El salt del cavall X29235_ca

Tenim un tauler de mida $m \times n$ on hi ha un cavall com els d'escacs col·locat en la posició (i, j). Per exemple, considerem el cas particular $m = 4$, $n = 5$, $i = 1$ i $j = 1$. Quan el cavall encara no s'ha mogut, marquem amb 1 la seva posició i amb 0 la resta de posicions. Així, representem aquesta configuració inicial amb la matriu:

```
00000
01000
00000
00000
```

Ara, permetem fer a el cavall un salt (o moviment). Recordeu que si un cavall està en la posició (i, j) del tauler, en un salt pot accedir a les posicions $(i + a, j + b)$ on $a \neq 0$ i $b \neq 0$ i $\mathtt{a} + \mathtt{b} = 3$ sempre que siguin posicions vàlides del tauler. Per tant, en un salt pot accedir a tot estirar a vuit posicions diferents, les que siguin vàlides entre $(i + 1, j + 2)$, $(i + 1, j - 2)$, $(i - 1, j + 2)$, $(i - 1, j - 2)$ i també entre $(i + 2, j + 1)$, $(i + 2, j - 1)$, $(i - 2, j + 1)$ i $(i - 2, j - 1)$. En el nostre exemple, si marquem amb un 2 les posicions a les que pot accedir queda la següent configuració:

```
00020
01000
00020
20200
```

Simulant un nou moviment de cavall (el segon), marquem amb 3 les noves posicions a les que pot accedir (i no ha pogut visitar abans) des de qualsevol de les marcades amb un 2. Obtenim:

```
00323
01030
30323
23200
```

Amb un altre moviment (el tercer) marquem amb 4 les noves posicions a les que pot accedir. Tenim:

```
04323
41434
34323
23204
```

Amb un salt més (el quart):

```
54323
41434
34323
23254
```

i aquesta és la configuració final perquè cap salt de cavall addicional permet visitar una posició nova. Fixeu-vos que una configuració final pot tenir zeros (és a dir, posicions sense visitar). Per exemple, en el tauler 2×7 amb posició inicial del cavall $(0, 0)$ la configuració final és:

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

Es demana un programa per calcular la configuració final del tauler a partir de la informació de la mida del tauler i de la posició inicial del cavall. El programa s'ofereix a baix gairebé complet mancant l'acció `move_update`. La vostra tasca és completar la següent especificació i proposar un codi que la resolgui. Es recomana que feu servir les funcions o accions auxiliars oportunes per aconseguir un codi llegible i de bona qualitat.

```
// 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 sa
//       testigo es true si tab ha cambiado y false en caso contrario
void move_update(... tab, ... k, ... testigo)
```

Per exemple, si `tab` és el tauler:

```
04323
41434
34323
23204
```

i fem `move_update(tab, 4, testigo)` obtenim com a nou tauler:

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

i el valor de `testigo` és `true` perquè han hagut canvis en `tab`. Si ara fessim `move_update(tab, 5, testigo)` llavors `tab` no canviaria i `testigo` seria `false`, indicant que que aquesta és la configuració final del tauler i no hi ha mes canvis.

```
#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;
    }
}
```

Punts examen: 2.500000 **Part automàtica:** 30.000000%

Entrada

L'entrada és una seqüència de casos. Cada cas consta de quatre nombres. Els dos primers m i n descriuen respectivament el nombre de files i columnes del tauler i són tots dos més grans que zero. Els dos últims i i j detallen la posició inicial del cavall al tauler. Sempre és $0 \leq i < m$ i $0 \leq j < n$.

Sortida

Per a cada cas, el valor del tauler de salts en la seva configuració final quan no hi pot haver més canvis seguit d'una línia en blanc.

Exemple d'entrada

```
4 5 1 1

3 4 1 1

2 7 0 0
```

Exemple de sortida

```
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
```

Observació

La caracterització de l'enunciat de que en un salt es pot accedir a les posicions $(i + a, j + b)$ on $a \neq 0$ i $b \neq 0$ i $\mathtt{a} + \mathtt{b} = 3$ sempre que siguin posicions vàlides del tauler es pot aprofitar per evitar una instrucció alternativa d'exploració amb molts casos.

Informació del problema

Autor :
Generació : 2020-06-16 17:36:14