El salto del caballo X29235_es

Tenemos un tablero de tamaño $m \times n$ donde hay un caballo como los de ajedrez colocado en la posición (i, j). Por ejemplo, consideremos el caso particular $m = 4$, $n = 5$, $i = 1$ y $j = 1$. Cuando el caballo todavía no se ha movido, marcamos con 1 su posición y con 0 el resto de posiciones. Así, representamos esta configuración inicial con la matriz:

```
00000
01000
00000
00000
```

Ahora, permitimos hacer al caballo un salto (o movimiento). Recordad que si un caballo está en la posición (i, j) del tablero, en un salto puede acceder a las posiciones $(i + a, j + b)$ donde $a \neq 0$ y $b \neq 0$ y $\mathtt{a} + \mathtt{b} = 3$ siempre que sean posiciones válidas del tablero. Por tanto, en un salto puede acceder a lo sumo a ocho posiciones diferentes, las que sean válidas entre $(i+1, j+2)$, $(i+1, j-2)$, $(i-1, j+2)$, $(i-1, j-2)$ y también entre $(i+2, j+1)$, $(i+2, j-1)$, $(i-2, j+1)$ y $(i-2, j-1)$. En nuestro ejemplo, si marcamos con un 2 las posiciones a las que puede acceder queda la siguiente configuración:

```
00020
01000
00020
20200
```

Simulando un nuevo movimiento de caballo (el segundo) , marcamos con un 3 las nuevas posiciones a las que puede acceder (y no ha podido visitar antes) desde cualquiera de las marcadas con un 2. Obtenemos:

```
00323
01030
30323
23200
```

Con otro movimiento (el tercero) marcamos con un 4 las nuevas posiciones a las que puede acceder. Tenemos:

```
04323
41434
34323
23204
```

Con un salto más (el cuarto):

```
54323
41434
34323
23254
```

y ésta es la configuración final porque ningún salto de caballo adicional permite visitar una posición nueva. Notad que una configuración final puede tener ceros (es decir, posiciones sin visitar). Por ejemplo, en el tablero 2×7 con posición inicial del caballo $(0, 0)$ la configuración final es:

```
1 0 0 0 3 0 0
0 0 2 0 0 0 4
```

Se pide un programa para calcular la configuración final del tablero a partir de la información del tamaño del tablero y de la posición inicial del caballo. El programa se ofrece abajo casi completo a falta de la acción `move_update`. Vuestra tarea es completar la siguiente especificación y proponer un código que la resuelva. Se recomienda que uséis las funciones o acciones auxiliares oportunas para lograr un código legible y de buena calidad.

```
// Pre: tab es configuracion del tablero cuando el caballo ha hecho k-1 saltos
//      k >= 1 indica que se ha de simular el k-esimo salto.
//
// Post: tab es la configuracion  del tablero cuando se ha hecho el k-esimo sal
//       testigo es true si tab ha cambiado y false en caso contrario
void move_update(... tab, ... k, ... testigo)
```

Por ejemplo, si `tab` es el tablero:

```
04323
41434
34323
23204
```

y hacemos `move_update(tab, 4, testigo)` obtenemos como nuevo tablero:

```
5 4 3 2 3
4 1 4 3 4
3 4 3 2 3
2 3 2 5 4
```

y el valor de `testigo` es `true` porque han habido cambios en `tab`. Si ahora hiciesemos `move_update(tab, 5, testigo)` entonces `tab` no cambiaría y `testigo` sería `false`, indicando que que esa es la configuración final del tablero y no hay mas cambios.

```
#include <iostream>
#include <vector>
using namespace std;


typedef vector<int> Fila;
typedef vector<Fila> Tablero;


//
//   UNA O MAS FUNCIONES O ACCIONES
//   SON NECESARIAS AQUI
```

```
//
//


void escribir_tablero(const Tablero& tab) {
    int m = tab.size();
    int n = tab[0].size();
    for (int i = 0; i < m; ++i) {
        cout << tab[i][0];
        for (int j = 1; j < n; ++j) cout << ' ' << tab[i][j];
        cout << endl;
    }
}


//inicializa el tablero tab a cero
void set_zero(Tablero& tab) {
    int m = tab.size();
    int n = tab[0].size();
    for (int i = 0; i < m; ++i)
        for (int j = 0; j < n; ++j)
            tab[i][j] = 0;
}


int main() {
    int m;
    while (cin >> m) {
        int n;
        cin >> n;
        Tablero tab(m, Fila(n));
        set_zero(tab);
        int i, j;
        cin >> i >> j;
        tab[i][j] = 1;
        bool testigo = true;
        int k = 1;
        while (testigo) {
            move_update(tab, k, testigo);
            ++k;
        }
        escribir_tablero(tab);
        cout << endl;
    }
}
```

Puntos examen: 2.500000 **Parte automática:** 30.000000%

Entrada

La entrada es una secuencia de casos. Cada caso consta de cuatro números. Los dos primeros m y n describen respectivamente el número de filas y columnas del tablero y son ambos mayores que cero. Los dos últimos i y j detallan la posición inicial del caballo en el tablero. Siempre es $0 \leq i < m$ y $0 \leq j < n$.

Salida

Para cada caso, el valor del tablero de saltos en su configuración final cuando no puede haber más cambios seguido de una línea en blanco.

Ejemplo de entrada

```
4 5 1 1

3 4 1 1

2 7 0 0
```

Ejemplo de salida

```
5 4 3 2 3
4 1 4 3 4
3 4 3 2 3
2 3 2 5 4

5 4 3 2
4 1 6 5
5 4 3 2

1 0 0 0 3 0 0
0 0 2 0 0 0 4
```

Observación

La caracterización del enunciado de que en un salto se puede acceder a las posiciones $(i + a, j + b)$ donde $a \neq 0$ y $b \neq 0$ y $\mathtt{a} + \mathtt{b} = 3$ siempre que sean posiciones válidas del tablero se puede aprovechar para evitar una instrucción alternativa de exploración con muchos casos.

Información del problema

Autor :
Generación : 2020-06-16 17:36:17