Bài V. đề bài:

V. HÌNH VUÔNG LỚN NHẤT

Cho một bảng số N hàng, M cột chỉ gồm 0 và 1. Bạn hãy tìm hình vuông có kích thước lớn nhất, sao cho các số trong hình vuông toàn là số 1.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 10$).

Mỗi test bắt đầu bởi 2 số nguyên N, M ($1 \leq N, M \leq 500$).

N dòng tiếp theo, mỗi dòng gồm M số mô tả một hàng của bảng.

Output:

Với mỗi test, in ra đáp án là kích thước của hình vuông lớn nhất tìm được trên một dòng.

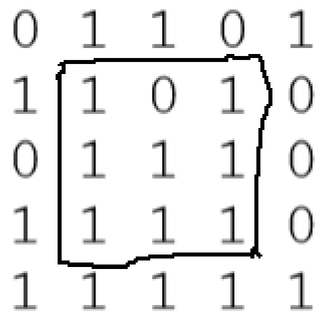
Test ví dụ:

Input:	Output
2 6 5 0 1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 2 2 0 0 0 0	3 0

I. Phân tích bài toán :
   - Mục là tìm ra hình vuông mà trong đó không có số 0 nào . đáp án in ra là độ dài cạnh của hình vuông.

II. Tiếp cận bài toán.
   1. Cách 1:
      - ta nghĩ ra đầu tiên nhất là duyệt từ tọa độ A(x , y) đến tọa độ B(x+k,y+k) với K >= 0.

        Ví dụ:



Điểm A sẽ có tọa độ là A(1,1) nếu lấy gốc là (0,0) và B(3,3) với k = 1.

- Và ta kiểm tra xem hình vuông đó có phải là hình vuông thỏa mãn đề ra là tất cả cái số trong đó đều là số 1 hay không và việc kiểm tra như này mất O(k^2).
- Cứ mỗi một x,y thì ta phải chạy từ k = 0 đến k + x < n và k + y < m. mất O(k^2) nữa.
- Vậy độ phức tạp của 1 điểm A(x,y) sẽ là O(k^2*k^2) = O(k^4).
- Nhưng điều đáng tồi tệ hơn nữa là ta phải duyệt tất cả cái điểm từ ô (0,0) đến o(n,m). vậy độ phức tạp để duyệt như thế lại là O(m*n).
- Tổng lại độ phức tạp của ta là O(m*n*k^4) và sấp sỉ = O(R^6) với R = max(n,m). quá lớn . nhưng sao lại là quá lớn vì ta thay giá trị lớn nhất của m và n vô ta được R^6= 500^6 = 5^6* 10^12 . mà trong khi đó 1 s của máy tính bình thường chạy khoảng không quá 10^8 phép tính toán căn bản , mà khi đó ta ước lượng thì con số khi gấp 5^6*10^4 lần. => độ phức tạp quá lớn.

2. Cách 2: Quy hoạch động.
   - Bài này có hint rất nhiều trên mạng nhưng các bạn đọc hint đó có hiểu được không tại sao nó lại như vậy. như vậy rất khó có thể giải quyết được những bài toán tương tự. vậy cơ của chúng ta khó lên được.

   Điều đầu tiên nhất của quy hoạch động là ta phải nhớ nhưng nhớ cái j mới được?
   - ở đây theo tính chất của 1 hình chữ nhật mà nói thì hình chữ nhật được tạo bởi 2 cạnh chiều dài và chiều rộng vậy chỉ cần chiều dài và chiều rông khác nhau thì hình chữ nhật chắc chắn cũng khác nhau . vậy mình gọi gốc O(0,0) là gốc vậy mỗi thằng I , j tương ứng là chiều dài khác nhau thì phải có kết quả khác nhau vậy nên ta phải dung mảng 2 chiều F[i][j] để lưu kết quả tối ưu nhất độ dài hình vuông lớn nhất của hình chữ nhật OA với A có tọa độ A(I,j) .
   - nhưng đều đáng lo ngại nhất nếu ta lưu như ở trên thì việc chúng ta liên hệ với cái hình chữ nhật bé hơn là rất là khó :V.
   - Vậy để liên hệ với nhau thì ta sẽ gọi F[i][i] là hình vuông lớn nhất mà kết thúc tại điểm I,j . ví dụ

$$\begin{matrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Hình vuông màu xanh là hình vuông lớn nhất mà kết thúc tại điểmđược khoanh đỏ đó. Vậy cuối ta đi xây dựng công thức liên hệ giữa các f[i][j].

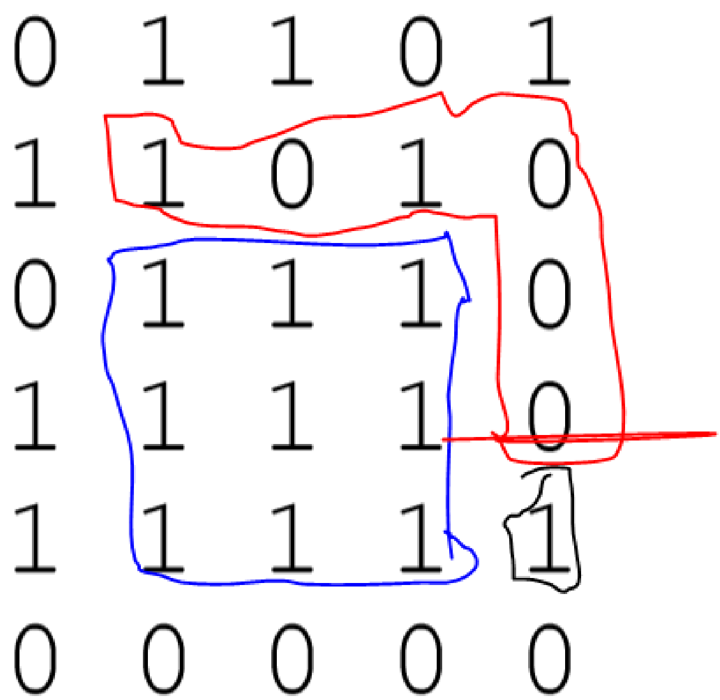
- Trường hợp thứ nhất thì mảng ta nhận được là vị trí thứ I , j của ta là số 0 vậy dĩ nhiên là cái F[i][j] = 0 rồi.
- Trường hợp thứ 2 thì vị trí I , j là số 1.
  - Nhiều bạn sẽ lầm tưởng là F[i][j] = f[i-1][j-1] + 1 chứ gì nhưng không phải. điều quan trọng nếu ta sử dụng hình vuông mà kết thúc ở I- 1, j - 1, thì ta phải 2 cạnh thêm vào là cạnh I chạy về lề trái và cạnh j chạy về lề phải có có thỏa mãn là trong khoảng suốt tiếp xúc với hình vuông thêm vào có xuất hiện số 0 nào không?

    Ví dụ:

$$\begin{matrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Ta muốn xét hình vuông kết thúc tại ô khoanh màu xanh thì giá trị f[i-1][j-1] = 2. Nhưng cái cạnh bên phải hình vuông lại không thỏa mãn vì nếu ta thêm điểm khoanh màu xanh vào hình vuông kết thúc ở tọa độ i-1 và j-1 thì là ta thêm toàn bộ điểm . nhưng nếu cả 2 cùng thỏa mãn thì tất nhiên f[i][j] có thể bằng f[i-1][j-1] + 1.

- Cũng tương tự vậy đối với f[i-1][j] và f[i][j-1] ta cũng kiểm tra xem có thỏa mãn kiểu vậy ko ?
  - Với f[i][j-1]



  - Với f[i-1][j].

$$\begin{matrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

- Nhưng với những việc này thì thường làm rất khó. Vậy nên ta phải tìm mối qua hệ của 3 thằng nằm cạnh A(I,j) xem có cách nào liên hệ được ko?
  Mình định hình vô nhưng mình vẽ không chuẩn nên mình xin phép nói mồm thui các bạn rang hiểu nếu giả sử mà nó để có thỏa mãn các vòng viền ngoài có đáp ứng được hay không là tùy thuộc vào 2 đứa còn lại ví dụ như hình vuông i-1,j-1 thì dĩ nhiên 2 cạnh đắp vào thì lại lại phục thuộc vào f[i-1][j] và f[i][j-1] tương ứng nếu cạnh của hình vuông mà 1 trong 2 đứa đó mà nhỏ hỏi f[i-1][j-1] chứng tỏ rằng số 0 nằm ở 1 trong 2 cạnh bên chứ không nằm bên trong hình vuông mà I -1, j -1 kiểm soát được. cũng tương tự đối với 2 thằng i-1,j và I,j-1. Vậy làm thế nào để chọn 1 trong 3 thằng mà 2 thằng còn lại lại không nhỏ hơn . đó chính xác thằng đó phải là min của 3 thằng.
- Vậy suy cho cùng thì công thức ở mục này sẽ là:
  f[i][j] = min(f[i-1][j-1], f[i-1][j], f[i][j-1]) + 1
  - Vậy tổng quát lại vấn đề f[i][j] = min(f[i-1][j-1], f[i-1][j], f[i][j-1]) + 1 nếu vị trí I,j là số 1 còn f[i][j] = 0, nếu ngược lại.
  - Còn 1 vấn đề nữa là giả ta quy hoạch động theo hướng nào các bạn làm thế nào cũng được nhưng phải luôn luôn có f[i-1][j-1], f[i-1][j], f[i][j-1] trước khi tính f[i][j] với mọi I,j. theo mình là chọn chạy theo hằng trước sau đó ta hết

hàng ta lên hàng mới => thỏa mãn => khời tạo nó đơn giản là tất cả f[i][j] = chính giá trị ở ô I,j. nếu I = 0 hoặc j = 0;

- Kết quả nhận được là max(f[i][j]) với mọi (0 <= I < n và 0<= j < m).