3N1?M4 Weird Notebook

Contents

1 Cuadrados y Matrices

1.1 Cuadrado Magico de Tamaño Impar

```
/* Odd-Size Magic Square
 *
 * La suma de los enteros en cada columna, fila o diagonal es
 * la misma (n**2 + 1) * n/2
 *
 * N - numero de filas/columnas (N es par)
 *
 * Se implementa el metodo Siamese (De la Loubere)
 *
 *  */

vector<vector<int>> magic_square(int N){
     vector<vector<int>> ans(N);
     for(auto& i: ans)
         i = vector<int>(N);

     int i=0, j=N/2;
     for(int num=1; num<=N*N; num++){
         ans[i][j] = num;

         int ii = (i-1+N) % N;
         int jj = (j+1+N) % N;

         if( ans[ii][jj] )
            i = (i+1) % N;
         else
            i = ii, j = jj;
     }

     return ans;
}
```

1.2 El Juego del 15 Generalizado a $N^2 - 1$

```
/* Fifteen Puzzle
 *
 * Aqui las posiciones son codificadas como permutaciones
 * desde 1 hasta n**2-1 y INF, donde INF es la casilla vacia
 *
 * Dadas dos posiciones, se puede llegar de una a otra si
 * ambas tienen la misma paridad
 *
 * */

/* Para N>=1000 usar otro metodo */
int inversions(vector<int>& P){
   int64 ans = 0ll;
   for(int i=0; i < (int)P.size(); i++)
       for(int j=i+1; j < (int)P.size(); j++)
           if( P[i] > P[j] ) ans++;

   return ans;
}

bool are_reachable(vector<int>& A, vector<int>& B){
    assert( A.size() == B.size() );

    return inversions(A) + inversions(B) % 2 == 0;
}
```

1.3 Recorrido del Caballo

```
/* Knight's Tour
 *
 * Regla de Warnsdorf para obtener un recorrido completo del caballo
 * sobre el tablero de ajedrez de talla N*N
 *
 * - Devuelve una matriz 1-indexada con las casillas numeradas de
 *   1 a N^2 segun el recorrido obtenido. Este recorrido no es unico.
 *
 * */

#define POSSIBLE (X > 0 && X<=N && Y > 0 && Y<=N)
typedef vector<vector<int>> matrix;

const int dx[] = {-1, -2, -2, -1,  1, 2,  2,  1};
const int dy[] = {-2, -1,  1,  2,  2, 1, -1, -2};

matrix warnsdorf(int N){
    matrix ans(N+1, vector<int>(N+1, 0));
    matrix C(N+1, vector<int>(N+1, 0));

    int x, y, X, Y, min_moves, min_x, min_y;

    for(int i=1; i<=N; i++) for(int j=1; j<=N; j++)
        for(int k=0; k < 8; k++){
            X = i + dx[k];
            Y = j + dy[k];

            if( POSSIBLE ) C[i][j]++;
    }

    x = 1, y = 1;
    for(int i=1; i<=N * N; i++){
        ans[x][y] = i;
        min_moves = INF;

        for(int j=0; j < 8; j++){
            X = x + dx[j];
            Y = y + dy[j];

            if( !POSSIBLE ) continue;

            C[X][Y]--;

            if( !ans[X][Y] && min_moves > C[X][Y] )
                min_x = X, min_y = Y, min_moves = C[X][Y];
        }

        x = min_x, y = min_y;
    }
    return ans;
}
```