Last update: June 8, 2022 Translated From: e-maxx.ru

Placing Bishops on a Chessboard

Find the number of ways to place K bishops on an $N \times N$ chessboard so that no two bishops attack each other.

Algorithm

This problem can be solved using dynamic programming.

Let's enumerate the diagonals of the chessboard as follows: black diagonals have odd indices, white diagonals have even indices, and the diagonals are numbered in non-decreasing order of the number of squares in them. Here is an example for a 5×5 chessboard.

$$\begin{matrix} \bf{1} & 2 & \bf{5} & 6 & \bf{9} \\ 2 & \bf{5} & 6 & \bf{9} & 8 \\ \bf{5} & 6 & \bf{9} & 8 & \bf{7} \\ 6 & \bf{9} & 8 & \bf{7} & 4 \\ \bf{9} & 8 & \bf{7} & 4 & \bf{3} \\ \end{matrix}$$

Let `D[i][j]` denote the number of ways to place `j` bishops on diagonals with indices up to `i` which have the same color as diagonal `i`. Then `i = 1...2N-1` and `j = 0...K`.

We can calculate `D[i][j]` using only values of `D[i-2]` (we subtract 2 because we only consider diagonals of the same color as i). There are two ways to get `D[i][j]`. Either we place all `j` bishops on previous diagonals: then there are `D[i-2][j]` ways to achieve this. Or we place one bishop on diagonal `i` and `j-1` bishops on previous diagonals. The number of ways to do this equals the number of squares in diagonal `i` minus `j-1`, because each of `j-1` bishops placed on previous diagonals will block one square on the current diagonal. The number of squares in diagonal `i` can be calculated as follows:

```
int squares (int i) {
    if (i & 1)
        return i / 4 * 2 + 1;
    else
        return (i - 1) / 4 * 2 + 2;
}
```

The base case is simple: `D[i][0] = 1`, `D[1][1] = 1`.

Once we have calculated all values of `D[i][j]`, the answer can be obtained as follows: consider all possible numbers of bishops placed on black diagonals `i=0...K`, with corresponding numbers of bishops on white diagonals `K-i`. The bishops placed on black and white diagonals never attack each other, so the placements can be done independently. The index of the last black diagonal is `2N-1`, the last white one is `2N-2`. For each `i` we add `D[2N-1][i] * D[2N-2][K-i]` to the answer.

Implementation

```
int bishop_placements(int N, int K)
{
    if (K > 2 * N - 1)
        return 0;

    vector<vector<int>> D(N * 2, vector<int>(K + 1));
    for (int i = 0; i < N * 2; ++i)
        D[i][0] = 1;
    D[1][1] = 1;
    for (int i = 2; i < N * 2; ++i)
        for (int j = 1; j <= K; ++j)
            D[i][j] = D[i-2][j] + D[i-2][j-1] * (squares(i) - j + 1);

    int ans = 0;
    for (int i = 0; i <= K; ++i)
        ans += D[N*2-1][i] * D[N*2-2][K-i];
    return ans;
}
```

Contributors:

tcNickolas (53.42%) jakobkogler (38.36%) adamant-pwn (8.22%)