lưới.
- ✗ M dòng tiếp theo, mỗi dòng ghi ba số nguyên a b c với ý nghĩa có Đường Đi một chiều từ a Đến b với Độ dài là c.

Dữ liệu: TRANS.OUT

- ✗ Dòng thứ nhất ghi một số thập phân với ít nhất 2 chữ số sau dấu phẩy là chi phí nhỏ nhất. Nếu không tồn tại tuyến Đường nào thỏa mãn ghi ra -1.
- ✗ Dòng thứ hai ghi số P là số lượng thành phố trên tuyến Đường Được chọn.
- ✗ Dòng thứ ba ghi P + 1 số lần lượt là các thành phố trên tuyến Đường Được chọn.

Giới hạn:

- ✗ $1 \leq N \leq 100$
- ✗ $1 \leq M \leq 9000$
- ✗ $1 \leq c \leq 2^{23}$

Ví dụ:

TRANS.INP	TRANS.OUT
3 3	1.00
1 2 1	3
2 3 1	1 2 3 1
3 1 1	

Gợi ý: Xét Đồ thị có hướng G với N Đỉnh và M cung nối. Gọi r là chi phí nhỏ nhất của tuyến Đường cần tìm. Xét Đồ thị $G(r)$ có trọng số các cung Được xây dựng $c := d - r$ (d là Độ dài các tuyến Đường ban Đầu). Giả sử có chu trình âm trên Đồ thị $G(r)$ - giả sử chu trình âm Đó gồm các cung c_1, c_2, \dots, c_m thì

$$\begin{aligned} c_1 + c_2 + \dots + c_m &< 0 \\ \bigvee d_1 + d_2 + \dots + d_m - m.r &< 0 \\ \bigvee r &> (d_1 + d_2 + \dots + d_m)/m = r_0 \end{aligned}$$

Suy ra r chưa phải là chi phí nhỏ nhất cần tìm vì có tuyến Đường qua các cung d_1, d_2, \dots, d_m có chi phí nhỏ hơn.

Vậy nếu r là chi phí nhỏ nhất thì Đồ thị $G(r)$ không có chu trình âm, Hay nói cách khác nếu Đồ thị $G(r)$ có chu trình âm thì chu trình âm Đó cho ta một chi phí tương ứng r_0 nhỏ hơn r . Từ Đó có thể suy ra nếu r là chi phí nhỏ nhất thì:

- + với mọi $r' > r$ thì $G(r')$ có chu trình âm
- + với mọi $r' < r$ thì $G(r')$ không có chu trình âm

Từ nhận xét trên có thể tìm ra Được chi phí nhỏ nhất bằng cách chặt nhị phân giá trị r , sau Đó kiểm tra Đồ thị $G(r)$ có chu trình âm hay không.

```
x := 0;
y := +∞;
while y - x > 0.01 do {Độ chính xác tới 2 chữ số phần thập phân}
begin
    r:=(x + y)/2;
    if check(r) then {Nếu Đồ thị G(r) có chu trình âm}
    begin
        {lưu chu trình âm Đề xuất tuyến Đường tối ưu hơn}
        y:=r;
    end
    else x:=r;
end;
writeln(r) {chí phí nhỏ nhất}
```

Luyện tập:

Bài toán: **Lợi nhuận tối Đa (PROFIT)**

Thuyền trưởng Sinbad có dự Định Đưa Đoàn tàu cùng thủy thủ Đoàn Đi trao Đổi hàng hoá vòng quanh các hòn Đảo. Sinbad quyết Định sẽ xuất phát tại một hòn Đảo bất kỳ, sau Đó Đi qua một số hòn Đảo và quay trở về hòn Đảo xuất phát (một hòn Đảo có thể Được thăm nhiều lần). Là một thuyền trưởng tài ba và Đây kinh nghiệm, Sinbad muốn chọn một lộ trình sao cho lợi nhuận thu Được là tối Đa. Lợi nhuận thu Được Đối với một hành trình Được tính dựa trên thương số giữa tổng lượng hàng hoá bán Được dọc theo hành trình và tổng Độ dài hành trình. Cho biết bản Đồ của vùng biển gồm N hòn Đảo Được Đánh số từ 1 Đến N và M tuyến hải trình giữa chúng cùng lượng hàng hoá dự kiến sẽ bán Được $C[I, J]$ nếu Đi từ Đảo I Đến Đảo J ($C[I, J] = C[J, I]$) (Các tuyến hải trình xem như hai chiều). Bạn hãy giúp Sinbad lập lộ trình tối ưu cho thủy thủ Đoàn.

Dữ liệu: PROFIT.INP

- ✂ Dòng Đầu tiên gồm 2 số N,M ($1 \leq N \leq 100$) lần lượt là số Đảo và số hải trình giữa chúng.
- ✂ M dòng tiếp theo, mỗi dòng gồm 4 số nguyên A,B,C,D ($1 \leq A, B \leq N$; $1 \leq C, D \leq 255$) cho biết tồn tại hải trình giữa hai hòn Đảo A,B trong Đó C là số hàng hoá dự kiến sẽ bán Được và D là Độ dài của hải trình.

Kết quả: PROFIT.OUT

- ✂ Gồm một số thực P duy nhất là lợi nhuận tối Đa ứng với phương án tối ưu mà bạn tìm Được. (Yêu cầu lấy chính xác 3 chữ số sau dấu phẩy).

Ví dụ:

PROFIT.INP	PROFIT.OUT
5 6	2.520
1 4 172 100	
4 5 184 100	
5 1 252 100	
1 2 12 100	
2 3 10 100	
3 1 6 100	

VẤN ĐỀ: TÔ MÀU ĐỒ THỊ

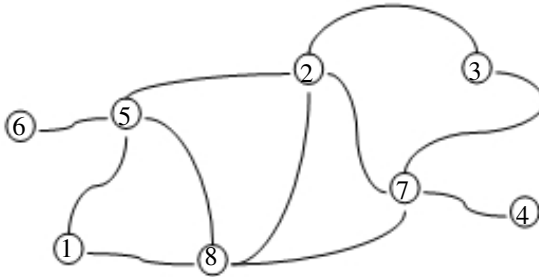
Bài toán:

Tô màu các Đỉnh.

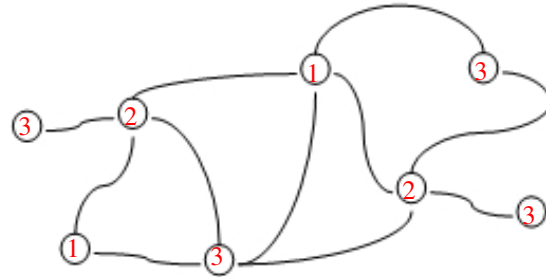
Cho Đồ thị vô hướng G. Tìm cách tô màu các Đỉnh của Đồ thị (mỗi Đỉnh một màu) sao cho thỏa mãn 2 Điều kiện sau:

- Hai Đỉnh có cạnh nối trực tiếp Được tô bởi 2 màu khác nhau.
- Số màu cần dùng là ít nhất.

(Số màu cần dùng ít nhất Được gọi là sắc số của Đồ thị.)



Đồ thị ban Đầu



Đồ thị Được tô bởi 3 màu

Hướng giải quyết:

Với dữ liệu bài toán nhỏ, duyệt nhánh cận là phương pháp cho kết quả tối ưu. Tuy nhiên cách trên sẽ không chạy Được với dữ liệu lớn. Chúng ta có một cách làm khác khá hiệu quả, cho kết quả tối ưu với phần lớn các test - Sử dụng phương pháp tham như sau:

Trước hết Định nghĩa bậc của một Đỉnh là số Đỉnh kề với nó và chưa Được tô màu.

Khởi tạo gán số màu cần dùng bằng 0.

Mọi Đỉnh Đều chưa Được tô.

Lặp

Tăng số màu lên 1 $color := color + 1$

B1: Tìm Đỉnh u có bậc lớn nhất.

B2: Tô màu cho Đỉnh u này màu color.

B3: Lặp

Tìm Đỉnh v có bậc lớn nhất có thể tô màu color.

Tô màu cho Đỉnh v màu color.

Cho Đến khi không còn Đỉnh v nào có thể tô màu color.

Cho Đến khi tô màu hết.

Mô tả thuật toán:

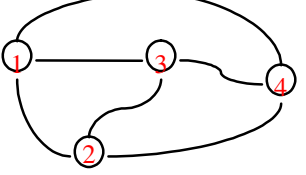
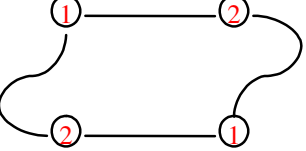
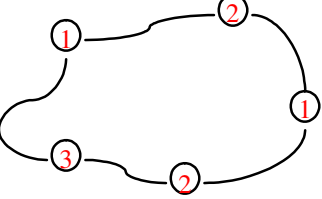
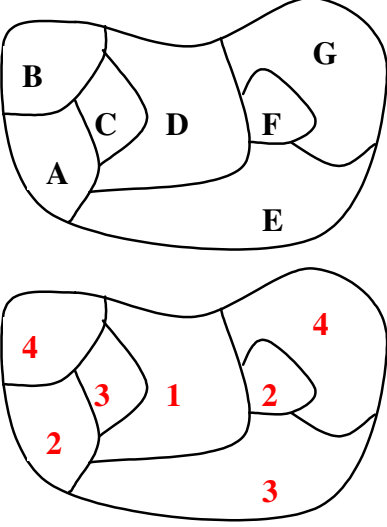
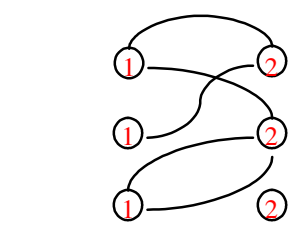
```
Function deg(u:integer):integer;
{Bậc của Đỉnh u}
Begin
  D:=0;
  For v:=1 to n do
    If a[u,v] and C[v]=0 then inc(d);
  deg:=d;
End;

Function canput(u,k:integer):boolean;
{có thể tô Đỉnh u bằng màu k hay không}
Begin
  canput:=false;
  For v:=1 to n do
    If a[u,v] and c[v]=k then exit;
  canput:=true;
End;

Procedure put(u,k:integer);
{Tô Đỉnh u bằng màu k}
Begin
  C[u]:=k;
End;

Procedure process;
Begin
  color:=0;
  Fillchar(c,sizeof(c),0);
  Repeat
    U:=0; maxdeg:=-1;
    For v:=1 to n do
      If C[v]=0 and deg(v)>maxdeg then
        Begin u:=v; maxdeg:=deg(v); end;
    If u<>0 then
      begin
        Inc(color);
        Put(u,color);
        Repeat
          V:=0; maxdeg:=-1;
          For t:=1 to n do
            If (c[t]=0) and canput(t,color) and (deg(t)>maxdeg) then
              Begin v:=t; maxdef:=deg(t); end;
          If v <> 0 then put(v,color);
        Until v = 0;
      End;
    Until u = 0;
  Writeln(color); {số màu tối thiểu cần dùng}
  For i:=1 to n do writeln(I,' : ',c[i])
End;
```

Trên Đây chúng ta Đã xét phương pháp tô màu với Đồ thị vô hướng tổng quát. Ngoài ra còn một số dạng Đồ thị Đặc biệt mà khi tô màu chúng ta phát hiện ra Được những tính chất khá thú vị:

	<ul style="list-style-type: none"> - Đồ thị Đầy Đủ n Đỉnh (là Đồ thị mà giữa 2 Đỉnh bất kì luôn có cạnh nối) có sắc số bằng n.
	<ul style="list-style-type: none"> - Đồ thị vòng với số Đỉnh chẵn có sắc số là 2.
	<ul style="list-style-type: none"> - Đồ thị vòng với số Đỉnh lẻ có sắc số là 3.
	<ul style="list-style-type: none"> - Đồ thị phẳng : là Đồ thị có thể biểu thị các Đỉnh và các cạnh trên một mặt phẳng sao cho các cạnh và các Đỉnh không có Điểm giao nhau nào khác ngoài các Đầu mút. Loại Đồ thị này có sắc số không lớn hơn 4. Do Đó có thể dùng không quá 4 màu Để tô màu một bản Đồ bất kì sao cho 2 vùng Đất khác nhau có màu khác nhau.
	<ul style="list-style-type: none"> - Đồ thị 2 phía bất kì có sắc số là 2. Có thể nhận ra cách tô màu Đơn giản nhất là tô tập Đỉnh bên X bằng màu 1 và tô tập Đỉnh bên Y bằng màu 2. Ngoài ra nếu một Đồ thị vô hướng bất kì có sắc số là 2 thì Đồ thị này có thể phân thành Đồ thị 2 phía.

Bài toán:

Tô màu các cạnh.

Cho Đồ thị vô hướng G . Tô màu các cạnh của Đồ thị bởi 2 màu sao cho:

- Đối với mỗi Đỉnh, nếu xét các cạnh có nối với Đỉnh Đó thì chênh lệch giữa số cạnh Được tô màu này và số cạnh Được tô màu kia không quá 1.
- Đối với toàn bộ Đồ thị, nếu xét tất cả các cạnh thì chênh lệch giữa số cạnh Được tô màu này với số cạnh Được tô màu kia không quá 1.

Tìm cách tô màu tất cả các cạnh thỏa mãn 2 Điều kiện trên.

Hướng giải quyết:

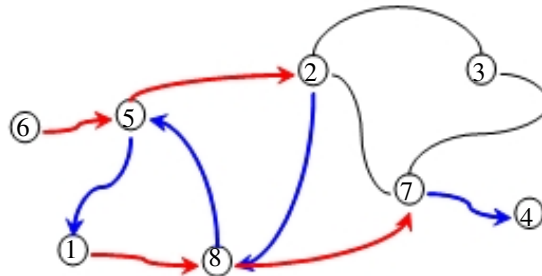
Gọi $\deg(u)$ là bậc của Đỉnh u : bằng số cạnh nối với Đỉnh u và chưa Được tô màu.

Gọi s bằng tổng bậc của tất cả các Đỉnh, dễ thấy s là một số chẵn. Do Đó số Đỉnh bậc lẻ (nếu có) trong Đồ thị luôn là một số chẵn.

Gọi $\text{start}(u)$ là một Đường Đi xuất phát từ Đỉnh u , lần lượt qua các cạnh chưa Được tô màu (mỗi cạnh chỉ qua 1 lần) và kết thúc tại Đỉnh v nếu từ v không thể Đi Được tiếp (tức là các cạnh nối với v hoặc Đã Đi qua hoặc Đã Được tô màu).

Gọi $\text{color1}(u)$ là một phép tô màu theo nguyên tắc: Đỉnh u là Đỉnh có bậc lẻ, dựa vào Đường Đi $\text{start}(u)$ tìm Được, lần lượt tô màu các cạnh trên Đường Đi bằng 2 màu xen kẽ nhau.

Gọi $\text{color2}(u)$ là một phép tô màu theo nguyên tắc: Đỉnh u là Đỉnh có bậc chẵn, dựa vào Đường Đi $\text{start}(u)$ tìm Được, lần lượt tô màu các cạnh trên Đường Đi bằng 2 màu xen kẽ nhau.



(Một phép tô màu $\text{color1}(6)$ ứng với Đường Đi $6 \rightarrow 5 \rightarrow 1 \rightarrow 8 \rightarrow 5 \rightarrow 2 \rightarrow 8 \rightarrow 7 \rightarrow 4$)

Ta chứng minh một số nhận xét quan trọng sau Đây.

1/ Nếu Đỉnh u có bậc lẻ thì $\text{start}(u)$ kết thúc tại một Đỉnh v có bậc lẻ khác.

Giả sử k là số lần Đi qua Đỉnh v trên Đường Đi $\text{start}(u)$. Định nghĩa một lần Đi qua Đỉnh v có nghĩa là từ một Đỉnh v' Đi Đến Đỉnh v rồi từ Đỉnh v Đi Đến Đỉnh v'' . Vì v là Đỉnh kết thúc nên $\deg(v)$ bằng số cạnh có nối với Đỉnh v trên Đường Đi $\text{start}(u)$.

- giả sử u trùng với v thì số cạnh có nối với Đỉnh v là $2k$ cộng thêm một cạnh xuất phát từ u và một cạnh kết thúc tại v . Suy ra $\deg(v) = 2k+2$ là một số chẵn hay $\deg(u)$ là một số chẵn - vô lý. Vậy u khác với v .
- vì u khác v nên số cạnh có nối với Đỉnh v là $2k$ cộng thêm một cạnh kết thúc tại v . Suy ra $\deg(v) = 2k+1$ là một số lẻ. Vậy Đỉnh kết thúc v là một Đỉnh bậc lẻ khác u .

2/ Nếu mọi Đỉnh của Đồ thị có bậc chẵn thì xuất phát từ một Đỉnh u bất kì, $\text{start}(u)$ kết thúc tại chính Đỉnh u .

Cũng chứng minh tương tự như trên, giả sử v là Đỉnh kết thúc khác u .

Gọi k là số lần Đi qua Đỉnh v thì $\deg(v) = 2k + 1$ là một số lẻ, hay v là một Đỉnh bậc lẻ - vô lý (vì mọi Đỉnh của Đồ thị đều có bậc chẵn).

Dựa vào 2 nhận xét trên ta có thuật toán tô màu như sau:

B1/Lặp

Tìm Đỉnh u có bậc lẻ trên Đồ thị.

Thực hiện phép $\text{color1}(u)$.

Bỏ các cạnh Đã Được tô ra khỏi Đồ thị

Cho Đến khi nào tô hết tất cả các cạnh.

B2/Lặp

Tìm Đỉnh u có bậc chẵn (khác 0) trên Đồ thị.

Thực hiện phép $\text{color2}(u)$.

Bỏ các cạnh Đã Được tô ra khỏi Đồ thị

Cho Đến khi nào tô hết tất cả các cạnh.

Tính Đúng Dẫn của thuật toán:

Ta thấy phép tô $\text{color2}(u)$ với u là Đỉnh bậc chẵn không làm thay Đổi chênh lệch 2 màu tại mỗi Đỉnh có cạnh Được tô.

Còn Đối với phép tô $\text{color1}(u)$ có u là Đỉnh bậc lẻ, ta nhận thấy nếu v là Đỉnh kết thúc thì phép tô làm cho chênh lệch giữa 2 màu tại 2 Đỉnh u và v tăng lên 1. Tuy nhiên sau khi bỏ các cạnh Đã tô, u và v trở thành 2 Đỉnh bậc chẵn. Cho nên mỗi Đỉnh u chỉ gọi bởi phép tô $\text{color1}(u)$ không quá 1 lần và mỗi Đỉnh v cũng chỉ là Đỉnh kết thúc trong các phép $\text{color1}(u)$ không quá 1 lần. Do Đó chênh lệch giữa 2 màu tại mỗi Đỉnh không quá 1 - Điều kiện 1 của bài toán Được thỏa mãn

Mặt khác khi tô màu các cạnh ta luôn dùng 2 màu xen kẽ nhau nên sau khi tô xong Điều kiện 2 của bài toán Được thỏa mãn.

Bài toán:

Tô màu các cạnh trên Đồ thị hai phía

Cho Đồ thị 2 phía G, tìm cách tô mỗi cạnh một màu sao cho:

- Các cạnh chung Đầu mút Được tô bởi các màu khác nhau.
- Số màu cần sử dụng là ít nhất.

Hướng giải quyết: Gọi maxDeg là giá trị bậc lớn nhất trong các Đỉnh. Để thấy số màu cần dùng không nhỏ hơn maxDeg . Ta sẽ chỉ ra một cách tô dùng maxDeg màu, và Đây chính là cách tô có số màu ít nhất:

Thực hiện maxDeg lượt tô, lượt thứ i dùng màu i Để tô. Do Đồ thị Đã cho là Đồ thị 2 phía cho nên luôn tìm Được một bộ ghép mà các Đỉnh có bậc bằng lớn nhất Luôn Được ghép. Tô các cạnh thuộc bộ ghép này bằng màu i. Sau Đó bỏ các cạnh này ra khỏi Đồ thị, bậc của các Đỉnh Được ghép thay Đổi. Thực hiện maxDeg lượt như vậy ta tô màu Được Đồ thị thỏa mãn yêu cầu bài toán.

Luyện tập:

Bài toán:

Xếp lịch thi (LICHTHI)

Một số trường Đại học tổ chức học theo tín chỉ. Nếu sinh viên tích lũy Đủ số chứng chỉ cho một số môn quy Định của một ngành là có quyền nhận bằng tốt nghiệp của ngành Đó. Đối với các Đại học như thế, việc học và thi không tổ chức theo lớp mà theo các môn học. Hàng năm nhà trường thông báo các môn sẽ học Để sinh viên tự Đăng ký học các môn học theo ngành mình chọn. Cuối năm nhà trường tổ chức thi cho các môn Đã giảng trong năm. Mỗi môn thi trong một ngày nhưng trong một ngày có thể tổ chức thi nhiều môn. Do một sinh viên có thể Đăng ký thi nhiều môn nên lịch thi cần phải bố trí Để nếu có một sinh viên Đăng ký thi hai môn nào Đó thì các môn Đó không Được thi cùng ngày.

Yêu cầu: Hãy xếp lịch thi sao cho tổng số ngày thi càng ít càng tốt.

Dữ liệu: LICHTHI.INP

- ✂ M, N : lần lượt là số thí sinh và số môn học ($M \leq 1000$, $N \leq 100$);
- ✂ M dòng tiếp theo ghi thông tin về Đăng ký thi của mỗi sinh viên, mỗi dòng gồm N số 0 hoặc số 1 cách nhau một dấu cách, số thứ j trong dòng thứ i là 1 nếu sinh viên thứ i Đăng ký học và thi môn thứ j và bằng 0 nếu sinh viên thứ i không Đăng ký thi môn thứ j.

Kết quả: LICHTHI.OUT

- ✂ Gồm N dòng mỗi dòng có một số tự nhiên. Số ở dòng thứ i là ngày thi tương ứng Đối với môn thứ i.

Ví dụ:

LICHTHI.INP	LICHTHI.OUT
10 8	1
1 0 1 1 1 0 0 0	1
0 1 1 0 1 1 0 0	2
1 0 0 0 1 0 1 1	3
0 1 1 0 1 0 1 0	4
1 0 0 0 1 1 0 1	3
0 0 1 1 1 0 0 0	3
0 0 0 1 1 0 0 1	2
0 1 1 0 1 1 0 0	
1 0 0 0 1 0 1 1	
0 1 0 0 1 1 0 1	

Gợi ý: Xây dựng Đồ thị N Đỉnh Đại diện cho N môn học, 2 Đỉnh có cạnh nối với nhau nếu tồn tại một học sinh Đăng kí thi cả 2 môn học tương ứng với 2 Đỉnh Đó. Tô màu cho Đồ thị ta Được phương án xếp lịch thi.

Bài toán:

Tập ổn Định trong cực Đại.

Cho Đồ thị vô hướng G. Định nghĩa một tập ổn Định trong là tập các Đỉnh thỏa mãn:

- Giữa 2 Đỉnh bất kì không có cạnh nối trực tiếp.
- Nếu thêm vào tập một Đỉnh khác thì tính chất trên không còn Đúng.

Yêu cầu tìm tập ổn Định trong cực Đại - tập ổn Định trong có nhiều nhiều Đỉnh nhất.

Dữ liệu:

- ✂ Dòng Đầu ghi giá trị n,m là số Đỉnh và số cạnh của Đồ thị.
- ✂ m dòng tiếp theo mỗi dòng ghi 2 số u, v cho biết có cạnh nối trực tiếp giữa 2 Đỉnh u và v.

Kết quả:

- ✧ Dòng Đầu ghi số Đỉnh trong tập ổn Định trong cực Đại.
- ✧ Dòng tiếp theo ghi chỉ số các Đỉnh trong tập.

Gợi ý: Duyệt nhánh cận là cách Đơn giản Để giải quyết tuy nhiên với dữ liệu lớn thì bài toán nêu trên chưa có lời giải tối ưu. Ta chỉ có thể tìm một thuật toán tham cho kết quả tốt nhất có thể.

Nếu ta tô màu các Đỉnh của Đồ thị thì các Đỉnh Được tô cùng một màu sẽ cho một tập ổn Định trong. Dựa vào Điều này ta tìm tập Đỉnh cùng màu có nhiều Đỉnh nhất. Tuy nhiên, trong một số trường hợp, tập các Đỉnh này chưa hẳn Đã là tập ổn Định trong cực Đại của Đồ thị. Nhưng ít nhất nó Đã cho ta một cận tốt trước khi sử dụng phương pháp duyệt.

Bài toán:

Trồng cây(GTREE)

Một khu vườn hình chữ nhật Được chia thành $M \times N$ ô Đất (M hàng và N cột). Người ta muốn trồng 2 loại cây cam và chanh tại một số ô trong khu vườn. Tìm cách trồng loại cây nào vào mỗi ô Được trồng sao cho Đảm bảo Điều kiện thâm mĩ: chênh lệch giữa số cây cam và số cây chanh không quá 1 khi:

- Khi nhìn theo bất cứ hàng nào.
- Khi nhìn theo bất cứ cột nào.
- Khi nhìn toàn bộ khu vườn.

Dữ liệu: GTREE.INP

- ✧ Dòng Đầu 2 giá trị M,N cách nhau một dấu cách.
- ✧ Tiếp theo là ma trận 0,1 có kích thước $M \times N$ biểu thị các vị trí cần trồng cây (những vị trí này nhận giá trị 1) và những vị trí không Được trồng cây (nhận giá trị 0).

Kết quả: GTREE.OUT

- ✧ Một cách trồng cây thỏa mãn biểu thị qua Ma trận 0,1,2 có kích thước $M \times N$ trong đó giá trị 0 biểu thị vị trí không Được trồng cây, giá trị 1 biểu thị vị trí trồng cây cam, giá trị 2 biểu thị vị trí trồng cây chanh.

Ví dụ:

GTREE.INP	GTREE.OUT
5 11	01020010002
01010010001	10200001000
10100001000	02010200001
01010100001	00000002010
00000001010	20000100000
00001000000	

Gợi ý: Xét Đồ thị 2 phía (X,Y), bên X có M Đỉnh Đại diện cho M hàng, bên Y có N Đỉnh Đại diện cho N cột. ô(x,y) Được trồng cây thì có cạnh nối (x,y) trên Đồ thị 2 phía. Thực hiện tô màu các cạnh bằng 2 màu ta Được một phương án trồng cây thỏa mãn.

Bài toán: Trang trí đèn Đèn (DEC)

Nhân dịp Giáng sinh và năm mới, một cửa hàng lắp Đặt một đèn Đèn có dạng bảng hình chữ nhật $m \times n$. Trên đèn Đèn có một số vị trí cần lắp Đặt bóng Đèn. Cửa hàng muốn trang trí đèn Đèn bằng các bóng Đèn màu. Để thật Đẹp mắt, cửa hàng muốn các bóng Đèn nằm trên cùng hàng hoặc cùng cột Đèn khác màu nhau! Nhưng Để Đảm bảo sự hài hòa, đèn Đèn không nên có quá nhiều màu.

Bạn hãy giúp cửa hàng trang trí đèn Đèn, sao cho số loại bóng Đèn màu cần sử dụng là ít nhất!

Dữ liệu: DEC.INP

- ✂ Dòng Đầu tiên chứa 2 số nguyên m, n ($1 \leq m, n \leq 100$)
- ✂ m dòng tiếp theo, mỗi dòng chứa n kí tự '0' hoặc '1' thể hiện đèn Đèn, kí tự '1' cho biết vị trí cần lắp Đặt bóng Đèn.

Kết quả: DEC.OUT

- ✂ Dòng Đầu tiên: chứa số nguyên p là số loại bóng Đèn màu ít nhất cần sử dụng.
- ✂ Dòng thứ i trong số m dòng tiếp theo chứa n số nguyên, số thứ j là chỉ số màu của bóng Đèn ở vị trí tương ứng, hoặc là 0 nếu ở vị trí tương ứng không cần lắp Đặt bóng Đèn. Các màu Được Đánh số từ 1 Đến p .

Ví dụ:

DEC.INP	DEC.OUT
5 10	6
1100111010	3 2 0 0 4 5 6 0 1 0
0011000101	0 0 2 3 0 0 0 4 0 1
0001011000	0 0 0 2 0 3 1 0 0 0
0110001010	0 3 1 0 0 0 4 0 2 0
1111010100	2 4 3 5 0 6 0 1 0 0

Bài toán: Sơn cửa (RPAINT)

Phú ông vừa mới xây dựng xong toà biệt thự của mình. Việc trước tiên phú ông muốn làm là Đi sơn các cánh cửa trong ngôi nhà của mình. Toà biệt thự Được mô tả là một kiểu kiến trúc kỳ quái không tuân theo bất kỳ quy luật tự nhiên nào. Nó bao gồm N phòng và 1 hệ thống hành lang nối các phòng với nhau. Mỗi một căn phòng thay vì chỉ có 1 cửa thì lại có tới A_i cánh cửa. Mỗi cánh cửa dẫn ra một hành lang nối tới một phòng khác. Hiện thời thị trường Đang khuyến loại sơn mua 1 tặng 1. Cụ thể là nếu bạn mua 1 hộp sơn màu trắng bạn sẽ Được tặng 1 hộp sơn màu Đen và ngược lại nếu bạn mua 1 hộp sơn màu Đen, bạn sẽ Được tặng 1 hộp sơn màu trắng. Phú ông vừa xây xong biệt thự nên cũng khá là túng tiền. Mà cơ bản tính khí cũng khá là keo kiệt, ông ta quyết Định sơn các cánh cửa theo cách sau Đây:

Nếu cánh cửa X của phòng Y dẫn ra hành lang thông với cánh cửa U của phòng V thì 2 cánh cửa này phải sơn 2 màu khác nhau.

Ngoài ra bên cạnh Đó là yêu cầu về thẩm mỹ, trong mỗi phòng thì số lượng cánh cửa sơn màu trắng và số lượng cánh cửa sơn màu Đen không Được chênh lệch quá 1.

Bên cạnh Đó ông ta muốn tổng chênh lệch giữa số lượng cánh cửa sơn màu trắng và số lượng cánh cửa sơn màu Đen là nhỏ nhất có thể Được vì như vậy Đỡ tốn tiền mua sơn. Bạn hãy giúp phú ông giải quyết bài toán kinh tế này .

Dữ liệu: RPAINT.INP

- ✂ Dòng thứ nhất ghi số N là số phòng trong tòa biệt thự.
- ✂ Dòng thứ i trong N dòng tiếp theo ghi thông tin về các cánh cửa của phòng thứ i. Đầu tiên là số cửa A_i . Tiếp theo là A_i số nguyên cho biết phòng i có các cánh cửa dẫn Đến phòng nào.

Kết quả: RPAINT.OUT

- ✂ Dòng thứ nhất, nếu có phương án thỏa mãn ghi ra “YES”, nếu không có ghi ra “NO SOLUTION”.
- ✂ Nếu dòng thứ nhất ghi “YES” thì tiếp theo là N dòng. Dòng i mô tả về cách sơn cửa của phòng thứ i gồm A_i kí tự ‘W’ / ‘B’ tương ứng với việc cánh cửa Được sơn trắng hoặc Đen. Thứ tự ghi trong output tương ứng với thứ tự ghi trong input. Mỗi kí tự ‘W’ / ‘B’ trong output Được ghi cách nhau Đúng một dấu cách.

Giới hạn:

- ✂ $1 \leq N \leq 10000$
- ✂ Tổng số hành lang ≤ 100000

Ví dụ:

RPAINT.INP	RPAINT.OUT
5	YES
3 2 3 4	B W B
3 1 3 5	W B W
4 1 2 4 5	B W W B
3 1 3 5	W B B
3 2 3 4	B W W

(Với cách sơn như thế này thì chênh lệch giữa 2 màu bằng 0)

Bài toán:

Thị trường (MAYORS)

Một Đất nước có N thành phố, vị trí của mỗi thành phố Được thể hiện bởi một Điểm trên mặt phẳng tọa Độ. Tổng thống vừa Đắc cử muốn bổ nhiệm mỗi thành phố một thị trưởng. Tổng thống muốn bổ nhiệm cả các thị trưởng nam và nữ. Với mỗi Đường thẳng ngang hoặc dọc (song song với trục tọa Độ), gọi Độ bình Đẳng giới là trị tuyệt Đối của hiệu giữa số thị trưởng nam và thị trưởng nữ của các thành phố nằm trên Đường thẳng Đó.

Để chứng tỏ tinh thần bình Đẳng giới của Đất nước, tổng thống muốn bổ nhiệm các thị trưởng sao cho tổng Độ bình Đẳng giới Đối với các Đường thẳng ngang và dọc là nhỏ nhất.

Bạn hãy viết chương trình giúp tổng thống thực hiện nhiệm vụ này.

VẤN ĐỀ: THUẬT TOÁN FORD BELLMAN KẾT HỢP QUEUE VÒNG.

Bài toán: Cho Đồ thị có hướng không có chu trình âm, tìm Đường Đi có trọng số nhỏ nhất giữa 2 Đỉnh s và t cho trước trên Đồ thị.

Hướng giải quyết: Thuật toán FORD BELLMAN Được sử dụng trong bài toán tìm Đường Đi trọng số nhỏ nhất trên Đồ thị không có chu trình âm, thuật toán mô tả như sau:

```
for v:=1 to n do d[v] := +∞; {d[v] là nhãn trọng số Đường Đi ngắn nhất từ s tới v}
d[s] := 0;
repeat
  stop:=true;
  for u:=1 to n do
    for v:=1 to n do
      if a[u,v] then {nếu có cung (u,v)}
        if d[v] > d[u] + c[u,v] then {Điều kiện Để tối ưu nhãn}
          begin
            d[v]:=d[u] + c[u,v];
            trace[v]:=u; {mảng truy vết Đường Đi}
            stop:=false;
          end;
until stop;
```

Tư tưởng của thuật toán FORD BELLMAN là làm tốt dần nhãn Đỉnh v (cho Đến khi tối ưu) sử dụng các nhãn Đỉnh u Đã Được làm tốt hơn trước Đó ($d[v] := d[u] + c[u,v]$). Điều Đó có nghĩa là nếu ta có tập Q chứa những Đỉnh u Đã Được làm tốt hơn ở bước trước Đó thì ta chỉ cần duyệt những Đỉnh u trong tập A này, sau Đó kiểm tra có thể sử dụng nhãn Đỉnh u Đó Để tối ưu thêm những Đỉnh v nào khác hay không. Nếu tối ưu Được nhãn Đỉnh v thì thêm v vào tập Q Để sử dụng cho lần tối ưu sau.

Tư tưởng kết hợp với Queue Được mô tả như sau:

```
Ban Đầu Queue chỉ có Đỉnh s
Lặp
  Lấy một Đỉnh u trong Queue
  Bỏ Đỉnh u trong Queue
  Duyệt mọi cung (u,v) xem có tối ưu Được nhãn Đỉnh v nào không, Nếu
    có thì thêm v vào Queue (nếu trong Queue chưa có v).
Cho Đến khi Queue rỗng.
```

Dễ thấy Queue có tối Đa n phân tử nên ta có thể dùng Queue vòng Để cài Đặt thuật toán

```
x:=0; y:=0; Queue[0]:=s; {khởi tạo Queue}
fillchar(inqueue,sizeof(inqueue),false); {mảng Đánh dấu Đỉnh Đang có
trong Queue}
inqueue[s]:=true;
for v:=1 to n do d[v]:=+∞;
d[s]:=0;
While x <> (y+1) mod n do
  Begin
    u:=queue[x];
    inqueue[u]:=false; {lấy Đỉnh u ra khỏi Queue}
    x:=(x +1) mod n;
    For v:=1 to n do
      If a[u,v] then {Nếu có cung (u,v)}
        If d[v] > d[u] + c[u,v] then
          Begin
            d[v]:=d[u] + c[u,v];
            trace[v]:=u;
            If not inqueue[v] then
              Begin
                y:=(y + 1) mod n;
                queue[y]:=v;
                inqueue[v]:=true;
              End;
            End;
          End;
    End;
  End;
```

Thuật toán FORD BELLMAN kết hợp Queue vòng có thời gian hoạt Động rất hiệu quả so với thuật toán FORD BELLMAN Đơn thuần. Nó không chỉ ứng dụng trong các bài tìm Đường Đi trọng số nhỏ nhất mà còn Được dùng trong các bài toán sử dụng tư tưởng FORD BELLMAN nói chung.

Luyện tập:

Bài toán:

Vị trí tốt nhất (BESTSPOT)

Bessie, luôn luôn muốn cuộc sống của mình tốt hơn , Đã thấy rõ rằng cô ta thật sự rất thích ghé thăm F ($1 \leq F \leq P$) cánh Đồng yêu thích F_i trong tổng số P ($1 \leq P \leq 500; 1 \leq F_i \leq P$) cánh Đồng (Được Đánh số từ 1-> P) thuộc sở hữu của nông dân John. Bessie biết rằng cô ấy có thể xác Định Được C ($1 \leq C \leq 8000$) con Đường hai chiều (Được Đánh số 1 .. C) kết nối tất cả các cánh Đồng trong toàn bộ nông trại. Ứng với mỗi con Đường P_i là thời gian Đi T_i ($1 \leq T_i \leq 892$) và nối 2 cánh Đồng a_i và b_i ($1 \leq a_i \leq P; 1 \leq b_i \leq P$).

Bessie muốn tìm cánh Đồng tốt nhất Để ngủ thỏa mãn bình quân thời gian Để Đi Đến F cánh Đồng yêu thích của cô ta là nhỏ nhất.

Dữ liệu: BESTSPOT.INP

- ✂ Dòng 1: 3 số nguyên P,F,C
- ✂ Dòng 2..F+1: Dòng i+2 chứa 1 số Nguyên F_i
- ✂ Dòng F+2..C+F+1 : Mỗi dòng chứa 3 số Nguyên a_i, b_i, F_i mô tả 1 con Đường 2 chiều là thời gian di chuyển giữa chúng.

Kết quả: BESTSPOT.OUT

- ✂ Gồm 1 dòng duy nhất là cánh Đồng Được chọn . nếu có nhiều kết quả , chọn cánh Đồng có chỉ số nhỏ nhất.

✂

Ví dụ:

BESTSPOT.INP	BESTSPOT.OUT
13 6 15	10
11	
13	
10	
12	
8	
1	
2 4 3	
7 11 3	
10 11 1	
4 13 3	
9 10 3	
2 3 2	
3 5 4	
5 9 2	
6 7 6	
5 6 1	
1 2 4	
4 5 3	
11 12 3	
6 10 1	
7 8 7	

001 00000 000 000 000 0000

VẤN ĐỀ: THUẬT TOÁN DIJTRA VỚI CÁC ĐỈNH ẢO

Làm quen với dạng toán này qua các ví dụ:

Bài toán: Tuyến bay (AIRLINES)

Có N thành phố và M Đường hàng không hai chiều giữa một số cặp thành phố nào Đó, các Đường bay Được quản lý bởi 16 hãng hàng không. Các thành phố Được Đánh số từ 1 tới N ($N \leq 100$) và các hãng Được Đánh số từ 1 tới 16.

Được biết chi phí bay trực tiếp giữa hai thành phố i, j bất kỳ (nếu như có Đường bay) là C . Nếu Đang Đi máy bay của một hãng Đến sân bay nào Đó rồi chuyển sang máy bay của hãng khác thì sẽ phải mất thêm một khoản phụ phí A .

Yêu cầu: Cho trước hai thành phố S và F , hãy tìm hành trình bay từ thành phố S Đến thành phố F với chi phí ít nhất. Với giả thiết rằng luôn luôn tồn tại cách bay từ S tới F .

Dữ liệu: AIRLINES.INP

- ✂ Dòng 1 ghi sáu số nguyên dương N, M, C, A, S, F . ($1 \leq A, C \leq 100$)
- ✂ M dòng tiếp theo, mỗi dòng có dạng $u \ v \ k_1 \ k_2 \dots$ cho biết rằng giữa thành phố u và thành phố v có Đường bay và k_1, k_2, \dots là số hiệu các hãng sở hữu Đường bay Đó

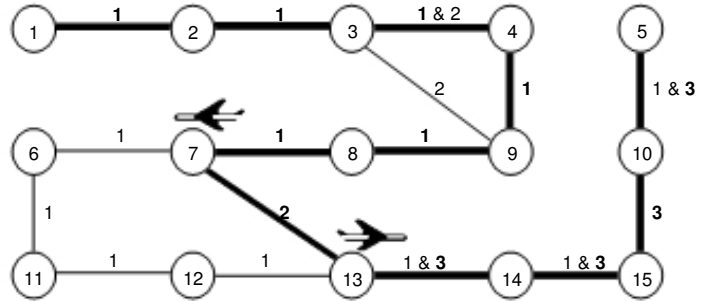
Kết quả: Ghi ra file văn bản AIRLINES.OUT. Trong Đó:

- ✂ Dòng 1: Ghi chi phí tối thiểu phải trả
- ✂ Các dòng tiếp theo, mỗi dòng ghi một bộ ba i, j, k . Thể hiện tại bước Đó sẽ bay từ thành phố i Đến thành phố j bởi máy bay của hãng k . Thứ tự các dòng phải theo Đúng thứ tự bay trong hành trình.

Các số trên một dòng của Input/Output file ghi cách nhau ít nhất một dấu cách.

Ví dụ: Với mạng lưới Đường không như dưới Đây: cần Đi từ thành phố 1 Đến thành phố 5. Chi phí Đường bay trực tiếp giữa hai thành phố bất kỳ $C = 3$, phụ phí chuyển tuyến $A = 2$. Các số ghi bên cạnh các Đường bay trực tiếp là tên các hãng sở hữu Đường bay Đó.

AIRLINES.INP	AIRLINES.OUT
15 16 3 2 1 5	37
1 2 1	1 2 1
2 3 1	2 3 1
3 4 1 2	3 4 1
3 9 2	4 9 1
4 9 1	9 8 1
5 10 1 3	8 7 1
6 7 1	7 13 2
6 11 1	13 14 3
7 8 1	14 15 3
7 13 2	15 10 3
8 9 1	10 5 3
10 15 3	
11 12 1	
12 13 1	
13 14 1 3	
14 15 1 3	



Gợi ý: Xây dựng Đồ thị các Đỉnh (u,k) với ý nghĩ hành khách Đã bay chuyến bay của hãng k Đến thành phố u.

Đỉnh (u, k) có cạnh nối với Đỉnh (u', k) nếu hãng máy bay k có tuyến bay từ u Đến u' .
cạnh này có trọng số là C .

Đỉnh (u, k) có cạnh nối với Đỉnh (u, k') nếu hãng máy bay k' có tuyến bay xuất phát từ u . cạnh này có trọng số là A .

Kết quả là Đường Đi ngắn nhất trong số các Đường Đi ngắn nhất từ Đỉnh (S,k) Đến (F,k') ($1 \leq k, k' \leq 16$)

Bài toán:

Chuyển hàng (CARGO)

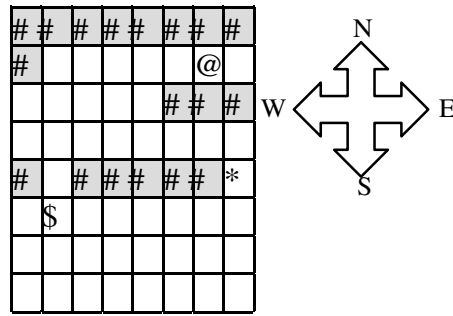
Bản Đồ một kho hàng hình chữ nhật kích thước $m \times n$ Được chia thành các ô vuông Đơn vị (m hàng, n cột: các hàng Đánh số từ trên xuống dưới, các cột Đánh số từ trái qua phải). Trên các ô của bản Đồ có một số ký hiệu:

- Các ký hiệu # Đánh dấu các ô Đã có một kiện hàng xếp sẵn, ○
- Một ký hiệu *: Đánh dấu ô Đang có một xe Đầy
- Một ký hiệu \$: Đánh dấu ô chứa kiện hàng cần xếp
- Một ký hiệu @: Đánh dấu vị trí ô mà cần phải xếp kiện hàng B vào ô Đó ○

Các ký hiệu dấu chấm ".": Cho biết ô Đó trống

Cần phải dùng xe Đẩy ở * Để Đẩy kiện hàng ở \$ Đến vị trí @ sao cho trong quá trình di chuyển cũng như Đẩy hàng, không chạm vào những kiện hàng Đã Được xếp sẵn. (Xe Đẩy có thể di chuyển sang một trong 4 ô chung cạnh với ô Đang Đứng). Nếu có nhiều phương án thì chỉ ra một phương án sao cho xe Đẩy phải di chuyển qua ít bước nhất.

Các hướng di chuyển Được chỉ ra trong hình dưới Đây



Dữ liệu: CARGO.INP

- ✂ Dòng 1: Ghi hai số nguyên dương m, n cách nhau một dấu cách ($m, n \leq 80$)
- ✂ m dòng tiếp theo, dòng thứ i ghi n ký hiệu trên hàng thứ i của bản Đồ theo Đúng thứ tự từ trái qua phải. Các ký hiệu Được ghi liền nhau

Kết quả: CARGO.OUT

- ✂ Dòng 1: Ghi số bước di chuyển xe Đẩy Để thực hiện mục Đích yêu cầu, nếu không có phương án khả thi thì dòng này ghi số -1
- ✂ Dòng 2: Nếu có phương án khả thi thì dòng này ghi các ký tự liền nhau thể hiện hướng di chuyển của xe Đẩy R (East, West, South, North). Các chữ cái thường (e,w,s,n) thể hiện bước di chuyển không Đẩy hàng, các chữ cái in hoa (E,W,S,N) thể hiện bước di chuyển có Đẩy hàng.

Ví dụ:

CARGO.INP	CARGO.OUT
8 8	23
# @ .	sswwwwwNNNwnEseN
. . ###	wnEEEE
# . ##### *	
.\$...	

CARGO.INP	CARGO.OUT
5 9	22
@ . . .	eeNNNsseeeennnnwww
. ## . ### . #	WWW
. . . # . .	
. ## \$ ### . #	
. * . . .	

Gợi ý: Để Đẩy kiện hàng Đến ô vuông (x,y) bất kì mà số bước Đi là ít nhất thì ở trạng thái cuối cùng, xe Đẩy hoặc nằm trên, dưới, trái hoặc phải Đối với kiện hàng (xe Đẩy và kiện hàng nằm kề nhau).

Xây dựng Đồ thị các Đỉnh (x,y,k) với ý nghĩa: xe Đẩy Đang ở phía k so với kiện hàng Đang ở ô (x,y) . ($1 \leq k \leq 4$ ứng với 4 hướng trên, dưới, phải, trái.)

Xây dựng các cạnh của Đồ thị: Đỉnh (x,y,k) có cạnh nối tới Đỉnh (x',y',k) nếu xe Đẩy có thể Đẩy kiện hàng ở ô (x,y) 1 nấc theo hướng k Để tới ô (x',y') . cạnh này có trọng số bằng 1.

Đỉnh (x,y,k) có cạnh nối tới Đỉnh (x,y,k') nếu xe Đẩy có thể di chuyển từ vị trí nằm ở phía k so với kiện hàng Đến vị trí mới nằm ở phía k' so với kiện hàng. Cạnh này có

trọng số bằng Độ dài Đường Đi ngắn nhất mà xe Đẩy phải di chuyển Để thay Đổi vị trí so với kiện hàng

Với (x_1, y_1) là vị trí của kiện hàng ban Đầu, (x_2, y_2) là vị trí của kiện hàng cần di chuyển Đến. Duyệt k_1, k_2 nhận giá trị 1 trong 4 hướng

Gọi $d(k_1, k_2)$ = khoảng cách ngắn nhất từ Đỉnh (x_1, y_1, k_1) Đến Đỉnh (x_2, y_2, k_2) + khoảng cách ngắn nhất Để xe Đẩy di chuyển từ vị trí ban Đầu (x_0, y_0) Đến vị trí nằm phía k_1 so với kiện hàng.

Giá trị nhỏ nhất của d chính là số bước di chuyển ngắn nhất của xe Đẩy

Bài toán:

Điệp viên (SPY)

Địa bàn hoạt Động của một Điệp viên là một khu phố mà ở Đó chỉ có các Đường phố ngang, dọc tạo thành một lưới ô vuông. Với mục Đích bảo mật, thay vì tên Đường phố, Điệp viên Đánh số các phố ngang từ 0 Đến m và các phố dọc từ 0 Đến n . ở một số ngã ba hoặc ngã tư có các trạm kiểm soát. Anh ta Đang Đứng ở nút giao của hai Đường (i_1, j_1) (j_1 - Đường ngang; i_1 - Đường dọc) và cần tới Điểm hẹn ở giao của hai Đường (i_2, j_2) . Để tránh bị theo dõi, Đường Đi phải không qua các trạm kiểm soát và cứ tới chỗ rẽ thì nhất thiết phải Đổi hướng Đi, thậm chí có thể sang Đường và Đi ngược trở lại. Việc Đổi hướng chỉ Được thực hiện ở ngã ba hoặc ngã tư. Hãy xác Định Đường Đi ngắn nhất tới Điểm hẹn hoặc cho biết không có Đường Đi Đáp ứng Được yêu cầu Đã nêu.

Dữ liệu: SPY.INP

- ✂ Dòng Đầu: $m \ n \ i_1 \ j_1 \ i_2 \ j_2$ ($0 \leq m, n \leq 100$)
- ✂ Các dòng sau: mỗi dòng 2 số i, j (toạ Độ trạm kiểm soát).

Kết quả: SPY.OUT

- ✂ Dòng Đầu: Độ dài Đường Đi ngắn nhất hoặc thông báo NO nếu không có Đường Đi.
- ✂ Các dòng sau: mỗi dòng 2 số i, j chỉ nút tiếp theo cần tới theo Đường Đi tìm Được, bắt Đầu là $i_1 \ j_1$ và kết thúc là $i_2 \ j_2$.

Ví dụ:

SPY.INP	SPY.OUT
4 5 0 0 5 4	13
0 1	0 0
0 4	1 0
2 2	1 1
2 3	1 0
4 0	2 0
5 2	2 1
5 3	3 1
-1	3 2
	4 2
	4 3
	3 3
	4 3
	4 4
	5 4

Gợi ý: Xét Đồ thị các Đỉnh (x,y,k) với ý nghĩa: Điệp viên vừa Đi từ hướng k tới và Đang Đứng tại vị trí (x,y) . (k nhận giá trị từ 1 Đến 4 tương ứng với 4 hướng).

Luyện tập

Bài toán:

Hành trình rẻ nhất (ERP)

Thành phố Peace vừa Đưa vào áp dụng một hệ thống Điện tử tự Động tính lệ phí sử dụng lệ phí Đường giao thông. Một hệ thống Được triển khai Để phát hiện xe của bạn rẽ trái, rẽ phải, Đi thẳng hoặc quay Đầu và mỗi thao tác như vậy phải trả một lệ phí tương ứng. Đó là 1 \$ cho mỗi lần rẽ trái, 5\$ cho mỗi lần rẽ phải, Đi thẳng về phía trước là miễn phí, quay Đầu xe là bị cấm, ngoại trừ tình huống ở cuối phố khi không còn có thể Đi thẳng, rẽ trái, hoặc rẽ phải Được nữa. Trong trường hợp ngoại lệ bắt buộc phải quay Đầu xe, bạn phải trả lệ phí 10\$. Bạn Được mời Để thiết kế và Đưa ra chỉ dẫn cho người Đi xe Đi sao cho phải trả lệ phí là ít nhất giữa 2 Điểm bất kỳ trong thành phố. Rất may hệ thống Đường giao thông của Peace có dạng bàn cờ.

ERP.INP	ERP.OUT
8 11	8
.#####..	
.#...#..	
.#...#..	
.#E#####..	
.#...	
.##F#...	

Ở ví dụ trên ký tự '#' Để chỉ ra Đoạn Đường phố, còn ký tự '.' chỉ ra Đoạn Đường không là Đường. Các Đoạn Đường phố Đều có thể Đi cả 2 chiều. Ký tự 'E' chỉ ra vị trí xuất phát của xe ô tô có Đầu xe hướng về phía Đông, còn ký tự 'F' chỉ ra vị trí kết thúc. Lộ trình phải trả là 8\$, trong Đó ta phải thực hiện 3 lần rẽ trái và 1 lần rẽ phải Để Đến Đích. Ta có thể thực hiện cách Đi rẽ phải 2 lần Để Đến Đích, nhưng cách Đó lệ phí phải trả là 10\$.

Chiều cao và chiều rộng của bản Đồ ít nhất là 4 và không quá 30. Bản Đồ có Đúng 1 Điểm xuất phát và 1 Điểm kết thúc. Luôn có Đường Đi từ Điểm xuất phát Đến Điểm kết thúc. Luôn có một khung gồm toàn ký tự '.' viền bản Đồ Để không thể vượt ra ngoài bản Đồ.

Dữ liệu: ERP.INP

- ✂ Dòng thứ nhất chứa 2 số nguyên dương h, w theo thứ tự là chiều cao h và chiều rộng của bản Đồ.
- ✂ Mỗi dòng trong h dòng tiếp theo chứa w ký tự. Mỗi ký tự chỉ là một trong số các ký tự sau:
 - '.': vị trí không có Đường
 - '#': vị trí có Đường của bản Đồ.
 - 'E': vị trí xuất phát, xe hướng Đầu về phía Đông.

- 'W': vị trí xuất phát, xe hướng Đầu về phía Tây.
- 'N': vị trí xuất phát, xe hướng Đầu về phía Bắc.
- 'S': vị trí xuất phát, xe hướng Đầu về phía Nam.
- 'F': vị trí kết thúc.

✂ Trong bản Đồ có Đúng 1 trong 4 ký tự 'E', 'W', 'N', 'S'. Và Đúng 1 ký tự F.

Kết quả: ERP.OUT

- ✂ Duy nhất một số là lệ phí của lộ trình rẻ nhất Đi từ vị trí xuất phát Đến vị trí kết thúc.

Bài toán:

Tạo xâu

Cho hai tập A và B chứa các xâu. $A=(p_1,p_2,\dots,p_n)$; $B=(q_1,q_2,\dots,q_m)$;
Một xâu S Được gọi là có thể nhận Được từ A nếu S có thể tạo thành bằng cách ghép các xâu trong A. Tương tự, một xâu S Được gọi là có thể nhận Được từ B nếu S có thể tạo thành bằng cách ghép các xâu trong B. Ví dụ:

$A=('12', '43', '5')$, $B=('124', '31', '25')$

$S='1254312'$ là xâu có thể nhận Được từ A nhưng không nhận Được từ B

$S='12425'$ là xâu có thể nhận Được từ B nhưng không nhận Được từ A

Bài toán Đặt ra là: Hãy tìm một xâu S nhận Được từ A và B với Độ dài ngắn nhất.

Bài toán:

N thang máy (LIFT)

Một tòa nhà cao tầng có N thang máy, mỗi thang máy nối liền Đúng 2 tầng với nhau và không dừng lại những tầng nằm giữa 2 tầng này. Vận tốc của các thang máy là như nhau: 5 giây qua một tầng.

Thời Điểm bắt Đầu, mỗi thang máy Đầu ở tầng thấp nhất và chúng cùng bắt Đầu di chuyển lên tầng trên. Sau khi tới tầng trên, ngay lập tức lại chuyển xuống tầng dưới, rồi lại lên tầng trên, cứ như thế...

Mình Đang ở tầng 1 (tầng thấp nhất) và muốn nhanh chóng lên tầng trên cùng của ngôi nhà. Anh ta thay Đổi thang máy chỉ trên những tầng chung của hai thang máy và nếu thang máy kia tại thời Điểm này cũng tới tầng này, việc chuyển thang máy khi Đó coi như không tốn thời gian nào.

Yêu cầu: Hãy viết chương trình tính thời gian ít nhất mà Mình có thể lên tới tầng trên cùng của tòa nhà.

Dữ liệu: LIFT.INP

- ✂ Dòng Đầu tiên chứa 2 số K và N, cách nhau dấu cách, là số tầng và số thang máy của tòa nhà ($2 \leq K \leq 1000$, $1 \leq N \leq 50000$).
- ✂ Trên mỗi một trong N dòng tiếp theo, mô tả một thang máy ghi hai số nguyên A và B cách nhau dấu cách, ($1 \leq A \leq B \leq K$), nghĩa là thang máy này di chuyển giữa hai tầng A và B.
- ✂ Không có hai thang máy nào khác nhau mà lại di chuyển giữa 2 tầng như nhau.

Chú ý: Dữ liệu vào luôn Đảm bảo có nghiệm.

Kết quả: LIFT.OUT

- ✂ Ghi trên dòng một, thời gian ít nhất (tính theo giây) mà Minh có thể lên tới tầng trên cùng của tòa nhà.

Ví dụ:

LIFT.INP	LIFT.INP	LIFT.INP
10 4	10 3	20 5
1 5	1 5	1 7
5 10	3 5	7 20
5 7	3 10	4 7
7 10		4 10
		10 20
LIFT.OUT	LIFT.OUT	LIFT.OUT
45	105	150

Bài toán:

Hành trình nhanh (JOURN)

Cho N thành phố. Có những con Đường một chiều nối một số thành phố với nhau. N thành phố Được Đánh số từ 1 Đến N. Vào những ngày lễ, xe cộ Đi theo Đúng chiều giao thông, và trong những ngày chẵn thì phải Đi theo hướng ngược lại. Độ dài các con Đường giữa hai thành phố là số nguyên Được Đo bằng khoảng thời gian Đi hết con Đường Đó (không phụ thuộc vào hướng Đi, tốc Độ của các loại xe cộ trên một con Đường là như nhau). Viết chương trình tìm tuyến từ thành phố A tới thành phố B mà thời gian tới Được B càng sớm càng tốt. Biết ngày Đầu tiên của hành trình là một ngày lễ. Thời gian Đi trong mỗi ngày không vượt quá 12 giờ. Đêm cần nghỉ trong một thành phố. Hành trình có thể Được tiếp tục trong ngày hôm sau.

Dữ liệu: JOURN.INP

- ✂ Dòng Đầu tiên chứa hai số lần lượt là số hiệu của thành phố A và thành phố B.
✂ Dòng thứ hai chứa N là số thành phố và K là số các con Đường ($1 \leq K \leq 1000$).
✂ K dòng còn lại chứa thông tin về các con Đường, mỗi dòng chứa ba số là số hiệu hai thành phố Được nối với nhau bởi một con Đường và khoảng thời gian tính bằng giờ Để Đi hết con Đường Đó, chiều của con Đường từ thành phố ghi trước Đến thành phố ghi sau.

Kết quả: JOURN.OUT

- ✂ Mỗi dòng thể hiện một con Đường lần lượt Đi trên hành trình bằng 4 số: số hiệu thành phố bắt Đầu, số hiệu thành phố kết thúc con Đường Đó, số ngày (là ngày thứ mấy kể từ khi bắt Đầu xuất phát từ A) và thời gian Đi trên con Đường Đó.

Ví dụ:

JOURN.INP	JOURN.OUT
1 3	1 5 1 10
6 7	5 4 1 1
1 2 9	4 3 3 4
1 6 2	
1 5 19	
5 4 1	
4 6 2	
4 3 4	
2 3 5	

Bài toán:

Sắp xếp (SORT)

Cho một dãy số. Bạn cần sắp xếp dãy số bằng cách Đổi chỗ các cặp phần tử. Chi phí Để Đổi chỗ phần tử hai ở vị trí i và vị trí j là C_{ij} .

Nhiệm vụ của bạn là tìm chi phí nhỏ nhất Để có thể sắp xếp dãy số theo thứ tự tăng dần.

Dữ liệu: SORT.INP

- ✂ Dòng Đầu tiên chứa dãy số cần sắp xếp, có số phần tử không vượt quá 7.
- ✂ Dòng thứ i trong số N dòng tiếp theo chứa N số nguyên, số thứ j cho biết C_{ij} , chi phí Để Đổi chỗ phần tử ở vị trí thứ i và vị trí thứ j . Biết N là số phần tử của dãy số, các phần tử Được Đánh số từ 1 Đến N từ trái sang phải. $0 \leq C_{ij} \leq 999$, $C_{ii}=0$ và $C_{ij}=C_{ji}$.

Kết quả: SORT.OUT

- ✂ In ra một số nguyên dương duy nhất: tổng chi phí nhỏ nhất Để sắp xếp dãy số theo thứ tự tăng dần.

Ví dụ:

SORT.INP	SORT.OUT
1 2 3 4 6 5	4
0 1 2 3 4 5	
1 0 1 2 3 4	
2 1 0 1 2 3	
3 2 1 0 1 2	
4 3 2 1 0 900	
5 4 3 2 900 0	

Gợi ý: Xem mỗi Đỉnh là một trạng thái hoán vị của N số (có tất cả $N!$ Đỉnh). Hai Đỉnh có cạnh nối với nhau nếu hai trạng thái hoán vị tương ứng chỉ khác nhau ở hai phần tử u và v bị Đổi chỗ cho nhau, trọng số của cạnh này bằng $C_{u,v}$. Tìm Đường Đi ngắn nhất từ Đỉnh ứng với dãy cần sắp xếp Đến Đỉnh ứng với dãy sau khi Đã sắp không giảm.

Chú ý: do số Đỉnh có thể lớn ($= 7!$) nên có thể dùng DIJTRA kết hợp cấu trúc dữ liệu Heap.

VẤN ĐỀ: MỘT SỐ ỨNG DỤNG THUẬT TOÁN DIJTRA

Bài toán:

Đường Đi giới hạn

Mạng lưới giao thông của thành phố biểu diễn bằng Đồ thị N Đỉnh (Đại diện cho N thành phố) và M cạnh (Đại diện cho M cây cầu). Mỗi cây cầu có giới hạn Độ cao tối Đa của xe khi chạy qua nó. Hay nói cách khác mỗi cạnh trong Đồ thị ứng với một trọng số nguyên dương là chiều cao của cây cầu tương ứng. Tìm Đường Đi từ s Đến t (cho trước) sao cho chiều cao của xe có thể Đi qua là lớn nhất có thể.

Gợi ý: Gọi $H[i]$ là chiều cao lớn nhất có thể của xe Đi từ s Đến i.

Khởi tạo gán $H[s] = +\infty$ và $H[i] = 0$ với $i \neq s$

Thuật toán sửa nhãn tương tự thuật toán DIJTRA

```
Repeat
  u:=0; max:=0;
  for v:=1 to n do
    if free[v] and (h[v] > max) then
      begin
        max:=h[v]; u:=v;
      end;
  if u=0 then break;
  free[u]:=false;
  for v:=1 to n do
    if a[u,v] then
      if h[v] < min(h[u],c[u,v]) then
        begin
          h[v]:=min(h[u],c[u,v]);
          trace[v]:=u;
        end;
  until false;
```

Bài toán:

Tổng số Đường Đi ngắn nhất

Cho Đồ thị vô hướng G với các cạnh có trọng số, 2 Đỉnh s,t cho trước. Tính số Đường Đi ngắn nhất từ s tới t. (2 Đường Đi khác nhau nếu thứ tự các Đỉnh trên 2 Đường Đi khác nhau)

Hướng giải quyết: Kết hợp DIJTRA và Quy hoạch Động

Theo thuật toán DIJTRA, $d[i]$ là khoảng cách ngắn nhất từ s Đến Đỉnh i. Ban Đầu $d[v] = +\infty$ ($\forall v \neq s$) và $d[s] = 0$.

Quy hoạch Động gọi $f[i]$ là số Đường Đi ngắn nhất từ s Đến Đỉnh i. Ban Đầu $f[v] = 0$ ($\forall v \neq s$) và $f[s] = 1$.

Trong chương trình DIJTRA:

Mỗi khi tìm Được Đường Đi mới có Độ dài ngắn hơn ($d[v] > d[u] + c[u,v]$) ta tiến hành thay Đổi $d[v] := d[u] + c[u,v]$ Đồng thời $f[v] := f[u]$.

Mỗi khi tìm Được 2 Đường Đi có Độ dài bằng nhau ($d[v] = d[u] + c[u,v]$) thay Đổi $f[v] := f[v] + f[u]$.

$f[t]$ là kết quả cần tìm.

Lập trình: Đoạn chương trình DIJTRA kết hợp Quy hoạch Động.

```
For v:=1 to n do d[v]:=maxlongint; d[s]:=0;
For v:=1 to n do f[v]:=0; f[s]:=1;
Fillchar(Free,sizeof(Free),true);

Repeat
  U:=0; mi:=maxlongint;
  For v:=1 to n do
    If (Free[v]) and (d[v]<mi) then
      Begin mi:=d[v]; u:=v; end;
  If u=0 then break;
  Free[u]:=false;
  For v:=1 to n do
    If Free[v] then
      If d[v]>d[u] + c[u,v] then
        Begin d[v]:=d[u] + c[u,v]; f[v]:=f[u]; end
      Else
        if d[v] = d[u] + c[u,v] then f[v]:=f[v] + f[u];
Until false;
```

Bài toán:

K Đường Đi ngắn nhất (PATHK)

Vùng Đất X có N thành phố ($4 \leq N \leq 800$) Được Đánh số từ 1 Đến N. Giữa hai thành phố có thể có Đường nối trực tiếp hoặc không. Các con Đường này Được Đánh số từ 1 Đến M ($1 \leq M \leq 4000$). Ban lãnh Đạo thể dục thể thao vùng X tổ chức cuộc chạy Đua tiếp sức thông minh theo quy luật sau:

- Thành phố xuất phát là thành phố 1, thành phố Đích là thành phố N.
- Mỗi Đội có K người dự thi. Lần lượt từng người chạy từ thành phố 1 về thành phố N.
- Khi một người chạy về tới thành phố N thì người tiếp theo mới Được phép xuất phát. Người thứ K về Đích tại thời Điểm nào thì thời Điểm Đó Được coi là thời Điểm về Đích của toàn Đội.
- Đường chạy của K Đội viên không Được giống nhau hoàn toàn.
- Có thể chạy lại những Đoạn Đã chạy qua.

Yêu cầu: Hãy viết chương trình tính thời gian nhỏ nhất Để một Đội hoàn thành cuộc chạy Đua tiếp sức nêu trên nếu các vận Động viên có tốc Độ chạy như nhau.

Dữ liệu: PATHK.INP

- ✂ Dòng Đầu ghi ba số K, N, M
- ✂ M dòng tiếp theo, mỗi dòng ghi 3 số nguyên i, j, w thể hiện một Đường Đi trực tiếp giữa hai thành phố i và j mất thời gian chạy là w (Đơn vị thời gian $1 \leq w \leq 9500$)

Kết quả: PATHK.OUT

- ✂ Dòng thứ nhất chứa một số nguyên duy nhất là thời gian chạy nhỏ nhất của một Đội.
- ✂ K dòng tiếp theo, mỗi dòng thể hiện một hành trình chạy của một vận Động viên trong Đội: là dãy các thành phố liên tiếp trên hành trình Đó.

Ví dụ:

PATHK.INP	PATHK.OUT
4 5 8	23
1 2 1	1 3 2 5
1 3 2	1 3 5
1 4 2	1 2 1 2 5
2 3 2	1 2 5
2 5 3	
3 4 3	
3 5 4	
4 5 6	

Gợi ý: Tư tưởng dựa vào thuật toán DIJTRA. Gọi $D[t,i]$ là Đường Đi ngắn thứ t trong K Đường Đi ngắn nhất từ Đỉnh 1 Đến Đỉnh i .

Mô tả thuật toán:

```

Khởi tạo  $D[t,i] = +\infty$  với mọi  $(i,j)$  ngoại trừ  $D[1,1]=0$ 
Lặp
    Tìm  $D[t,i]$  nhỏ nhất và  $(t,i)$  chưa Được Đánh dấu.
    Đánh dấu  $(t,i)$ .
    Sử dụng  $(t,i)$  Để sửa các nhãn  $D[t',i']$  với Điều kiện
        - có cạnh  $(i,i')$ 
        -  $D[t',i'] > D[t,i] + a[i,i']$ 
    Khi Đó cần gán lại mọi Đường Đi ngắn thứ  $t$  tới Đỉnh  $i$  thành Đường
    Đi ngắn thứ  $t+1$  và cập nhật Đường Đi ngắn thứ  $t$  tới  $i$  bằng Đường Đi
    qua  $i'$  trước rồi qua cạnh  $(i,i')$  tới  $i$ .
Cho Đến khi Đánh dấu hết các  $(t,i)$ .
    
```

Bài toán:

Số phụ thuộc - Olympic Balan 2003 (SUMS)

Cho tập số nguyên A gồm n phần tử, $A=\{a_1, a_2, \dots, a_n\}$. Số k Được gọi là phụ thuộc vào tập A , nếu k Được tạo thành bằng cách cộng các phần tử của tập A (mỗi phần tử có thể cộng nhiều lần).

Ví dụ cho $A=\{2,5,7\}$. Các số như 2, 4(2+2), 12(5+7 hoặc 2+2+2+2+2) Được gọi là phụ thuộc vào tập A . Số 0 cũng gọi là phụ thuộc vào tập A .

Yêu cầu: Cho một dãy B , hãy kiểm tra xem b_i có phải là số phụ thuộc vào tập A hay không .

Dữ liệu: SUMS.INP

- ✂ Dòng Đầu tiên chứa số nguyên n ($1 \leq n \leq 5000$).
- ✂ N dòng tiếp theo chứa các phần tử của tập A , $a_1 < a_2 < \dots < a_n$ ($1 \leq a_i \leq 50000$). ✂
- Dòng thứ $N+2$ chứa số nguyên m ($1 \leq m \leq 10000$).
- ✂ M dòng tiếp theo chứa dãy số nguyên b_1, b_2, \dots, b_m ($0 \leq b_i \leq 1000000000$).

Kết quả: SUMS.OUT

- ✂ Gồm m dòng, dòng thứ i ghi ra TAK nếu b_i là số phụ thuộc vào tập A và NIE nếu không phải là số phụ thuộc.

Ví dụ:

SUMS.INP	SUMS.OUT
3	TAK
2	NIE
5	TAK
7	TAK
6	NIE
0	TAK
1	
4	
12	
3	
2	

Gợi ý : Tư tưởng áp dụng thuật toán DIJTRA với lý thuyết Đồng dư trong toán học.

Xét Đồ thị gồm a_1 Đỉnh Đại diện cho các số dư khi chia một số cho a_1 . Các Đỉnh này có số hiệu từ $0, 1, \dots, a_1 - 1$.

Lần lượt lấy số dư của phép chia a_2, a_3, \dots, a_n cho a_1 . Giả sử $a_k \bmod a_1 = x$ thì sẽ có các cạnh nối $(0, x), (1, x+1), \dots$ với trọng số a_k . Chú ý trường hợp 2 số $a_k < a_k'$ có cùng số dư khi chia cho a_1 thì chỉ xét a_k .

Gọi $d(i)$ là Đường Đi ngắn nhất từ Đỉnh 0 Đến Đỉnh i còn lại ($1 \leq i \leq a_1$) - Đây cũng chính là tổng nhỏ nhất có thể tạo Được từ n số Đã cho thỏa mãn Điều kiện khi chia cho a_1 có số dư là i . Do Đó b_i là số phụ thuộc nếu $b_i \geq d(b_i \bmod a_1)$

Luyện tập:

Bài toán: **Bảo tồn Động vật hoang dã (MOVE)**

Một khu bảo tồn Động vật có n Địa Điểm và các Đường Đi hai chiều nối các Địa Điểm Đó, Địa Điểm thứ i có nhiệt Độ là t_i , giữa hai Địa Điểm bất kỳ có nhiều nhất là một Đường Đi nối chúng.

Người ta muốn di chuyển một loài Động vật quý hiếm từ Địa Điểm A tới Địa Điểm B, tuy nhiên nếu chênh lệch về nhiệt Độ giữa hai Địa Điểm liên tiếp trên Đường Đi là quá cao thì loài Động vật này rất có thể bị chết.

Yêu cầu: Hãy chỉ ra một hành trình mà Độ lệch nhiệt Độ lớn nhất giữa hai Địa Điểm liên tiếp bất kỳ trên Đường Đi là cực tiểu.

Dữ liệu: MOVE.INP

- ✂ Dòng 1: Chứa ba số n, A, B ($2 \leq n \leq 200; A \neq B$)
- ✂ Dòng 2: Chứa n số tự nhiên t_1, t_2, \dots, t_n ($\forall i: 0 \leq t_i \leq 20000$)
- ✂ Các dòng tiếp theo, mỗi dòng chứa hai số nguyên dương u, v cho biết giữa hai Địa Điểm u và v có Đường Đi nối chúng.

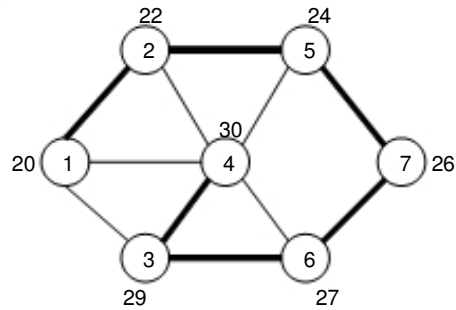
Kết quả: MOVE.OUT

- ✂ Dòng 1: Ghi Độ lệch nhiệt Độ lớn nhất giữa hai Địa Điểm liên tiếp bất kỳ trên Đường Đi tìm Được, nếu không tồn tại Đường Đi thì dòng này ghi số -1.
- ✂ Trong trường hợp tìm Được Đường Đi thì dòng 2 ghi hành trình tìm Được, bắt Đầu từ Địa Điểm A, tiếp theo là những Địa Điểm Đi qua, kết thúc là Địa Điểm B. Các Địa Điểm phải Được liệt kê theo Đúng thứ tự Đi qua trên hành trình

- ✂ Các số trên một dòng của Input/ Output file Được ghi cách nhau ít nhất một dấu cách.

Ví dụ:

MOVE.INP	MOVE.OUT
7 1 4	2
20 22 29 30 24 27 26	1 2 5 7 6 3 4
1 2	
1 3	
1 4	
2 4	
2 5	
3 4	
3 6	
4 5	
4 6	
5 7	
6 7	



Bài toán:

Đường Đến trường (SCHOOL)

Ngày 27/11 tới là ngày tổ chức thi học kỳ I ở trường ĐH BK. Là sinh viên năm thứ nhất, Hiếu không muốn vì Đi muộn mà gặp trục trặc ở phòng thi nên Đã chuẩn bị khá kỹ càng. Chỉ còn lại một công việc khá gay go là Hiếu không biết Đi Đường nào tới trường là nhanh nhất. Thường ngày Hiếu không quan tâm tới vấn Đề này lắm cho nên bây giờ Hiếu không biết phải làm sao cả . Bản Đồ thành phố là gồm có N nút giao thông và M con Đường nối các nút giao thông này. Có 2 loại con Đường là Đường 1 chiều và Đường 2 chiều. Độ dài của mỗi con Đường là một số nguyên dương. Nhà Hiếu ở nút giao thông 1 còn trường ĐH BK ở nút giao thông N. Vì một lộ trình Đường Đi từ nhà Hiếu tới trường có thể gặp nhiều yếu tố khác như là gặp nhiều Đèn Đỏ , Đi qua công trường xây dựng, ... phải giảm tốc Độ cho nên Hiếu muốn biết là có tất cả bao nhiêu lộ trình ngắn nhất Đi từ nhà tới trường. Bạn hãy lập trình giúp Hiếu giải quyết bài toán khó này.

Dữ liệu: SCHOOL.INP

- ✂ Dòng thứ nhất ghi hai số nguyên N và M.
- ✂ M dòng tiếp theo, mỗi dòng ghi 4 số nguyên dương K, U, V, L. Trong Đó
- ✂ K = 1 có nghĩa là có Đường Đi một chiều từ U Đến V với Độ dài L.
- ✂ K = 2 có nghĩa là có Đường Đi hai chiều giữa U và V với Độ dài L.

Kết quả: SCHOOL.OUT

- ✂ Ghi hai số là Độ dài Đường Đi ngắn nhất và số lượng Đường Đi ngắn nhất. Biết rằng số lượng Đường Đi ngắn nhất không vượt quá phạm vi int64 trong pascal hay long long trong C++.

Giới hạn:

- ✂ $1 \leq N \leq 5000$
- ✂ $1 \leq M \leq 20000$
- ✂ Độ dài các con Đường ≤ 32000

Ví dụ:

SCHOOL.INP	SCHOOL.OUT
3 2	4 1
1 1 2 3	
2 2 3 1	

Bài toán:

Thành phố trung tâm (CENTRE)

Theo thống kê cho biết mức Độ tăng trưởng kinh tế của nước Peace trong năm 2006 rất Đáng khả quan. Cả nước có tổng cộng N thành phố lớn nhỏ Được Đánh số tuần tự từ 1 Đến N phát triển khá Đồng đều. Giữa N thành phố này là một mạng lưới gồm M Đường Đi hai chiều, mỗi tuyến Đường nối 2 trong N thành phố sao cho không có 2 thành phố nào Được nối bởi quá 1 tuyến Đường. Trong N thành phố này thì thành phố 1 và thành phố N là 2 trung tâm kinh tế lớn nhất nước và hệ thống Đường Đảm bảo luôn có ít nhất một cách Đi từ thành phố 1 Đến thành phố N .

Tuy nhiên, cả 2 trung tâm này đều có dấu hiệu quá tải về mật Độ dân số. Vì vậy, Đức vua Peaceful quyết Định chọn ra thêm một thành phố nữa Để Đầu tư thành một trung tâm kinh tế thứ ba. Thành phố này sẽ tạm ngưng mọi hoạt Động thường nhật, cũng như mọi luồng lưu thông ra vào Để tiến hành nâng cấp cơ sở hạ tầng. Nhưng trong thời gian sửa chữa ấy, phải bảo Đảm Đường Đi ngắn nhất từ thành phố 1 Đến thành phố N không bị thay Đổi, nếu không nền kinh tế quốc gia sẽ bị trì trệ.

Vị trí và Đường nối giữa N thành phố Được mô tả như một Đồ thị N Đỉnh M cạnh. Hãy giúp nhà vua Đếm số lượng thành phố có thể chọn làm trung tâm kinh tế thứ ba sao cho thành phố Được chọn thỏa mãn các Điều kiện ở trên

Dữ liệu: CENTRE.INP

- ✂ Dòng Đầu tiên ghi 2 số nguyên dương N và M là số thành phố và số tuyến Đường.
- ✂ Dòng thứ i trong số M dòng tiếp theo ghi 3 số nguyên dương x_i , y_i và d_i với ý nghĩa tuyến Đường thứ i có Độ dài d_i và nối giữa 2 thành phố x_i , y_i .

Output: CENTRE.OUT

- ✂ Dòng Đầu tiên ghi số tự nhiên S là số lượng các thành phố có thể chọn làm trung tâm kinh tế thứ ba.
- ✂ S dòng tiếp theo, mỗi dòng ghi 1 số nguyên dương là số thứ tự của thành phố Được chọn (In ra theo thứ tự tăng dần)

Giới hạn:

- ✂ $2 \leq N \leq 30000$
- ✂ $1 \leq M \leq 100000$
- ✂ $1 \leq d_i \leq 1000$

Ví dụ:

CENTRE.INP	CENTRE.OUT
6 6	2
1 2 1	4
2 3 1	5
3 6 1	
1 4 100	
4 5 100	
5 6 100	

Gợi ý: Gọi $F1[i]$ là số Đường Đi ngắn nhất từ 1 Đến i . $F2[i]$ là số Đường Đi ngắn nhất từ i Đến N . Điều kiện Để i Được chọn làm trung tâm kinh tế là $F1[i] + F2[i] > F1[n]$ ($= F2[1]$). Chú ý vì số Đỉnh của Đồ thị lớn nên sử dụng thuật toán DIJTRA + Heap.

*** KẾT THÚC ***

VẤN ĐỀ: LUỒNG MINCOST

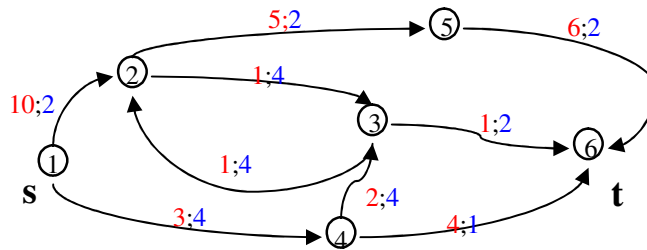
Bài toán:

Luồng cực Đại với chi phí nhỏ nhất (MINCOST)

Cho một mạng là Đồ thị có hướng n Đỉnh, m cung. Mỗi cung của mạng có một khả năng thông qua và một cước phí vận chuyển nhất Định.

Trong thực tế có thể hình dung Đồ thị như một mạng lưới vận chuyển hàng hóa. Mỗi cung (u,v) có khả năng thông qua $a_{u,v}$ có nghĩa là lượng hàng hóa thông qua cung này không Được vượt quá $a_{u,v}$. Mỗi cung (u,v) có một cước phí vận chuyển $c_{u,v}$ nghĩa là nếu vận chuyển k Đơn vị hàng ($k \leq a_{u,v}$) trên cung này thì mất một chi phí $k \cdot c_{u,v}$.

Hãy tìm một phương án vận chuyển, nghĩa là hãy xác Định trên mỗi cung của mạng cần vận chuyển bao nhiêu hàng (phù hợp với khả năng thông qua của mạng) sao cho vận chuyển Được lượng hàng **lớn nhất** từ nguồn (Đỉnh s cho trước) về Đích (Đỉnh t cho trước) với tổng chi phí vận chuyển là nhỏ nhất.



Dữ liệu: MINCOST.INP

- ✂ Dòng Đầu là n, m, s, t : Số Đỉnh, số cung, Đỉnh nguồn, Đỉnh Đích.
- ✂ m dòng tiếp theo mỗi bao gồm u, v, c, d cho biết có cung (u,v) với chi phí là c và khả năng thông qua là a .

Kết quả: MINCOST.OUT

- ✂ Dòng Đầu, ghi lượng hàng tối Đa vận chuyển Được và tổng chi phí vận chuyển.
- ✂ Một số dòng tiếp ghi u, v, i cho biết vận chuyển i Đơn vị hàng từ trên cung (u,v) .

Giới hạn:

- ✂ $n \leq 100$
- ✂ $d_{ij} \leq 30000$
- ✂ $c_{ij} \leq 10^9$
- ✂ Phạm vi tính toán là Longint.

Ví dụ:

MINCOST.INP	MINCOST.OUT
6 9 1 6	5 57
1 2 10 2	1 2 1
1 4 3 4	1 4 4
2 3 1 4	2 5 2
2 5 5 2	3 2 1
4 3 2 4	3 6 2
3 2 1 4	4 3 3
3 6 1 2	4 6 1
4 6 4 1	5 6 2
5 6 6 2	

Hướng giải quyết: Dựa vào thuật toán tìm luồng cực Đại FORD-FULKERSON.

Chú ý rằng thuật toán FORD-FULKERSON sử dụng $f_{i,j}$ để biểu thị giá trị luồng hiện tại thông qua trên cung (i,j) ($f_{i,j} = -f_{j,i}$).

$f_{i,j} > 0$ có nghĩa là có luồng giá trị $f_{i,j}$ thông qua cung (i,j) . Khi Đó cung (i,j) gọi là cung thuận

$f_{i,j} < 0$ có nghĩa là có luồng giá trị $-f_{i,j}$ thông qua cung (j,i) . Khi Đó cung (i,j) gọi là cung nghịch

Để tìm luồng MINCOST, dựa vào thuật toán FORD-FULKERSON ,tại mỗi bước tìm Đường tăng luồng, ta tìm Đường có chi phí ít nhất từ s Đến t thỏa mãn các cung trên Đường Đi chưa Được bão hòa (tức là có thể tăng thêm giá trị luồng trên các cung này). Chi phí trên các cung (i,j) tại mỗi bước tìm Được quy Định bằng $c_{i,j}$ nếu (i,j) hiện tại Đang là cung thuận và bằng $-c_{i,j}$ nếu (i,j) Đang là cung nghịch. Ta tăng giá trị luồng dựa theo Đường Đi chi phí ít nhất này cho Đến khi không tìm Được Đường tăng luồng nào nữa thì ta Được luồng MINCOST cực Đại.

Toàn bộ chương trình bài MINCOST

```
const inp='MINCOST.INP';
      out='MINCOST.OUT';
      maxn=100;
      maxc=1000000000;
type ii=longint;
var fi,fo:text;
    n,s,t:ii;
    d:array[1..maxn] of ii;
    trace:array[1..maxn] of ii;
    c,a,f:array[1..maxn,1..maxn] of ii;

Procedure OpenFile;
begin
  assign(fi,inp);
  reset(fi);
  assign(fo,out);
  rewrite(fo);
end;
```

```
Procedure CloseFile;  
begin  
    close(fi);  
    close(Fo);  
end;  
  
Procedure Enter;  
var i,x,y,m:ii;  
begin  
    readln(fi,n,m,s,t);  
    fillchar(a,sizeof(a),0);  
    for i:=1 to m do  
        readln(fi,x,y,c[x,y],a[x,y]);  
end;  
  
function Getd(x:ii):ii;  
begin  
    if x>=0 then Getd:=1 else Getd:=-1;  
end;  
  
function min(x,y:ii):ii;  
begin  
    if x<y then min:=x else min:=y;  
end;  
  
function FindPath:boolean;  
{Tìm Đường tăng luồng}  
{sử dụng thuật toán FORD BELLMAN Để tìm Đường tăng luồng có chi phí ít nhất}  
var u,v:ii;  
    stop:boolean;  
begin  
    for v:=1 to n do d[v]:=maxc;  
    d[s]:=0;  
    repeat  
        stop:=true;  
        for u:=1 to n do  
            for v:=1 to n do  
                if a[u,v] - f[u,v] > 0 then  
                    if d[v] > d[u] + c[u,v]*Getd(f[u,v]) then  
                        {cung (u,v) có trọng số c[u,v] nếu (u,v)  
                        là cung thuận  $\forall f[u,v] \geq 0$ }  
                        begin  
                            trace[v]:=u;  
                            d[v] := d[u] + c[u,v]*Getd(f[u,v]);  
                            stop:=false;  
                        end;  
            until stop;  
            if d[t]<maxc then FindPath:=true else FindPath:=false;  
end;
```

```
function findel:ii;
{Tìm lượng luồng có thể tăng trên Đường Đi chi phí ít nhất Đã tìm Được}
var u,v,mi:ii;
begin
  v:=t;
  mi:=maxc;
  while v<>s do
  begin
    u:=trace[v];
    if f[u,v] >= 0 then {Nếu (u,v) là cung thuận}
      mi:=min(mi,a[u,v] - f[u,v])
    else {(u,v) là cung nghịch}
      mi:=min(mi,-f[u,v]);
    v:=u;
  end;
  findel:=mi;
end;

procedure IncFlow;
{tăng luồng}
var pp,u,v:ii;
begin
  pp:=findel;
  v:=t;
  while v<>s do
  begin
    u:=trace[v];
    inc(f[u,v],pp);
    dec(f[v,u],pp);
    v:=u;
  end;
end;

Procedure Process;
var boo:boolean;
    u,v,k,sum:ii;
begin
  fillchar(f,sizeof(f),0);
  repeat
    boo:=FindPath;
    if boo then IncFlow else break;
  until false;

  k:=0;
  for v:=1 to n do
    if f[s,v]>0 then inc(k,f[s,v]);

  sum:=0;
  for u:=1 to n do
    for v:=1 to n do
      if f[u,v]>0 then inc(sum,f[u,v]*c[u,v]);
  writeln(fo,k,' ',sum);

  for u:=1 to n do
    for v:=1 to n do
      if f[u,v]>0 then writeln(fo,u,' ',v,' ',f[u,v]);
end;
```

```
begin
  OpenFile;
  Enter;
  Process;
  CloseFile;
end.
```

Luyện tập:

Bài toán:

Trao Đổi thông tin (KWAY)

Cho một mạng thông tin gồm n trạm và m Đường nối hai chiều giữa các trạm. Trạm s là trạm chỉ huy, trạm f là trạm Điều khiển. Sau một lần bị tin tặc tấn công lấy mất dữ liệu từ trạm chỉ huy chuyển Đến trạm Điều khiển, chỉ huy mạng quyết Định chia thông tin chuyển Đi thành k Đơn vị thông tin Để chuyển theo k Đường Đến trạm Điều khiển. Mà hai Đường truyền bất kỳ không Được chung bất kỳ một Đường nào.

Hãy tìm cách truyền k Đơn vị thông tin sao cho tổng chi phí là nhỏ nhất.

Dữ liệu: KWAY.INP

- ✂ Dòng Đầu là n, m, k, s, f ($n \leq 100$).
- ✂ m dòng tiếp là u, v, c cho biết có Đường từ $u \rightarrow v$ và $v \rightarrow u$ với chi phí là c .

Kết quả: KWAY.OUT

- ✂ Dòng Đầu ghi -1 nếu không thể chuyển k Đơn vị thông tin theo cách trên, ngược lại ghi chi phí Để chuyển.
- ✂ k dòng tiếp lần lượt ghi cách chuyển của từng Đơn vị thông tin. Số Đầu là số lượng trạm trên Đường truyền, tiếp Đó là dãy các trạm trên Đường truyền (bắt Đầu từ s , kết thúc ở f)

Chú ý: Phạm vi tính toán là Longint.

Ví dụ:

KWAY.INP	KWAY.OUT
8 11 3 1 8	11
1 2 1	4 1 2 3 8
1 4 1	5 1 5 3 6 8
1 5 1	5 1 4 2 7 8
2 3 1	
2 4 1	
2 7 1	
3 8 1	
3 6 1	
3 5 1	
6 8 1	
7 8 1	

VẤN ĐỀ: CÁC CÔNG THỨC HÌNH HỌC

1. Lưu dữ liệu hình học trong Pascal.

Thông thường việc tính toán trong hình học Được thực hiện trên phạm vi số thực. Một số kiểu số thực thường Được sử dụng như sau:

Kiểu	Giới hạn	Chữ số có nghĩa	Kích thước (Byte)
Single	1.5e-45..3.4e38	7-8	4
Real	2.9e-39..1.7e38	11-12	6
Double	5.0e-324..1.7e308	15-16	8
Extended	3.4e-4932..1.1e4932	19-20	10

Tùy theo yêu cầu về Độ chính xác mà chúng ta sử dụng kiểu dữ liệu cho phù hợp. Chú ý rằng các phép toán với số thực luôn có sai số, do Đó khi so sánh hai kết quả là số thực thì hai giá trị chênh nhau không vượt quá một số eps (rất nhỏ Được mặc Định từ trước) thì có thể xem hai giá trị Đó bằng nhau.

2. Các khai báo kiểu dữ liệu.

• Khai báo Điểm

```
Type point=record  
  x,y:real;  
end;
```

• Khai báo Đoạn thẳng

```
Type Line=record  
  p1,p2:point  
End;
```

• Khai báo Đường thẳng ($ax + by + c = 0$)

```
Type StrLine=record  
  a,b,c:real;  
End;
```

3. Phương trình Đường thẳng.

• Cách lập phương trình Đường thẳng Đi qua 2 Điểm (x_1, y_1) , (x_2, y_2) là:

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

```
Procedure Extract(p1,p2:point; var d:StrLine);  
begin  
  Var d : strLine  
  d.a := p1.y - p2.y;  
  d.b := p2.x - p1.x;  
  d.c:=p1.x*p2.y - p2.x*p1.y;  
end;
```

- Phương trình Đường thẳng trùng với trục tung là $x = 0$
- Phương trình Đường thẳng trùng với trục hoành là $y = 0$

4. Khoảng cách, góc.

- Khoảng cách giữa 2 Điểm (x_1, y_1) và (x_2, y_2) là: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
- Khoảng cách giữa Điểm (x_0, y_0) và Đường thẳng $ax + by + c = 0$ là:

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

- Góc giữa 2 Đường thẳng $a_1x + b_1y + c_1 = 0$ và $a_2x + b_2y + c_2 = 0$ là

$$\cos \alpha = \frac{|a_1a_2 + b_1b_2|}{\sqrt{a_1^2 + b_1^2} \sqrt{a_2^2 + b_2^2}}$$

- Hai Đường thẳng vuông góc với nhau khi $a_1a_2 + b_1b_2 = 0$

5. Vị trí tương Đối.

- Vị trí tương Đối của Điểm Đối với Đường thẳng, Đoạn thẳng:
 - Điểm (x_0, y_0) nằm trên Đường thẳng $ax + by + c = 0$ khi và chỉ khi $ax_0 + by_0 + c = 0$.
 - Nếu gọi $F(x, y) = ax + by + c$ thì $ax_0 + by_0 + c = 0 \Leftrightarrow F(x_0, y_0) = 0$
 - Hai Điểm (x_1, y_1) và (x_2, y_2) nằm khác phía so với Đường thẳng $ax + by + c = 0$ khi và chỉ khi: $(ax_1 + by_1 + c) \cdot (ax_2 + by_2 + c) < 0 \Leftrightarrow F(x_1, y_1) \cdot F(x_2, y_2) < 0$
 - Một Điểm (x_0, y_0) nằm trên Đoạn thẳng nối hai Điểm (x_1, y_1) và (x_2, y_2) nếu
 - $F(x_0, y_0) = 0$
 - $\min(x_1, x_2) \leq x_0 \leq \max(x_1, x_2)$
 - $\min(y_1, y_2) \leq y_0 \leq \max(y_1, y_2)$
- Vị trí tương Đối giữa 2 Đường thẳng (d_1): $a_1x + b_1y + c_1 = 0$ và (d_2): $a_2x + b_2y + c_2 = 0$

Xét

$$D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1b_2 - a_2b_1$$

$$D_x = \begin{vmatrix} -c_1 & b_1 \\ -c_2 & b_2 \end{vmatrix} = b_1c_2 - b_2c_1 \quad D_y = \begin{vmatrix} a_1 & -c_1 \\ a_2 & -c_2 \end{vmatrix} = a_1c_2 - a_2c_1$$

- Nếu $D \neq 0$ thì hai Đường thẳng giao nhau tại $(x_0, y_0) = \left(\frac{D_x}{D}, \frac{D_y}{D} \right)$

- Nếu $D = 0$ thì

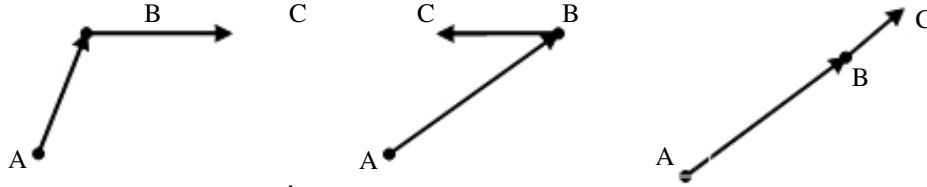
- Nếu $D_x \neq 0$ hay $D_y \neq 0$ thì hai Đường thẳng song song với nhau
- Nếu $D_x = D_y = 0$ thì hai Đường thẳng trùng nhau

6. Giao Điểm.

- Hai Đoạn thẳng A_1A_2 và B_1B_2 cắt nhau tại Điểm không phải là Đầu mút khi và chỉ khi:
 - A_1 và A_2 khác phía so với B_1B_2
 - B_1 và B_2 khác phía so với A_1A_2

- Hai Đường thẳng cắt nhau khi $D \neq 0$, giao Điểm là $(x_0, y_0) = \left(\frac{D_x}{D}, \frac{D_y}{D} \right)$

7. Hàm rẽ phải, rẽ trái.



Hàm CCW cho biết khi Đi từ $A \rightarrow B \rightarrow C$ là rẽ trái hay rẽ phải. Hàm trả về giá trị -1 nếu rẽ phải ($k < 0$), 1 nếu rẽ trái ($k > 0$) và 0 nếu Đi thẳng. Trong Đó:

$$k = \begin{vmatrix} (x_b - x_a) & (x_c - x_b) \\ (y_b - y_a) & (y_c - y_b) \end{vmatrix} = (x_b - x_a)(y_c - y_b) - (y_b - y_a)(x_c - x_b)$$

```

Procedure    ccw(p1,p2,p3:point):integer;
Var u1,u2,v1,v2,T:longint;
Begin
  u1:=p2.x - p1.x; v1:=p2.y - p1.y;
  u2:=p3.x - p2.x; v2:=p3.y - p2.y;
  T:= u1*v2 - v1*u2;
  If T = 0 then abs(ccw) < eps then ccw:=0
  Else if t > eps then ccw:= 1
  Else ccw:=-1
End;
```

Có thể dùng hàm ccw Để kiểm tra: 2 Đoạn thẳng A_1A_2 và B_1B_2 có giao Điểm khác Đầu mút khi và chỉ khi $ccw(A_1,A_2,B_1) * ccw(A_1,A_2,B_2) < 0$

8. Tam giác.

- Diện tích tam giác

$$S = \frac{1}{2} a h_a = \sqrt{p(p-a)(p-b)(p-c)} = \frac{1}{2} b c \sin A = p r = \frac{\overline{abc}}{4R}$$

Trong Đó:

- a, b, c là Độ dài 3 cạnh của tam giác
- p là nửa chu vi tam giác: $p = \frac{1}{2}(a + b + c)$
- r là bán kính Đường tròn nội tiếp, R là bán kính Đường tròn ngoại tiếp

9. Đa giác.

- Diện tích Đa giác gồm các Điểm $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$ là:

$$S = \sum_{i=1}^n \frac{(x_i - x_{i+1})(y_i - y_{i+1})}{2} \quad (\text{Trong Đó } (x_{n+1}, y_{n+1}) = (x_1, y_1))$$

- Nếu Đa giác với các Điểm có tọa Độ nguyên thì có công thức liên hệ giữa diện tích Đa giác và số Điểm nguyên nằm bên trong Đa giác như sau:

$$S = n + \frac{\overline{m}}{2} - 1$$

Trong Đó:

- n là số Điểm nguyên nằm bên trong Đa giác
- m là số Điểm nguyên nằm trên các cạnh của Đa giác

Vấn đề : Một dạng bài quy hoạch động

Đây là một dạng bài quy hoạch động thường gặp , phương pháp thường là tính $f[i,j]$ qua $f[i,k]$ và $f[k+1,j]$ Trong đó $i \leq k \leq j$ (độ phức tạp thường là $O(n^3)$), nhưng có 1 số dạng bài có thể rút xuống n^2 .) hoặc tính $f[i,j]$ qua $f[i,j+1]$ và $f[i,j-1]$,(độ phức tạp là n^2)

Bài Toán : **<nguồn spoj.pl/problems/TRT>**

FJ (Farmer John) muốn bán N miếng bánh làm từ sữa các con bò. ($1 \leq N \leq 2000$). FJ bán mỗi ngày 1 miếng bánh và muốn nhận được số tiền lớn nhất từ việc bán những cái bánh đó trong một khoảng thời gian có hạn. Mỗi miếng bánh có giá trị cao nhờ nhiều nguyên nhân :

- Các miếng bánh được xếp trong một băng dài được đánh số từ 1 đến N. Mỗi ngày, FJ có thể lấy 1 miếng bánh ở đầu này hoặc đầu kia của băng đó).
- Cũng giống như rượu và pho mát, các miếng bánh càng có tuổi thọ cao thì càng có giá trị.
- Giá trị các miếng bánh cũng không cố định. Miếng bánh thứ i có giá trị $V(i)$ ($1 \leq V(i) \leq 1000$).
- Mỗi chiếc bánh có giá trị phụ thuộc vào tuổi của nó. Mỗi con bò sẽ nhận được $a \cdot V(i)$ với miếng bánh thứ i có a tuổi.

Cho các giá trị $V(i)$. Tìm giá trị lớn nhất FJ có thể nhận được từ việc bán những chiếc bánh.

Chiếc bánh được bán đầu tiên sẽ có tuổi là 1. Các miếng bánh sau đó sẽ nhiều hơn miếng bánh trước 1 tuổi.

INPUT

Dòng thứ nhất : Một số tự nhiên N duy nhất.

Dòng thứ 2 đến N+1 : Dòng thứ i chứa giá trị của $V(i)$

OUTPUT

Một số duy nhất là giá trị lớn nhất mà FJ thu được từ việc bán bánh.

Sample :

Input:

5
1
3
1
5
2

Output:

43

Thuật Toán : QHĐ :

$F[i,j]$ là số tiền lớn nhất thu được của đoạn $i \rightarrow j$.

Dễ thấy số tuổi của mỗi bánh ở 2 đầu lúc đó là $n - (j - i) = T$

$F[i,j] = \text{Max}(F[i,j - 1] + V[j] * T, F[i - 1,j] + V[i] * T)$.

Có thể cải tiến thành mảng 1 chiều.

KQ : $F[1,N]$

Bài Toán LSFIGHT

< nguồn <http://vn.spoj.pl/problems/LSFIGHT/> >

Trong kỳ thi Marathon 08 năm nay các vCoders phải tham gia một môn thi đấu đối kháng giữa 2 người. Sau vòng loại, ban tổ chức sẽ chọn ra N thí sinh có số điểm cao nhất và đánh số từ 1 đến N.

Một số vấn Đề Đáng chú ý trong môn tin học

Các thí sinh này phải xếp lần lượt theo thứ tự thành 1 vòng tròn (người thứ N đứng cạnh người thứ 1). Sau đó sẽ chọn ra 2 thí sinh bất kì đang đứng cạnh nhau trong vòng tròn để thi đấu, thí sinh nào thua sẽ bị loại và buộc phải đi ra vòng tròn, trở về hàng ghế khán giả. Cuộc đấu cứ tiếp tục như thế đến khi chỉ còn một người ở lại và cũng chính là người thắng cuộc.

Tuy nhiên ban tổ chức muốn biết trước xem có bao nhiêu người có khả năng thắng cuộc và đó là những người nào. Biết trước ai sẽ thắng trong mỗi trận đấu, bạn hãy giúp ban tổ chức nhé ^^

Dữ liệu

- Dòng đầu là số nguyên dương N ($3 \leq N \leq 500$)

- N dòng sau là ma trận A[i, j], A[i, j] = 0 nếu thí sinh i thua thí sinh j và A[i, j] = 1 nếu ngược lại.

Biết rằng luôn đảm bảo A[i, i] = 1 với mọi i và A[i, j] + A[j, i] = 1 với $i \neq j$. Các số viết cách nhau ít nhất 1 dấu cách.

Kết quả

- Dòng đầu là số nguyên dương M - số lượng thí sinh có khả năng thắng cuộc

- M dòng sau mỗi dòng ghi một số là chỉ số của thí sinh có khả năng thắng cuộc theo thứ tự tăng dần của chỉ số.

Ví dụ

Dữ liệu

```
7
1 1 1 1 1 0 1
0 1 0 1 1 0 0
0 1 1 1 1 1 1
0 0 0 1 1 0 1
0 0 0 0 1 0 1
1 1 0 1 1 1 1
0 1 0 0 0 0 1
```

Kết quả

```
3
1
3
6
```

Thuật Toán :

Quy hoạch động

$F(i, j)$ = true nếu có thể tồn tại cách đấu sao cho các đối thủ giữa i và j đều bị loại (tính theo chiều

kim Đồng hồ trên vòng tròn).

$F(i, j) = \text{true}$ nếu tồn tại k giữa i và j (giữa là tính theo vòng tròn) mà $F(i, k) = F(k, j) = \text{true}$ và k thua một trong 2 người i và j .

Từ đây dễ dàng suy ra được đáp số.

Độ phức tạp $O(n^3)$.

Bài Toán : OPTCUT

< Nguồn <http://vn.spoj.pl/problems/OPTCUT/> >

Bạn cần chặt một thanh gỗ ra thành n đoạn, mỗi đoạn có độ dài a_i . Các đoạn được chặt phải có độ dài **theo đúng thứ tự** a_1, a_2, \dots, a_n từ trái sang phải.

Tại mỗi bước, bạn có thể chặt một nhát chia một thanh gỗ làm hai, và chi phí cho nhát chặt này bằng độ dài của thanh gỗ trước khi chặt.

Thứ tự chặt khác nhau sẽ cho ra tổng chi phí khác nhau khi chặt thanh gỗ thành n đoạn yêu cầu.

Ví dụ bạn cần chặt một thanh gỗ độ dài 20 ra thành 4 đoạn độ dài 3, 5, 2 và 10 theo thứ tự.

Khi chặt từ trái sang phải:

20 chặt thành 3 và 17, chi phí 20.

17 chặt thành 5 và 12, chi phí 17.

12 chặt thành 2 và 10, chi phí 12.

Tổng chi phí: 49

Khi chặt từ phải sang trái:

20 chặt thành 10 và 10, chi phí 20.

10 chặt thành 8 và 2, chi phí 10.

8 chặt thành 3 và 5, chi phí 8.

Tổng chi phí: 38

Bạn hãy tìm cách chặt có tổng chi phí nhỏ nhất.

Dữ liệu

Dòng 1: n ($1 \leq n \leq 2000$)

Dòng 2: n số nguyên dương a_1, a_2, \dots, a_n , biết rằng độ dài của thanh gỗ $a_1 + a_2 + \dots + a_n \leq 500000$

Kết quả

Một số nguyên duy nhất là chi phí nhỏ nhất tìm được.

Ví dụ

Dữ liệu

4
3 5 2 10

Kết quả

37

Thuật Toán : Gọi $Cat[i,j]$ là điểm cắt sao cho $f[i,j]$ đạt min , có thể tính $cat[I,j]$ thông qua $Cat[i+1,j]$

Và $cat[i,j-1]$. c/m $Cat[i,j] \geq Cat[i+1,j]$, tương tự có thể Cm $Cat[i,j] \leq cat[i,j-1]$.

Cách chứng minh có thể gọi là 1 trò chơi tối ưu.

Bài Toán :DTGAME <nguồn <https://vn.spoj.pl/problems/DTGAME/>>

Tiền bạc vẫn luôn là thứ có giá trị đối với mỗi con người, kể cả với *pirate*. Vì vậy, khi hòn đảo xinh đẹp của tên cướp biển khét tiếng này sắp bị... giải tỏa, hắn đã tranh thủ vơ vét cho đến đồng tiền vàng cuối cùng. Trên hòn đảo có N mỏ vàng nằm cạnh nhau trên một đường thẳng, đánh số từ 1 đến N . Mỏ vàng i có giá trị là p_i . Đội thi công sẽ có nhiệm vụ san lấp hết N mỏ vàng này.

Nhưng tại mỗi thời điểm, *pirate* cố gắng ngăn cản việc san lấp bằng cách xây một bức tường ngăn cách hai mỏ vàng liên tiếp nào đó, vậy là cụm mỏ còn lại được chia làm 2 phần. Đội thi công sẽ chọn một trong hai phần đó để san lấp hết, thế là *pirate* giữ lại được phần còn lại và hắn sẽ nhận được số đồng tiền vàng đúng bằng tổng giá trị của những mỏ còn lại đó. Công việc cứ tiếp tục cho đến khi chỉ còn lại một mỏ vàng duy nhất. Đội thi công biết điều đó, cho nên họ đã có chiến thuật san lấp để cực tiểu hóa số tiền của pirate. Với lòng tham không đáy, pirate quyết tâm lấy càng nhiều vàng càng tốt. Hãy giúp con người khốn khổ vì tiền này

Input

Dữ liệu vào gồm nhiều dòng:

- Dòng 1: Một số nguyên dương N ($1 \leq n \leq 2000$).
- Dòng 2...n+1: Mỗi dòng ghi giá trị p_i của từng mỏ vàng theo thứ tự từ 1 đến n ($1 \leq p_i \leq 1000$).

Output

Dữ liệu ra gồm 1 dòng duy nhất ghi ra số đồng tiền vàng lớn nhất có thể vơ vét được.

Example

Input:

5
8
6
2
4
2

Output: 10

Giải thích: Đầu tiên pirate chia mỏ vàng thành 2 phần [1,2] và [3,4,5]. Xe ủi sẽ san lấp phần [1,2] và pirate nhận được $p_2 + p_3 + p_4 = 8$ đồng tiền vàng. Tiếp theo chia [3,4,5] thành [3] và [4,5]. Xe ủi san lấp [4,5] và pirate thu thêm $p_2 = 2$ đồng tiền vàng. Công việc kết thúc vì chỉ còn 1 mỏ, vậy tổng cộng thu được là $8 + 2 = 10$ đồng tiền vàng. Không có cách nào giúp hắn thu thêm tiền.

Thuật toán : đây là 1 bài tương tự như bài OPTCUT , các bạn có thể tham khảo cách giải ở trên .

Luyện tập .

Bài Toán : **NKPOLY** < nguồn <http://vn.spoj.pl/problems/NKPOLY/>>

Đức vua vương quốc XYZ tổ chức kén rể cho cô công chúa duy nhất của mình. Vì vậy, ông đặt ra những yêu cầu rất cao cho con rể tương lai. Để có thể trở thành con rể của ngài, các chàng trai thi nhau thể hiện mình. Sau khi vượt qua những phần thi đòi hỏi sức khỏe, lòng dũng cảm, ... họ sẽ gặp phải một thử thách vô cùng khó khăn, đó là phần thi về sự nhanh nhạy và thông minh. Đức vua sẽ cho mỗi người một miếng bìa chéo bất kì không có điểm chung khác các đầu mút. Với cách vẽ như vậy, chúng ta sẽ thu được $N-2$ hình tam giác. Đức vua yêu cầu họ hãy tìm 2 cách chia:

- Một cách chia sao cho tam giác có diện tích lớn nhất trong $N-2$ tam giác là lớn nhất.
- Một cách chia sao cho tam giác có diện tích lớn nhất trong $N-2$ tam giác là nhỏ nhất.

Một số vấn Đề Đáng chú ý trong môn tin học

Sau khi nhà vua đưa ra hình dạng của đa giác lồi, họ sẽ chỉ có 1 giây để đưa ra đáp án của mình.

Người đưa ra đáp án đúng nhất và nhanh nhất sẽ được chọn làm phò mã. Bạn cũng là một người đã lọt vào vòng thi này. Hãy chứng tỏ khả năng của mình đi!

Dữ liệu

- Dòng đầu tiên ghi số nguyên N là số đỉnh của đa giác.
- Trong n dòng sau, mỗi dòng ghi một cặp số nguyên là tọa độ các đỉnh của đa giác. Các đỉnh được liệt kê theo chiều kim đồng hồ.

Kết quả

- Dòng thứ nhất ghi diện tích của tam giác lớn nhất trong trường hợp 1.
- Dòng thứ hai ghi diện tích của tam giác lớn nhất trong trường hợp 2.

Các giá trị diện tích có độ chính xác 1 chữ số thập phân.

Giới hạn

- $4 \leq N \leq 200$.
- Các tọa độ là các số nguyên có trị tuyệt đối không quá 10^6 .

Ví dụ

Dữ liệu:

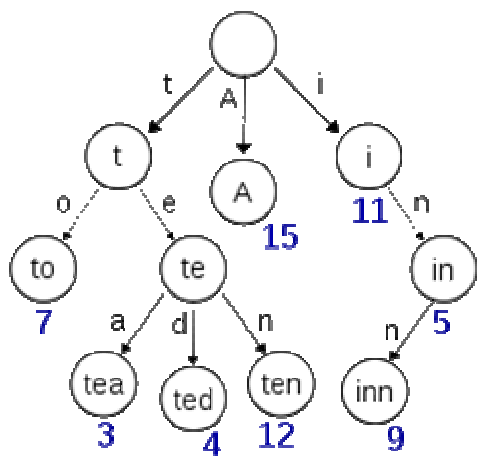
```
5
0 0
0 2
1 4
2 2
2 0
```

Kết quả

```
4.0
2.0
```

Vấn đề : TRIE TREE và ứng dụng.

Trie tree hay còn gọi là cây tiền tố , được sử dụng rất nhiều trong các bài toán so sánh chuỗi . Định nghĩa đơn giản và dễ hiểu nhưng nó có ứng dụng khá rộng .



Hình bên là 1 cây trie tương ứng cho các từ "A", "to", "tea", "ted", "ten", "i", "in", and "inn".

Một số bài tập ứng dụng

Bài toán : CHAIN2 < <http://vn.spoj.pl/problems/CHAIN2/>>

Chuỗi từ có độ dài n là một dãy các từ w_1, w_2, \dots, w_n sao cho với mọi $1 \leq i < n$, từ w_i là tiền tố của từ w_{i+1} . Nhắc lại từ u có độ dài k là tiền tố của từ v có độ dài l nếu $l > k$ và các ký tự đầu tiên của v trùng với từ u . Cho tập hợp các từ $S = \{s_1, s_2, \dots, s_m\}$. Tìm chuỗi từ dài nhất có thể xây dựng được bằng cách dùng các từ trong tập hợp S (có thể không sử dụng hết các từ).

Dữ liệu

Dòng đầu tiên chứa số nguyên m ($1 \leq m \leq 250000$). Mỗi dòng trong số m dòng sau chứa một từ trong tập S . Biết rằng mỗi từ có độ dài không quá 250000 ký tự và tổng độ dài của các từ không

vượt quá 250000 ký tự.

Kết quả

In ra một số duy nhất là độ dài của chuỗi từ dài nhất xây dựng được từ các từ trong tập đã cho.

Ví dụ

Dữ liệu

3
a
ab
abc

Kết quả

3

Dữ liệu

5
a
ab
bc
bcd
add

Kết quả

2

Thuật Toán : Bài CHAIN2 em dùng Trie cho các từ, sau đó chỉ cần QHĐ trên Trie để tìm nhánh có nhiều nút là kết thúc của các từ nhất. Code tham khảo :

```
uses math;
```

```
const
```

```
    FINP      =    ";
```

```
    FOUT      =    ";
```

```
type
```

```
    trie      =    ^node;
```

```
    node      =    record
```

```
        u,f   :    longint;
```

```
        c     :    array['a'..'z'] of trie;
```

```
end;
```

var

f1,f2 : text;

root : trie;

s : string;

procedure openF;

begin

assign(f1,FINP); reset(f1);

assign(f2,FOUT); rewrite(f2);

end;

procedure closeF;

begin

close(f1);

close(f2);

end;

procedure add(var a:trie); inline;

var

c:char;

begin

new(a);

a^.u:=0;

a^.f:=0;

for c:='a' to 'z' do

a^.c[c]:=nil;

end;

procedure inp;

var

n,i:longint;

p:trie;

begin

readln(f1,n);

add(root);

for n:=1 to n do

begin

readln(f1,s); p:=root;

for i:=1 to length(s) do

begin

if p^.c[s[i]]=nil then add(p^.c[s[i]]);

p:=p^.c[s[i]];

end;

p^.u:=1;

end;

end;

procedure dfs(a:trie); inline;

var

c:char;

begin

a^.f:=a^.u;

for c:='a' to 'z' do

if a^.c[c]<>nil then

begin

dfs(a^.c[c]);

```
    a^.f:=max(a^.f,a^.c[c]^f+a^.u);  
  
    end;  
  
end;
```

```
procedure solve;
```

```
begin  
    dfs(root);  
    writeln(f2,root^.f);  
end;
```

```
begin
```

```
    openF;  
    inp;  
    solve;  
    closeF;
```

```
end.
```

Bài luyện tập **SEC** < <http://vn.spoj.pl/problems/SEC/>>

Bessie định dẫn đàn bò đi trốn. Để đảm bảo bí mật, đàn bò liên lạc với nhau bằng cách tin nhắn nhị phân. Từng là một nhân viên phản gián thông minh, John đã thu được M ($1 \leq M \leq 50,000$) tin nhắn mật, tuy nhiên với tin nhắn i John chỉ thu được b_i ($1 \leq b_i \leq 10,000$) bit đầu tiên.

John đã biên soạn ra 1 danh sách N ($1 \leq N \leq 50,000$) các từ mã hóa mà đàn bò có khả năng đang sử dụng. Thật không may, John chỉ biết được c_j ($1 \leq c_j \leq 10,000$) bit đầu tiên của từ mã hóa thứ j . Với mỗi từ mã hóa j , John muốn biết số lượng tin nhắn mà John thu được có khả năng là từ mã hóa j này. Tức là với từ mã hóa j , có bao nhiêu tin nhắn thu được có phần đầu giống với từ mã hóa j này. Việc của bạn là phải tính số lượng này.

Tổng số lượng các bit trong dữ liệu đầu vào (tổng các b_i và c_j) không quá 500,000.

QUY CÁCH NHẬP DỮ LIỆU

- Dòng 1: 2 số nguyên: M và N
- Dòng 2..M+1: Dòng i+1 mô tả tin nhắn thứ i thu được, đầu tiên là b_i sau đó là b_i bit cách nhau bởi dấu cách, các bit có giá trị 0 hoặc 1.
- Dòng M+2..M+N+1: Dòng M+j+1 mô tả từ mã hóa thứ j, đầu tiên là c_j sau đó là c_j bit cách nhau bởi dấu cách.

VÍ DỤ

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

GIẢI THÍCH VÍ DỤ

Có 4 tin nhắn và 5 từ mã hóa. Các tin nhắn thu được có phần đầu là 010, 1, 100 và 110. Các từ mã hóa có phần đầu là 0, 1, 01, 01001, và 11.

QUY CÁCH GHI KẾT QUẢ

- Dòng 1..M: Dòng j: Số lượng tin nhắn mà có khả năng là từ mã hóa thứ j

VÍ DỤ

```
1
3
1
1
2
```

GIẢI THÍCH

0 chỉ có khả năng là 010 -> 1 tin nhắn. 1 chỉ có khả năng là 1, 100, hoặc 110 -> 3 tin nhắn. 01 chỉ có thể là 010 -> 1 tin nhắn. 01001 chỉ có thể là 010 -> 1 tin nhắn. 11 chỉ có thể là 1 hoặc 110 -> 2 tin nhắn.

Cho một danh sách từ **L** và một từ **w**. Nhiệm vụ của bạn là phải tìm một từ trong **L** tạo thành một "vần hoàn hảo" với **w**. Từ **u** này là duy nhất xác định bởi các thuộc tính sau:

- Nó nằm trong **L**.
- Nó khác **w**.
- Phần hậu tố chung của chúng dài nhất có thể.
- **u** là từ có thứ tự từ điển nhỏ nhất thoả mãn các điều trên

Chú ý

Một tiền tố của một từ là một chuỗi có thể thu được bằng cách lặp lại việc xoá kí tự cuối cùng của từ. Tương tự, một hậu tố của một từ là một chuỗi mà có thể thu được bằng cách lặp lại việc xoá kí tự đầu tiên của từ.

Ví dụ với từ: *different*.

Từ này vừa là tiền tố, vừa là hậu tố của chính nó. một tiền tố dài nhất khác của nó *different*, và một hậu tố dài nhất khác của nó là *ifferent*. Chuỗi *rent* cũng là một hậu tố khác nhưng ngắn hơn. Chuỗi *eent* và *iffe* đều không phải là tiền tố hay hậu tố của từ *different*.

Gọi **u** và **v** là 2 từ khác nhau. Ta nói rằng **u** có thứ tự từ điển nhỏ hơn **v** nếu hoặc **u** là một tiền tố của **v**, hoặc nếu *i* là vị trí đầu tiên mà chúng khác nhau, và kí tự thứ *i* của **u** đứng trước kí tự thứ *i* của **v** trong bảng chữ cái.

Ví dụ, *dog* nhỏ hơn *dogs*, từ này lại nhỏ hơn *dragon* (Vì *o* nhỏ hơn *r*).

Dữ liệu

Có 2 phần. Phần thứ nhất chứa danh sách từ **L**, mỗi từ trên 1 dòng. Mỗi từ chỉ chứa các chữ cái thường tiếng Anh và không có 2 nào từ giống nhau.

Phần thứ nhất kết thúc bằng một dòng trống.

Tiếp theo là phần 2, với mỗi câu hỏi cho từ **w** trên một dòng.

Bạn có thể chắc chắn rằng trong cả 2 phần của dữ liệu vào, độ dài của mỗi từ không quá 30. Và số lượng từ trong mỗi phần không quá 250000.

Kết quả

Với mỗi câu hỏi, viết ra trên một dòng từ mà tạo thành vần hoàn hảo với nó. Kết quả phải viết bằng chữ cái thường.

Ví dụ

Dữ liệu

perfect
rhyme
crime
time

crime
rhyme

Kết quả

time
crime

Trong câu hỏi thứ 2, có 2 từ có cùng độ dài hậu tố với rhyme (là crime và time), từ có thứ tự từ điển nhỏ hơn đã được chọn.

Chủ Đề : Duyệt bằng cách chia đôi tập hợp

Trong thực tế ta thường gặp dạng bài phải xét tất cả các cấu hình để có thể xác định cấu hình tối ưu. Phương pháp thông thường được sử dụng là duyệt 2^n có cận . Với n cỡ 32 thì chi phí sẽ là $2^{32} \sim 4$ tỷ (Máy tính loại trung bình có thể phải chạy mấy vài phút) , ko thể đảm bảo chạy trong thời gian cho phép !

Có 1 thuật toán tối ưu hơn , đó là ta sẽ chia tập hợp trong đề bài thành 2 phần . Duyệt tất cả các trạng thái mỗi phần . Sau đó Tiếp tục Duyệt hay Qhd. Chi phí sẽ nằm ở cỡ $2^{(n/2)} * n \sim 1$ triệu , có thể đảm bảo chạy trong thời gian cho phép .

Bài tập ví dụ.

Bài toán : VECTOR < <http://vn.spoj.pl/problems/VECTOR/>>

Trong mặt phẳng tọa độ có N véc tơ. Mỗi một véc tơ được cho bởi hai chỉ số x và y . Tổng của hai véc tơ (x_i, y_i) và (x_j, y_j) được định nghĩa là một véc tơ $(x_i + x_j, y_i + y_j)$. Bài toán đặt ra là cần chọn một số véc tơ trong N véc tơ đã cho sao cho tổng của các véc tơ đó là véc tơ (U, V) .

Yêu cầu: Đếm số cách chọn thoả mãn yêu cầu bài toán đặt ra ở trên.

Input

Dòng thứ nhất ghi số N ($0 \leq N \leq 30$).

N dòng tiếp theo, dòng thứ i ghi các số nguyên x_i, y_i lần lượt là hai chỉ số của véc tơ thứ i . ($|x_i|, |y_i| \leq 100$). Dòng cuối cùng ghi số hai số nguyên U, V ($|U|, |V| \leq 10^9$).

Output

Gồm một số duy nhất là số cách chọn thoả mãn.

Example

Input:

```
4
0 0
-1 2
2 5
3 3
2 5
```

Output:

```
4
```

Gợi ý : Dễ dàng nhận xét với đề bài , có một thuật toán trâu bò , đó là duyệt tất cả các tổ hợp của tập vector đã cho , với mỗi trạng thái tìm được , tính tổng giá trị của các vector được chọn , từ đó tối ưu kết quả. Như đã nói ở trên , với cách làm này độ phức tạp sẽ là 2^n ($n \leq 30$) sẽ ko chạy trong thời gian cho phép .

Cách tối ưu hơn là : chia N véc tơ thành 2 phần mỗi phần gồm $N/2$ véc tơ.

Duyệt phần 1, được $2^{(N/2)}$ véc tơ tổng, lưu vào mảng 32767 phần tử.

Sort lại mảng này theo chỉ số x rồi y .

Một số vấn Đề Đáng chú ý trong môn tin học

Duyệt phần 2, với mỗi véc tơ tổng (x, y) duyệt được của phần 2, tìm xem có bao nhiêu véc tơ tương ứng $(U-x, V-y)$ trong phần 1. Ta có thể tìm bằng chia nhị phân.

Độ phức tạp $O(2^{(N/2)} * (N/2))$

Bộ nhớ $O(2^{(N/2)})$

Bài Toán : COIN34 < <http://vn.spoj.pl/problems/COIN34/> >

Bạn có 34 đồng xu có giá trị như sau:

xu(1) có giá trị 2

xu(2) có giá trị 3

xu(3) có giá trị 5

for $n = 4$ to 34

xu(n) có giá trị $(xu(n-1) + xu(n-2) + xu(n-3))$

Bạn hãy dùng nhiều đồng xu nhất để mua một món hàng có giá là X!

Dữ liệu

Dòng đầu tiên là số test (không quá 1000). Mỗi dòng tiếp theo chứa một số nguyên X ($1 \leq X \leq 2000000000$).

Kết quả

Với mỗi test, in ra "Case #" + số hiệu test + ": " + số lượng lớn nhất đồng xu cần dùng. Nếu không có cách nào để đạt giá trị X thì in ra -1.

Ví dụ

Dữ liệu

```
4
1
5
8
9
```

Kết quả

Case #1: -1

Case #2: 2

Case #3: 2

Case #4: -1

Gợi ý : Bài tập này có nhiều điểm giống với bài VECTOR , nhưng do dữ liệu đề bài nên ta sẽ chia các đồng xu đã cho thành 2 phần .

Phần 1 : các đồng xu thứ 1,2...20.

Phần 2: các đồng xu thứ 21,22,...34

Duyệt phần 1 , được 2^{20} “tổng “, ta lưu số đồng tiền nhiều nhất để đạt được

Mỗi tổng vào mảng F. Duyệt phần : duyệt tất cả các tổng , với mỗi tổng sinh ra , gọi đó là S , với mỗi test , ta tìm số đồng xu nhiều nhất ở phần 1 mà tổng giá trị của chúng là (X-S). từ đó tìm đc kết quả .

Bài Toán : CHNREST <http://vn.spoj.pl/problems/CHNREST/>

Hàng năm vì muốn có không khí ấm cúng và cũng để tiết kiệm nên bạn thường tổ chức sinh nhật ở nhà. Tuy nhiên trước sinh nhật năm nay vài hôm bạn đã thi đậu vào đội tuyển tin học quốc gia. Đây là một sự kiện đặc biệt có ý nghĩa nên bạn quyết định mừng ngày sinh nhật của mình tại một nhà hàng Trung Quốc sang trọng và bạn tự nhủ lần này nhất định phải tiêu xài rộng tay hơn. Mọi việc chuẩn bị đã gần xong nhưng còn một vấn đề làm bạn khá nhức đầu, đó là làm sao chọn được những món ăn mà mọi người cùng thích.

Nhà hàng có M món ăn khác nhau và thú vị ở chỗ là mỗi món ăn rất nhiều nên có thể đủ cho bao nhiêu người cũng được, vì thế vấn đề là gọi món nào chứ không phải mỗi món gọi bao nhiêu. Có tất cả N người đến dự tiệc sinh nhật (bao gồm cả bạn trong đó). Bạn đã tìm hiểu được danh sách những món ăn yêu thích của từng người và bạn muốn rằng đối với mỗi người phải có ít nhất 2 món mà họ thích. Tuy nhiên sau khi ăn xong còn nhiều tiết mục hấp dẫn khác nên bạn cũng muốn rằng bất kỳ ai cũng không có quá 2 món ăn yêu thích trong danh sách được đặt trước. Và vấn đề cuối cùng, đây là tiền của bố mẹ nên cũng không nên tiêu xài quá đáng.

Yêu cầu

Một số vấn Đề Đáng chú ý trong môn tin học

Hãy cho biết số tiền ít nhất phải trả để gọi một thực đơn thỏa mãn các yêu cầu trên.

Dữ liệu

- Dòng đầu tiên chứa hai số M, N
- Dòng thứ hai chứa M số P_i là giá của món thứ i .
- Trong N dòng cuối cùng, dòng thứ k ghi danh sách các món yêu thích của người thứ k .

Kết quả

- Gồm một số duy nhất là kết quả của bài toán, hoặc
- in ra -1 nếu không có cách gọi món nào thỏa mãn.

Ví dụ

Dữ liệu:

5 3

100 150 300 425 200

1 2 4

1 3 4 5

1 4 5

Kết quả:

450

Giới hạn

- $M \leq 30$.

- $N \leq 10$.

Gợi ý : với $m/2$ món ăn đầu tiên , ta duyệt tất cả các tổ hợp của nó,

với mỗi tổ hợp sinh ra thì số món ăn ưa thích trong list của mỗi người sẽ là (a_1, a_2, \dots, a_n) ,

rõ ràng chỉ xét những bộ nào có tất cả các $a_i \leq 2$, với mỗi bộ thì mã hóa ~thành ` số tự nhiên

tương tự cho nhóm 2 , với mỗi trạng thái (b_1, \dots, b_n) sẽ tìm đc trạng thái tương ứng bên nhóm 1

(sao cho $a_1 + b_1 = 2, \dots, a_n + b_n = 2$)

Luyện Tập : LQDDIV <http://vn.spoj.pl/problems/LQDDIV/>

Cho N người ($2 \leq N \leq 32$) , mỗi người có một số a_i ($1 \leq a_i \leq 10^9$) được gọi là độ tin cậy

Cần phân chia n người này vào 2 tập sao cho:

- Mỗi người thuộc đúng một tập
- Chênh lệch tổng độ tin cậy của 2 phần là bé nhất

Input

Dòng đầu chứa số nguyên N

Dòng tiếp theo chứa N số : số thứ i là độ tin cậy của người thứ i

Output

Ghi ra hai số u và v với u là độ chênh lệch nhỏ nhất và v là số cách phân chia

Example

Input:

5

1 5 6 7 8

Output:

1 3

Chú thích : Độ chênh lệch ít nhất của 2 phần là 1

Có 3 cách phân chia .3 cách phân chia nhóm 1 là (3,5) ,(1,3,4) và (1,2,5)

Chủ đề : Một Số Bài Toán Về Cây Khung

Với các bài toán về cây khung , ta thường gặp 2 dạng bài chủ yếu , đó là các bài toán có liên quan đến Cây khung nhỏ nhất , và các bài toán có liên quan đến cây khung lớn nhất .

Hai thuật toán thường được sử dụng là PRIM và KRUSCAL (bạn đọc có thể tham khảo trong tài liệu của thầy Lê Minh Hoàng hoặc Hồ Sỹ Đàm, xin ko nhắc lại lý thuyết ở đây)

Các bài tập.

Bài toán IOIBIN < <https://vn.spoj.pl/problems/IOIBIN/>>

Có N thùng nước được đánh số từ 1 đến N, giữa 2 thùng bất kỳ đều có một ống nối có một van có thể khóa hoặc mở. Ở trạng thái ban đầu tất cả các van đều đóng.

Bạn được cho một số yêu cầu, trong đó mỗi yêu cầu có 2 dạng:

Dạng X Y 1 có ý nghĩa là bạn cần mở van nối giữa 2 thùng X và Y.

Dạng X Y 2 có ý nghĩa là bạn cần cho biết với trạng thái các van đang mở / khóa như hiện tại thì 2 thùng X và Y có thuộc cùng một nhóm bình thông nhau hay không? Hai thùng được coi là thuộc cùng một nhóm bình thông nhau nếu nước từ bình này có thể chảy đến được bình kia qua một số ống có van đang mở.

Input

Dòng đầu tiên ghi một số nguyên dương P là số yêu cầu.

Trong P dòng tiếp theo, mỗi dòng ghi ba số nguyên dương X, Y, Z với ý nghĩa có yêu cầu loại Z với 2 thùng X và Y.

Output

Với mỗi yêu cầu dạng X Y 2 (với Z = 2) bạn cần ghi ra số 0 hoặc 1 trên 1 dòng tùy thuộc 2 thùng X và Y không thuộc hoặc thuộc cùng một nhóm bình.

Example

Input:

```
9
1 2 2
1 2 1
3 7 2
```

2 3 1
1 3 2
2 4 2
1 4 1
3 4 2
1 7 2

Output:

0
0
1
0
1
0

Giới hạn

- $1 \leq N \leq 10000$
- $1 \leq P \leq 50000$

Gợi ý : Đây là bài tập cơ bản - ứng dụng thuật toán KRUSKAL . Nếu 2 bình cùng thuộc 1 nhóm Bình Thì chúng sẽ cùng thuộc 1 gốc cây.

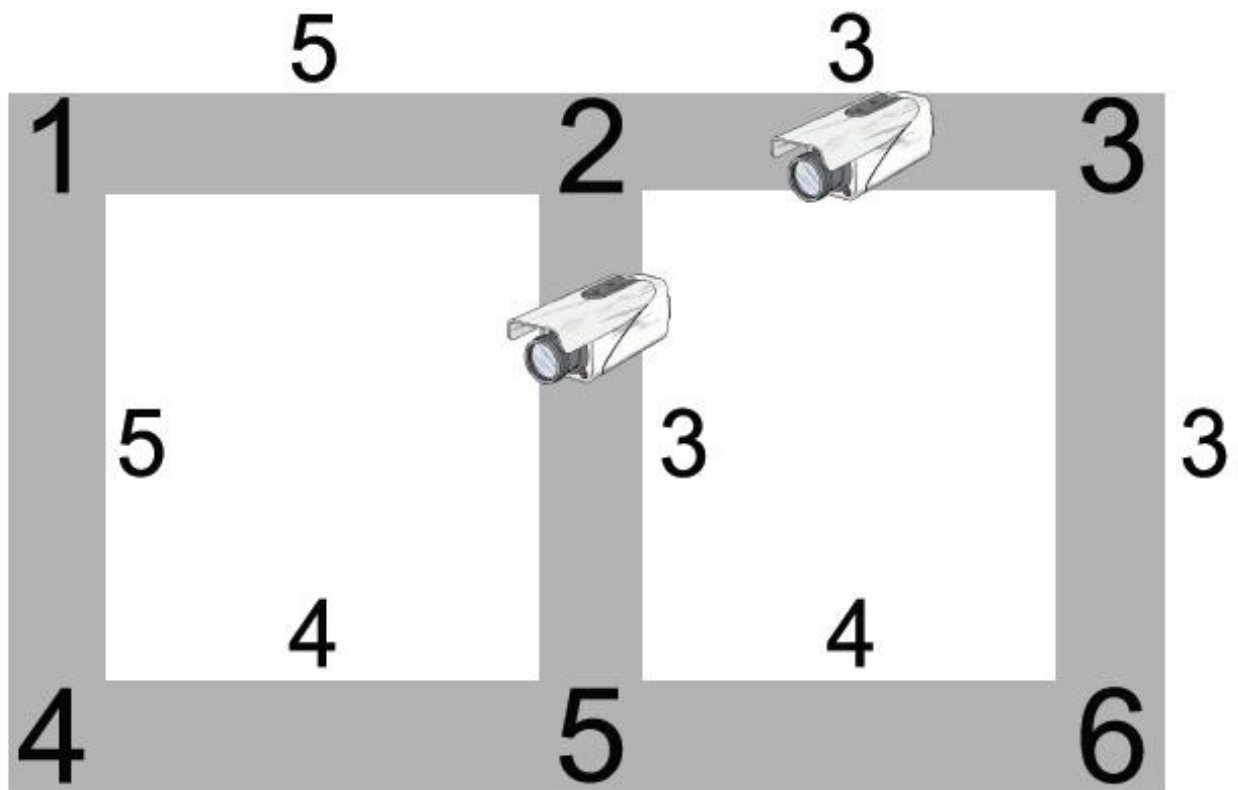
Bài Toán : NKRACING < <https://vn.spoj.pl/problems/NKRACING/>>

Singapore sẽ tổ chức một cuộc đua xe Công Thức 1 vào năm 2008. Trước khi cuộc đua diễn ra, đã xuất hiện một số cuộc đua về đêm trái luật. Chính quyền muốn thiết kế một hệ thống kiểm soát giao thông để bắt giữ các tay đua phạm luật. Hệ thống bao gồm một số camera đặt trên các tuyến đường khác nhau. Để đảm bảo tính hiệu quả cho hệ thống, cần có ít nhất một camera dọc theo mỗi vòng đua.



Hệ thống đường ở Singapore có thể được mô tả bởi một dãy các nút giao thông và các đường nối hai chiều (xem hình vẽ). Một vòng đua bao gồm một nút giao thông xuất phát, tiếp theo là đường đi bao gồm ít nhất 3 tuyến đường và cuối cùng quay trở lại điểm xuất phát. Trong một vòng đua, mỗi tuyến đường chỉ được đi qua đúng một lần, theo đúng một hướng.

Chi phí để đặt camera phụ thuộc vào tuyến đường được chọn. Các số nhỏ trong hình vẽ cho biết chi phí để đặt camera lên các tuyến đường. Các số lớn xác định các nút giao thông. Camera được đặt trên các tuyến đường chứ không phải tại các nút giao thông. Bạn cần chọn một số tuyến đường sao cho chi phí lắp đặt là thấp nhất đồng thời vẫn đảm bảo có ít nhất một camera dọc theo mỗi vòng đua. Viết chương trình tìm cách đặt các camera theo dõi giao thông sao cho tổng chi phí lắp đặt là thấp nhất.



Dữ liệu

- Dòng đầu tiên chứa 2 số nguyên n, m ($1 \leq n \leq 10000, 1 \leq m \leq 100000$) là số nút giao thông và số đường nối. Các nút giao thông được đánh số từ 1 đến n .
- m dòng tiếp theo mô tả các đường nối, mỗi dòng bao gồm 3 số nguyên dương cho biết hai đầu nút của tuyến đường và chi phí lắp đặt camera. Chi phí lắp đặt thuộc phạm vi $[1, 1000]$.

Kết quả

In ra 1 số nguyên duy nhất là tổng chi phí lắp đặt thất nhất tìm được.

Ví dụ

Dữ liệu:

6 7

1 2 5

2 3 3

1 4 5

4 5 4

5 6 4

6 3 3

5 2 3

Kết quả

6

Gợi ý: Ta Sẽ Tìm cây khung lớn nhất của đồ thị (thuật toán tương tự thuật toán tìm cây khung nhỏ nhất nhưng sort các cạnh ngược lại),

tổng trọng số của các cạnh không thuộc cây khung lớn nhất sẽ là kết quả bài toán.

Với dữ liệu đề bài , thuật toán để tìm cây khung lớn nhất sẽ là kruskal.

Bài 3 Sơ Đồ Mạng Điện

Một công ty ần thay thế toàn bộ hệ thống dây điện mắc nối tiếp giữa N phòng làm việc. Cho biết sơ đồ Mạng lưới điện hiện có của N căn phòng được biểu diễn bởi ma trận $A[i,j]$ trong đó $A[i,j]$ chính là độ dài dây điện nối giữa hai căn phòng i,j . Nếu không có dây điện thì giữa 2 phòng (i,j) thì $A[i,j] = 0$ và $a[j,i] = 0$. Hiện tại N phòng đều có dây điện nối đến hệ thống nhưng hệ thống đường dây này quá nhiều. Hãy tính

tổng độ dài ít nhất của dây dẫn cần sử dụng sao cho cả N phòng đều có điện.

Dữ liệu vào cho trong file SD.INP

+ dòng đầu tiên là số N

+ N dòng sau mô tả ma trận $A(n,n)$, mỗi dòng n số cách nhau 1 dấu cách.

Kết quả ghi ra file SD.OUT

+ dòng đầu là tổng độ dài các đường dây cần dùng

+các dòng sau mỗi dòng 2 số ui vi thể hiện có dây điện nối 2 phòng ui và vi ($i=1,2,\dots$)

Ví dụ

SD.INP

4

0 3 4 2

3 0 3 2

4 3 0 1

2 2 1 0

SD.OUT

5

3 4

1 4

2 4

Thuật toán : Xây dựng cây khung nhỏ nhất .

Bài toán NK CITY <http://vn.spoj.pl/problems/NKCITY/>

Nước Anpha đang lập kế hoạch xây dựng một thành phố mới và hiện đại. Theo kế hoạch, thành phố sẽ có N vị trí quan trọng, được gọi là N trọng điểm và các trọng điểm này được đánh số từ 1 tới N . Bộ giao thông đã lập ra một danh sách M tuyến đường hai chiều có thể xây dựng được giữa hai trọng điểm nào đó. Mỗi tuyến đường có một thời gian hoàn thành khác nhau.

Các tuyến đường phải được xây dựng sao cho N trọng điểm liên thông với nhau. Nói cách khác, giữa hai trọng điểm bất kỳ cần phải di chuyển được đến nhau qua một số tuyến đường. Bộ giao thông sẽ chọn ra một số tuyến đường từ trong danh sách ban đầu để đưa vào xây dựng sao cho điều kiện này được thỏa mãn. Do nhận được đầu tư rất lớn từ chính phủ, bộ giao thông sẽ thuê hẳn một đội thi công riêng cho mỗi tuyến đường cần xây dựng. Do đó, thời gian để hoàn thành toàn bộ các tuyến đường cần xây dựng sẽ bằng thời gian lâu nhất hoàn thành một tuyến đường nào đó

Yêu cầu: Giúp bộ giao thông tính thời gian hoàn thành các tuyến đường sớm nhất thỏa mãn yêu cầu

đã nêu.

Dữ liệu

Dòng chứa số N và M ($1 \leq N \leq 1000$; $1 \leq M \leq 10000$).

M tiếp theo, mỗi dòng chứa ba số nguyên u , v và t cho biết có thể xây dựng tuyến đường nối giữa trọng điểm u và trọng điểm v trong thời gian t . Không có hai tuyến đường nào nối cùng một cặp trọng điểm.

Kết quả

Một số nguyên duy nhất là thời gian sớm nhất hoàn thành các tuyến đường thỏa mãn yêu cầu đã nêu.

Ví dụ

Dữ liệu

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

Kết quả

```
3
```

Gợi ý : Xây dựng cây khung theo thuật toán KRUSKAL , sort các cạnh theo thứ tự từ bé đến lớn , cho đến khi nạp đủ N thành phố vào cây khung Thì kết quả là cạnh lớn nhất trong cách cạnh trong cây khung vừa tìm được (nói cách khác đó là cạnh cuối cùng được kết nạp vào Cây khung tr khi tất cả N thành phố được nạp vào cây khung).

Chủ Đề : Tìm Kiếm Nhị Phân và Ứng dụng

Trong thực tế , ta thường gặp dạng bài toán tìm thời điểm kết thúc sớm nhất (hay muộn nhất) của một công việc , hay tìm chi phí bé nhất (hay lớn nhất) ,... với các yêu cầu ràng buộc trong đề bài.

Khi đó ta có 1 thuật toán rất hiệu quả , đó là chặt nhị phân . Nó được mô tả như sau.

Yêu cầu : Cần tìm giá trị V thỏa mãn yêu cầu đề bài.

Thuật toán được mô tả như sau : { Ví dụ sau mô tả việc tìm V sao cho V min }

Dau := 0; cuoi := Vc;

While dau <> cuoi do

Begin

Mid := (dau+ cuoi) div 2; {chặt nhị phân}

If Kiemtra(mid) {neu mid thoa man} then dau:= mid+1 else cuoi:=mid;

End;

Bài toán : LIS (<http://vn.spoj.pl/problems/LIS/>)

Cho một dãy gồm N số nguyên ($1 \leq N \leq 30000$). Hãy tìm dãy con tăng dài nhất trong dãy đó. In ra số lượng phần tử của dãy con. Các số trong phạm vi longint.

Input

- Dòng đầu tiên gồm số nguyên N.
- Dòng thứ hai gồm N số mô tả dãy.

Output

Gồm một số nguyên duy nhất là đáp số của bài toán

Example

Input:

5
2 1 4 3 5

Output:

3

Gợi ý : Gọi k là độ dài cực đại của dãy con tăng và ký hiệu $H[1..k]$ là dãy có ý nghĩa sau : $H[i]$ là số hạng nhỏ nhất trong các số hạng cuối cùng của các dãy con tăng có độ dài i. Đương nhiên $H[1] < H[2] < \dots < H[k]$. Mỗi khi xét thêm một giá trị mới trong dãy A thì các giá trị trong dãy H và giá trị k cũng tương ứng thay đổi.

```
Res:=1; H[1]:=1;
For i:=2 to n do
Begin
  If A[i] < a[h[1]] then h[1]:=i
  else if a[i] > a[h[res]] then
  Begin
    Inc(Res); H[res]:=i;
  End
  else
  Begin
    d:=1; c:=Res;
    While d<>c do
    begin
      mid:=(d+c+1) div 2;
      If A[i] > a[h[mid]] then d:=mid else c:=mid-1;
    End;
    Mid:=(d+c) div 2;
    If (A[h[mid]] < a[i]) and (a[i]<a[h[mid+1]]) then
      h[mid+1]:=i;
  End;
End;
Writeln(fo,Res);
```

Bài toán : ASSIGN1(<https://vn.spoj.pl/problems/ASSIGN1/>)

Có n người, n việc ($1 < n \leq 200$). Người thứ i thực hiện công việc j mất $C[i,j]$ đơn vị thời gian. Giả sử tất cả bắt đầu vào thời điểm 0, hãy tìm cách bố trí mỗi công việc cho mỗi người sao cho thời điểm hoàn thành công việc là sớm nhất có thể.

Input

- Dòng đầu: N
- Tiếp theo là ma trận $C[i,j]$. (thuộc kiểu Integer)

Output

- Ghi thời điểm sớm nhất hoàn thành.

Example

Input:

```
4
10 10 10 2
10 10 3 10
4 10 10 10
10 5 10 10
```

Output:

```
5
```

Gợi ý : Chặt nhị phân thời điểm kết thúc sớm nhất . với mỗi giá trị đang xét , kiểm tra xem liệu có tồn tại Một cách ghép N thợ với N việc hay ko.(Sử dụng kiến thức ập ghép)

Bài toán : YUGI(<https://vn.spoj.pl/problems/YUGI/>)

Các bạn đã đọc bộ truyện tranh Nhật Bản Yugi-oh chắc hẳn ai cũng cực kì yêu thích trò chơi bài Magic. Bộ bài và chiến thuật chơi quyết định đến sự thắng thua của đối thủ(mà sự thắng thua thì còn liên quan đến cả tính mạng >_<). Vì thế tầm quan trọng của bộ bài là rất lớn. Một bộ bài tốt không chỉ bao gồm các quân bài mạnh mà còn phụ thuộc vào sự hỗ trợ tương tác giữa các quân bài. Bộ bài của Yugi là một bộ bài có sự bổ sung, hỗ trợ cho nhau rất tốt, điều này là 1 trong các nguyên nhân khiến Kaiba luôn là kẻ chiến bại.

Tình cờ Kaiba đã tìm được 1 quân bài ma thuật mà chức năng của nó là chia bộ bài hiện có của đối thủ ra làm K phần, mỗi phần có ít nhất 1 quân bài (điều này làm giảm sức mạnh của đối thủ).

Kaiba quyết định áp dụng chiến thuật này với Yugi. Hiện tại Yugi có trong tay N quân bài, 2 quân bài i, j có sức mạnh tương tác $a(i,j)$ ($a(i,j) = a(j,i)$). Kaiba muốn chia các quân bài thành K phần theo quy tắc sau:

- Giả sử K phần là P_1, P_2, \dots, P_k thì độ giảm sức mạnh giữa 2 phần u, v là $b(u,v) = \min(a(i,j))$ với i thuộc P_u, j thuộc P_v .
- Độ giảm sức mạnh của bộ bài là $S = \min(b(u,v))$ với $1 \leq u, v \leq K$.

Kaiba muốn chia K phần sao cho S lớn nhất

Input

- Dòng đầu là 2 số $N, K (2 \leq K \leq N \leq 200)$
- N dòng tiếp theo mỗi dòng là N số $a(i, j)$ ($a(i, j) \leq 32767$; nếu $i = j$ thì $a(i, j) = 0$)

Output

Gồm 1 dòng duy nhất là S lớn nhất

Example

Input:

```
4 3
0 1 2 3
1 0 2 3
2 2 0 3
3 3 3 0
```

Output:

```
2
```

Gợi ý : Chặt nhị phân , với mỗi giá trị V (sức mạnh) đang xét , nhóm tất cả các quân bài (i, j) thành 1 nhóm nếu

+ $I \# j$.

+ $a[i, j] \leq V$.

Để phân nhóm có thể sử dụng BFS.

Kiểm tra xem liệu số nhóm cần thiết để có giá trị V là bao nhiêu ? Nhỏ hơn K hay lớn hơn K , từ đó giảm dần khoảng cần xét .

bài toán : MTWALK(<https://vn.spoj.pl/problems/MTWALK>)

Cho một bản đồ kích thước $N \times N$ ($2 \leq N \leq 100$), mỗi ô mang giá trị là độ cao của ô đó ($0 \leq$ độ cao ≤ 110). Bác John và bò Bessie đang ở ô trên trái (dòng 1, cột 1) và muốn đi đến cabin (dòng N , cột N). Họ có thể đi sang phải, trái, lên trên và xuống dưới nhưng không thể đi theo đường chéo. Hãy giúp bác John và bò Bessie tìm đường đi sao cho chênh lệch giữa điểm cao nhất và thấp nhất trên đường đi là nhỏ nhất.

Dữ liệu

- Dòng 1: N
- Dòng 2..N+1: Mỗi dòng chứa N số nguyên, mỗi số cho biết cao độ của một ô.

Kết quả

Một số nguyên là chênh lệch cao độ nhỏ nhất.

Ví dụ

Dữ liệu

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

Kết quả

2

Gợi ý : Ta có nhận xét các giá trị độ cao nằm trong khoảng $[1, 110]$, vì thế ta có thuật toán chặt nhị phân như sau :

Giả sử ta đang xét giá trị V (Chênh lệch chiều cao)

- Vì chênh lệch tối ưu đang xét là V , suy ra nếu giá trị chiều cao nhỏ nhất trong đường đi tìm được là Min , thì giá trị chiều cao lớn nhất phải nhỏ hơn hoặc bằng $Max = Min + V$.

- Do chiều cao các ô ≤ 110 , nên việc thử lần lượt các giá trị dưới (giá trị Min) có thể chạy trong thời gian cho phép .

-Với mỗi giá trị V , ta loang tìm 1 đường đi thỏa mãn 2 điều kiện trên. Độ phức tạp $M \log M * N^2$ (M là giá trị)

VẤN ĐỀ : XẾP LỊCH CÔNG VIỆC

Bài toán xếp lịch công việc là một trong những bài toán phổ biến nhất trong thực tế : xếp lịch thi đấu, xếp lịch thi công công trình ,xếp lịch sửa chữa máy, xếp lịch học tập...

Yêu cầu chủ yếu của bài toán lập lịch là tìm ra phương án tối ưu nhất phù hợp với dữ liệu trong thời gian cho phép. Các phương pháp chủ yếu để giải quyết bài toán lập lịch chủ yếu là : sắp xếp topo, thuật toán Johnson, phương pháp heuristic, duyệt có đặt cận , qhi hoạch động , sử dụng luồng và đồ thị hai phía ..

Chúng ta sẽ đề cập đến từng thuật toán qua các bài toán dưới đây.

Phương pháp 1: **Duyệt có đặt cận**

Việc quan trọng nhất trong phương pháp này chính là tìm cận. Việc tìm cận hợp lí với thời gian cho phép ta có thể tìm được kết quả tương đối tối ưu.

Đặc điểm của phương pháp này là thường sắp xếp dữ liệu theo một khóa nào đó để thuận lợi cho việc duyệt tìm kết quả.

Bài toán 1: Cho n công việc , với mỗi công việc cho biết số tiền thu C_i được khi

hoàn thành , thời gian T_i thực hiện công việc, thời điểm R_i tối đa để kết thúc. Hãy xếp lịch thực hiện công việc tối ưu.

Dữ Liệu: XL01.INP :

- Dòng đầu là số n
- N dòng sau là 3 số nguyên T_i , R_i , C_i .

Kết quả : XL01.OUT :

Một số vấn Đề Đáng chú ý trong môn tin học

- Dòng đầu là tổng số tiền thu được.
- Các dòng sau gồm 4 số : số hiệu , thời điểm bắt đầu ,thời điểm kết thúc , giá trị công việc.
- Ví dụ:

XL01.INP	XL01.OUT
10	329
1 4 9	5 0 1 25
5 8 86	1 1 2 89
4 11 83	3 2 6 83
5 7 84	6 6 9 61
1 2 25	10 9 14
3 11 61	71
6 11 33	
4 7 28	
3 10 1	
5 14 71	

Hướng dẫn :

Sắp xếp các công việc theo thứ tự tăng của thời điểm kết thúc.

Chúng ta sẽ duyệt với cận : Tổng tiền đã thu được khi thực hiện thêm công việc thứ k + tổng tiền các công việc còn lại > kết quả tối ưu đã có .

Xây dựng một stack gồm 4 trường: số hiệu công việc tại đỉnh, thời gian bắt đầu,kết thúc, tiền công.

Nếu tổng tiền > kết quả tối ưu thì cập nhật.

Giả sử thực hiện công việc thứ k : nạp k vào stack,ta tính số tiền của các công việc còn lại = (tổng số tiền các công việc chưa thực hiện – số tiền của các công việc không thể thực hiện được nếu thực hiện công việc k).

Nếu tổng tiền + tiền còn lại > kết quả tối ưu thì duyệt tiếp ngược lại gỡ k ra khỏi stack và sửa lại giá trị tiền còn lại .

Kết quả của bài toán là các giá trị còn lại trong stack.

Bài toán 2: Có N công việc, mỗi công việc thực hiện trên một trong 2 máy A hoặc B. Thời gian thực hiện trên máy A là $A[i]$, trên máy B là $B[i]$. Hãy thực hiện hoàn tất các công việc sao cho thời gian hoàn thành là sớm nhất.

Dữ Liệu: XL02.INP

- Dòng đầu là số N.
- N dòng sau là mỗi dòng là 2 số $a[i], b[i]$.

Kết quả: XL02.OUT

- Dòng đầu là thời gian hoàn thành công việc
- Dòng thứ 2 là các công việc hoàn thành trên máy A
- Dòng thứ 3 là các công việc hoàn thành trên máy B

Hướng dẫn :

Xây dựng mảng một chiều $rest[i] = \text{tổng}(\min(a[k], b[k]))$ với $k=i+1..n$ với ý nghĩa $rest[i]$ là thời gian ít nhất cần có để thực hiện các công việc còn lại là $i+1..n$.

Giả sử đã làm xong công việc k, thời đã làm trên máy A là $sumA$, thời gian đã làm trên máy B là $sumB$ thì điều kiện để có thể chọn công việc k là :

$$(sumA + sumB + res[k+1]) / 2 \leq lsum \text{ với } lsum \text{ là kết quả tối ưu đã có.}$$

Chứng minh: Gọi thời gian để làm hết các công việc còn lại là T thì $T \geq res[k+1]$, thời gian còn làm trên máy A là TA , thời gian còn làm trên máy B là TB thì $T = TA + TB$. Hai lần thời gian thực hiện các công việc là :

$$sumA + sumB + tA + tB = sumA + sumB + T \geq sumA + sumB + res[k+1] \Rightarrow \text{đpcm.}$$

Bài toán 3: Cho n công việc cần được làm trên M máy. Thời gian hoàn thành công việc là $t[i]$. Mỗi công việc phải thực hiện liên tục trên một máy. Mỗi máy J không thể hoạt động liên tục quá $Q[j]$ giờ ($1 \leq j \leq M$), nếu muốn máy j hoạt động trở lại thì phải cho nó nghỉ $R[j]$ giờ. Tìm thời gian nhỏ nhất hoàn thành N công việc trên M máy.

Dữ liệu: XL03.INP gồm :

- Dòng đầu là 2 số N, M ($N \leq 15; M \leq 10$);
- Dòng thứ 2 là N số $T[i]$ ($0 < T[i] \leq 10$);
- Dòng thứ 3 là M số $Q[j]$ ($0 < Q[j] \leq 24$);
- Dòng thứ 4 là M số $R[j]$ ($0 < R[j] \leq 6$);

Kết quả: XL03.OUT :

- Dòng thứ nhất là Tmin thời gian hoàn thành N công việc.
- M dòng sau mỗi dòng j gồm các nhóm có 3 số: số hiệu công việc, thời gian bắt đầu, thời gian kết thúc tương ứng là lịch làm việc của máy j.

Ví dụ:

XL03.INP	XL03.OUT
10 3	14
1 2 3 4	8 0 4 6 5
5 4 3 4	9 4 10 14
5 3	10 0 3 7
4 3 8	4 7 3 8
1 1 2	11 2 12
	14
	1 0 1 9 1
	6 5 8 13

Hướng dẫn:

Duyệt theo công việc. Tạo một phương án đơn giản ban đầu với thời gian thực hiện xong N công việc là l_{tg} để làm cân ngay khi duyệt phương án tiếp theo. Khi xây dựng một phương án nếu đã duyệt xong k-1 công việc, tiếp tục chọn một máy làm công việc thứ k mà thấy thời gian để làm các công việc là từ 1 đến k là t_g > l_{tg} thì không thực hiện tiếp phương án này nữa. Điều kiện để chọn máy j nhận công việc k là: thời gian làm công việc k phải không vượt quá thời cho phép làm liên tục của máy j ưu tiên chọn những máy I nào vừa thực hiện xong sớm nhất công việc nay trước k mặt khác máy I phải là máy thực hiện công việc k tốt nhất (không có máy nào thực hiện xong công việc k sớm hơn nó). Để đảm bảo thời gian chạy cần đặt cận thời gian.

Bài toán 4: Có M môn học được đánh số từ 1..M và N khóa học đánh số 1..N. Mỗi khóa học đáp ứng 1 số môn học. Để tốt nghiệp học sinh cần hoàn thành tất cả các môn học. Hãy giúp chọn ra 1 số ít nhất các khóa học để tốt nghiệp.

Dữ liệu: XL04.INP :

- Dòng đầu là 2 số N,M (M,N<200);
- Trong N dòng sau mỗi dòng gồm M số: số 0 nếu khóa học j không đáp ứng được môn học I, 1 nếu ngược lại.

Một số vấn Đề Đáng chú ý trong môn tin học

Kết quả: XL04.OUT :

- Dòng đầu là số khóa học
- Dòng sau là số hiệu các khóa học.

Ví Dụ:

XL04.INP	XL04.OUT
4 3	2
0 0 1	2 3
1 0 0	
0 1 1	
1 0 0	

Hướng dẫn:

Cận: số khóa học được chọn < số khóa học tối ưu

Nếu đã đáp ứng đủ M môn học thì cập nhật kết quả.

Duyệt các khóa học với độ sâu phù hợp: mỗi lần duyệt ta chỉ đề cử một số khóa có nhiều môn học được thỏa mãn nhất tức là mỗi lần duyệt ta sắp xếp giảm các khóa học theo số lượng các môn học chưa hoàn tất mà các khóa học có thể đáp ứng được. Ngoài ra ta còn phải đặt cân j thời gian để đảm bảo thời gian chạy.

Bài toán 5: Robot quét vôi

Có 9 căn phòng được đánh số từ 1 đến 9 đã được quét vôi với các màu trắng xanh và vàng. Có 9 robot được đánh số từ 1 đến 9 phụ trách quét vôi các phòng. Mỗi robot chỉ quét một số phòng nhất định. Việc quét vôi được thực hiện nhờ một chương trình cài đặt tuân theo các qui tắc sau :

Một số vấn Đề Đáng chú ý trong môn tin học

- Nếu phòng đang màu trắng thì quét thành màu xanh
- . Nếu phòng đang màu xanh thì quét thành màu vàng
- . Nếu phòng đang màu vàng thì quét thành màu trắng

Mỗi lần chỉ được phép gọi một robot ra quét với nhưng mỗi robot có thể gọi nhiều lần. Robot được gọi sẽ quét với tất cả các phòng nó được phụ trách. Hãy tìm một phương án quét với sao cho cuối cùng tất cả các phòng đều màu trắng và tổng số lượt phòng phải quét với là ít nhất.

Dữ liệu : XL05.INP

- 9 dòng đầu, dòng thứ I mô tả danh sách các phòng mà robot thứ I phải quét với dưới dạng một xâu kí tự số liền nhau.
- Dòng cuối cùng là 1 xâu 9 kí tự mô tả trạng thái của các phòng. Kí tự 'V' tương ứng với màu vàng, kí tự 'X' tương ứng với màu xanh, kí tự 'T' tương ứng với màu trắng.

Kết quả: XL05.OUT

- Nếu không có phương án ghi ra -1.
- Nếu có phương án thì dòng đầu là thứ tự gọi các robot, dòng sau là số lượt phòng phải quét với.

Ví Dụ:

XL05.INP	XL05.OUT
159	2455688
123	21
357	
147	
532	

369	
456	
789	
258	
XTVVXVTXT	

Hướng dẫn :_

Duyệt đệ quy theo số lần quét liên tục của 1 robot.

Nhận xét:

- mỗi robot chỉ có thể gọi liên tục 0 đến 2 lượt
- Nếu mã hóa các kí tự 'T' là 0, 'X' là 1, 'V' là 2. Màu mỗi phòng sẽ trở thành màu trắng khi số hiệu màu hiện tại của nó + số lượt phòng được quét với là 1 số chia hết cho 3.
Từ đó có thể dễ dàng giải quyết được bài toán này.

Bài toán 6: Xếp vali

Cho một va li có thể chứa W đơn vị trọng lượng. Có N loại đồ vật (số lượng không hạn chế), đồ vật I có trọng lượng $A[i]$ và giá trị $C[i]$. Hỏi nên chọn số lượng mỗi đồ vật là bao nhiêu để giá trị thu được là lớn nhất.

Dữ liệu: XL06.INP

- Dòng thứ nhất là 2 số N và W
- N dòng tiếp theo : Dòng i+1 ghi 2 số $A[i], C[i]$.

Kết quả: XL06.OUT

- Dòng thứ nhất là tổng giá trị của vali.
- Các dòng tiếp theo mỗi dòng 2 số : số I (số hiệu vật được chọn) và số lượng vật i.

Ví dụ:

XL06.INP	XL06.OUT
4 100	110
50 50	2 1
19 20	3 1
80 90	
21 25	

Hướng dẫn :

Xây dựng mảng $r[i]=c[i]/a[i]$ là giá trị trung bình của mỗi đơn vị khối lượng của đồ vật thứ i.

Sắp xếp các mảng $a[i], c[i], r[i]$ theo chiều tăng của $r[i]$.

Tiến hành duyệt theo số lượng mỗi đồ vật.

Giả sử xét đến đồ vật thứ i khối lượng vali đã xếp là weight, giá trị hiện tại là cost. Tính số lượng max số đồ vật thứ I có thể bỏ vào vali là T.

Cận: Có thể cho vào X ($0 \leq X \leq T$) vật I nếu :

$$\text{Cost} + (R[i+1] * (W - (\text{weight} + a[i] * x))) > \text{Lcost}.$$

Phương pháp 2: Duyệt kết hợp heuristic

Trong các bài toán lập lịch xuất hiện một lớp bài toán NP là những bài toán không có thuật giải tốt trong thời gian cho phép. Chính vì vậy vấn đề đặt ra là tiến gần sát đến kết quả tối ưu càng gần càng tốt. Việc đoán nhận điều kiện thích hợp (heuristic) là rất quan trọng để đạt được kết quả gần sát tối ưu.

Các heuristics thường là sắp xếp dữ liệu theo một khóa nào đó, với mỗi kiểu sắp xếp tiến hành duyệt tham lam để được kết quả tối ưu nhất có thể

Bài toán 7: Chia N việc cho M máy

Cho N công việc, công việc i hoàn thành trong thời gian $t[i]$. Các công việc được thực hiện trên M máy công suất như nhau, mỗi máy đều có thể thực hiện được công việc bất kì trong N việc) mỗi công việc được làm liên tục trên một máy cho đến khi xong. Hãy tổ chức máy thực hiện đủ N công việc sao cho thời gian hoàn thành càng nhỏ càng tốt.

Dữ liệu: XL07.INP

- Dòng đầu gồm 2 số nguyên N, M
- Dòng tiếp theo là N số T_1, T_2, \dots, T_n

Kết quả: X107.OUT :

- Dòng đầu là thời gian hoàn thành n công việc
- M dòng sau dòng $i+1$ ghi các số hiệu công việc thực hiện trên máy i

Một số vấn Đề Đáng chú ý trong môn tin học

Ví dụ:

XL08.INP	XL08.OUT
6 3	8
2 5 8 1	3
5 1	2 1 4
	5 6

Hướng dẫn:

Bước 1: Sắp xếp các công việc giảm dần theo thời gian hoàn thành

Bước 2 : Lấy M công việc đầu phân công cho M máy, Mỗi máy làm một việc

Gọi MaxT là thời điểm hoàn thành M công việc này.

Bước 3: Repeat

Repeat

Chọn máy có thời gian làm nhỏ hơn MaxT, xếp thêm 1 công việc mới cho máy này.

Until :Nếu công việc mới vào thì không còn máy nào có thời gian dự kiến nhỏ hơn MaxT.

Cập nhật lại giá trị MaxT

Until Sắp xếp hết việc.

Bài toán 8:

Đóng gói

Có N đồ vật và một số gói hàng. Mỗi đồ vật có kích cỡ $V[i]$, các gói hàng có thể gói tối đa một kích cỡ To . Hãy chọn 1 số ít nhất các gói hàng để đóng gói N đồ vật trên.

Dữ liệu: XL08.INP.

Một số vấn Đề Đáng chú ý trong môn tin học

- Dòng đầu là số N,V0 ($N \leq 100$).
- Dòng thứ 2 là N số nguyên dương V1,V2..Vn.

Kết quả: XL08.OUT

- Dòng đầu là số S số gói hàng ít nhất cần chọn
- S dòng sau mỗi dòng là 1 số số nguyên chỉ số hiệu của đồ vật xếp trong gói i

Ví dụ:

XL08.INP	XL08.OUTPUT
5 5	3
3 3 2 2	1 3
2	2 4
	5

Hướng dẫn:

Cách 1:

Heuristic 1 : Sắp giảm theo kích thước đồ vật

- Theo thứ tự đó lần lượt cho xếp vào gói. Một gói tiếp tục nhận thêm đồ vật nếu tổng kích thước các đồ vật trong nó và đồ vật mới nhận còn chưa vượt quá Vo
- Nếu một máy nào đó không thể nhận tiếp công việc thì tăng số máy thêm 1.
Heuristic 2: Tương tự heuristic 1 nhưng mỗi lần chọn gói còn trống nhiều nhất.Nếu không còn gói nào gói được nữa thì tăng thêm 1 gói mới.

Cách 2:

Nhận xét số máy luôn \leq số đồ vật tức là $M \leq N$

Cho nên ta có thể chia nhỏ phân giá trị của M.Với mỗi giá trị đang xét ta sẽ tính kích thước lớn nhất .

Bài toán 9: Có N công việc cần thực hiện trên một máy, mỗi việc đòi hỏi đúng 1 giờ chạy máy. Với mỗi việc ta biết thời hạn cuối cùng phải nộp và tiền thu lao thu được nếu nộp kết quả đúng thời hạn. Chỉ có một máy, hãy lập lịch thực hiện một số công việc trên máy sao cho tổng số tiền thu được là lớn nhất (nếu có hai cách cùng thu số tiền như nhau thì chọn cách có ít thời gian hơn). Thời gian và tiền thưởng là các số nguyên dương.

Dữ liệu: XL09.INP

- Dòng đầu là số $N \leq 100$
- N dòng sau : dòng i+1 gồm 2 số ,số thứ nhất là thời hạn cuối cùng phải nộp,số thứ 2 là tiền thu lao được hưởng nếu nộp đúng hạn

Kết quả: XL09.OUT

- Dòng đầu ghi hai số : tổng tiền thưởng và thời gian làm các công việc đã chọn
- Các dòng sau là thứ tự thực hiện các công việc

Ví dụ:

XL09.INP	XL09 .OUT
10	111
4 4	7
5 1	8
2 21	3
3 5	10
12 27	7
5 4	9
4 24	2
2 18	5

5 6	
4 14	

Hướng dẫn:

Bước 1: Sắp giảm theo tiền thưởng công việc

Bước 2 : Lần lượt lấy các công việc đã sắp xếp đưa vào trục thời gian phân bố công việc theo cách sau:

- Từ thời điểm kết thúc mỗi công việc I lui về 1 đơn vị thời gian được đoạn thời gian thực hiện công việc i. Nếu trên trục còn trống đoạn này thì xác nhận đoạn thời gian này để thực hiện công việc i.
- Nếu đoạn này đã bố trí công việc khác thì lui nó về phía gốc một và xếp vào 1 đoạn còn trống
- Nếu không có đoạn còn trống nào thì bỏ công việc I xếp công việc khác
- Cuối cùng dời các đoạn về phía gốc để thu được thời gian thực hiện ít nhất

Phương pháp 3:

Các thuật toán riêng

1.Lập lịch trên hai máy

Bài toán 11: Có N chi tiết máy đánh số từ 1..N.Mỗi chi tiết cần phải gia công lần lượt trên 2 máy A và B. Thời gian thực hiện chi tiết i trên máy A là A[i] trên máy B là B[i]

Xếp lịch thực hiện gia công các chi tiết sao cho thời gian hoàn thành là sớm nhất.

Dữ liệu: XL07.INP :

- Dòng đầu là số N ($N \leq 10000$).
- Dòng thứ 2 gồm n số A[i]
- Dòng thứ 3 gồm n số B[i]

Kết quả: XL07.OUT :

- Dòng đầu số S thời điểm sớm nhất hoàn thành công việc
- Dòng thứ 2 ghi thứ tự thực hiện công việc.

Ví dụ:

XL11.INP	XL11.OUT
4	26
3 3	1 4 2 5
4 3	3
6 2	
5 7	
6 3	

a) Định lí Johnson (1954)

Phương án $V = \{v[1], v[2], \dots, v[N]\}$ lần lượt thực hiện các chi tiết $v[1], v[2], \dots, v[N]$ với thời gian nhỏ nhất khi và chỉ khi $\min(A_{v[k]}, B_{v[k+1]}) \leq \min(B_{v[k]}, A_{v[k+1]})$ $k=1, 2, \dots, N-1$,

Từ định lí trên có thể chứng minh sự đúng đắn của thuật toán sau:

b) Thuật toán :

+ Bước 1 : Chia các chi tiết thành 2 nhóm :

Nhóm 1 : Gồm các chi tiết mà $A_i \leq B_i$

Nhóm 2 : Gồm các chi tiết mà $A_i > B_i$

+ Bước 2 : Xếp nhóm 1 theo chiều tăng của A_i , xếp nhóm 2 theo chiều giảm của B_i .

+ Bước 3 : Nối nhóm 2 vào sau nhóm 1.

Bài toán 12: Lập lịch trên 3 máy .

Xét bài toán gia công N chi tiết trên 3 máy theo thứ tự A, B, C với bảng thời gian $A_i, B_i, C_i, i=1, 2, \dots, n$ thỏa mãn:
 $\max B_i \leq \min A_i$ hoặc $\max B_i \leq \min C_i$

Hướng dẫn:

Lịch gia công tối ưu trên 3 máy sẽ trùng với lịch gia công tối ưu trên 2 máy: máy thứ nhất với thời gian $A_i + B_i$ và máy thứ hai với thời gian $B_i + C_i$.

2. Thuật toán More

Bài toán 13: Lập lịch giảm thiểu trễ hạn

Có n công việc đánh số từ 1 đến n và một máy để thực hiện chúng. Biết:

$-p_i$ là thời gian cần thiết để hoàn thành công việc i .

- d_i là thời hạn hoàn thành công việc i .

Máy bắt đầu hoạt động từ thời điểm 0. Mỗi công việc cần được thực hiện liên tục từ lúc bắt đầu cho tới khi kết thúc, không cho phép ngắt quãng. Giả sử c_i là thời điểm hoàn thành công việc i . Khi đó, nếu $c_i > d_i$ ta nói công việc i bị hoàn thành trễ hạn, còn nếu $c_i \leq d_i$ thì ta nói công việc i được hoàn thành đúng hạn.

Yêu cầu: Tìm trình tự thực hiện các công việc sao cho số công việc hoàn thành trễ hạn là ít nhất.

Dữ liệu

- Dòng đầu tiên chứa số nguyên dương n ($0 < n \leq 1000$).
- Dòng thứ hai chứa n số nguyên dương p_1, p_2, \dots, p_n ($0 < p_i \leq 1000$).
- Dòng thứ ba chứa n số nguyên dương d_1, d_2, \dots, d_n ($0 < d_i \leq 1000$).

Kết quả

- Dòng đầu tiên ghi số lượng công việc bị hoàn thành trễ hạn theo trình tự tìm được.
- Dòng tiếp theo ghi n số nguyên dương là chỉ số của các công việc theo trình tự thực hiện tìm được.

Ví dụ

Dữ liệu

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

Kết quả

```
2
1 3 4 6 2 5
```

Thuật toán : More

Bước 1: Sắp xếp theo thứ tự tăng dần của thời điểm bàn giao

Bước 2 :Duyệt từ đầu cho đến khi gặp công việc quá hạn đầu tiên (Giả sử là công việc thứ k)

Tìm từ đầu cho đến công việc thứ k , công việc nào có t_i lớn nhất (Giả sử đó là công việc thứ m).

Nếu công việc này đã được chuyển một lần rồi thì dừng chương trình, còn nếu không thì ta chuyển công việc này xuống cuối. Rồi trở lại bước 2.

3.Lập lịch có thưởng phạt

Bài toán 14: Có N công việc đánh số từ 1 đến N cần bố trí thực hiện trên một máy. Biết:

- P_i là thời gian cần thiết để làm công việc i
- D_i là thời điểm cuối cùng phải hoàn thành công việc i.
- H_i là hệ số thưởng phạt của công việc i.

Mỗi công việc cần được thực hiện liên tục từ lúc bắt đầu đến khi kết thúc, không cho phép ngắt quãng. Thời gian chuyển từ công việc này sang công việc khác là không đáng kể. Giả sử T_i là thời điểm hoàn thành công việc I, khi đó giá trị thưởng phạt của công việc I là $H_i(D_i - T_i)$. Biết thời điểm có thể thực hiện công việc là 0, hãy tìm trình tự thực hiện N công việc sao cho tổng giá trị thưởng phạt của các công việc là lớn nhất.

Dữ liệu: XL14.INP

- Dòng đầu ghi số nguyên dương N ($N < 100001$).
- Dòng thứ hai ghi N số nguyên dương P_1, P_2, \dots, P_N .
- Dòng thứ ba ghi N số nguyên dương D_1, D_2, \dots, D_N .
- Dòng thứ tư ghi N số nguyên dương H_1, H_2, \dots, H_N .

Kết quả: XL14.OUT

- Dòng đầu là số giá trị thưởng phạt tìm được.
- Dòng thứ hai ghi N số nguyên dương là trình tự thực hiện công việc.

XL11.INP	XL11.OUT
4	-17
1 2 3 4	1 2 3 4
3 2 1 3	
1 2 1 2	

Một số vấn Đề Đáng chú ý trong môn tin học

XL11.INP	XL11.OUT
5	6
2 2 3	2 5 3 1 4
4 3	
3 2 10	
11 9	
1 2 2	
1 3	

Hướng dẫn: Gọi $V[1], V[2], \dots, V[N]$ là thứ thực hiện các công việc. Gọi giá trị thực hiện thường phạt của công việc $V[i]$ là $Q_{v[i]}$ và giả sử công việc $V[k-1]$ kết thúc tại thời điểm t thì tổng giá trị thường phạt là :

$$S1 = \sum_{k=1}^N Q_{v[k]} + H_{v[i]}(D_{v[i]} - t - P_{v[i]}) + H_{v[i+1]}(D_{v[i+1]} - t - P_{v[i+1]} - P_{v[i]}) + \sum_{k=i+2}^N Q_{v[k]}.$$

Khi trao đổi hai công việc $V[i]$ và $V[i+1]$ thì tổng giá trị thường phạt của N công việc là :

$$S2 = \sum_{k=1}^N Q_{v[k]} + H_{v[i+1]}(D_{v[i+1]} - t - P_{v[i+1]}) + H_{v[i]}(D_{v[i]} - t - P_{v[i+1]} - P_{v[i]}) + \sum_{k=i+2}^N Q_{v[k]}.$$

$$\text{Do đó để } S1 \geq S2 \text{ thì } H_{v[i+1]}P_{v[i]} \leq H_{v[i]}P_{v[i+1]} \Leftrightarrow H_{v[i+1]}/P_{v[i+1]} \leq H_{v[i]}/P_{v[i]}.$$

Áp dụng nhận xét trên, bằng cách xét lần lượt các công việc ở vị trí i trong dãy ($i=1, 2, \dots, N-1$) lần lượt so sánh với các công việc ở vị trí j ($j=i+1, i+2, \dots, N$) sẽ dẫn đến kết luận:

$$\text{Dãy } V[1], V[2], \dots, V[N] \text{ tốt nhất } \Leftrightarrow H_{v[i+1]}/P_{v[i+1]} \leq H_{v[i]}/P_{v[i]}$$

Thuật toán : sắp xếp các công việc theo giá trị tăng của $H_{v[i]}/P_{v[i]}$ thì dãy các công việc sau khi sắp xếp là dãy kết quả tối ưu từ đó ta có thể tính được tổng số tiền thưởng phạt.

Phương pháp 4: Quy hoạch động

Phương pháp quy hoạch động thường áp dụng cho những bài toán có dữ liệu vừa phải, và thường tìm được kết quả tối ưu trong thời gian chạy ngắn.

Tuy nhiên đây là một phương pháp khó, việc tìm công thức quy hoạch động không dễ chút nào. Chính vì vậy khi làm bài thi nếu không tìm được công thức quy hoạch động thì nên chuyển sang hướng đệ quy hoặc tham lam.

Bài toán 15: Sau khi thành công với phi vụ xuất khẩu bò sữa Farmer John đã có được một số tiền rất lớn và ông quyết định xây nhà. Bộ khung của ngôi nhà đã được hoàn thành, đó là một bộ khung có kiểu dáng rất đẹp. Farmer John quyết định trang trí nó thành ngôi nhà đẹp nhất trong làng. Để trang trí được cần trải qua N công đoạn, hoàn thành công đoạn I cần phải trả cho thợ P_i tiền. Ngoài ra nếu công đoạn j đã hoàn thành trước công đoạn I thì muốn thợ hoàn thành công việc I thì cần tốn S_{ij} tiền nữa (để đảm bảo khi thực hiện công đoạn I thợ không phá vỡ hoặc làm hỏng công đoạn). Ví dụ khi trang trí nội thất, nếu trong nhà chưa lắp cửa kính chỉ phải trả tiền sơn tường là 10 triệu đồng, nhưng khi đã sơn tường cần phải trả thêm 2 triệu đồng nữa). Là một con người luôn quý trọng đồng tiền mình làm ra Farmer John không muốn lãng phí một cách sai lầm. Vì vậy ông đã tìm đến bạn đề nghị lập kế hoạch trang trí sao cho số tiền phải trả là ít nhất. Là một lập trình viên không ngại khó khăn hãy viết chương trình tính số tiền ít nhất FJ phải trả.

Dữ Liệu : XL15.INP

Dòng đầu là số nguyên dương N ($0 < N < 16$).

N dòng tiếp theo mỗi dòng chứa đúng N số nguyên : số nguyên thứ I là giá trị gốc của việc I , số nguyên thứ j ($j > i$) là số tiền S_{ij} phải trả thêm cho công việc i .

Các giá trị nguyên không âm $< 10^6$.

Kết quả: XL15.OUT

một số duy nhất là giá trị nhỏ nhất tìm được.

Ví dụ:

XL15.INP	XL15.OUT
----------	----------

Một số vấn Đề Đáng chú ý trong môn tin học

2	30
10 10	
9000 10	

XL15.INP	XL15.OUT
3	47
14 23 0	
5 14 0	
1000	
9500 14	

Hướng dẫn:

Gọi H là tập hợp các công việc đã làm. $C[H]$ là chi phí nhỏ nhất để hoàn thành H . Do số các tập con của tập N công việc là $2^N, N \leq 15$ nên có thể dùng một số nguyên 2 byte để đại diện cho một tập con. Trong số nguyên này, bit thứ i của nó bằng 1 nghĩa là công việc i đã hoàn thành, ngược lại bit thứ $i=0$ nghĩa là chưa hoàn thành. Ta có thể đồng nhất kí hiệu h là số nguyên tương ứng với tập H nói trên.

Thực hiện QHĐ theo số h tăng dần từ 1 đến $2^N - 1$:

- Khởi trị $C[h]=0$ với mọi $h=0..2^N-1$. Riêng $C[2^{i-1}]=s[i,i]$ vì tập công việc H tương ứng với $h=2^{i-1}$ chỉ có một công việc duy nhất là i .

- Giả sử đã xét đến số h , lần lượt xét các bit i là bit 1 từ phải qua trái: Nếu bỏ bit i trong số h (cho bit $i=0$) ta được số k thì :

$$C[h] = \min\{C[k] + s[i,i] + s[j,i] \mid j \text{ là số hiệu các bit 1 trong số } k\}$$

Từ đó ta sẽ có được kết quả.

Bài toán 16: Hội trường

Nhà trường có một phòng hội trường. Có những yêu cầu muốn sử dụng phòng hội trường này, mỗi yêu cầu cho biết thời điểm bắt đầu và thời điểm kết thúc. Nhà trường có thể chấp nhận hoặc từ chối đối với một yêu cầu.

Yêu cầu: hãy giúp nhà trường chọn các yêu cầu sử dụng hội trường sao cho tổng thời gian hội trường được sử dụng là lớn nhất.

Dữ liệu

Dòng đầu tiên chứa một số nguyên dương n ($n \leq 10000$), số yêu cầu.

Mỗi dòng trong số n dòng tiếp theo chứa 2 số nguyên dương p và k ($0 \leq p < k \leq 30000$), mô tả một yêu cầu bắt đầu tại thời điểm p và kết thúc tại thời điểm k .

Kết quả

Gồm một dòng duy nhất là tổng thời gian lớn nhất mà hội trường được sử dụng

Ví dụ

XL16.INP	XL16.OUT
12	12
1 2	
3 5	
0 4	
6 8	
7 13	
4 6	
9 10	
9 12	
11 14	
15 19	
14 16	
18 20	

Hướng dẫn:

- Sắp xếp các công việc theo tăng dần của thời điểm kết thúc.
- Sử dụng mảng $T[i] \ i=0 \dots 30001$ là số hiệu của công việc cuối cùng có thời điểm kết thúc là i .
- Sử dụng mảng $F[i] \ i=0 \dots 30001$ với ý nghĩa tính đến thời điểm I máy đã thực hiện được $F[i]$ giờ. Ban đầu khởi tạo bằng 0.
- Gọi $D[i], C[i]$ lần lượt là thời điểm bắt đầu và kết thúc của công việc i .
Ta QHD như sau:

For $i=1 \dots 30001$ {xét các thời điểm}

Begin

$F[i] = \text{Max}(F[i-1], F[D[k]] + C[k] - D[k])$ với k là số hiệu các công việc có $C[k] = i$.

End.

$F[30001]$ là kết quả của bài toán.

Bài toán 17: Dịch bệnh lở mồm long móng đang hoành hành vì lo lắng đàn bò bị bệnh nên Farmer John đã bán rẻ tất cả số bò và quyết định đổi nghề. Lấy toàn bộ số tiền bán bò farmer John mở một cửa hàng cho thuê máy cày. Tại thời điểm 0, ông John nhận được đơn đặt hàng thuê sử dụng máy cày của N khách hàng. Các khách hàng được đánh số từ 1 đến N . Khách hàng I cần sử dụng máy từ thời điểm D_i đến thời điểm C_i (D_i, C_i là các số nguyên và $0 < D_i < C_i < 10^9$) và sẽ trả tiền sử dụng máy là P_i ($P_i < 10^7$). Hai đơn đặt hàng I và J được nhận nếu khoảng thời gian được nhận nếu thời gian sử dụng không giao nhau. Vì mới bước vào nghề nên Farmer John không biết nên nhận nhưng đơn nào để thu được lợi nhuận lớn nhất, là một lập trình viên bạn hãy giúp ông ấy tính xem lợi nhuận lớn nhất là bao nhiêu và danh sách các đơn đặt hàng được nhận.

Dữ liệu:

XL17.INP

- Dòng đầu là số N ($0 < N \leq 1000$)
- Dòng thứ $i+1$ trong N dòng sau là 3 số D_i, C_i, P_i .

Kết quả:

XL17.OUT

- Dòng đầu gồm 2 số : số đơn đặt hàng được nhận và tổng tiền công thu được.
- Dòng thứ 2 ghi số hiệu của các đơn đặt hàng được nhận.

Một số vấn Đề Đáng chú ý trong môn tin học

XL17.INP	XL17.OUT
3	2 180
150 500	2 3
150	
1 200	
100	
400 800	
80	

XL17.INP	XL17.OUT
4	2 1100
400 821	2 4
800	
200 513	
500	
100 325	
200	
600 900	
600	

Hướng dẫn:

- Sắp xếp các khoảng thời gian thuê sử dụng máy theo điểm đầu của nhưng khoảng này.
- Gọi $Nh[i]$ là số tiền tối ưu thu được khi xét xong các công việc từ 1 đến i (theo thứ tự đã sắp xếp).
- Sau đó sửa nhãn từ $Nh[i]$ đến $Nh[n]$ như sau : Gán $Nh[i]=P_i$.Để sửa nhãn $Nh[i]$, tìm công việc j có nhả lớn nhất trong các công việc từ 1 đến $i-1$ mà thời điểm hoàn thành của công việc j sớm hơn thời điểm bắt đầu của công việc i , sử dụng công thức:

$$Nh[i]=\text{Max}(Nh[i],Nh[j]+P_i, C_j < D_i)$$

Để tìm được thứ tự thực hiện công việc thì sử dụng thêm mảng $trace[i]$, gán $trace[i]=j$ nếu j là đơn làm cho $Nh[i]$ max.

Phương pháp 5: Sử dụng đồ thị 2 phía và Luồng

Để có thể ứng dụng đồ thị hai phía và luồng vào bài toán lập lịch các bạn cần nắm vững kiến thức và cách một số bài toán cơ bản sau đây:

- Tìm cặp ghép có lực lượng cực đại (cặp ghép có nhiều cạnh nhất).
- Tìm cặp ghép có trọng số cực đại, trọng số cực tiểu
- Tìm Luồng cực đại trên mạng
- Tìm lát cắt nhỏ nhất.

Bài toán: Có N thợ và M việc. Với mỗi thợ cho biết danh sách các công việc mà anh ta có thể làm được. Hãy phân M việc cho N thợ sao cho mỗi thợ chỉ làm một việc mỗi việc chỉ giao cho 1 thợ làm và số việc làm được là lớn nhất.

Dữ liệu:

- Dòng đầu 2 số N, M ($0 < N, M < 101$).
- N dòng sau mỗi dòng gồm M số 0 hoặc 1. Bằng 0 nếu thợ I không làm được việc j, bằng 1 nếu thợ I có thể làm được việc J.

Kết quả:

- Dòng đầu là số S, số công việc lớn nhất làm được.
- S dòng sau mỗi dòng gồm 2 số I, j biểu diễn phân công việc j cho thợ i.

Ví dụ:

XL18.INP	XL18.OUT
4 4	4
0 1 1 1	1 4
0 1 0 1	2 2
1 0 1 1	3 1
0 0 1 1	4 3

Hướng dẫn: Xây dựng đồ thị 2 phía, một phía gồm N đỉnh tượng trưng cho N thợ, một phía là M đỉnh tượng trưng cho M việc, có cạnh nối i-j nếu $a[i,j]=1$. Bài toán trở thành tìm cặp ghép có lực lượng cực đại.

Bài toán : Phân công hoàn thành sớm nhất

Có n người, n việc ($1 < n \leq 200$). Người thứ i thực hiện công việc j mất $C[i,j]$ đơn vị thời gian. Giả sử tất cả bắt đầu vào thời điểm 0, hãy tìm cách bố trí mỗi công việc cho mỗi người sao cho thời điểm hoàn thành công việc là sớm nhất có thể.

Dữ liệu: XL19.INP

- Dòng đầu: N
- Tiếp theo là ma trận $C[i,j]$. (thuộc kiểu Integer)

Kết quả: XL19.OUT

- Ghi thời điểm sớm nhất hoàn thành.

XL19.INP	XL19.OUT
4	5
10 10 10	
2	
10 10 3	
10	
4 10 10	
10	
10 5 10	
10	

Hướng dẫn:

- Chặt nhị phân thời điểm sớm nhất hoàn thành công việc gọi đó là CM.

Left:=1;Right:=Maxint;

Repeat

Cm:=(left+right) div 2.

- Tiến hành tìm cặp ghép cực đại với điều kiện: cạnh I,j có thể ghép vào nếu $C[I,j] \leq Cm$.

Nếu có thể tìm được cặp ghép có N cạnh thì giá trị lưu giá trị kết quả bằng Cm đồng thời giảm
Right:=Cm-1 ngược lại Left:=Cm+1.

Until Left>Right.

Bài toán: Có M sinh viên và N chuyên đề cho các sinh viên này ($M, N < 100$). Cho ma trận $A[1..M, 1..N]$ trong đó $A[i, j] = 0$ nếu sinh viên I có nguyện vọng học chuyên đề j, $A[i, j] = 2$ nếu sinh viên không có nguyện vọng học chuyên đề.

Cho dãy P gồm M số P_i ($1 \leq i \leq M$): P_i là số chuyên đề sinh viên I cần phải học.

Hãy tìm các bố trí sinh viên học đủ các chuyên đề phù hợp nguyện vọng của họ đồng thời các chuyên đề có số lượng sinh viên theo học chênh lệch nhau càng ít càng tốt.

Dữ Liệu: XL20.INP

- Dòng đầu ghi hai số M, N.
- M dòng sau là ma trận A (mỗi dòng ghi N số).
- Dòng cuối cùng là M số P_1, P_2, \dots, P_M .

Kết quả: XL20.OUT

- Dòng đầu ghi số sinh viên theo chuyên đề có đông sinh viên tham gia nhất.
- M dòng tiếp theo : Dòng thứ I ($1 \leq i \leq M$) ghi P_i số là các chuyên đề do lịch xếp cho sinh viên I theo học.

Ví dụ:

XL20.INP	XL20.OUT
8 8	4
2 0 2 0 0	5 6 7
0 0 2	2 5 7 8
0 0 2 2 0	7 8
2 0 0	1 2 3 5
2 2 0 0 0	1 4 5 8
0 0 0	1 3 6
0 0 0 2 0	3 4 8
0 2 2	3 5 6 7
0 2 2 0 0	
2 2 0	
0 0 2 0 0	
0 0 0	
0 2 0 0 0	
2 0 0	
2 0 0 2 0	

0 0 2	
3 4 2 4 4	
3 3 4	

Hướng dẫn:

- Chặt nhị phân số học sinh tham gia chuyên đề đồng nhất gọi đó là k.

- Xét luồng trên đồ thị hai phía (phía trái N đỉnh: mỗi đỉnh là số hiệu của sinh viên ,phía phải M đỉnh : mỗi đỉnh là số hiệu nguyên vọng),thêm 2 đỉnh T và S .Trọng số trên các cung (t,i) với $i=1,2,\dots,N$ nguyên vọng của sinh viên I.Trọng số trên các cung (j,S) với $j=N+1,N+2,\dots,N+m$ đều bằng k.Các cung (I,j) có trọng số bằng 1 nếu $a[I,j-N]=0$,ngược lại nếu $a[I,j-N]=2$ thì trọng số cung bằng 0.

Nếu thì tìm được luồng cực đại thì thỏa mãn .

Luyện tập:

Bài toán 21: (Dựa theo đề thi chọn đội tuyển quốc gia năm 1997-1998).

Có N công việc đánh dấu từ 1..N .Mỗi công việc I có thời gian hoàn thành là T_i , thời điểm bắt đầu thực hiện là B_i ,thời điểm tối đa phải kết thúc là C_i .Hãy lập lịch thực hiện được nhiều công việc nhất.

Dữ liệu: XL11.INP.

Một số vấn Đề Đáng chú ý trong môn tin học

- Dòng đầu là số N
- N dòng sau mỗi dòng gồm 3 số Ti,Bi,Ci.

Kết quả: XL11.OUT

- Dòng đầu là số M số công việc xếp được
- M dòng sau mỗi dòng gồm 3 số : số hiệu công việc là được,thời điểm hoàn thành,thời điểm kết thúc

Ví dụ:

XL21.INP	XL21.OUT
8	4
25 22 47	3 8 `12
3 22 47	4 16 41
4 8 20	2 41 44
25 16 41	5 45 54
9 45 63	
24 47 71	
16 27 47	
24 39 63	

Bài toán 22: Lập lịch sửa chữa ô tô

Một cơ sở sửa chữa ô tô có nhận n chiếc xe để sửa. Do các nhân viên làm việc quá lười nhác nên đã đến hạn trả cho khách hàng mà vẫn chưa tiến hành sửa được chiếc xe nào. Theo hợp đồng đã ký kết từ trước, nếu bàn giao xe thứ i quá hạn ngày nào thì sẽ phải trả thêm một khoản tiền phạt là A[i].

Ông chủ cơ sở sửa chữa quyết định sa thải toàn bộ công nhân và thuê nhân công mới. Với lực lượng mới này, ông ta dự định rằng để sửa chiếc xe thứ i sẽ cần $B[i]$ ngày. Vấn đề đặt ra đối với ông là phải lập lịch sửa tuần tự các chiếc xe sao cho tổng số tiền bị phạt là ít nhất.

Yêu cầu: Hãy lập lịch sửa xe giúp cho ông chủ cơ sở sửa chữa ô tô.

Dữ liệu

- Dòng 1: Chứa số n ($n \leq 10000$)
- Dòng 2: Chứa n số nguyên dương $A[1], A[2], \dots, A[n]$ ($1 \leq A[i] \leq 10000$)
- Dòng 3: Chứa n số nguyên dương $B[1], B[2], \dots, B[n]$ ($1 \leq B[i] \leq 100$)

Kết quả

- Dòng 1: Ghi số tiền bị phạt tối thiểu
- Dòng 2: Ghi số hiệu các xe sẽ tiến hành sửa chữa, theo thứ tự từ xe được sửa đầu tiên đến xe sửa sau cùng

Ví dụ

Input:

4

1 3 4 2

3 2 3 1

Output:

44

4 2 3 1

Xong công việc 4 vào cuối ngày 1 \Rightarrow phải trả $2 * 1 = 2$.
Xong công việc 2 vào cuối ngày 3 \Rightarrow phải trả $3 * 3 = 9$.
Xong công việc 3 vào cuối ngày 6 \Rightarrow phải trả $6 * 4 = 24$.
Xong công việc 1 vào cuối ngày 9 \Rightarrow phải trả $1 * 9 = 9$.
Vậy tổng cộng phải trả 44 .

Bài toán 23: Đàn bò đồng đánh

Đại dịch lở mồm long móng đã qua đi Farmer John lại quay lại nghề chăn nuôi bò sữa cho ông. Lúa bò này của ông có khả năng cho sữa rất tốt nhưng cũng rất đồng đánh. Chúng chỉ ăn cỏ ở cánh đồng chúng thích và chỉ cho sữa nếu được ăn ngay tại cánh đồng đó (chúng không ăn cỏ khô).

Farmer John có N cánh đồng cỏ, hiện tại có k con bò đồng đánh đang ở trên k cánh đồng. Hãy giúp Farmer John đưa k con bò đến các cánh đồng mà các chú bò thích ăn.

Hiện tại Farmer John đang đứng ở cánh đồng thứ 0, và để đưa con bò thứ j đến cánh đồng I thì John phải đến cánh đồng j đón con bò tại đó rồi đưa về đồng cỏ i. Trên đường Farmer John có thể mang theo nhiều con bò.

Là một người rất quý trọng thời gian nên Farmer John muốn đi thật nhanh.

Bạn hãy giúp ông ấy tìm cách đưa các con bò sao cho tốn ít thời gian nhất.

Dữ Liệu: XL23.INP

- Dòng đầu là 2 số N, K ($K < N < 1000$);
- K dòng sau mỗi dòng gồm 2 số là vị trí của con bò đang đứng và cánh đồng cỏ mà nó yêu thích.

Kết quả: XL24.INP

- Dòng đầu là thời gian ngắn nhất cần dùng.
- Dòng thứ 2 gồm k số là thứ tự các con bò đến cánh đồng nó thích.

Ví dụ:

XL23.INP	XL23.OUT
10 5	12
1 3	
2 4	
4 3	
5 10	
7 10	

Vấn đề về xử lý số nguyên(số nguyên lớn, hàm mod).

I) Xử lý số nguyên lớn.

Trong tin học có nhiều lúc chúng ta phải làm việc với những con số lớn, thậm chí là rất lớn, khi đó các kiểu dữ liệu chuẩn của ngôn ngữ lập trình không thể biểu diễn được, buộc ta phải tìm các cách biểu diễn khác. Để biểu diễn 1 số nguyên lớn, các bạn có thể dùng xâu hoặc mảng, mỗi phần tử của xâu hoặc mảng biểu diễn 1 chữ số của số cần tìm. Ở đây chúng ta sẽ xét ở trên mảng. 1 số lớn được khái báo = 1 kiểu record như sau đây.

Type bignum=record

D:longint;

CS:array[0..maxcs] of byte;

End;

D là số lượng chữ số của nó, mảng cs biểu diễn các chữ số của số đang xét. Để thuận tiện, ta sẽ lưu các chữ số ngược lại so với thứ tự viết. VD số 123456 thì mảng CS sẽ lần lượt là CS[1]=6;..CS[D]=1. Lưu các chữ số kiểu này sẽ dễ dàng khi thực hiện các phép tính trên số lớn.

Bài toán:

Cho 2 số nguyên A và B có không quá 1000 chữ số, tính A+B, A-B, A*B.

Hướng dẫn:

1) Phép cộng, ta thực hiện như phép cộng đã được học ở trường, tức là cộng từ hàng đơn vị đến hàng chục... Ta cần có 1 biến nhớ để lưu nhớ, sau khi đến hàng tiếp theo thì phải cộng cả nhớ vào nữa. Thủ tục mẫu:

Procedure cong(a,b:bignum);

Var n,i,nho:longint;

Begin

If a.d>b.d then n:=a.b else n:=b.d;

Nho:=0;

For i:=1 to n do

Begin

```
    Nho:=nho+a.cs[i]+b.cs[i];  
c.cs[i]:=nho mod 10;  
nho:=nho div 10;  
    End;  
    c.d:=n;  
    if nho>0 then  
        begin  
inc(c.d);  
c.cs[c.d]:=nho;  
        end;  
    End.
```

Kết quả là số C.

2) Phép trừ, giả sử số bị trừ > số trừ. Ta cũng cần 1 biến nhớ để thực hiện phép trừ như chúng ta thường làm. Đến hàng I, nếu $nho + b[i] > a[i]$ thì $c[i] := 10 + a[i] - nho - b[i]$ và gán $nho := 1$, ngược lại, tức là $nho + b[i] \leq a[i]$ thì $c[i] := a[i] - nho - b[i]$ và $nho := 0$. Các bạn tự viết thử tục.

3) Phép nhân: phép nhân thì có phức tạp hơn 1 chút.

```
Procedure nhan(a,b:bignum);  
Var I,j:longint;  
Begin  
    For i:=1 to a.d do  
        For j:=1 to b.d do  
            Inc(c.cs[i+j],a.cs[i]*b.cs[j]);  
    For i:=1 to a.b+b.d do c.cs[i]:=c.cs[i+1];  
    For i:=a.b+b.d downto 1 do if c.cs[i]<>0 then break;  
    c.d:=I;  
End.
```

Sau khi cài đặt được 3 thủ tục trên, ta đã có thể hoàn thành bài toán, chỉ lưu ý ở phép trừ, nếu $A < B$ thì ta cần in ra 1 số âm, vì vậy trước hết in ra dấu trừ rồi đổi chỗ A và B, sau đó thực hiện trừ như bình thường. Hàm so sánh 2 số lớn cũng không đơn giản, các bạn có thể tự cài đặt.

Bây giờ 1 vấn đề đặt ra là, nếu có 1 bài toán mà mỗi số có khoảng 10000 chữ số, và các bạn phải thực hiện các phép tính $+$, $-$, $*$ nhiều lần, như vậy thì nếu ta vẫn thực hiện như các thủ tục trên đây thì không thể chấp nhận được về mặt thời gian. Vì vậy chúng ta cần có sự cải tiến. Để ý là ta đang thực hiện các phép tính trên hệ cơ số 10, vậy thì ta sẽ nghĩ đến thực hiện những phép tính trên các hệ cơ số lớn hơn, và để không làm phức tạp hóa vấn đề thì những cơ số này là những bội số của 10, các bạn có thể tăng lên cơ số 10^9 (nếu dùng kiểu longint để lưu trữ), hoặc thậm chí là 10^{17} , 10^{18} (nếu các bạn dùng int64 hoặc qword để lưu). Chỉ cần chỉnh sửa những thủ tục trên ở chỗ mod 10 và div 10 thành mod coso và div coso là xong. Nhưng khi in ra cần có 1 chút chỉnh sửa. Do đây là cơ số lớn nên mỗi phần tử cần phải có đủ $\log_{10}(\text{coso})$ chữ số (trường hợp $\text{coso}=10$ là có đủ 1 chữ số), nhưng trong PASCAL những chữ số 0 vô nghĩa ở đầu sẽ bị bỏ đi, nên khi write 1 phần tử nào đó, ta phải bổ sung những chữ số 0 vào đầu đến khi đủ số chữ số rồi mới ghi ra (trừ phần tử đầu tiên vì những số 0 trước phần tử này không có nghĩa).

Bài toán: Đếm dây

Đếm số dãy tăng dần (độ dài dãy phải ≥ 2) có các phần tử nguyên dương mà tổng đúng bằng n

Giới hạn

$n \leq 4000$

Dữ liệu vào

- Ghi duy nhất số n

Dữ liệu ra

- Ghi ra số cách

Ví Dụ

Input:

5

Output:

2

Giải thích: Có 2 dãy là (1,4) và (2,3)

Đây là 1 bài toán QHĐ. Bài này quy về bài toán tìm số nghiệm của phương trình $x_1 + x_2 + \dots + x_k = n$ với $x_1 < x_2 < x_3 \dots < x_k$. Gọi $F[n, k]$ là kq bài toán.

TH1: Tất cả các số đều xi đều > 1 , lúc đó $F[n, k] = F[n - k, k]$.

TH2: $x_1=1, \Rightarrow F[n,k]:=F[n-1,k-1]$ (loại x_1 ra).

Kết hợp 2 trường hợp này lại ta có công thức:

$$F[n,k]:=F[n-k,k]+F[n-1,k-1].$$

Vậy là ta cần tìm các $F[n,k]$ thỏa mãn $k*(k+1) \leq n$ và cộng lại vào biến kết quả, kết quả bài toán là rất lớn, cho dù các bạn có tìm ra công thức QHĐ nhưng vẫn cài đặt ở hệ cơ số 10 thì cũng không thể giải quyết trọn vẹn bài toán. Vì thế nên giải pháp của bài này là dùng hệ cơ số lớn. Cơ số càng lớn thì càng rút gọn thời gian chạy chương trình lại.

II) Hàm mod trong các bài toán tin học.

Trong tin học có nhiều bài toán cần động đến xử lý số lớn, nhưng khi ý đồ của tác giả chỉ là tìm ra thuật toán đúng để giải quyết bài toán chứ không coi trọng việc cài đặt số lớn, lúc đó thường thì tác giả sẽ yêu cầu ghi ra số dư của kết quả khi chia cho 1 cơ số bất kỳ nào đó. Vì vậy mà ta cần tìm hiểu những tính chất của hàm mod để có thể đảm bảo kết quả mình đưa ra là chính xác.

Giả sử có 2 số x và y , với cơ số là n , có những tính chất sau đây:

1. $(x+y) \bmod n = (x \bmod n + y \bmod n) \bmod n$.

2. $(x-y) \bmod n = (x \bmod n - y \bmod n) \bmod n$; (ở đây có thể hiểu 1 cách đơn giản là nếu $x \bmod n - y \bmod n < 0$ thì cộng thêm n vào kết quả).

3. $(x*y) \bmod n = (x \bmod n) * (y \bmod n) \bmod n$.

Ngoài ra còn có thể có nhiều tính chất khác, nhưng chừng này cũng đủ để các bạn dùng trong hầu hết các bài toán thường gặp.

Nếu bài Đếm dãy ở trên chỉ yêu cầu chúng ta in ra phần dư của kết quả khi chia cho 123456789 thì mỗi lần tính $F[i,j]$ ta lại lấy $F[i,j] \bmod 123456789$. Khi cộng vào biến kết quả ta cũng lấy kết quả $\bmod 123456789$. Do ở đây chỉ thực hiện phép $+$ nên theo tính chất 1 thì cuối cùng kết quả ta tìm được là chính xác.

Bài toán: Số huyền bí (Mystery).

Đất nước Văn Lang thời cổ xưa đã có những hiểu biết tân tiến về số học. Tương truyền rằng, vua Hùng Vương thứ 17 cùng các trưởng lão trong triều đình đã phát minh ra các số huyền bí. Các số này giúp chỉ dẫn đường vào kho tàng của đất nước.

Theo các chứng tích khảo cổ, các nhà khoa học kết luận rằng số huyền bí cơ sở a bằng tích của $(3^d - 1)$ với mọi ước số $d > 0$ của a .

Bờm thích số học đồng thời cũng rất thích tìm hiểu lịch sử đất nước. Bạn hãy giúp Bờm tính số huyền bí cơ sở a ($1 \leq a \leq 10^9$). Do kết quả có thể rất lớn, bạn chỉ cần in ra phần dư của số huyền bí cơ sở a khi chia cho 20122007.

Dữ liệu

Gồm một số nguyên a duy nhất.

Kết quả

In ra số nguyên duy nhất là phần dư của số huyền bí cơ sở a khi chia cho 20122007.

Ví dụ

Dữ liệu:
10

Kết quả
7291779

Hướng dẫn:

Với bài này thì ta cần tìm tất cả các ước số của a , sau đó lấy số dư của (3^d-1) khi chia cho 20122007, sau đó áp dụng tính chất 3 để lấy kết quả. Vấn đề mấu chốt ở đây là làm sao tính số dư của 3^d-1 cho 20122007. Nếu dùng 1 vòng for từ 1-> d thì sẽ không thể đảm bảo thời gian vì có những ước rất lớn. Để ý tính chất 3, ta có 1 thuật toán tính trong $\log(d)$ như sau: Nếu d chẵn là thì $3^d=3^{(d \div 2)}*3^{(d \div 2)}$. Nếu d lẻ thì $3^d=3*3^{(d \div 2)}*3^{(d \div 2)}$. Vì vậy ta chỉ cần phải tính số dư của $3^{(d \div 2)}$ là có thể tính được cho d , vậy 1 thủ tục đệ quy là rất thích hợp trong trường hợp này. Sau đó áp dụng tính chất 2 để lấy phần dư của (3^d-1) do ở đây xuất hiện dấu trừ.

Tree Num(TREENUM)

Một số được gọi là số tree_num khi nó được là tổng của các lũy thừa cơ số 3 với số mũ không âm tăng dần.

Ví dụ $30=3^1+3^3$; $325=3^0+3^4+3^5$ là những số tree_num.

Yêu cầu :Tìm số tree_num thứ n

Input

Dòng đầu chứa số nguyên dương ntest là số test($ntest \leq 30000$)

Ntest dòng sau:mỗi dòng chứa số nguyên dương n ($0 \leq n \leq 10^{19}$).

Output

Ntest dòng mỗi dòng chứa số tree_num thứ n

Sample

Input:

5

1

2

7

3

6

Output:

1

3

13

4

12

Thuật toán

Nhìn vào định nghĩa của số TREE_NUM thứ n , ta nhận thấy: Nếu thay cơ số 3 bằng cơ số 2 thì số tree_num thứ n chính là số n . Từ đó ta có thuật toán sau:

Phân tích n ra hệ nhị phân. Giả sử $n = 2^x1 + 2^x2 + \dots + 2^xk$ thì số tree_num được tính bằng công thức $3^x1 + 3^x2 + \dots + 3^xk$. Như vậy ta có thể sử dụng xử lý bit trong bài toán này, nếu bit thứ j là 1 thì cộng 3^j và kết quả. Nhưng bài toán chưa dừng lại ở đây.

$N \leq 10^{19}$ tức là n có tối đa 64 bit. Ta lại phải sử dụng số lớn để giải quyết bài này, do khi chuyển từ 2 sang 3 thì kết quả đã lớn hơn rất nhiều. Nếu ta cộng trừ trên cơ số 10 thì khó có thể chạy trong thời gian cho phép với số lượng test lớn (≤ 30000). Vậy bài này chúng ta nên thực hiện xử lý với cơ số lớn, mảng lũy thừa của 3 cần được tính trước để tránh tình trạng tính lặp lại nhiều lần, gây lãng phí và làm chậm chương trình.

Space settlement(SPACESET)

Đã lâu lắm rồi, ở một thiên hà xa xôi ... Một đế chế đã xây dựng nên một trạm không gian vĩ đại. Trạm không gian này có m tầng, và mỗi tầng gồm có n khu vực. Các khu vực được nối với nhau bởi các tuyến đường để các tàu không gian có thể bay qua.

Một tầng là một vòng tròn với n khu vực được đặt trên đó. Vòng tròn cung cấp các tuyến đường bay cho các tàu không gian. Trong vòng một tầng, việc di chuyển chỉ được thực hiện dọc theo vòng tròn đó. Vòng tròn định hướng đường bay sẽ luôn tồn tại, bất chấp số lượng khu vực ở trong tầng đó.

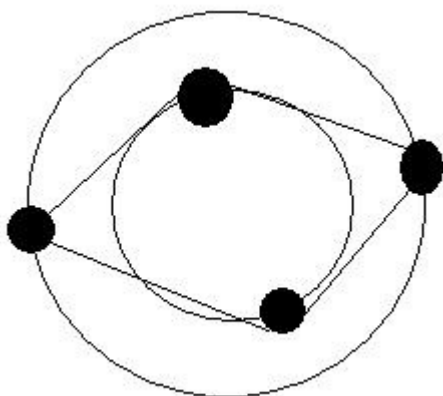
Các tầng được tổ chức thành một hệ thống. Một sự kết nối hoàn hảo được cung cấp giữa các tầng. Tại mỗi khu vực trong một tầng sẽ có các tuyến đường nối tới mọi khu vực ở tầng ngay trên và ngay dưới nó. Tầng trên cùng và dưới cùng cũng được nối theo cách tương tự.

Chàng phi hành gia thích phiêu lưu -- Han Solo muốn chở mọi người giữa các khu vực. Tàu không gian của Solo có thể bay theo một hành trình có chính xác k điểm dừng, không hơn không kém.

Cho giá trị của k , n và m . Bạn phải tính số hành trình trong trạm không gian mà Solo có thể dùng để thực hiện công việc của mình.

Lưu ý

1. Một khu vực có thể được thăm bao nhiêu lần tùy ý trong 1 hành trình.
2. Một hành trình được coi là một dãy các đỉnh sao cho tồn tại một cạnh giữa 2 đỉnh liên tiếp. Hai hành trình được coi là khác nhau nếu chúng khác nhau tại một đỉnh bất kỳ.
3. Tầng trên cùng và dưới cùng chỉ được nối với nhau nếu như chúng khác nhau.
4. Hình ảnh tượng trưng



Trong hình, các dấu chấm màu đen thể hiện các khu vực. Sự kết nối được biểu diễn bằng các vòng tròn và đường thẳng.

Input

Dòng đầu chứa T , số lượng test. Mỗi test chứa các giá trị của m , n và k .

Output

Chứa T dòng, mỗi dòng cho 1 test. Mỗi dòng sẽ ghi phần dư của tổng số hành trình có thể sử dụng khi chia cho 12345678.

Example

Input:

```
5
1 1 1
1 2 1
```

3 3 56
4 3 4
691 60 97764

Output:

1
2
8019378
6144
470730

Constraints

Dataset 1: $T \leq 100$, $m \leq 1000$, $n \leq 1000$, $k \leq 100000$ Time limit: 5s

Dataset 2: $T \leq 100$, $m \leq 10^8$, $n \leq 10^8$, $k \leq 10^8$ Time limit: 5s

Thuật toán:

Mỗi hành trình có k điểm dừng. Điểm đầu tiên trong hành trình có $m \cdot n$ khả năng để chọn, sau đó từ điểm thứ 2 trở đi, mỗi điểm có $(2 \cdot n + 2)$ khả năng để chọn (số điểm ở vòng kẻ dưới + số điểm ở vòng kẻ trên + số điểm kẻ điểm đó trên vòng tròn hiện tại).

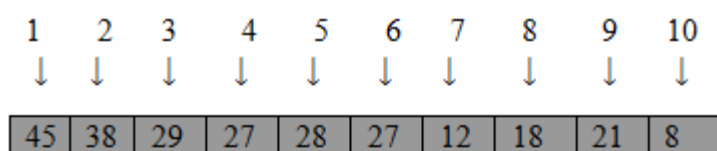
Từ đó ta có công thức tính sau: Kết quả $:= (m \cdot n \cdot (2 \cdot n + 2)^{(k-1)}) \bmod 12345678$.

Đây là 1 trong những bài toán mà áp dụng tính chất thứ 3 của hàm mod rất nhiều lần. Dựa vào tính chất này các bạn có thể tính ra kết quả không mấy khó khăn. Chỉ xin lưu ý các bạn 1 vài trường hợp đặc biệt khi $m, n \leq 2$. Lúc này thì công thức của chúng ta có 1 chút thay đổi (các bạn tự tìm công thức của những trường hợp này). 1 điều nữa là nếu các bạn áp dụng tính chất 3 những không cẩn thận với kiểu dữ liệu có thể gây tràn số.

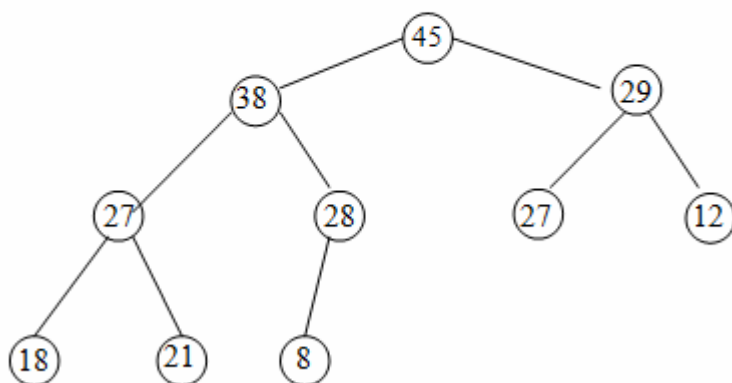
Vấn Đề HEAP VÀ ỨNG DỤNG

-Heap là 1 trong những cấu trúc dữ liệu đặc biệt quan trọng trong tin học, nó giúp chúng ta có thể giải được nhiều bài toán với thời gian cho phép. Sau đây ta sẽ tìm hiểu chi tiết về HEAP.

-Đầu tiên heap là 1 cây nhị phân có quan hệ giữa nút cha và 2 nút con của nó. Loại heap max là heap mà mọi có khóa lớn hơn hoặc bằng các nút con của nó(nếu nó không phải là nút lá). Trong loại heap này thì ta có thể dễ dàng nhận thấy khóa của gốc là khóa lớn nhất. Loại heap min thì được định nghĩa tương tự nhưng ngược lại. Sau đây là hình ảnh của 1 heap max.



(Hình 1: Biểu diễn mảng của một Heap)



(Hình 2: Biểu diễn cây của một Heap)

-Từ đây chúng ta chỉ xét trên heap max, heap min làm tương tự, chú ý là với heap min thì nút cha có khóa bé hơn nút con nên khi thực hiện các so sánh thì phải đảo chiều.

-Do heap là 1 cây nhị phân, nên cách đơn giản nhất để biểu diễn nó là dùng 1 mảng 1 chiều n phần tử(với n là số lượng phần tử lớn nhất cần quản lí). 2 nút con của nút k là $2*k$ và $2*k+1$, nút cha của nút k là $k \div 2$. Gọi nheap là số phần tử hiện tại của heap.

Ta cần thực hiện những thao tác sau với 1 heap:

- Upheap: Khi có 1 nút có giá trị khóa tăng lên thì có thể cấu trúc heap không còn được bảo toàn nữa, nó có thể phải ở 1 vị trí khác(do nó lớn hơn nên có thể phải gần gốc hơn). Cần có 2 biến cha và con để thực hiện quá trình chỉnh sửa heap, ta tưởng tượng nút con lúc đầu là vị trí cần chỉnh sửa, nếu nút cha của nó có giá trị khóa < giá trị khóa của nó thì đổi chỗ nút con cho nút cha, nút con mới

lúc này là nút cha cũ. Cứ tiếp tục như thế đến khi gặp 1 vị trí mà nút cha có giá trị khóa \geq nó hoặc vị trí nút con hiện tại = 1 (nút gốc).

procedure upheap(v:longint);

Var cha, con, x:longint;

Begin

Con:=v;x:=h[v].key;

While con \geq 2 do

Begin

Cha:=con div 2;

If h[cha].key \geq x then break;

H[con]:=H[cha];

Con:=cha;

End;

H[con]:=v;

End.

- b) Downheap: Ngược lại với upheap, có thể có giá trị khóa của phần tử nào đấy giảm đi thì phải tiến hành điều chỉnh lại heap. Đầu tiên nút cha ở vị trí cần thay đổi. Nếu giá trị khóa ở nút cha < giá trị khóa lớn nhất trong các nút con thì tiến hành đổi chỗ nút cha và nút con có giá trị khóa lớn nhất này. Nút cha mới là nút con cũ. Cứ tiếp tục như thế cho đến khi gặp 1 vị trí mà giá trị khóa của nút cha \geq giá trị khóa của nút con lớn nhất hoặc là nút đó không có nút con nữa. Các bạn có thể tham khảo thủ tục sau đây.

Procedure downheap(v:longint);

Var cha, con, x:longint;

Begin

Cha:=v;x:=h[v].key;

While cha*2 \leq n do

Begin

Con:=cha*2;

If (con \leq nheap)and(h[con].key<h[con+1].key) then inc(con);

If h[con].key \leq x then break;

H[cha]:=h[con];

Cha:=con;

End;

H[cha]:=v;

End.

Nếu trong quá trình xử lí, các bạn cần biết số hiệu các phần tử thì thêm vào 1 mảng sh nữa, lưu ý là khi nào có sự thay đổi vị trí của 1 nút thì cũng thay đổi giá trị của mảng sh theo để đảm bảo $sh[h[v]]:=v$;

Khi các bạn đã có trong tay 2 thủ tục upheap và downheap thì các thao tác còn lại có thể quy về 2 thủ tục này:

-Thêm 1 phần tử vào heap: Cho phần tử này nằm ở vị trí $nheap+1$ rồi thực hiện thao tác upheap.

-Lấy 1 phần tử ra khỏi heap(thường là ta cần lấy phần tử ở gốc vì đây là phần tử nhỏ nhất hoặc lớn nhất trong 1 đoạn): Gán phần tử cuối vào phần tử cần loại bỏ, đồng thời $dec(nheap)$, sau đó coi như phần tử này bị thay đổi giá trị, giá trị khóa mà tăng lên thì thực hiện upheap còn không thì thực hiện downheap.

Sau khi ta đã cài đặt được các thủ tục cần có của 1 heap, ta sẽ tìm hiểu ứng dụng của nó khi giải các bài toán trong tin học. Như ta thấy, heap max hay heap min có thể cho ta lấy phần tử lớn nhất hoặc phần tử nhỏ nhất trong 1 đoạn. Vì vậy nó rất hữu dụng trong các bài toán mà ta cần phải truy vấn các phần tử lớn nhất hoặc các phần tử nhỏ nhất trên 1 đoạn(chẳng hạn như các bài toán QHĐ thì ta thường cần tìm những giá trị này). Hơn nữa, mỗi thủ tục của heap có độ phức tạp không quá $O(\log n)$ (điều này các bạn có thể tự chứng minh), vì vậy heap giúp những bài toán loại này có thể giải quyết đơn giản hơn trong thời gian cho phép.

Đường đi ngắn nhất

Cho 1 đồ thị có n đỉnh, m cạnh ($1 \leq n, m \leq 10^5$), tìm độ dài đường đi ngắn nhất giữa 2 đỉnh s và t của đồ thị.

Thuật toán:

Bài này với dữ liệu nhỏ ($n \leq 5000$) thì các bạn dùng thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh s . Nhưng với $n \leq 10^5$ mà thuật toán Dijkstra cổ điển có độ phức tạp n^2 thì không khả thi, vậy ta phải tìm cách cải tiến.

Trong thuật toán Dijkstra, mỗi lần ta cần tìm 1 đỉnh trong số các đỉnh chưa được đánh dấu sao cho đỉnh này có giá trị đường đi hiện tại là nhỏ nhất. Điều này làm ta nghĩ đến dùng heap. Dùng 1 heap min lưu chỉ số của các đỉnh (giá trị đường đi ngắn nhất hiện tại lưu trong mảng d). Mỗi lần tìm đỉnh min, ta lại lấy đỉnh ở gốc ra, khi cập nhật nếu có đỉnh nào được thay đổi giá trị thì ta thực hiện upheap(hoặc thêm vào nếu phần tử này chưa có trong heap). Để thực hiện cập nhật mảng d thì nên lưu đồ thị bằng danh sách liên kết hoặc lưu kiểu Forward-star. Độ phức tạp thuật toán là $m \log(n)$. Thuật toán Dijkstra kết hợp cấu trúc heap là 1 trong những thuật toán thường gặp nhất với cấu trúc dữ liệu quan trọng bậc nhất này.

Một chút về Huffman Tree(HEAP1)

Một người nông dân muốn cắt 1 thanh gỗ có độ dài L của mình thành N miếng , mỗi miếng có độ dài là 1 số nguyên dương $A[i]$ ($A[1] + A[2] + \dots + A[N] = L$) . Tuy nhiên để cắt một miếng gỗ có độ dài là X thành 2 phần thì ông ta sẽ mất X tiền . Ông nông dân này không giỏi tính toán lắm , vì vậy bạn được yêu cầu lập trình giúp ông ta cho biết cần để dành ít nhất bao nhiêu tiền thì mới có thể cắt được tấm gỗ như mong muốn .

*Lưu ý : Kết quả có thể vượt **longint** (trong Pascal) và vượt **long** (trong C++) đấy nhé .*

Input

Dòng 1 : 1 số nguyên dương T là số bộ test .

T nhóm dòng tiếp theo mô tả các bộ test , mỗi nhóm dòng gồm 2 dòng :

Dòng 1 : số nguyên dương N ($1 \leq N \leq 20000$) .

Dòng 2 : N số nguyên dương $A[1], \dots, A[N]$. ($1 \leq A[i] \leq 50000$)

Output

Kết quả mỗi test ghi ra trên 1 dòng , ghi ra 1 số nguyên dương duy nhất là chi phí tối thiểu cần để cắt tấm gỗ .

Example

Input:

1
4
1 2 3 4

Output:

19
Đầu tiên cắt miếng gỗ thành 2 phần có độ dài 6 và 4 . Sau đó cắt tiếp miếng có độ dài 6 -> 3 và 3 .
Cắt 1 miếng 3 thành 2 phần có độ dài 1 , 2 . Như vậy chi phí là $10 + 6 + 3 = 19$.

Thuật toán:

Đầu tiên cho tất cả các phần độ dài trên vào 1 heap min, thực hiện vòng lặp $n-1$ lần như sau: Lấy 2 phần tử đầu tiên(tức là 2 phần tử có giá trị nhỏ nhất trong heap hiện tại) ra, sau đó thêm 1 phần tử mới có giá trị = tổng 2 phần tử vừa lấy ra cho vào heap. Như thế sau $n-1$ lần, heap chỉ còn 1 phần tử và giá trị của phần tử này chính là kết quả cần tìm, phần chứng minh xin nhường bạn đọc. Một lưu ý nữa là phải khai báo mảng heap kiểu `int64` vì kết quả có thể vượt quá `longint`.

KMIN

Cho 2 dãy số nguyên A và B. Với mọi số $A[i]$ thuộc A và $B[j]$ thuộc B người ta tính tổng nó. Tất cả các tổng này sau khi được sắp xếp không giảm sẽ tạo thành dãy C.

Nhiệm vụ của bạn là: Cho 2 dãy A, B. Tìm K số đầu tiên trong dãy C

Input

Dòng đầu tiên gồm 3 số: M, N, K

M dòng tiếp theo gồm M số mô tả dãy A

N dòng tiếp theo gồm N số mô tả dãy B

Output

Gồm K dòng tương ứng là K phần tử đầu tiên trong dãy C

Example

Input:

```
4 4 6
1
2
3
4
2
3
4
5
```

Output:

```
3
4
4
5
5
5
```

Giới hạn

- $1 \leq M, N, K \leq 50000$
- $1 \leq A_i, B_i \leq 10^9$

Thuật toán:

Với $m, n \leq 50000$ thì ta không thể tạo 1 mảng 50000^2 rồi sắp xếp lại được, vì vậy chúng ta cần những thuật toán tinh tế hơn, 1 trong những thuật toán đơn giản mà hiệu quả nhất là dùng heap.

Đầu tiên ra sắp xếp 2 mảng A và B không giảm, dĩ nhiên phần tử đầu tiên chính là $A[1]+B[1]$, vấn đề là phần tử thứ 2 là $A[2]+B[1]$ hay $A[1]+B[2]$, ta xử lí như sau:

Trước hết ta tạo 1 heap min gồm các phần tử $A[1]+B[1], A[1]+B[2], \dots, A[1]+B[n]$, mỗi khi lấy ra phần tử ở gốc (tức là phần tử có giá trị nhỏ nhất trong heap hiện tại), giả sử phần tử đó là $A[i]+B[j]$, ta lại đẩy vào heap phần tử mới là $A[i+1]+B[j]$ (nếu $i=m$ thì không đẩy thêm phần tử nào vào heap). Cứ thế cho đến khi ta lấy ra đủ k phần tử cần tìm.

Độ phức tạp thuật toán trong trường hợp lớn nhất là $O(50000 \cdot \log(50000))$, có thể chạy trong thời gian 1 s, bộ nhớ lưu trữ tỷ lệ với n.

Ball game

Hôm qua AnhdQ mới mời cô gái của mình đi ăn món gà KFC (vốn là sở trường của hai người). Hai người quyết định gọi phần ăn Chicky (dành cho trẻ em :d), vừa tiết kiệm, mà còn được tặng kèm món đồ chơi ngộ nghĩnh có tên là "Bắt gà" :d Sau một hồi loay hoay, AnhdQ phát hiện ra rằng trò chơi của bọn trẻ con nước ngoài này khá là giống với trò chơi gù của trẻ con Việt Nam :d (tất nhiên là hiện đại hơn ^^). Sau đó AnhdQ quyết định rủ nàng đi xem phim ở MegaStar : "> trên đường ghé qua khu StarBowl, cầm trên tay món đồ "Bắt gà", AnhdQ chợt nảy ra một trò chơi thú vị kết hợp giữa những quả bóng bowling và trò chơi gù dân gian, và chàng quyết định hai nhóc Alice và Bob sẽ là những vị khách đầu tiên tham gia trò chơi mới này! Chàng tạm đặt tên cho trò chơi là "Trò chơi với những trái bóng":

AnhdQ làm sẵn một cái máng có N lỗ xếp liên nhau được đánh số từ 1..N và một đường rãnh giúp đưa các quả bóng vào các lỗ này. Đường rãnh này có thiết kế đặc biệt như sau:

- Chỉ có thể ném bóng vào từ một trong hai đầu của đường rãnh.
- Tại một thời điểm chỉ có một quả bóng được ném vào đường rãnh.
- Nếu bóng được ném vào từ đầu nào thì đường rãnh sẽ đưa quả bóng tới lỗ **trống gần đầu đó nhất**.
- Thời gian đưa bóng tới một lỗ bất kì là không đáng kể.

AnhdQ làm ra M quả bóng đặc biệt cho Alice và Bob chơi, những quả bóng này có đặc điểm như sau:

- Chúng có cùng hình dạng, kích thước; vậy nên có thể rơi vào bất kì lỗ nào trên máng.
- Mỗi quả bóng có một độ xoáy (có thể giống hoặc khác nhau); nếu một quả bóng có độ xoáy là x thì nó sẽ xoáy trên miệng lỗ mà nó rơi vào trong x phút trước khi rơi hẳn xuống lỗ.

Lại nói về cái máng:

- Mỗi lỗ trên máng có sức chứa vô hạn.
- Một lỗ được gọi là **trống** nếu không có quả bóng nào đang xoáy trên miệng lỗ đó; tại thời điểm một quả bóng dừng việc xoáy và bắt đầu rơi vào lỗ thì lỗ đó cũng được coi là trống.
- Trò chơi bắt đầu ở thời điểm 0, Alice đứng ở đầu rãnh phía bên trái (gần lỗ thứ nhất) còn Bob đứng ở đầu rãnh phía bên phải (gần lỗ thứ N).
- Tại các thời điểm khác nhau, Alice hoặc Bob ném một trong M quả bóng vào rãnh.
- Trò chơi kết thúc khi một trong các trường hợp sau xảy ra:
 - ++ Alice và Bob ném hết M quả bóng; khi đó hai nhóc **hòa nhau**; thời gian của ván chơi là thời điểm mà quả bóng cuối cùng ngừng xoáy và rơi xuống lỗ.
 - ++ Alice hoặc Bob ném một quả bóng vào rãnh trong lúc không có lỗ nào trên máng còn trống; khi đó nhóc nào ném quả bóng đó sẽ là người **thua cuộc** :d; thời gian của ván chơi là thời điểm mà nhóc thua cuộc ném quả bóng đó.

AnhDQ đi chơi về và thấy hai nhóc đang ngồi thở hổn hển vì ném bóng :)) AnhDQ liền hỏi hai nhóc xem ai thắng thì hai nhóc cười toe toét nói là.. không nhớ @) Bó tay, AnhDQ liền gắng hỏi chúng về diễn biến ván chơi, hai nhóc chạnh chèo nhau kể ra các lần ném bóng của mình, được mô tả bởi hai số u, v, trong đó u là thời điểm ném quả bóng và v là độ xoáy của quả bóng đó.

Nhưng chúng kể cũng chẳng có thứ tự gì hết, càng làm AnhDQ đau đầu hơn, chàng cố ghi lại thông tin mà chúng kể và nhờ các VOJ-er(s) tài giỏi giúp chàng phân định kết quả của ván chơi xem sao. Chàng rất hồi hộp về trò chơi mới này của mình.

Dữ liệu

- Dòng đầu tiên chứa hai số N, M.
- M dòng tiếp theo, mỗi dòng bắt đầu bởi một chữ cái 'A' hoặc 'B', tương ứng với lời kể của Alice hoặc Bob; tiếp theo là hai số u, v tương ứng.

Kết quả

- Nếu ván chơi kết thúc với kết quả hòa, in ra như sau:
 - ++ Dòng đầu tiên in ra **DRAW**.
 - ++ Dòng thứ hai in ra **Game lasts: T minute(s)**; trong đó T là thời gian của ván chơi.
 - ++ M dòng tiếp theo in ra "biên bản" của ván chơi; mỗi dòng ghi **Alice takes the hole: H** hoặc **Bob takes the hole: H** tương ứng với một lượt ném bóng của Alice hoặc Bob; trong đó H là lỗ trống tương ứng mà quả bóng của lượt ném đó rơi vào. Biên bản này viết ra theo **thứ tự thời gian**.
- Nếu ván chơi kết thúc với một nhóc thua cuộc, in ra như sau:
 - ++ Dòng đầu tiên in ra **Alice loses at her turn: R** nếu Alice thua cuộc, hoặc in ra **Bob loses at his turn: R** nếu Bob thua cuộc; với ý nghĩa: nhóc tương ứng thua cuộc sau lần ném thứ R của mình.
 - ++ Dòng thứ hai in ra **Game lasts: T minute(s)**; trong đó T là thời gian của ván chơi.

Ví dụ

Dữ liệu:

2 2

A 1 10

B 2 20

Kết quả:

DRAW

Game lasts: 22 minute(s)

Alice takes the hole: 1

Bob takes the hole: 2

Dữ liệu:

1 2

A 1 10

B 2 20

Kết quả:

Bob loses at his turn: 1

Game lasts: 2 minute(s)

Giới hạn

- $N, M \leq 10^5$.
- $u, v \leq 10^9$.

Thuật toán:

Đầu tiên ta sắp xếp lại thứ tự thời gian của các cuộc ném bóng. Bài này ta dùng 3 heap với các ý nghĩa như sau: 1 heap min, 1 heap max lưu chỉ số các lỗ còn trống. 1 heap quản lí các ô đang có bóng ở xoáy ở trên, heap này là 1 heap min với giá trị khóa là thời gian quả bóng rơi xuống lỗ. Đầu tiên 2 heap min và max đầy(tất cả các ô còn trống). Cho 1 vòng for qua các thứ tự ném, đến 1 lượt ta phải làm 2 việc sau:

1: Loại bỏ từ heap quản lí các ô có bóng những ô nào có thời gian bóng rơi xuống lỗ \leq thời gian ném bóng của lượt này, sau đó thêm vào heap min và heap max những ô tìm được, những ô này lần lượt được lấy ra từ gốc của heap đang xét.

2: Thực hiện ném: Nếu ném từ bên trái, dựa vào heap min để tìm ra ô trống có chỉ số nhỏ nhất, ném bên phải thì ngược lại. Khi đã tìm được thì ta tiến hành loại bỏ khỏi 2 heap này phần tử đã tìm được(do nó bắt đầu có bóng xoáy phía trên) và thêm vào heap kia 1 phần tử gồm 2 thông tin, chỉ số của lỗ và thời gian bóng rơi xuống lỗ.

Nếu đến 1 lượt ném mà tất cả các ô đều có bóng thì trò chơi dừng lại. Có 1 lưu ý là ta cần quản lí số hiệu của các ô trong 2 heap để tiện cho việc loại bỏ.

INTERVAL TREE

Bài viết này muốn giới thiệu với các bạn 1 công cụ rất hữu dụng được sử dụng nhiều trong các bài toán trên dãy số, hoặc được quy về các bài toán xử lý trên dãy số, đặc biệt là các bài toán có nhiều công việc cần xử lý và nhiều truy vấn xen kẽ nhau. Công cụ được nói đến ở đây chính là INTERVAL TREE(hay còn gọi là segmet tree), tức là cây đoạn.

Interval tree là cây đoạn, vậy thì nó là 1 cây, và các nút của nó thì lưu thông tin của 1 đoạn xác định. Interval tree là 1 cây nhị phân mà mỗi nút không phải là lá đều có đúng 2 nút con. Nếu nút A lưu thông tin của đoạn từ $i..j$ thì 2 nút con của nó, A1 và A2, lần lượt lưu thông tin của các đoạn $i..m$ và $m+1..j$, với $m=(i+j) \text{ div } 2$ là phần tử giữa của đoạn.

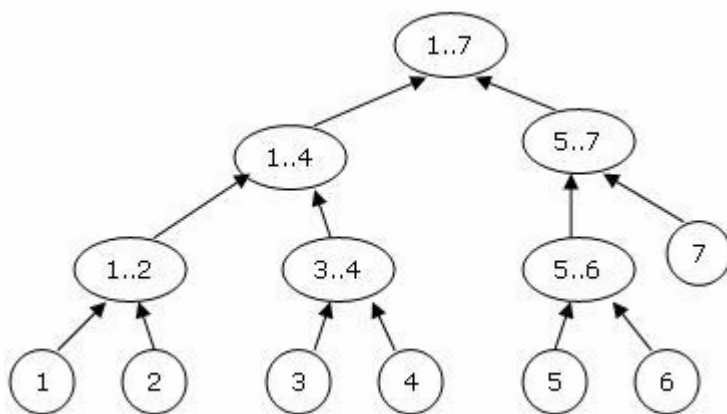
Vì đây là cây nhị phân, lại có đầy đủ 2 nút con, để đơn giản thì ta có thể biểu diễn cây chỉ = 1 mảng 1 chiều, với ý nghĩa như sau:

Nút 1 là nút gốc, nút k (nếu không phải là nút lá) thì có 2 con là các nút $k*2$ (nút con trái) và $k*2+1$ (nút con phải).

Nút 1 sẽ lưu thông tin của đoạn $1..n$ (với n là độ dài dãy số).

Vậy nút 2 sẽ lưu thông tin đoạn $1..(n+1) \text{ div } 2$ và nút 3 sẽ lưu thông tin đoạn $(n+1) \text{ div } 2+1..n$.

Tương tự theo định nghĩa thì ta có thể biết được nút thứ k sẽ lưu thông tin của 1 đoạn xác định nào đấy. Sau đây là hình ảnh 1 cây INTERVAL TREE với $n=7$:



Có 1 vấn đề là với 1 đoạn n phần tử, thì mảng biểu diễn cây sẽ có bao nhiêu phần tử là đủ, người ta đã chứng minh được cây interval tree chỉ cần tối đa là $4*n-5$ phần tử, vì vậy khi khai báo 1 cây interval tree, ta thường khai báo là 1 mảng $1..4*n$.

Trên interval tree có 2 thao tác cần xử lý, 1 là cập nhật thay đổi cho đoạn $i..j$ và 2 là lấy thông tin cần có của đoạn $i..j$, nhưng mỗi nút của interval tree chỉ lưu những đoạn xác định, vì vậy 2 thao tác này có thể cần tác động đến nhiều nút.

Để dễ hình dung, ta lấy 1 ví dụ cụ thể:

Cho 1 dãy n phần tử ($n \leq 10^5$), ban đầu mỗi phần tử có giá trị 0. Có $q \leq 10^5$ truy vấn, mỗi truy vấn là 1 trong 2 thao tác sau: 1 là gán giá trị $v (v > 0)$ cho phần tử ở vị trí i , 2 là tìm giá trị lớn nhất trong đoạn $i..j$ bất kì.

Cách đơn giản nhất là ta có 1 mảng $A[1..100000]$, với thao tác 1 thì gán $A[i]=v$, thao tác 2 thì cho 1 vòng for từ i đến j để tìm max, nhưng cách này trong trường hợp có nhiều truy vấn thứ 2 thì sẽ chạy rất lâu, trường hợp xấu nhất là $n*q$. Dễ thấy là cách này không thể chạy trong thời gian cho phép.

Cách dùng interval tree như sau:

Với truy vấn 1, ta sẽ cập nhật là kết quả max cho tất cả các đoạn mà chứa phần tử thứ i , những đoạn còn lại thì không làm gì cả vì không ảnh hưởng đến.

Với truy vấn 2, ta cần tìm kết quả max trong số tất cả những đoạn nằm gọn trong $i..j$, tức là những đoạn có điểm đầu $\geq i$ và điểm cuối $\leq j$.

Gọi $T[1..400000]$ of longint là mảng để lưu trữ cây interval tree. $T[i]$ là giá trị lớn nhất trong đoạn mà nút k lưu giữ thông tin.

Khi gặp truy vấn 1, ta làm như trong thủ tục sau:

procedure truyvan1(k,l,r,i:longint);

var m:longint;

begin

1.if (l<i)or(r>i) then exit;

2 if (l=r) then

begin

T[k]:=v;exit;

end;

3.m:=(l+r) div 2;

4.truyvan1(k*2,l,m,i);

5.truyvan1(k*2+1,m+1,r,i);

6.T[k]:=max(T[k*2],T[k*2+1]);

end;

Ta sẽ phân tích thủ tục truyvan1:

-Thủ tục này có các tham số k, l, r, i với ý nghĩa: l và r là điểm đầu và điểm cuối của đoạn mà nút k lưu trữ thông tin, i chính là phần tử cần gán giá trị v . -Trong thủ tục có 1 biến m để xác định phần tử nằm giữa đoạn $l..r$.

-Câu lệnh 1 có ý nghĩa là ta sẽ bỏ qua không làm gì với các đoạn không chứa phần tử thứ i ($l > i$ hoặc $r < i$)

-Câu lệnh 2 tức là, khi đã đến nút thứ k biểu diễn đoạn $i..i$ thì ta chỉ cần gán $T[k]=v$, vì tất nhiên sau khi gán, v sẽ là giá trị lớn nhất của đoạn $i..i$.

-Câu lệnh 3 xác định phần tử nằm giữa $l..r$.

-Câu lệnh 4 là 1 lời gọi đệ quy, để ý các tham số thì dễ dàng nhận ra, câu lệnh này gọi đến nút con trái của nút k , tức nút $k*2$ để ta cập nhật cho nút đó.

-Câu lệnh 5 tương tự câu lệnh 4 nhưng là gọi để cập nhật cho nút con phải.

-Câu lệnh 6: Sau khi đã cập nhật cho 2 nút con trái và phải thì đã xác định được giá trị lớn nhất từ $l..m$ và $m+1..r$, vậy thì để xác định giá trị lớn nhất của đoạn $l..r$ ta chỉ cần lấy giá trị lớn nhất của 2 đoạn kia $\Rightarrow T[k] := \max(T[k*2], T[k*2+1])$.

Khi gọi thủ tục truy vấn 1, ta sẽ gọi từ nút gốc, tức là gọi `truyvan1(1,1,n,i)`;

Vậy là đã xong yêu cầu thứ nhất, nhưng cái ta cần để xem chương trình có hoạt động tốt hay không lại là kết quả của yêu cầu thứ 2, vì vậy thủ tục `truyvan1` là để giúp thủ tục `truyvan2` sau đây tìm được kết quả với đoạn $i..j$ bất kì.

Procedure `truyvan2(k,l,r,i,j:longint)`;

var `m:longint`;

begin

1.if $(l > j)$ or $(r < i)$ then exit;

2.if $(i \leq l)$ and $(j \geq r)$ then

begin

res:=max(res,T[k]);exit;

end;

3:m:=(l+r) div 2;

4:truyvan2(k*2,l,m,i,j);

5:truyvan2(k*2+1,m+1,r,i,j);

end;

Phân tích: Thủ tục truy vấn 2 này có các tham số với ý nghĩa như thủ tục 1, có thêm tham số j vì cần lấy kết quả trên đoạn $i..j$.

Mỗi đoạn $l..r$ sẽ có 3 khả năng với đoạn $i..j$.

a: $l..r$ không giao $i..j$, trường hợp này bỏ qua không làm gì cả(câu lệnh 1).

b: $l..r$ nằm gọn trong $i..j$, trường hợp này thì ta chỉ cần tối ưu kết quả khi so sánh với $T[k]$ vì: Ta đã xác định được phần tử lớn nhất trong đoạn $l..r$ nhờ thủ tục 1, và do $l..r$ nằm gọn trong $i..j$ nên không cần đi vào 2 nút con của nó.(câu lệnh 2).

c: $l..r$ không nằm gọn trong $i..j$ nhưng có 1 phần giao với $i..j$, trường hợp này thì ta sẽ đi vào 2 nút con của nó để lấy kết quả, đến khi nào gặp phải 2 trường hợp đầu.(câu lệnh 4 và 5).

Suy nghĩ kĩ sẽ thấy thủ tục truy vấn 2 không bỏ sót cũng như tính thừa phần tử nào ngoài đoạn $i..j$.

Khi gọi thủ tục truyvan2 ta cũng gọi từ nút gốc truyvan2(1,1,n,i,j). Sau thủ tục thì in ra res là kết quả cần tìm.

Người ta cũng chứng minh được rằng, độ phức tạp của mỗi thủ tục truyvan1 và truyvan2 không quá $\log(n)$. Vì vậy thuật toán dùng interval tree cho bài này có độ phức tạp không quá $q \cdot \log(n) = 10^5 \cdot \log(10^5)$, có thể chạy trong thời gian cho phép.

Đây là 1 trong những ví dụ cơ bản nhất về interval tree, hi vọng các bạn đã có thể hiểu và cài đặt nó. Trong khi nghiên cứu về các bài tập sau đây, ta sẽ tìm hiểu 1 vài kiểu sử dụng interval tree khác. Đây cũng là 1 trong những cấu trúc dữ liệu thường được sử dụng trong các kì thi Olympic Tin học trên thế giới.

.

Xếp hàng(NKLINEUP)

Hàng ngày khi lấy sữa, N con bò của bác John ($1 \leq N \leq 50000$) luôn xếp hàng theo thứ tự không đổi. Một hôm bác John quyết định tổ chức một trò chơi cho một số con bò. Để đơn giản, bác John sẽ chọn ra một đoạn liên tiếp các con bò để tham dự trò chơi. Tuy nhiên để trò chơi diễn ra vui vẻ, các con bò phải không quá chênh lệch về chiều cao.

Bác John đã chuẩn bị một danh sách gồm Q ($1 \leq Q \leq 200000$) đoạn các con bò và chiều cao của chúng (trong phạm vi $[1, 1000000]$). Với mỗi đoạn, bác John muốn xác định chênh lệch chiều cao giữa con bò thấp nhất và cao nhất. Bạn hãy giúp bác John thực hiện công việc này!

Dữ liệu

- Dòng đầu tiên chứa 2 số nguyên N và Q .
- Dòng thứ i trong số N dòng sau chứa 1 số nguyên duy nhất, là độ cao của con bò thứ i .

- Dòng thứ i trong số Q trong tiếp theo chứa 2 số nguyên A, B ($1 \leq A \leq B \leq N$), cho biết đoạn các Con bò từ A đến B .

Kết quả

Gồm Q dòng, mỗi dòng chứa 1 số nguyên, là chênh lệch chiều cao giữa con bò thấp nhất và cao nhất thuộc đoạn tương ứng.

Ví dụ

Dữ liệu:

6 3

1

7

3

4

2

5

1 5

4 6

2 2

Kết quả

6

3

0

Thuật toán:

Đây là 1 bài toán xử lí trên dãy số và cần truy cập đến những đoạn $A..B$ bất kì trong dãy, vì vậy interval tree là 1 trong những lựa chọn không tồi. Bài này chúng ta có 1 điều may mắn là không cần phải cập nhật lại chiều cao của các con bò, vì vậy thông tin trong cây interval tree là cố định và ta sẽ tạo cây interval tree dựa trên mảng chiều cao của các con bò. Mỗi đoạn thì ta cần in ra chênh lệch độ cao con bò cao nhất và con bò thấp nhất, vì vậy chúng ta cần tìm được giá trị lớn nhất và giá trị nhỏ nhất trong các phần tử từ A đến B . Ta có thể dùng 1 cây interval tree với mỗi nút lưu 2 thông tin, giá trị lớn nhất và giá trị nhỏ nhất trong đoạn mà nó biểu diễn, cũng có thể dùng 2 cây

interval tree, 1 cây dùng để lưu giá trị lớn nhất, cây còn lại là giá trị nhỏ nhất. Ở đây ta gọi 2 cây này là maxd và mind. Phần tạo ta có thể làm rất đơn giản dựa trên ý tưởng sau:

Nếu $l=r$ thì $Maxd[k]=Mind[k]=A[l]$;

Nếu không thì $Maxd[k]=\max(Maxd[k*2],Maxd[k*2+1])$ và $Mind[k]=\min(Mind[k*2],Mind[k*2+1])$;

Khi muốn tìm kết quả thì dựa vào mảng Maxd để tìm GTLN trên đoạn A..B, dùng mảng Mind để tìm GTNN trên đoạn A..B, việc này làm tương tự như trong ví dụ của bài viết giới thiệu về Interval tree phía trên. Chú ý là khi tìm max hay tìm min ta đều phải đi đến những nút giống nhau (do đi đến những nút nào thì chỉ phụ thuộc A và B) nên mỗi lần tìm chỉ cần gọi chung 1 thủ tục.

Giá trị lớn nhất(QMAX)

Cho một dãy gồm n phần tử có giá trị ban đầu bằng 0.

Cho m phép biến đổi, mỗi phép có dạng (u, v, k): tăng mỗi phần tử từ vị trí u đến vị trí v lên k đơn vị.

Cho q câu hỏi, mỗi câu có dạng (u, v): cho biết phần tử có giá trị lớn nhất thuộc đoạn [u, v]

Giới hạn

- n, m, q ≤ 50000
- $k > 0$
- Giá trị của một phần tử luôn không vượt quá $2^{31}-1$

Input

- Dòng 1: n, m
- m dòng tiếp theo, mỗi dòng chứa u, v, k cho biết một phép biến đổi
- Dòng thứ m+2: p
- p dòng tiếp theo, mỗi dòng chứa u, v cho biết một phép biến đổi

Output

- Gồm p dòng chứa kết quả tương ứng cho từng câu hỏi.

Example

Input:

6 2

1 3 2

4 6 3

1

3 4

Output:

3

Thuật toán:

Bài này thì ta có m phép biến đổi tăng dãy trước rồi mới yêu cầu tìm giá trị lớn nhất trong các đoạn chứ không phải xen kẽ nhau, vì vậy ta sẽ tìm cách xây dựng dãy số sau m phép biến đổi. Khi có được mảng giá trị sau m phép biến đổi, công việc còn lại của ta là tạo 1 cây interval tree với mỗi nút lưu giá trị lớn nhất của đoạn mà nó quản lí trong khi các phần tử của mảng đã xác định, với mỗi truy vấn tìm GTLN thì ta có thể làm không mấy khó khăn.

Vấn đề bây giờ là xây dựng dãy số sau m phép biến đổi.

Ta có thể sử dụng 1 kĩ thuật đơn giản nhưng rất hiệu quả như sau.

Giả sử mảng ta cần có là mảng $A[0..n+1]$, lúc đầu $A[i]=0$ với mọi i .

Mỗi yêu cầu u,v,k tức là tăng các phần tử từ vị trí u đến vị trí v lên k đơn vị, ta làm như sau:
 $A[u]:=A[u]+k; A[v+1]:=A[v+1]-k;$

Sau khi đọc xong m phép biến đổi và làm như trên, cuối cùng là tính mảng A :

For $i:=1$ to n do

$A[i]:=A[i]+A[i-1];$

Các bạn có thể tự chứng minh tính đúng đắn hay có thể viết đoạn chương trình này ra và kiểm nghiệm lại. Như vậy ta đã có thể giải quyết trọn vẹn bài toán.

Giá trị lớn nhất ver2(QMAX2)

Giống bài "Giá trị lớn nhất" ở trên.

Input

- n: số phần tử của dãy ($n \leq 50000$).
- m: số lượng biến đổi và câu hỏi ($m \leq 100000$).
- +) biến đổi có dạng: 0 x y value
- +) câu hỏi có dạng : 1 x y.

Output

Ghi ra trả lời cho lần lượt từng câu hỏi.

Example

Input:

6 3

0 1 3 3

0 4 6 4

1 1 6

Output:

4

Thuật toán:

Bài này khác bài QMAX ở trên là các phép biến đổi và các câu hỏi xen kẽ nhau, vì vậy ta không thể tạo trước 1 cây interval được. Để giải quyết bài toán này, ta cần phải thực hiện cập nhật cũng như lấy kết quả xen kẽ nhau. Nhưng ở đây thì các công việc này không hề đơn giản.

Vừa có được kết quả, ta cũng cần phải đảm bảo 2 công việc này không được có độ phức tạp vượt quá $\log(n)$.

Để làm bài này, ta gọi $F[k]$ là giá trị lớn nhất trong đoạn mà nút k quản lí, trong lúc cập nhật, muốn đảm bảo thủ tục này không vượt quá $\log(n)$ thì khi đi đến 1 nút mà nằm gọn trong đoạn x..y thì ta không được đi vào các nút con của nó nữa(nếu không sẽ đi đến tất cả các đoạn có trong đoạn x..y và độ phức tạp tỷ lệ với n). Nếu như vậy, ta cần có 1 mảng phụ T để lưu lại giá trị cần tăng lên cho tất cả các phần tử của đoạn này, khi ta truy vấn đến 1 nút k đồng thời ta tăng $T[k]$ lên cho $T[k*2]$, $T[k*2+1]$ và $F[k]$. (do $T[k]$ là giá trị cần tăng lên cho tất cả những phần tử của đoạn nút k quản lí, nên các nút con của nó cũng phải tăng lên $T[k]$). Sau khi đã cập nhật cho mảng F và các nút con, ta gán lại $T[k]=0$ để tránh trường hợp cộng lại nhiều lần. Nếu đã đến đoạn nằm gọn trong đoạn x..y thì ta tăng mỗi biến $F[k]$, $T[k*2]$, $T[k*2+1]$ lên v. Khi muốn lấy kết quả, ta vẫn làm như bài QMAX, nhưng đến mỗi nút con, ta vẫn cần thực hiện tăng giá trị $T[k]$ cho $T[k*2]$, $T[k*2+1]$ và $F[k]$ để lấy được kết quả. Tức là khi đến được 1 nút nào đó(trong bất kì thao tác nào, cập nhật hay

lấy kết quả) thì đều thực hiện như vậy. Điều này đảm bảo là ta có thể lấy được giá trị chính xác, đây là kiểu cây IT có thông tin truyền từ nút cha xuống nút con. Các bạn có thể tham khảo chương trình để thấy rõ hơn.

Bật đèn(LITES)

Bác John giữ cho đàn bò thông minh bằng cách để chúng chơi các đồ chơi phát triển trí tuệ. Một trong các trò chơi là các ngọn đèn trong chuồng. Mỗi trong số N ($2 \leq N \leq 100,000$) con bò được đánh số từ $1..N$ có treo một ngọn đèn màu.

Vào đầu buổi tối, tất cả đèn đều tắt. Đàn bò điều khiển các ngọn đèn bằng N công tắc; bấm công tắc i đổi trạng thái của đèn i từ tắt sang bật hoặc ngược lại.

Đàn bò đọc và thực thi một danh sách gồm M ($1 \leq M \leq 100,000$) thao tác mô tả bởi một trong hai số nguyên ($0 \leq \text{thao tác} \leq 1$).

Thao tác thứ nhất (mô tả bởi số 0) theo sau bởi hai số nguyên S_i và E_i ($1 \leq S_i \leq E_i \leq N$) cho biết công tắc đầu và công tắc cuối. Đàn bò sẽ bấm mỗi công tắc từ S_i đến E_i đúng một lần.

Thao tác thứ hai (mô tả bởi số 1) yêu cầu đàn bò đến xem có bao nhiêu ngọn đèn giữa S_i và E_i ($1 \leq S_i \leq E_i \leq N$) đang bật. Hãy giúp bác John đảm bảo rằng đàn bò trả lời đúng bằng cách xử lý danh sách và trả về các kết quả đúng.

Dữ liệu

* Dòng 1: Hai số nguyên cách nhau bởi khoảng trắng: N và M

* Dòng 2.. $M+1$: Mỗi dòng chứa một thao tác với ba số nguyên cách nhau bởi khoảng trắng: thao tác, S_i , và E_i

Kết quả

* Dòng 1.. số truy vấn : Với mỗi truy vấn, in ra kết quả là một số nguyên trên một dòng.

Ví dụ

Dữ liệu:

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

Kết quả:

```
1
2
```

Thuật toán:

Ta có thể sử dụng INTERVAL TREE để làm như sau:

Bài này có tư tưởng khá giống với bài QMAX2, thông tin được truyền từ nút cha xuống nút con bất cứ khi nào có thể. Ta có 1 mảng B kiểu logic, $B[i]=true$ nghĩa là những đèn trong đoạn mà nút thứ i quản lý cần thay đổi trạng thái. Khi đến 1 nút, nếu $b[i]=true$ thì cần thay đổi trạng thái của $B[i*2]$ và $B[i*2+1]$ đồng thời gán lại $B[i]=false$ (đã thực hiện thay đổi rồi thì phải gán lại false), nếu gọi mảng $sl[i]$ là số lượng đèn đang bật trong đoạn do nút i quản lý thì nếu gặp $b[i]=true$ phải tiến hành sửa $sl[i]$, $sl[i]$ hiện tại là giá trị của số đèn đang tắt (do khi thay đổi trạng thái thì bật chuyển thành tắt). Vậy chỉ cần 1 phép trừ là có thể tính được số lượng đèn đang bật trong đoạn. Các công việc này cần làm ở đầu ở cả 2 thủ tục cập nhật và lấy kết quả.

Khi thực hiện thao tác cập nhật kết quả cho đoạn x..y thì nếu gặp những đoạn nằm gọn trong x..y thì ta lại thực hiện thay đổi trạng thái như trên. Phần lấy kết quả thì không khó, các bạn chỉ cần nhớ là trong thủ tục lấy kết quả cũng cần thực hiện truyền thông tin từ nút cha cho nút con (bất cứ khi nào có thể).

A Marble Game(MARBLE)

Trong những ngày hè rảnh rỗi, ktuan thường chơi bắn bi trên một bảng hình vuông gồm $N \times N$ ô vuông nhỏ. Trò chơi được thực hiện như sau:

- Ban đầu, ktuan đặt K vật cản vào K ô vuông của bảng.
- Sau đó, ktuan thực hiện lần lượt Q lượt chơi. Ở lượt chơi thứ i, ktuan lần lượt bắn D_i viên bi từ ngoài bảng vào một trong 4 đường biên của bảng. Kích thước của mỗi viên bi đúng bằng kích thước của một ô vuông nhỏ. Viên bi sẽ đi qua các ô thuộc cùng một hàng / cột cho đến khi đi ra ngoài bảng hoặc gặp một vật cản hay viên bi khác. Nếu có vật cản hay viên bi khác ở ngay ô đầu tiên thì viên bi đó không được đưa vào bảng.
- Ở mỗi lượt bắn, ktuan ghi lại tổng số ô mà các viên bi đã đi qua.

Bạn hãy viết chương trình mô phỏng lại trò chơi và với mỗi lượt bắn, in ra tổng số ô mà các viên bi của lượt bắn đó đã đi qua.

Dữ liệu

- Dòng đầu ghi 3 số N, K, Q.
- K dòng sau, mỗi dòng ghi một cặp số (u,v) thể hiện toạ độ (dòng, cột) của một vật cản.

- Q dòng sau, mỗi dòng ghi 4 giá trị c, D, u, v. Ký tự c có thể là 'L', 'R', 'T', hoặc 'B' cho biết viên bi được đưa vào từ biên trái, phải, trên hoặc dưới của bảng. (u,v) thể hiện toạ độ ô đầu tiên mà viên bi được đưa vào. Đây phải là một ô nằm trên biên của bảng ứng với ký tự c. D là số lượng viên bi sẽ bắn ở lượt chơi này.

Kết quả

- Với mỗi lượt chơi, in ra tổng số ô mà các viên bi của lượt đó đã đi qua

Ví dụ

Dữ liệu

5 1 3

3 3

L 2 3 1

T 1 1 1

B 5 5 5

Kết quả

3

2

25

Giải thích

- Viên bi đầu tiên của lượt 1 sẽ đi qua 2 ô (3,1) và (3,2) trước khi gặp vật cản ở ô (3,3)
- Viên bi tiếp theo của lượt 1 sẽ đi qua ô (3,1) trước khi gặp viên bi ở ô (3,2). Vậy tổng số ô bị đi qua ở lượt này là 3.
- Viên bi đầu tiên của lượt 2 sẽ đi qua 2 ô (1,1) và (2,1) trước khi gặp viên bi ở ô (3,1).
- Mỗi viên bi của lượt cuối cùng đều không gặp vật cản và sẽ đi ra ngoài bảng sau khi đi qua 5 ô.

Giới hạn

- $N \leq 50000$, $K \leq 10$, $Q \leq 100000$
- Trong 1/3 số test, N và Q không vượt quá 1000.
- Thời gian: 3-10s/test.

Thuật toán:

-Với mỗi lần bắn, ta cần phải tìm được ô chướng ngại vật đầu tiên. Bài này cũng có tư tưởng tương tự bài QMAX2, thông tin truyền từ nút cha xuống nút con, nhưng 1 điều đặc biệt là chúng ta chỉ cần thông tin của những đoạn 1 phần từ (do bắn chỉ trên 1 đường thẳng). Vì vậy thông tin chỉ được truyền xuống mà không truyền lên. Thao tác tìm chướng ngại vật đầu tiên trên đường bắn thì không có gì khó khăn.(cứ đi mãi cho đến nút quản lí hàng/cột hiện tại). Phần còn lại là phải cập nhật sau

khi bắn, thao tác này làm như QMAX2, phải truyền thông tin cho các nút con bất cứ khi nào có thể. Vậy ta cần 4 cây INTERVAL TREE, H1,H2,C1,C2 với ý nghĩa:

H1: Quản lí theo hàng, giả sử nút k quản lí từ hàng i đến hàng j thì H1[k] là chỉ số cột nhỏ nhất trong các chỉ số cột có chướng ngại vật từ hàng i đến hàng j. H1 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn L.

H2: Tương tự H1 nhưng thay bằng chỉ số lớn nhất. H2 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn R.

C1: Tương tự H1 nhưng C1 quản lí các cột và C1[k] là chỉ số hàng nhỏ nhất có chướng ngại vật từ cột i đến cột j. C1 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn T.

C2: Tương tự C1 nhưng là chỉ số hàng lớn nhất. C2 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn B.

Các bạn cũng có thể thay 4 cây bằng 2 cây(H và C) nhưng mỗi cây có 2 thông tin là chỉ số nhỏ nhất và lớn nhất của cột/hàng.

Mỗi khi có thao tác bắn L và R, sau khi xác định được hàng i sẽ thêm những viên bi từ vị trí nào đến vị trí nào thì thực hiện cập nhật trên 2 mảng H1, H2. Tương tự với 2 thao tác còn lại, lúc này ta cập nhật trên 2 mảng C1, C2. Cuối cùng xin lưu ý các bạn những trường hợp mà không có chướng ngại vật trên hàng hoặc cột, khi đó ta không cần cập nhật, và kết quả chính là d*n. Các bạn cũng cần chú ý những trường hợp mà số ô đưa vào nhiều hơn số ô tối đa có thể đưa vào hàng/cột, lúc này ta chỉ kết quả cho những ô đưa được vào bảng thôi(1 số ô bị bắn ra ngay khi gặp chướng ngại vật ở biên). Bài này các bạn nên khai báo kết quả kiểu int64 để tránh bị tràn số.

BINARY INDEX TREE(BIT)

Có 1 loại cấu trúc dữ liệu thường được sử dụng để thay thế cho INTERVAL TREE trong 1 vài trường hợp, đó chính là BINARY INDEX TREE(cây chỉ số nhị phân), thường viết tắt là cây BIT.

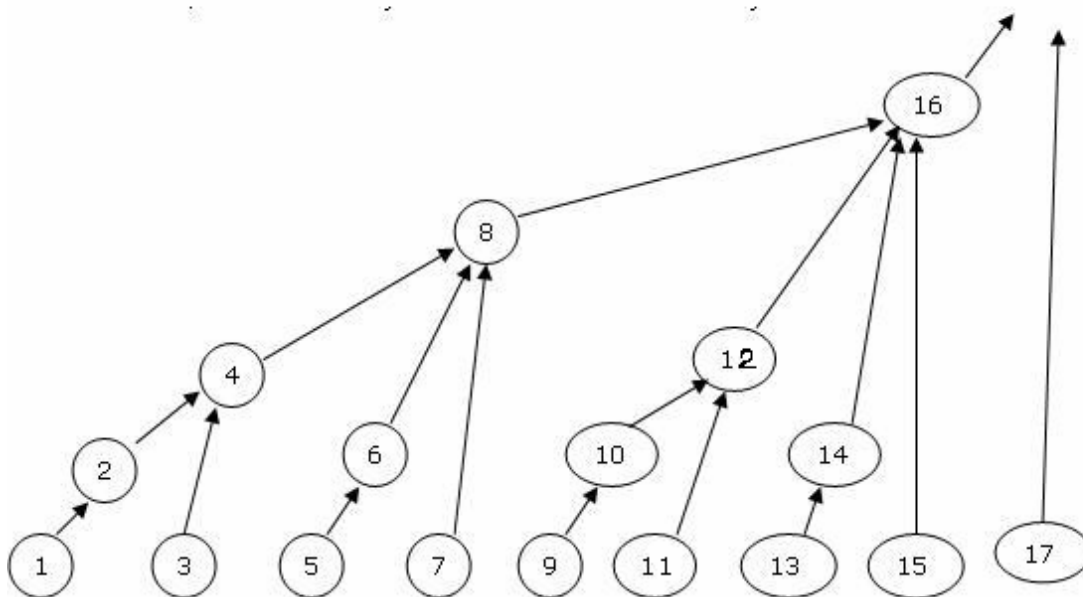
Tên của nó là cây chỉ số nhị phân, vì vậy mà ta liên tưởng ngay đến sự biểu diễn các số trong máy tính bằng hệ nhị phân, gồm các bit 0 và 1.

Cây BIT được định nghĩa 1 cách đệ quy như sau:

-Nếu i lẻ thì $cha[i]=i+1$;

-Nếu i chẵn là thì $cha[i]:=cha[i \div 2]*2$;

Sau đây là hình ảnh của 1 cây BIT.



Nhìn vào định nghĩa này ta có thể thấy, $cha[i]$ là cố định, phụ thuộc trực tiếp vào giá trị của i . Mỗi cây BIT chỉ cần 1 mảng 1 chiều $O(N)$ để lưu trữ, ít hơn INTERVAL TREE khoảng 4 lần.

Ta cần thực hiện 2 thao tác cơ bản sau từ nút i .

1: Tính $cha[i]$. Định nghĩa đệ quy là như vậy, nhưng để dễ hình dung, các bạn có thể tưởng tượng, $cha[i]$ sẽ được tính bằng cách thay số 0 ở trước nhóm những số 1 kề nhau phải nhất bằng số 1, đồng thời thay nhóm những số 1 kề nhau này thành nhóm các số 0.

VD: $cha[5]=6$ vì $5=101$, thay số 0 trước số 1 phải nhất $=1$, đồng thời số 1 phải nhất trở thành số 0. Vậy ta có $110=6$.

Cha[38]=40 vì $38=100110$, sau khi thực hiện thay như trên, ta có dãy bit lúc này là $101000=40$.

Tương tự như thế ta có cha[6]=8($110 \rightarrow 1000$). Vấn đề là làm sao để thực hiện thao tác này hiệu quả nhất, nếu làm như thông thường thì ta phải mất $\log(i)$ trong mỗi thao tác tính cha[i], nhưng ta có thể sử dụng 1 số công thức toán học đã được chứng minh. 1 công thức rất đơn giản, hiệu quả và dễ nhớ thường được sử dụng là: $\text{cha}[i]=i+i \text{ and } (-i)$;

2: Truy xuất đến nút j là nút lớn nhất nhỏ hơn i nhưng không phải là nút con cháu của i.

Để truy xuất đến j, ta thực hiện thay bit 1 phải nhất của i bằng bit 0. VD như nếu $i=6=110$ thì $j=100=4$. Điều này ta cũng có thể làm chỉ bằng 1 phép tính. Đây cũng là 1 công thức rất đơn giản và hiệu quả: $J=i-i \text{ and } (-i)$.

Ta thấy mỗi truy xuất ta chỉ sử dụng các phép toán bit hoặc là phép +, -. Chính vì thế mà cây BIT hoạt động rất nhanh.

Bây giờ ta sẽ tìm hiểu vai trò của 2 thao tác trên. Trong cây BIT thì thông tin được lưu trữ ở 1 nút là của nó và tất cả các nút con. Vì vậy thao tác đầu tiên, truy xuất đến cha[i] là để cập nhật thông tin.

Như thế thì thao tác thứ 2 chắc chắn là dùng để lấy thông tin. Nhưng thông tin được lấy trong cây BIT là thông tin từ 1 đến i, chứ không phải lấy trên đoạn i..j bất kì. Mỗi lần truy xuất đến nút j, ta lại tổng hợp những thông tin cần thiết. Cứ thế cho đến khi nút j ta tính được =0.

Để các bạn dễ hình dung, ta sẽ xét bài toán sau đây:

Cho 1 dãy n phần tử, thực hiện 2 thao tác sau.

1: Gán giá trị v cho phần tử ở vị trí i. ($1 \leq i \leq n$).

2: Tính tổng từ a[i] đến a[j] bất kì ($1 \leq i \leq j \leq n$).

Giới hạn:

$N \leq 100000$;

Số lượng thao tác: $M \leq 100000$.

Lúc đầu coi như các phần tử có giá trị=0.

Hướng dẫn:

Nếu làm theo cách thông thường, dễ dàng nhận ra độ phức tạp thuật toán trong trường hợp xấu nhất là $O(n*m)$, không thể thực hiện trong thời gian cho phép.

Bây giờ ta sẽ tìm hiểu cách sử dụng cây BIT vào cho bài này.

Gọi T là mảng chứa cây BIT, lưu thông tin, A là mảng giá trị các phần tử.

- Với thao tác 1. Giả sử giá trị hiện tại của $a[i]$ là u thì ta coi như thao tác này tăng $a[i]$ lên $v-u$. Khi cập nhật ta phải cập nhật cho nút i và tất cả các tổ tiên của i , cho đến khi vượt quá n . Thủ tục update như sau:

Procedure update(i,v:longint);

Begin

While i<=n do

Begin

T[i]:=T[i]+v-a[i];

I:=i+i and(-i);

End;

End;

- Với thao tác thứ 2, do cây BIT chỉ có thể lấy thông tin của đoạn $1..i$, nên để tính tổng của $a[i]$ đến $a[j]$, ta sẽ tính tổng từ 1 đến j , sau đó trừ đi tổng từ 1 đến $i-1$. Sau đây là thủ tục để tính tổng từ 1 đến vị trí i bất kì.

Procedure getsum(i:longint);

Begin

Sum:=0;

While i>0 do

Begin

Sum:=sum+T[i];

I:=i-i and(-i);

End;

End;

Vậy là ta đã có trong tay 2 thủ tục chính, phần còn lại các bạn có thể dễ dàng cài đặt. Độ phức tạp thuật toán là $O(m \cdot \log(n))$. Qua đây ta thấy cây BIT rất dễ cài đặt, các thủ tục nhìn rất sáng sủa, những công thức tính cũng dễ nhớ, vì vậy mà nó rất hay được dùng trong những trường hợp có thể thay thế cho INTERVAL TREE. Ta cũng dễ dàng nhận thấy, những bài nào làm được bằng cây BIT thì cũng làm được = INTERVAL TREE, nhưng điều ngược lại thì không đúng. Vì vậy phải cần nhắc thật kĩ trước khi các bạn định lựa chọn 1 trong 2 kiểu cấu trúc dữ liệu này.

Coder Rating(CRATE)

Cho danh sách N lập trình viên ($1 \leq N \leq 300000$), đánh số lần lượt từ 1 đến N . Mỗi người đều tham gia cả hai giải thi đấu: Giải THPT và giải Mở rộng. Với mỗi lập trình viên, bạn sẽ được cung cấp điểm số của giải Mở rộng A_i và điểm số của giải THPT H_i (Các điểm số đều là số nguyên không âm và không vượt quá 100000). Lập trình viên i được coi là giỏi hơn lập trình viên j khi và chỉ khi cả 2 điểm số của lập trình viên i đều lớn hơn hoặc bằng điểm số tương ứng của lập trình viên j , trong đó có ít nhất 1 điểm số phải lớn hơn. Hãy tính xem với mỗi lập trình viên i thì có bao nhiêu lập trình viên mà i giỏi hơn.

Input

Dòng đầu tiên chứa số nguyên N .

N dòng tiếp theo, dòng thứ $i+1$ chứa 2 số nguyên A_i và H_i .

Output

Dòng i chứa số lượng lập trình viên mà lập trình viên i giỏi hơn.

Example

Input:

```
8
1798 1832
862 700
1075 1089
1568 1557
2575 1984
1033 950
1656 1649
1014 1473
```

Output:

```
6
0
2
4
7
1
5
1
```

Thuật toán:

Bài này chúng ta có thể sử dụng cây BIT giải quyết như sau:

-Sắp xếp cả mảng A và mảng H theo chiều tăng dần của mảng H . (tức là khi so sánh thì chỉ so sánh trên mảng A nhưng khi đổi chỗ thì phải đổi chỗ cả 2 mảng để đảm bảo $A[i]$ và $H[i]$ luôn là điểm của cùng 1 người). Nếu $H[i] = H[j]$ thì sắp tăng theo $A[i]$ và $A[j]$.

-Những lập trình viên nào có cùng số điểm của cả 2 môn thì ta chỉ lấy 1 lần, tức là chỉ lấy đại diện để xét, những người cùng số điểm khác thì có kết quả bằng kết quả của đại diện(do yêu cầu lập trình viên i giỏi hơn lập trình viên j là phải có ít nhất 1 môn có điểm cao hơn). Khi lấy đại diện, ta phải có thêm 1 mảng D để đếm số lập trình viên có số điểm = số điểm đại diện này. Sau khi thực hiện xong, bài toán của chúng ta bây giờ chỉ là đếm số lập trình viên j từ vị trí 1 đến i-1 có $A[j] \leq A[i]$. (Do $B[j] \leq B[i]$ và $A[j] \leq A[i]$, hơn nữa không thể xảy ra đồng thời 2 dấu =, vì vậy mà lập trình viên i sẽ giỏi hơn lập trình viên j).

Các bạn dùng cây BIT cập nhật cũng như lấy kết quả không mấy khó khăn. Sau khi lấy được kết quả hiện thời, các bạn phải đưa về kết quả bài toán yêu cầu, tức là phải đưa ra theo thứ tự từ 1 đến n(do khi sắp chúng ta đã thay đổi thứ tự của các lập trình viên, chương trình cần dùng 1 mảng v với ý nghĩa $v[i]$ là vị trí lúc đầu của lập trình viên thứ i, mảng v cũng được đổi chỗ trong quá trình sắp xếp).

Dãy nghịch thế độ dài K(KINV)

Cho dãy N số nguyên dương $A[1], \dots, A[N]$ là một hoán vị của $1, 2, 3, \dots, N$.
Một dãy nghịch thế độ dài k là 1 dãy $A[j_1] > A[j_2] > A[j_3] \dots > A[j_k]$ với $j_1 < j_2 < j_3 \dots < j_k$. Hãy đếm xem có tất cả bao nhiêu dãy nghịch thế độ dài k.

Input

Dòng 1 : 2 số nguyên dương N và k ($2 \leq N \leq 10000, 2 \leq k \leq 10$).
Dòng 2 : N số nguyên dương $A[1] \dots A[N]$.

Output

Giả sử T là số lượng dãy nghịch thế có độ dài k, hãy ghi ra $T \bmod 10^9$.

Example

Input:

3 2
3 2 1

Output:

3

Thuật toán:

Bài này có thể giải quyết = quy hoạch động kết hợp sử dụng cấu trúc dữ liệu, ở đây ta sử dụng cây BIT. Gọi $F[i, j]$ là số lượng dãy nghịch thế độ dài i và kết thúc ở vị trí j. Vậy $F[i, j]$ sẽ được tính từ các $F[i-1, j_1]$ với j_1 từ 1 đến j-1 thỏa mãn $a[j_1] > a[j]$. Khi tính $F[i, j]$ thì làm tương tự NKINV, nhưng lúc cập nhật thì không phải là cộng thêm 1 mà cộng thêm $F[i, j]$.

Kết quả là tổng của các $F[k, i]$ với i từ 1 đến n.

Dãy nghịch thế(NKINV)

Cho một dãy số $a_1.. a_N$. Một nghịch thế là một cặp số u, v sao cho $u < v$ và $a_u > a_v$. Nhiệm vụ của bạn là đếm số nghịch thế.

Dữ liệu

- Dòng đầu ghi số nguyên dương N .
- N dòng sau mỗi dòng ghi một số a_i ($1 \leq i \leq N$).

Kết quả

Ghi trên một dòng số M duy nhất là số nghịch thế.

Giới hạn

- $1 \leq N \leq 60000$
- $1 \leq a_i \leq 60000$

Ví dụ

Dữ liệu:

3
3
1
2

Kết quả

2

Thuật toán:

Bài này các bạn có thể sử dụng INTERVAL TREE hoặc BINARY INDEX TREE để giải quyết đều được, ở đây chúng ta tìm hiểu cách sử dụng cây BIT.

Đến 1 vị trí i nào đó, ta cần tính được có bao nhiêu vị trí j thuộc đoạn từ $i+1$ đến n mà $a[j] < a[i]$. Cho 1 vòng for downto từ n về 1, đến mỗi vị trí i , đầu tiên ta tính số vị trí bằng cách lấy thông tin với nút ban đầu là nút thứ $a[i]-1$ (do ta tìm tất cả những vị trí có $a[j] < a[i]$), cập nhật giảm dần về 0, đến mỗi nút lại cộng vào kết quả giá trị ở nút đó. Khi cập nhật, ta cập nhật với nút ban đầu là nút $a[i]$, tiến hành cập nhật đến khi nào quá giá trị max trong dãy(ở đây có thể lấy $\text{max}=60000$ cũng được), tới mỗi nút ta cộng thêm 1 vào giá trị ở nút đó.

Bài này các bạn cũng có thể vừa đọc vừa xử lí, với mỗi vị trí i , tính số vị trí j từ 1 đến $i-1$ có $a[j] > a[i]$. Để làm điều này, với mỗi vị trí i , các bạn phải làm ngược với cách đầu tiên, tức là khi lấy kết quả thì nút ban đầu là $a[i]+1$ và ta truy vấn đến cha của nó, đến khi quá giá trị max , còn khi cập nhật thì nút ban đầu là $a[i]$ và cập nhật giảm dần về 0.

Trong cả 2 trường hợp thì bộ nhớ $O(60000)$ và độ phức tạp thuật toán là khoảng $O(n \cdot \log(60000))$

NKBOBILE

Các trạm di động thế hệ 4 ở Tampere hoạt động như sau. Cả khu vực được chia thành các ô vuông. Các ô vuông tạo thành một bảng $S \times S$ với các dòng và cột đánh số từ 0 đến $S-1$. Mỗi ô vuông chứa một trạm di động. Số điện thoại di động hoạt động trong một ô vuông có thể thay đổi vì điện thoại có thể chuyển sang ô vuông khác hoặc thay đổi trạng thái bật/tắt. Khi có thay đổi, mỗi trạm di động sẽ thông báo cho trung tâm. Tại một số thời điểm, trung tâm cần truy vấn tổng số điện thoại đang hoạt động trong một vùng diện tích hình chữ nhật nào đó.

Hãy giúp trung tâm nhận các thông báo từ trạm và trả lời các truy vấn.

Dữ liệu

Gồm nhiều dòng, mỗi dòng chứa các chỉ thị cho chương trình. Có 3 loại chỉ thị 0, 1, 2, 3 nhận các tham số và thực hiện các nhiệm vụ tương ứng như sau:

Chỉ thị	Tham số	Ý nghĩa
0	S	Khởi tạo bảng $S \times S$ chứa toàn bộ số 0. Chỉ thị này chỉ được cho một lần duy nhất ở đầu chương trình.
1	X Y A	Cộng A và số điện thoại hoạt động ở ô vuông (X, Y). A có thể âm hoặc dương.
2	L B R T	Truy vấn tổng số điện thoại hoạt động ở các ô vuông (X, Y) với $L \leq X \leq R, B \leq Y \leq T$
3	.	Kết thúc chương trình. Chỉ thị này chỉ được cho một lần duy nhất ở cuối chương trình.

Giá trị của một ô luôn không âm tại mỗi thời điểm. Các chỉ số bắt đầu từ 0, nghĩa là với một bảng 4×4 , ta có $0 \leq X \leq 3, 0 \leq Y \leq 3$.

Kết quả

Với mỗi chỉ thị loại 2, in ra một dòng gồm một số nguyên dương trả lời cho truy vấn tương ứng.

Giới hạn

- $1 \leq S \leq 1024$
- Giá trị của một ô tại mọi thời điểm luôn thuộc phạm vi $[0, 32767]$.
- $-32768 \leq A \leq 32767$
- Số chỉ thị thuộc phạm vi $[3, 60002]$.
- Tổng số điện thoại trên toàn bộ bảng không vượt quá 2^{30} .

Ví dụ

Dữ liệu

```
0 4
1 1 2 3
2 0 0 2 2
1 1 1 2
1 1 2 -1
2 1 1 2 3
3
```

Kết quả

```
3
4
```

Thuật toán:

Chúng ta sẽ tìm hiểu thuật toán dùng BIT2D(dùng cây BIT 2 chiều) cho bài này.

Nếu cây BIT thông thường là 1 chiều thì BIT2D dùng 1 mảng 2 chiều, khi cập nhật hay lấy kết quả ta thực hiện trên cả 2 chiều. Lúc cập nhật hay lấy kết quả, đầu tiên ta đi đến những nút thuộc chiều thứ nhất, với mỗi nút này, ta lại đi đến những nút thuộc chiều thứ 2. Có thể hiểu đơn giản là BIT2D là 1 cây BIT, mỗi nút của cây BIT này quản lí 1 cây BIT khác.

Trở lại với bài toán, để dễ dàng và thuận tiện hơn, ta sẽ đánh số lại chỉ số dòng và cột bắt đầu từ số 1 chứ không phải số 0.

-Với mỗi thao tác cập nhật ở ô X,Y ta lại cập nhật 2 chiều về phía S, đến mỗi nút lại cộng giá trị ở nút đó thêm A.

-Lấy kết quả: Do đặc điểm của cây BIT là chỉ lấy kết quả bắt đầu từ 1. Vì vậy với mỗi thao tác lấy kết quả L,B,R,T ta lại tính trung gian như sau:

$Kq[L,B,R,T] := Kq[1,1,R,T] - Kq[1,1,R,B-1] - Kq[1,1,L-1,T] + Kq[L-1,B-1]$ (các bạn có thể tự chứng minh tính đúng đắn). Với mỗi thao tác lấy kết quả từ 1,1 đến X,Y ta cũng lấy theo 2 chiều đi về 0, đến mỗi nút lại cộng vào kết quả giá trị ở nút đó. Các bạn có thể tham khảo chương trình mẫu để hiểu rõ hơn.

Như vậy ta thấy, BIT2D không khác nhiều so với BIT thông thường, chỉ khác là các bạn quản lí trên 2 chiều lồng nhau.

Team Selection(NKTEAM)

Các trưởng đoàn đội tuyển tin học vùng Balkan muốn chọn ra những thí sinh mạnh nhất trong khu vực từ N thí sinh ($3 \leq N \leq 100000$). Các trưởng đoàn tổ chức 3 kỳ thi, mỗi thí sinh sẽ tham dự cả 3. Biết rằng không có 2 thí sinh nào có cùng điểm số trong mỗi kỳ thi. Ta nói thí sinh A **giỏi hơn** thí sinh B nếu A được xếp hạng trước B trong cả 3 kỳ thi. Một thí sinh A được gọi là **xuất sắc** nếu như không có thí sinh nào giỏi hơn A.

Yêu cầu: Hãy giúp các trưởng đoàn đếm số thí sinh xuất sắc.

Dữ liệu vào

- Dòng thứ nhất chứa 1 số nguyên dương N .
- 3 dòng sau, mỗi dòng chứa N số nguyên dương cách nhau bởi khoảng trắng, là chỉ số của các thí sinh theo thứ tự xếp hạng từ cao đến thấp của kỳ thi tương ứng.

Kết quả

Gồm 1 số nguyên duy nhất cho biết số thí sinh xuất sắc.

Ví dụ

Dữ liệu mẫu

```
3
2 3 1
3 1 2
1 2 3
```

Kết quả

```
3
```

Không có thí sinh nào giỏi hơn thí sinh khác nên cả 3 thí sinh đều xuất sắc.

Dữ liệu mẫu

```
10
2 5 3 8 10 7 1 6 9 4
1 2 3 4 5 6 7 8 9 10
3 8 7 10 5 4 1 2 6 9
```

Kết quả

```
4
```

Thí sinh 1, 2, 3, 5 là những thí sinh xuất sắc.

Thuật toán:

Nếu tiếp cận trực tiếp bài toán với các thông số của đã cho thì rất khó khăn. Sau đây ta sẽ chuyển dữ liệu về 1 dạng tương đương mà ta có thể giải quyết bài toán dễ dàng hơn:

Gọi A,B,C là 3 mảng 1 chiều với ý nghĩa như sau: $A[i], B[i], C[i]$ là thứ tự của người thứ i trong 3 kì thi tương ứng. Ta có thể xây dựng 3 mảng này rất dễ, chẳng hạn như $A[i]=j$ khi số thứ j của hàng chứa điểm môn thi đầu tiên là i , các mảng B và C xây dựng tương tự. Sau khi xây dựng được 3 mảng này, bài toán được phát biểu lại là tìm số thí sinh mà không có thí sinh nào có giá trị của cả 3 mảng nhỏ hơn nó. Từ đó ta có thuật toán sử dụng Binary Index Tree như sau:

-Gọi mảng T là mảng lưu thông tin của cây BIT.

-Đầu tiên sắp xếp tăng dần 1 mảng bất kì, 2 mảng còn lại cũng phải đổi chỗ theo để đảm bảo $A[i], B[i], C[i]$ luôn là thứ tự của 1 học sinh trong 3 kì thi. Ở đây giả sử ta sắp xếp mảng A. Sau khi sắp xếp mảng A, ta chỉ cần tìm số vị trí i mà không có vị trí j nào từ 1 đến $i-1$ có $B[j]<B[i]$ và $C[j]<C[i]$ (Do $A[j]<A[i]$ nên nếu có vị trí i thỏa mãn điều kiện trên thì đã có 1 thí sinh xếp cao hơn thí sinh ở vị trí i trong cả 3 kì thi.

-Lúc cập nhật, ta cập nhật giá trị min của mảng B, nhưng lại cập nhật trên mảng C. Nghĩa là ta sẽ cập nhật với nút bắt đầu là $C[i]$, cập nhật đến khi quá n , khi đến mỗi 1 nút nào x đó mà giá trị $T[x]>B[i]$ thì gán lại $T[x]:=B[i]$. Vậy lúc đầu ta gán $T[i]=n+1$ với mọi i từ 1 đến n .

-Lúc lấy kết quả, ta sẽ có nút bắt đầu là $C[i]$, bắt đầu đi về 0, nếu đến 1 nút x nào đó mà có $T[x]<B[i]$ thì thí sinh ở vị trí thứ i không là thủ khoa được. Ta sẽ chứng minh i không phải là thủ khoa: Do khi đã tìm được nút x có $T[x]<B[i]$, giả sử $T[x]=B[j]$ nào đó thì ta có $j<i$ (thì mới cập nhật được lên, do ta cho for từ 1 đến n), vì vậy $A[j]<A[i]$. Thêm nữa, $C[j]<C[i]$ do quá trình cập nhật trên cây BIT của ta, nếu $C[j]>C[i]$ thì khi đi từ $C[i]$ đi về phía 0 sẽ không bao giờ có thể đến 1 nút x có $T[x]=B[j]$ (do khi cập nhật thì ta cập nhật lên, tức là đi về phía n). Vậy có 1 thí sinh j có cả 3 chỉ số đều nhỏ hơn của thí sinh i nên thí sinh i không phải là thủ khoa.

-Khi đã có thủ tục kiểm tra xem i có phải thủ khoa hay không thì bài toán cơ bản đã hoàn tất, chỉ có thêm 1 lưu ý nhỏ, khi sắp xếp mảng, có thể dùng thuật toán $O(n)$ để sắp, do các phần tử có giá trị khác nhau và đều không vượt quá n .

VẤN ĐỀ: CÁC LỖI TRONG PASCAL (ST)

1. Lỗi cú pháp là những lỗi phát sinh do lập trình viên viết sai những quy Định về văn phạm của hệ thống hoặc ngôn ngữ. Thí dụ các lỗi sau Đây là những lỗi cú pháp:
 - $(a + b * 2$: thiếu dấu Đóng ngoặc
 - `BEGIM`: Định viết `BEGIN`, sai N
2. Xử lý lỗi. Lỗi cú pháp Được phát hiện trong quá trình dịch. Turbo Pascal 5.0 báo lỗi cú pháp theo nguyên tắc "Mỗi lần chỉ báo một lỗi".
Nếu gặp lỗi ta cần trở về chế Độ soạn thảo, tìm vị trí xuất hiện lỗi, sửa lại lỗi Đó rồi dịch lại chương trình.
Sau khi báo lỗi, Turbo Pascal sẽ chờ ta bấm phím ESC Để trở về chế Độ soạn thảo.
Con trỏ của màn hình soạn thảo sẽ Đặt ở cạnh vị trí xuất hiện lỗi, thông thường quá Đi 1 ký tự.
3. Các thông báo lỗi thường gặp và gợi ý khắc phục.
Lỗi 1. Out of memory: vượt ra ngoài miền nhớ. Chương trình dịch thiếu miền nhớ. Vài gợi ý khắc phục:
 - a. Nếu mục `COMPILE/DESTINATION` (nơi Đặt chương trình Đích tức là chương trình Đã Được dịch ghi trong tệp cùng tên với chương trình nguồn nhưng có phần mở rộng là `.EXE`) Đang Đặt là `MEMORY` hãy Đổi thành `DISK` bằng cách bấm phím `ENTER` (còn gọi là phím `RETURN`).
 - b. Nếu mục `OPTIONS/COMPILER/LINK` Đang Đặt là `MEMORY` hãy Đổi thành `DISK` bằng cách bấm phím `ENTER`.
 - c. Bỏ các chương trình thường trú, chẳng hạn `SIDKICK`, `NC` v.v.
 - d. Thử ra khỏi `TURBO.EXE`, dịch lại chương trình của bạn với lệnh `TCP` như sau:
`TCP <tên chương trình của bạn>`
Chương trình dịch `TCP` này chiếm ít miền nhớ hơn.
Nếu cả 4 biện pháp nói trên Đều không mang lại kết quả tức là chương trình của bạn quá lớn. Hãy chia nó thành các Đơn thể nhỏ hơn.

Các mã lỗi và ý nghĩa:

2. Identifier expected: mong gặp Định danh.
3. Unknown identifier: Định danh chưa Được khai báo. Hãy khai báo Định danh này ở Đầu thủ tục hoặc chương trình.
4. Duplicate identifier: Định danh Được khai báo 2 lần trở lên.
5. Syntax error: Lỗi cú pháp. Gặp một ký tự sai hoặc viết sai một hằng.
6. Error in real constant: Viết sai hằng thực.
7. Error in integer constant: Viết sai hằng nguyên.
Chú ý rằng khai báo `CONST c = 1234` sẽ cho ta một hằng `c` kiểu nguyên. Muốn có một hằng kiểu thực, ta viết `CONST c = 1234.0`. Những hằng có giá trị nằm ngoài khoảng - 2147483648..2147483647 cần Được khai báo theo kiểu thực, thí dụ:
`CONST c = 12345678912.0;`

8. String constant exceeds line: giá trị của chuỗi ký tự quá dài, xem lại có thiếu dấu Đóng/mở (dấu nháy Đơn) hằng văn bản không?
9. Too many nested files: quá nhiều tệp lồng nhau. Chương trình dịch cho phép lồng nhau không quá 5 tệp.
10. Unexpected end of file: Cần gặp dấu kết tệp. Lỗi 10 có thể xuất hiện trong các trường hợp sau:
 - Trong trường trình các cặp BEGIN và END không cân Đối.
 - Tệp khác Được gọi lồng tại một vị trí không hợp lệ.
 - Chú thích chưa Được Đóng bằng dấu } hoặc *).
11. Line too long: Dòng dài quá Bộ soạn thảo cho phép phát sinh các dòng dài tối Đa 249 ký tự trong khi chương trình dịch chỉ làm việc với các dòng dài tối Đa 126 ký tự. Lời khuyên: Không nên viết các dòng dài quá 60 ký tự.
12. Type identifier expected: Cần có Định danh kiểu
13. Too many open files Có thể mở tối Đa 20 tệp. Muốn vậy trong tệp CONFIG. SYS cần có dòng khai báo FILES = 20 Lời khuyên: Nên mở chừng 15 tệp là vừa nếu có nhu cầu. Hãy Đưa dòng sau Đây: FILES = 15 vào tệp CONFIG.SYS. Khởi Động lại máy rồi gọi lại TP.
14. Invalid file name: Sai tên tệp. Tên tệp Được Đặt theo quy Định của DOS.
15. File not found: Không tìm thấy tên trong thư mục. Trong một số trường hợp TP cần có các tệp phụ trợ, thí dụ EGAVGA. BGI cho màn hình trong chế Độ Đồ thị. Bạn cần Đặt những tệp này vào thư mục hiện hành hoặc chỉ rõ Đường dẫn Để truy nhập tới chúng.
16. Disk full: Đĩa Đầy hết chỗ ghi. Hãy xoá Đi một số tệp không cần dùng nữa hoặc dùng Đĩa mềm mới (cho ổ A, B).
17. Invalid compiler directive: Đặt sai chế Độ dịch Lời khuyên: Không nên lạm dụng chế Độ dịch. Các chế Độ ngầm Định là Đủ cho bạn. Hãn hữu hãy Đặt tạm thời một chế Độ dịch, sau Đó hãy trả lại trạng thái cũ cho nó. Thí dụ, mẫu sau Đây thường dùng Để kiểm tra sự tồn tại của một assign (f, fname); { \$ I - } reset (f); { \$ I +) IF IORESULT = 0 THEN { tệp tồn tại }... ELSE {tệp không tồn tại}... Trong Đó f là biến tệp, fname là chuỗi chứa tên tệp.
18. Too many files: Quá nhiều tệp. Chương trình Đòi hỏi quá nhiều tệp. Sự Đòi hỏi thái quá này có thể phát sinh ra các nguyên nhân sau: - Đặt nhiều chế Độ khiến chương trình dịch phải mở nhiều tệp. - Các module gọi móc nối, liên hoàn, mỗi module lại Đòi hỏi một số tệp. Lời khuyên: Ngay từ Đầu hãy chọn cấu trúc tối thiểu cho chương trình. Cần có một danh sách các kiểu, biến và các tệp tổng thể mà chương trình gọi tới.
19. Undefined type in pointer definition: Kiểu chưa Định nghĩa khi Định nghĩa con trỏ. Thí dụ, khai báo var P: ^ptype; Nếu trước Đó ta chưa Định nghĩa kiểu ptype thì sẽ sinh lỗi 19.
20. Variable identifier expected: Cần một Định danh cho biến Thí dụ FOR = 4 TO 20 DO sẽ sinh lỗi 20.
21. Error in type: Lỗi về kiểu. Thí dụ, TYPE * = 3 .. 7
22. Structure too large: Kiểu cấu trúc có kích thước quá lớn. Cỡ tối Đa của cấu trúc trong TP là 65520 byts

23. Set base type of range: Kiểu tập cơ sở vượt quá giới hạn. Giới hạn của kiểu Đoạn dùng làm tập cơ sở là 0 ... 255, giới hạn của kiểu liệt kê dùng làm tập cơ sở là 256 phần tử. Thí dụ khai báo sau Đây sẽ sinh lỗi 23: Var S: set of -8 .. 300;
24. File components may not be files: Thành phần của tệp không thể là tệp. Thí dụ, khai báo TYPE FTEXT = FILE OF TEXT; Lỗi 24 vì TEXT là kiểu tệp văn bản cho nên FTEXT sẽ là kiểu tệp của tệp.
25. Invalid string length: Chiều dài chuỗi không hợp lệ. Chiều dài hợp lệ nằm trong khoảng 0 .. 255. Lời khuyên: Trong TP mỗi chuỗi x dùng phần tử x [0] chứa chiều dài của x. Không nên thay Đổi tùy tiện giá trị của x[0]. Đoạn trình sau Đây dù có lợi cho bạn cũng không bao giờ nên lạm dụng. (*Lấy 3 ký tự Đầu của chuỗi x *) VAR x: string; BEGIN X:= 'abcdef'; x[0]:= chr(3); writeln (x); readln; END.
26. TYPE mismatch: Kiểu không tương thích. các nguyên nhân sinh lỗi có thể là: - Biểu thức Được gán cho biến không Đúng kiểu. Thí dụ
- ```
VAR x: char;
BEGIN
 x:= 127 * 8
END.
```
- Truyền tham trị cụ thể không Đúng kiểu khi gọi thủ tục và hàm. Thí dụ

```
FUNCTION F (x: integer): string;
BEGIN
 Writeln (F ('ERROR')) (* sinh lỗi 26 *)
END.
```

- Sử dụng chỉ cần mạng sai kiểu4

- Dùng toán hạng không Đúng kiểu trong biểu thức. Thí dụ: x:= 5 + chr ©;

27. Invalid subrange base type: Kiểu cơ sở cho kiểu Đoạn không hợp lệ. Chỉ Được dùng các biểu thức bậc hàm kiểu cơ sở.

28. Lower bound greater than upper bound: Giới hạn dưới lớn hơn giới hạn trên. Thí dụ a; ARRAY [5.. -5] of ...

29. Ordinal type expected: Cần một kiểu thứ bậc. Trong trường hợp này không Được dùng các kiểu Real, string, Record, PROCEDURE hoặc pointer.

30. Integer constant expected: Cần một hằng nguyên

31. Constant expected: Cần một hằng

32. Integer or rcal constant expected: Cần một hằng nguyên hoặc thực

33. Type identifier expected: Cần một Định danh kiểu

34. Invalid function resunt type: Kiểu kết quả của hàm không hợp lệ. Hàm chỉ cho kết quả về thuộc kiểu Đơn, string và pointer. Trường hợp sau Đây sẽ là lỗi:

```
TYPE
 Phân số = RECORD
 Tử: integer;
 Mẫu: Integr
 END;
FUNCTION Rút gọn (x: phân số): phân số;
```

35. Label identifier expected: Cần chọn nhãn Lờ khuyên: Tốt nhất là nên tránh dùng toán tử GOTO và do Đó bạn không phải dùng Đến nhãn.
36. BEGIN expected: Thiếu BEGIN
37. END expected: Thiếu END
38. Integer expression expected: Cần biểu thức nguyên
39. Ordinal expression expected: Cần biểu thức thứ bậc
40. Boolean expression expected: Cần biểu thức kiểu BOOLEAN
41. Operand types do not match operator: Kiểu toán hạng không phù hợp với toán tử
42. Error in expression: Biểu thức sai. Thường gặp trường hợp sử dụng ký tự lạ hoặc quên viết dấu phép toán trong biểu thức.
43. Illegal assignment: Gans Gán không hợp lệ. Không Được gán trị cho biến tệp hoặc biến không Định kiểu. Không Được gán trị cho Định danh hàm ở ngoài thân của hàm Đó.
44. Field identifier expected: Cần một Định danh thường. Lỗi phát sinh khi sử dụng bản ghi (RECORD) không có trường hợp Đi kèm.
45. Object file too large: Tệp Đóch quá lớn. TP không thể ghép các tệp kiểu OBJ lớn hơn 64 KB (1kilo byte = 1024 bytes)
46. . Undefined external: Chưa khai báo các hàm hoặc thủ tục ngoài. Các hàm hoặc thủ tục ngoài cần Được khai báo ở mục PUBLIC trong các tệp OBJ hoặc trong các tệp hợp ngữ ASM. Trong chế Độ { \$ L ... } cần liệt kê Đầy Đủ tên các tệp OBJ.
47. Invalid object file record: Tệp OBJ không có cấu trúc chuẩn mực theo quy Định của TP.
48. Code segment too large: Đoạn mã quá lớn. Các chương trình hợp thành phải có kích thước không quá 65520 bytes.
49. Data segment too large: Đoạn dữ liệu quá lớn.
50. Do expected: Thiếu Do trong các cấu trúc FOR hoặc WHILE hoặc WITH
51. Invalid PUBLIC definition: Khai báo sai trong mục PUBLIC
52. Invalid EXTRN definition: Quá nhiều khai báo EXTRN. TP không sử lý quá 256 khai báo EXTRN trong mỗi tệp OBJ.
53. Too many EXTRN definitions Quá nhiều Định nghĩa EXTERN
54. OF expected: Thiếu OF trong TYPE, CASE, FILE, SET, ARRAY
55. INTERFACE expected: Thiếu mục INTERFACE
56. Invalid relocatable reference: Tham chiếu không hợp lệ
57. THEN expected: Thiếu THEN trong IF
58. TO or DOWN TO expected: Thiếu To hoặc DOWN TO trong FOR
59. Undefined forward: chưa khai báo trước.
60. Too many procedures: Quá nhiều thủ tục (hoặc hàm). TP cho phép tối Đa 512 thủ tục (PROCEDURE) và hàm (FUNCTION) trong một module. Nếu có quá nhiều bạn cần tách chương trình thành hai hoặc nhiều module.
61. Invalid typecast: chuyển Đổi kiểu không Đúng
62. Division by zero: chưa có 0
63. Invalid file type: Kiểu tệp không Đúng. Mỗi kiểu tệp làm việc với một số thủ tục dành riêng cho kiểu Đó. Thí dụ tệp TEXT làm việc với thủ tục readln, tệp Định kiểu làm việc với thủ tục seek. Nếu gọi thủ tục readln cho tệp Định kiểu hoặc gọi thủ tục seek cho tệp text sẽ sinh lỗi 63.

- 64. Can not Read or write vaziabls of this type: Không thể Đọc hoặc ghi biến thuộc kiểu này. Lưu ý rằng - Read và Readln có thể Đọc từ tệp vào các biến kiểu char, integer, real, Boolean hoặc string.
- 65. Pointer variable expected: Cần một biến con trỏ.
- 66. String variable expected: Cần một biến string.
- 67. String expression expected; Cần một biểu thức string.
- 68. Unit not found: Không tìm thấy Đơn thể (Unit). Cần khai báo chúng trong mục USES và Đặt Đường dẫn trong mục OPTIONS của môi trường TP.
- 69. Unit name mismatch: Tên Đơn thể không Đúng.
- 70. Unit version mismatch: version Đơn thể không phù hợp.
- 71. Duplicate Unit name: Tên Đơn thể trùng lặp

72. Lỗi Unit file format error: Tập Đơn thể (có Đuôi TPU) không có dạng quy Định.
73. Implementation expected: Thiếu phần IMPLEMENTATION trong Đơn thể.
74. Constant and case types do not match: Kiểu hằng và kiểu biểu thức trong CASE không phù hợp với nhau.
75. Record variable expected: Cần một biến kiểu RECORD.
76. Constant out of range: Hằng vượt quá miền. Lỗi có thể gặp trong các tình huống sau:  
- Chỉ dẫn mảng vượt ra ngoài giới hạn của mảng - Gọi thủ tục và hàm với các tham trị cụ thể vượt ra ngoài giới hạn Đã khai báo.
77. File variable expected: Cần biến tệp.
78. Pointer expression expected: Cần một biểu thức kiểu con trỏ.
79. Integer or Real expssion expected: Cần một biểu thức kiểu INTEGER hoặc REAL.
80. Labelnot within current block: Nhãn không có trong khối Đang xét.
81. Label already defined: Đã có nhãn.
82. Undefined label in processing statement part: Không thấy nhãn hiệu trong Đoạn Đã xử lý. Nhãn Đã Được khai báo, Đã thấy toán tử GOTO có tham Đối là nhãn Đó nhưng không thấy nhãn trong Đoạn văn bản. Mỗi nhãn cần xuất hiện ít nhất là 3 lần theo sơ Đồ sau:  
    LABEL L; (\* 1 \*)  
    ... (\* 2 \*) L: FOR i:= TO ...  
  
    GOTO L; (\* 3 \*)  
Nếu nhãn L không có mặt ở (\* 2 \*) sẽ sinh lỗi 82.
83. Invalid @ argument: Dùng các toán tử @ với toán hạng không hợp lệ. Toán hạng hợp lệ cho @ là các Định danh của biến, thủ tục và hàm.
84. Unit expected: Cần có từ khoá UNIT.
85. ";" expected: Cần có dấu chấm phẩy.
86. ":" expected: Cần có dấu hai chấm.
87. ", " expected: Cần có dấu phẩy
88. "(" expected: Cần có dấu mở ngoặc Đơn
89. ")" expected: Cần có dấu Đóng ngoặc Đơn
90. "=" expected: Cần có dấu bằng.
91. ":=" expected: Cần dấu gán
92. "[" or "(" expected: Cần mở dấu ngoặc vuông khi khai báo hoặc chỉ Định phần tử của mảng, tập.
93. "]" or ")" expected: Cần dấu Đóng ngoặc vuông khi khai báo hoặc chỉ Định phần tử của mảng hoặc tập.
94. "." expected: Cần dấu chấm khi chỉ Định một trường trong bản ghi
95. ".." expected: Cần dấu nhiều chấm khi liệt kê giới hạn mảng, tập.
96. Too many variables: Quá nhiều biến. Nguyên nhân xuất hiện lỗi có thể là: - Tổng kích thước của các biến toàn cục vượt quá 64 KB - Tổng kích thước của các biến cục bộ mô tả trong thủ tục hoặc hàm vượt quá 64 KB.
97. Invalid FOR control variable: Biến Điều khiển trong toán tử FOR không Đúng. Phải dùng biến kiểu liệt kê.
98. Integer variable expected: Cần biến nguyên.

99. Files are not allowed here: Không Được dùng tệp ở Đây. Hằng Định kiểu không thể là tệp.
100. String. length, mismatch: Chiều dài xâu không phù hợp với số lượng các phần tử của mảng ký tự.
101. Invalid ordering of fields: Trật tự các trường (trong bản ghi) không Đúng.
102. String constant expected: Cần một hằng kiểu xâu.
103. Integer or real variable expected: Cần biến nguyên hoặc thực (biến số).
104. Ordinal variable expected: Cần biến (kiểu) thứ tự.
105. INLINE error: Lỗi khi sử dụng toán tử INLINE ( < ). Toán tử này không Được dùng với các con trỏ tới biến. Các con trỏ loại này luôn luôn có kích thước bằng một từ.
106. Character expression expected: Cần biểu thức kiểu ký tự.
107. Too many relocation items: Quá nhiều phần tử Động. Kích thước của bảng Động cho tệp. EXE vượt quá 64 KB.
108. Not enough memory to run program: Không Đủ miền nhớ Để thực hiện chương trình.
109. Can not find EXE file: Không tìm thấy tệp EXE.
110. Can not run a Unit: Không Được phép thực hiện một modul. Modul là một thư viện các khai báo (biến, thủ tục, hàm) Để ghép với các chương trình sử dụng chúng chứ không dùng Để thực hiện Độc lập.
111. Compilation aborted: Huỷ việc dịch (bằng lệnh ^ BREAK).
112. CASE constant out of range: Hằng của toán tử CASE không nằm trong giới hạn. - 32768 đến 32767
113. Error in statement: Câu lệnh sai. Lỗi không xảy ra khi viết sai ký tự Đầu tiên của câu lệnh.
114. Can not call an interrupt procedure: Không thể gọi thủ tục ngắt.
115. Must have an 8087 to compile this: Cần có bộ xử lý 8087 Để dịch chương trình này nếu có khai báo chế Độ dịch { \$ N + }. hãy thử dùng tổ hợp { \$ N +, E + }.
116. Must be in 8087 mode compile this: Phải dùng chế Độ 8087 Để dịch chương trình này. Không Được sử dụng các kiểu SINGLE, DOUBLE, EXTENDED, và COMP trong chế Độ { \$ N - }.
117. Target address not found: Không tìm thấy Địa chỉ Đã cho.
118. Include files are not allowed here: Không Được dùng tệp lồng ở Đây.
119. TMP file format error: Lỗi khuôn dạng tệp TMP. Tệp có Đầu Đúng là TMP nhưng nội dung không Đúng dạng Đó.
120. NIL expected: Cần giá trị bằng NIL.
121. Invalid qualifier: Định danh không hợp lệ. Có thể do các nguyên nhân sau: -Viết một biến Đơn có kèm chỉ dẫn như biến mảng. - Viết một biến Đơn hoặc mảng theo trường như biến bản ghi. - Sử dụng biến không thuộc kiểu con trỏ như một biến con trỏ.
122. Invalid variable reference: Tham chiếu không hợp lệ tới biến. Bản thân biến Được khai báo Đúng kiểu con trỏ nhưng nội dung của nó không phải là Địa chỉ. Thường gặp trong khi gọi một hàm kiểu một con trỏ nhưng quên dùng ký hiệu ^.
123. Too many symbols: Có quá nhiều ký hiệu. Chương trình có tổng số ký hiệu lớn hơn 64 KB. Cần chia thành những module nhỏ hơn nếu dịch trong chế Độ { \$ D + }.
124. Statement part too large: Phần lệnh toán tử quá lớn. Turbo Pascal dành khoảng 24 KB cho phần toán tử. Nếu phần này quá lớn thì nên cắt chúng thành nhiều thủ tục

hoặc hàm. Nói chung một người lập trình tốt thường quan tâm Đến tổ chức cấu trúc.  
125. Module has no debug information: Trong module không có thông tin bắt lỗi do Đó  
khi thực hiện chương trình Đã dịch mà gặp lỗi Turbo bascal sẽ không chỉ ra Được toán



từ sinh lỗi. Nên dịch lại chương trình theo chế Độ { \$ D + } hoặc sử dụng mục COMPILER/FIND ERROR Để tìm lỗi.

126. Files must be var parameters: Tham biến hình thức kiểu tệp trong thủ tục hoặc hàm phải Được khai báo theo chế Độ truyền theo biến (có tiếp Đầu VAR).
127. Too many conditional symbols: Không Đủ miền nhớ Để xác Định các ký hiệu Điều kiện. Thử bỏ bớt một số ký hiệu hoặc giảm chiều dài của chúng.
128. Misplaced conditional directive: Thiếu các khai báo Điều kiện, chẳng hạn trong chương trình có { \$ ELSE } hoặc { \$ ENDIF } nhưng lại không có { \$ IFDEF }, { \$ IFNDEF } hoặc { \$ WIFOPT }.
129. ENDIF directive missing: Thiếu khai báo Ở \$ENDIF}.
130. Error in inital conditonal defines: Định nghĩa Điều kiện sai. Các Điều kiện ban Đầu Đặt trong OPTIONS/COMPILER/CONDITIONAL DEFILES không Đúng. Nếu viết nhiều Điều kiện cần ngăn chúng bằng dấu chấm phẩy.
131. Header does not match previous definitio: Phần Đầu không phù hợp với khai báo trước Đó. Có thể do các nguyên nhân sau: - Đầu của thủ tục và hàm của phần INTERFACE và phần IMPLEMENTATION không phù hợp với nhau. - Đầu của thủ tục và hàm khai báo trước (bằng từ chỉ thị FORWARD) không phù hợp với phần mô tả.
132. Criticaldisk error: Lỗi nặng về Đĩa, thí dụ ở ổ Đĩa chưa Đóng hoặc chưa Được lắp Đĩa.
133. Can not evaluate this expression: không thể tính Được giá trị của biểu thức, thí dụ  $\text{CONTS } C = \text{SIN}(2)$ ; sẽ sinh lỗi và khi khai báo hằng Đã dùng hàm SIN.
134. Exprsseion incorrectly terminated: Biểu thức kết thúc sai.
135. Invalid format specifier: Dạng thức Đặt tả sai.
136. Invalid inderect reference: Không sử dụng Đúng phương thức truy nhập trực tiếp.
137. Structured variable arenot allowed here: Không Được dùng biến cấu trúc ở Đây. Lỗi phát sinh khi lập trình viên thực hiện các phép toán không Được phép dùng cho các biến cấu trúc, thí dụ như nhân hai bản ghi.
138. Can not evaluate withut system unit: Cấm tính toán biểu thức khi chưa gọi module SYSTEM. Muốn làm Điều này ta phải Đưa module SYSTEM vào tệp TURBO.TPL.
139. Can not access this symbol: Không truy nhập Được tới ký hiệu này.
140. Invalid floating-point operation: Sử dụng sai thao tác với số thực. Có thể xảy ra hiện tượng tràn ô nhớ (kết quả tính toán vượt quá khả năng biểu diễn của ô nhớ) hoặc phép chia cho số 0.
141. Can not compile overlay to memory: Không thể dịch overlay trên miền nhớ RAM (mà phải Đặt chế Độ dịch ra Đĩa (Disk)).
142. Procedure or function variable expected: Cần sử dụng biến dạng thủ tục hoặc hàm. Tham biến hình thức của một thủ tục hoặc hàm có thể là một thủ tục hoặc hàm.

#### Các mã lỗi và ý nghĩa khi làm việc với file:

1. File not found: Không thấy tệp. Lỗi phát sinh khi sử dụng các thủ tục RESET, APPEND, RENAME hoặc ERASE mà không tìm thấy tệp cần thao tác
3. Path not found: Không thấy Đường dẫn. Lỗi xuất hiện khi:

- Các thủ tục RESET, APPEND, RENAME hoặc ERASE không tìm Được Đường dẫn tới tệp cần thao tác.

- Các thủ tục CHDIR, MKDIR, hoặc RMDIR nhận Được tham trị sai về thư mục hoặc Đường dẫn tới thư mục con cần sử lý.
- 4. Too many open files: mở quá nhiều tệp. Xuất hiện khi gọi các thủ tục RESET, REWRITE, hoặc APPEND. Để có thể tăng số lượng tệp cần mở hãy sửa hoặc thêm dòng FILES = xx Trong tệp CONFIG.SYS rồi khởi Động lại máy.xx là khối lượng tối Đa các tệp cần mở, thí dụ 20.
- 5. File access denied: Không Được phép truy nhập tệp. Lỗi xuất hiện khi: Các thủ tục RESET và APPEND nhận Được tham biến tệp ở chế Độ chỉ Được Đọc trong khi tham biến FILEMODE lại Được phép ghi.
  - Gọi thủ tục REWRITE khi thư mục Đã Đầy hoặc khi tham biến tệp Được khai báo ở chế Độ chỉ Được Đọc.
  - Gọi thủ tục RENAME với tên tệp là tên thư mục hoặc tên tệp mới Đã có trong thư mục.
  - Gọi thủ tục ERASE với tên tệp là tên thư mục hoặc tham chiếu tới tệp chỉ Được Đọc.
  - Gọi thủ tục MKDIR với tên thư mục Đã có hoặc tên thư mục Đã Đầy.
  - Gọi thủ tục RMDIR với thư mục chưa rỗng hoặc sai Đường dẫn.
  - Gọi các thủ tục READ hoặc BLOCKREAD khi chưa mở tệp.
  - Gọi các thủ tục WRITE hoặc BLOCKWRITE khi chưa mở tệp Để ghi.
- 6. Invalid file handle: Kênh cho tệp chưa thông
- 12. Invalid file access code: Mã truy nhập tệp bị sai. Lỗi xuất hiện khi gọi các thủ tục RESET hoặc APPEND với tham biến FILEMODE chứa giá trị không chuẩn.
- 15. Invalid drive number: Số hiệu ổ Đĩa bị sai. Xuất hiện khi gọi GETDIR.
- 16. Can not remove current directory: Không Được xoá thư mục hiện hành. Xuất hiện khi gọi RMDIR với thư mục Đang mở.
- 17. Cannot renameacross drives: Không Được Đổi tên với hai ổ Đĩa khác nhau. Xuất hiện khi gọi RENAME với hai tệp nằm ở hai ổ Đĩa khác nhau.
- 100. Disk read error: Lỗi Đọc Đĩa. Xuất hiện khi gọi READ từ tệp Định kiểu Để Đọc khi tệp Đã Được duyệt hết.
- 101. Disk write error: Lỗi viết vào Đĩa. Xuất hiện khi gọi close, write, writel hoặc flush với Đĩa Đầy.
- 102. File not assigned: Chưa gán tên cho tệp bằng thủ tục ASSIGN mà Đã gọi các thủ tục RESET, REWRITE, APPEND, RENAME hoặc ERASE.
- 103. File not open: Tệp chưa mở. Xuất hiện khi gọi close, rean, write, seek, eof, fileposfilesize, flush, blockread, blockwrite với tệp chưa Được mở
- 104. File not open for input: Tệp chưa Được mở Để Đọc. Xuất hiện khi gọi READ, READLN, EOF, EOLN. SEEKEOF, hoặc SEEKEOLN với tệp văn bản (kiểu text) chưa Được mở Để Đọc.
- 105. File not open for output: Tệp chưa Được mở Để ghi. Xuất hiện khi gọi WRITE hoặc WRITEL với tệp văn bản chưa Được mở Để ghi
- 106. Invalid numeric format: Dạng số sai. Xuất hiện khi gọi READ hoặc READLN một số từ tệp văn bản.

- 150. Disk is write protected: Đĩa chống ghi. Lỗi chống ghi ở Đĩa bị che kín.
- 151. Unknown unit: Module lạ
- 152. Drive not ready: Ổ Đĩa chưa sẵn sàng.
- 153. Unknown command: Lệnh lạ.
- 154. ARC error in data (Cycle Redundant Code -): Lỗi ở dữ liệu ban Đầu.
- 155. Bad drive request structure length: Chiều dài cấu trúc Đĩa không Đúng.
- 156. Disk seek error: Đầu - Đọc ghi ở chỗ Đĩa bị Đặt sai
- 157. Unknown media type: Kiểu vật mang (tin) không rõ
- 158. Sector not found: Không tìm Được Sector của Đĩa.
- 159. Printer out of paper: hết giấy in ở máy in
- 160. Device write fault: Lỗi khi dùng thiết bị ghi
- 161. Device read fault: lỗi khi dùng thiết bị Đọc
- 162. Hardware failure: Có sự cố phần cứng.

### **Các lỗi rất nặng:**

- 200. Division by zero: chia cho số 0.
- 201. Range check error: Lỗi kiểm tra giới hạn.
- 202. Stack overflow error: Ngăn xếp Đầy. Khi chương trình thực hiện cần có vùng nhớ Để tổ chức biến Động và gọi các thủ tục và hàm. Vùng nhớ này Được tổ chức theo nguyên tắc ngăn xếp. Nếu thiếu chỗ thì sinh lỗi 202.
- 203. Heap overflow error: Chùm Đầu. Khi chương trình thực hiện cần cấp phát vùng nhớ Động cho các thủ tục new hoặc getmem. Các vùng nhớ này Được quản lý theo nguyên tắc chùm (như bộ rễ của một cây). Nếu thiếu chỗ sẽ sinh lỗi 203.
- 204. Invalid pointer operation: Thao tác con trỏ không Đúng. Lỗi xuất hiện khi gọi DISPOSE hoặc FREEMEM với giá trị NIL hoặc với Địa chỉ nằm ngoài giới hạn dành cho phần bộ nhớ thao tác.
- 205. Floating point overflow: Toàn giá trị thực (real)
- 206. Floating point underflow: Lệch hẫng ở giá trị thực (Real). Lỗi này xuất hiện khi sử dụng các bộ Đồng xử lý loại 8087/80287/80387
- 207. Invalid floatingpoint operation: Thao tác sai với các giá trị thực (Real). Lỗi xuất hiện trong các trường hợp:
  - Tham trị của các hàm TRUNC hoặc ROUND không thể biến Đổi Được trong phạm vi của kiểu LONGINT (từ -2147483648 Đến +2147483647)
  - hàm lấy căn bậc 2 SQRT có tham trị âm
  - Hàm logarit LN chứa tham trị không dương
  - Ngăn xếp của bộ Đồng xử lý bị tràn.
- 208. Overlay manager not installed: Chưa Đặt phần hệ Điều khiển Overlay. Thường là thiếu thủ tục OVRINIT (khởi tạo - Đặt chế Độ Overlay) hoặc thủ tục này không Được thực hiện Đầy Đủ.
- 209. Overlay file read error: Lỗi khi Đọc tệp overlay.



## KINH NGHIỆM THI CỬ

Các cuộc thi là nơi Để bạn chứng tỏ trình Độ và khả năng của mình. Công sức học tập bấy lâu chỉ quyết Định trong 3 tiếng làm bài thi. Do Đó, kinh nghiệm là thứ không thể thiếu nếu muốn Đạt kết quả tốt.

### 1. Kiến thức

Cuộc thi nào cũng Đòi hỏi kiến thức trong một miền giới hạn nhất Định, và hãy Đảm bảo rằng bạn Đã chuẩn bị tốt những kiến thức Đó. Các cuộc thi lớn thường có lượng kiến thức giới hạn không hề nhỏ. Do Đó trước khi thi bạn phải tạo cho mình một cái nhìn tổng quát về lượng kiến thức Đã có.

Cần chú ý rằng việc bổ sung ,tìm hiểu kiến thức trong quá trình học tập khác với trong quá trình chuẩn bị cho kì thi. Việc vội Đầu bổ sung kiến thức mới ngay trước kì thi là hoàn toàn không nên. Vì như thế chúng ta sẽ bị ảnh hưởng bởi lượng kiến thức Đó và làm lu mờ Đi một lượng lớn kiến thức Đã thu thập Được bấy lâu.

### 2. Tâm lí

Nhiều người thường nói nào là trước ngày thi không nên xem nhiều phim, không Được chơi game, nghe nhạc ... Nhưng những hạn chế này chỉ làm cho bạn cảm thấy thêm sức ép mà thôi. Trước ngày thi khoảng 2 ngày, bạn cứ làm những gì bạn thích và Đôi lúc cần phải quên Đi việc Đối mặt với kì thi trước mắt. Cố gắng tạo một tâm lí thoải mái nhất.

Bạn cũng Đừng quá lo về lượng kiến thức hao hụt khi bạn giải trí. Đối với môn Tin học, chúng ta không học theo kiểu “*học thuộc lòng*”, vì thế kiến thức không thể mất Đi trong ngày một ngày hai Được.

### 3. Cách làm bài

Đối với môn Tin, sẽ có 180 phút Để làm 3 bài thi. Khi cầm một tờ Đề, những phút Đầu tiên, bất cứ ai cũng sẽ thấy hồi hộp. Hãy Đọc bài Đầu tiên (vì Đây thường là bài dễ, cho học sinh kiểm Điểm). Nếu vẫn chưa lấy lại Được bình tĩnh cần thiết thì có thể code cấu trúc form cho chương trình, rồi tạo trước file input.

Thực hiện làm từng bài theo chiến thuật: “*bài dễ làm trước, khó làm sau*”; nguyên tắc “*làm bài nào chắc chắn bài Đó*”.

Việc phân bố thời gian Để giải quyết 3 bài trong 180 phút cũng cực kì quan trọng. Bài 1 thường là bài cho Điểm nên cố gắng hoàn thành trong thời gian càng ngắn càng tốt. Hai bài còn lại mỗi bài thường dành 10 - 15 phút suy nghĩ Để tìm ra thuật toán tốt. Nếu

không Được tìm thì chuyển sang hướng cài Đặt Duyệt ăn 60% số Điểm (Đối với môn tin học thì mỗi bài có giới hạn 60% số test với dữ liệu nhỏ và các bạn có thể ăn Được số test này với thuật toán Duyệt nếu cài Đặt khéo). Sau Đó nếu còn dư thời gian có thể quay lại cải tiến tiếp, phải luôn luôn cải tiến chương trình cho Đến khi nào hết thời gian, cố gắng ăn càng nhiều test, Đúng càng nhiều trường hợp càng tốt. Chưa cần biết các test hay trường hợp Đó có mang lại nhiều Điểm hay không.

Đặc biệt chú ý Đến việc Đặt tên file chương trình và file INPUT, OUTPUT. Nên làm những việc này trước khi tiến hành lập trình. Nên lập trình trực tiếp bằng Free Pascal và Để chế Độ báo các lỗi như tràn mảng hay tràn bộ nhớ. Chú ý các giới hạn của Đề bài Để khai báo mảng hay biến số.

Và khi thời gian thi còn lại 10 phút, nên kiểm tra lại mọi thứ Để Đảm bảo không có sai sót Đáng tiếc nào về những file sắp ghi lên Đĩa.

Các bạn nên luyện tập cách làm trên với các Đề thi thử Để tránh lúng túng trong các kì thi quan trọng.

### **Kinh nghiệm thi cử**

#### **Làm bài thật nhanh**

Suy nghĩ thì phải cẩn thận, nhưng khi đặt tay xuống làm bài thì phải làm bài thật nhanh, không chần chừ do dự. Nên nhớ đây là một yếu tố rất quan trọng, nếu không nhiều khả năng sẽ bị thiếu thời gian! Luyện tập làm bài nhanh mỗi khi có thể để đến khi đi thi sẽ quen.

#### **Đọc kỹ và suy nghĩ tất cả các bài trước khi làm**

Nên dành một khoảng thời gian để suy nghĩ tất cả các bài toán, sau đó mới quyết định sẽ làm bài nào trước. Có rất nhiều trường hợp, một bài toán tưởng chừng đơn giản nhưng đến khi làm gần xong mới phát hiện là bài khó. Vì vậy cần tỉnh táo tránh lãng phí thời gian.

#### **Làm bài dễ trước**

Hoàn thành bài dễ trước để đảm bảo một số điểm cho mình và tạo cảm giác yên tâm khi làm những bài còn lại.

#### **Kiểm tra mỗi bài sau khi làm xong**

Sau khi làm xong mỗi bài, cần phải kiểm tra lại chương trình trước khi chuyển sang bài khác.

Để kiểm tra chương trình, cách tốt nhất là test thật nhiều. Có thêm một test đúng, xác suất đúng đắn của chương trình sẽ tăng lên đáng kể.

Dùng các test kích thước nhỏ, tự sinh ra để kiểm tra tính đúng đắn của thuật toán; các test lớn, nhận giá trị lớn nhất và giá trị ngẫu nhiên để kiểm tra thời gian thực hiện của thuật toán, phát hiện các lỗi như range check, stack overflow, arithmetic overflow.

Có những bài toán mà có thể kiểm tra kết quả một cách đơn giản (ví dụ bài Dò mìn). Khi đó đừng ngại viết chương trình tạo test hoặc chương trình kiểm tra kết quả nếu cần thiết.

Với những bài toán mà có thuật toán đúng đắn rất đơn giản (tuy nhiên thời gian thực hiện lớn hơn), hãy viết một chương trình như vậy dùng để kiểm tra kết quả. Chẳng hạn, bài toán cặp điểm gần nhất: cho  $N$  điểm trên mặt phẳng, tìm hai điểm có khoảng cách gần nhất. Dùng chương trình  $O(N^2)$  để so sánh với chương trình  $O(N \log N)$ , nếu hai chương trình cho kết quả như nhau ở các bộ test thì xác suất đúng đắn của chương trình  $O(N \log N)$  là rất cao. Tương tự, nếu có hai phương pháp khác nhau nhiều để giải bài toán mà cùng cho kết quả như nhau thì xác suất đúng của chương trình là tương đối cao.

#### **Đọc kỹ đề bài**

Đọc lại một đề bài nhiều lần, đọc cả trước khi làm, sau khi làm xong, phát hiện xem mình có nhầm lẫn, thiếu sót chi tiết gì không!



### Sửa lỗi chương trình

Khi chương trình chạy cho ra kết quả sai, cần sửa lỗi (debug) chương trình. Cần tận dụng hiệu quả các công cụ debug của Free Pascal như watches, break point, sử dụng tốt các phím tắt F4, F7, F8. Thông thường chương trình sẽ bị một vài lỗi nhỏ như viết sai tên biến trong một biểu thức, thiếu một đoạn xử lý,...

Có thể xảy ra trường hợp phát hiện cả thuật toán sai, nhất là đối với những bài có thuật toán đúng đắn thì điều này là hoàn toàn có thể. Khi đó đừng hốt hoảng! Cần bình tĩnh tính toán xem xác suất đúng đắn của chương trình có cao không, có nên làm tiếp bài toán đó hay không? Đừng bao giờ để bị vướng vào một bài toán quá lâu, nếu thấy lo lắng vì đã dành nhiều thời gian cho một bài mà vẫn cho kết quả sai, hãy tạm thời chuyển sang bài khác.

Trong trường hợp cần thay đổi toàn bộ chương trình, nhớ backup chương trình cũ vào một bản khác, phòng khi cần sử dụng lại.

### Phong cách viết chương trình

Phong cách viết chương trình là điều rất quan trọng khi đi thi. Với phong cách viết chương trình tốt sẽ có lợi thế hơn. Nhưng điều này lại chỉ có thể có khi luyện tập nhiều. Phong cách viết chương trình tốt không có nghĩa là những cách viết để cho chương trình chạy nhanh, mà là đảm bảo để làm bài nhanh, sửa lỗi nhanh, giảm xác suất sai của chương trình và giảm xác suất nhầm lẫn cho người viết.

### Đừng đòi hỏi perfect

Trong trường hợp không thể nghĩ ra thuật toán đúng đắn cho một bài toán (hoặc bản thân bài toán cũng không có thuật toán đúng đắn, điều này rất có thể xảy ra trong kỳ thi quốc gia), cố gắng tìm cách đạt được một số điểm nhất định. Một chương trình đạt 80 hay 90 điểm hầu như không khác gì nhiều so với một chương trình perfect. Đôi khi không đáng để viết thêm một đoạn chương trình rất dài chỉ để đạt thêm 10-20 điểm, nhưng đôi khi chỉ cần thêm một hai dòng cận hay điều kiện thì thậm chí có thể đạt thêm một số điểm rất lớn. Điều quan trọng là cần bình tĩnh để đánh giá những gì nên làm và không nên làm.

### Đạt được 50% số điểm

Các bài toán thường đưa ra 50% (hoặc 30-40%) số test có giới hạn nhỏ, có thể dùng những thuật toán đơn giản hoặc duyệt. Tốt nhất nên thêm một điều kiện để kiểm tra nếu dữ liệu thuộc giới hạn này thì dùng chương trình đơn giản hoặc duyệt để chạy.

### Sao lưu bài làm

Thỉnh thoảng hãy sao lưu bài làm vào nhiều thư mục, nhiều ổ đĩa khác nhau phòng khi có sự cố kỹ thuật. Hãy tạo 3 thư mục cho 3 bài làm. Lưu các bài sẽ nộp vào một nơi riêng, tránh nộp nhầm bài.

### Đánh giá, quyết định thông minh

Có nhiều thứ cần đánh giá: mức độ khó của mỗi bài, xác suất đúng đắn của thuật toán nghĩ ra, thời gian cần thiết để làm một bài, kiểm tra kết quả và tạo test có khó không? Từ đó quyết định làm bài nào trước, có nên tiếp tục cải tiến một bài không, cải tiến như thế nào sẽ hiệu quả hơn, kiểm tra chương trình như thế nào để chắc chắn nhất. Hãy linh hoạt

### **Khi lập trình**

Khi đi thi thường sẽ sử dụng compiler và IDE Free Pascal.

### **Đặt chỉ dẫn bộ nhớ**

Trong trường hợp chương trình lặp vào thủ tục đệ quy nhiều lần, cần tăng kích thước stack bằng cách đặt chỉ dẫn bộ nhớ:

```
{ $M 2000000,0,2000000 }
```

### **Phong cách viết chương trình**

Nếu chương trình sử dụng ít biến, có thể đặt tên các biến, mảng ngắn gọn đơn giản để tăng tốc độ làm bài, ví dụ a, b, x, y, ... mảng quy động là f. Tuy nhiên nếu chương trình sử dụng nhiều biến, các biến mang ý nghĩa khác nhau thì hãy đặt tên các biến dài và mang ý nghĩa của chúng để tránh nhầm lẫn. Nên nhớ việc nhầm lẫn giữa các biến là vấn đề rất khó chịu. Nên đặt tên các biến bằng tiếng Việt và viết chương trình bằng chữ in thường để tăng tốc độ làm bài.

Giữa các câu lệnh, nên xuống dòng để tận dụng các công cụ debug của Free Pascal như các phím F4, F7, F8.

### **Sử dụng chuỗi**

Nếu cần sử dụng chuỗi có độ dài vượt quá 255 ký tự cần đặt chỉ dẫn {H+} ở đầu chương trình.

### **Cắt dán**

Để sử dụng các phím tắt Ctrl-C, X, V cho cắt dán chữ như Windows, chọn Options – Keyboard & mouse – đánh dấu Microsoft convention.

Chúc các bạn thành công!