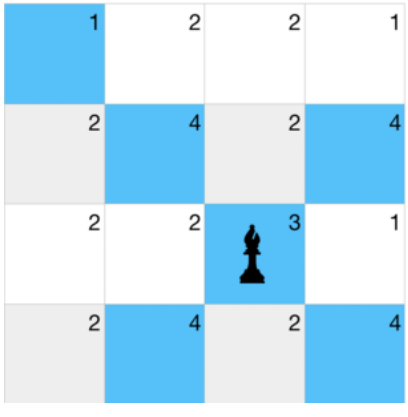
Problema: X-SUM

1 - **Objetivo**

Dada uma matriz linha(L)xcoluna(C), que representa um tabuleiro de xadrez, posicionar a peça Bispo de tal forma que a soma das casas em que a peça possa se mover seja o maior possível, sendo que cada posição na matriz possui um valor (peso).



2 - **Análise**

Primeiramente para solucionar este problema temos que a partir de uma posição dada no tabuleiro devemos somar os valores das diagonais que atravessam a posição.

Em seguida devemos pensar em como realizar essa soma de forma eficiente. Por exemplo, ao varrer a matriz, célula por célula, se decidirmos realizar a soma em cada caso irá gastar muito tempo de execução e dará erro de tempo limite de execução. Pensando neste problema uma maneira eficiente será varrer a matriz apenas uma vez e guardar a soma de cada uma das diagonais em um vetor.

Em uma matriz temos duas famosas diagonais, a principal e a secundária. Essas diagonais estão representadas pela tabela a seguir respectivamente pelas células demarcadas em verde e vermelho.

	0 0	0 1	0 2	0 3	
	1 0	1 1	1 2	1 3	
	2 0	2 1	2 2	2 3	
	3 0	3 1	3 2	3 3	

Os valores presentes na tabela representam a linha e a coluna, por exemplo na primeira célula temos os valores 0 0, tal que o primeiro zero representa a linha e o segundo a coluna.

Na diagonal secundária observamos que a soma dos índices sempre dão o mesmo valor, e isso se repete em todas as outras diagonais paralelas à ela. Com este padrão conseguimos associar qualquer posição do elemento à sua posição em um vetor que guardará a soma. Podemos observar que existem 7 diagonais em uma matriz 4x4, ou seja, a quantidade de linhas mais colunas menos 1.

Através desta análise podemos associar cada célula com uma posição do vetor em que armazenará a soma dos valores.

De forma análoga podemos observar a diagonal principal, temos que a diferença dos valores, coluna menos linha, sempre dão no mesmo valor. Vale observar que se pegarmos sempre o valor da coluna menos linha, na parte triangular superior da matriz este valor será positivo, e no triângulo inferior será negativo.

Com isso temos um padrão para identificar a posição no vetor, tanto da principal quanto da secundária.

Vamos analisar a célula 0 (linha) 1 (coluna). Para encontrarmos o índice da diagonal secundária basta somar os valores, 0 e 1, resultando no índice 1, como discutido anteriormente. Temos então que associar o índice 1 no vetor das “diagonais principais”, se realizarmos a diferença de coluna menos a linha também obtemos o valor 1, ou seja, valor positivo que nos fornecerá exatamente a posição no vetor. Vale observar que isso se repete para todos os valores das diagonais na parte superior da matriz.

Entretando isso não ocorre nas diagonais paralelas abaixo da diagonal principal. Vamos analisar a célula 1 (linha) 0 (coluna). Temos como resultado o índice de valor 1 na diagonal secundária e -1 para a diagonal principal seguindo a

mesma lógica, porém o índice a ser encontrado não deverá ser 1 para igual no caso da célula anteriormente analisada. Ao analisar a célula anterior identificamos que as posições no vetor já estão ocupados nos valores 0 à 3, então, ao recebemos o valor negativo dessa diferença, pegarmos o valor absoluto e somar o valor da quantidade de colunas existentes na matriz menos um e conseguimos o índice 4 no exemplo da célula 1 0. Vejamos que isso se repete em todas as outras diagonais paralelas.

Assim conseguimos uma fórmula fechada:

- Secundária: basta somar os índices, linha e coluna.
- Primária: basta realizar a diferença dos índices, coluna menos linha. Caso seja maior ou igual à zero, basta acessar a posição no vetor, caso contrário, pegamos o valor absoluto e somamos à quantidade de colunas menos 1 e teremos o índice correto.

3 - **Solução**

Para encontrar a maior soma basta passar por cada célula da matriz, realizar a soma das diagonais, subtrair uma vez o valor da célula, pois este valor está duplicado na soma, e por fim guardar o maior valor da soma a partir da célula com algum anterior encontrado anteriormente.

A seguir está a proposta da solução na linguagem C++:

```
#include <bits/stdc++.h>

using namespace std;

int main() {

  int Q; cin >> Q;
  for (int T = 0; T < Q; T++) {
    int L, C; cin >> L >> C;
    int P[L+C-1] = {0};
    int S[L+C-1] = {0};
    int m[L][C];

    for (int i = 0; i < L; i++) {
      for (int j = 0; j < C; j++) {
        cin >> m[i][j];
        if(j-i >= 0)
          P[j-i] += m[i][j];
        else
          P[C-1 + abs(j-i)] += m[i][j];
        S[i+j] += m[i][j];
      }
```

```
    }
    int bigger = 0;

    for (int i = 0; i < L; i++)  {
      for (int j = 0; j < C; j++) {
        int res = 0;
        res += S[i+j];
        res -= m[i][j];
        if(j-i >= 0)
          res += P[j-i];
        else
          res += P[C-1 + abs(j-i)];
        bigger = max(bigger, res);
      }
    }
    cout << bigger << '\n';
  }
  return 0;
}
```