688. Knight Probability in Chessboard

Difficulty:Medium

Total Accepted:2.4K

Total Submissions:6.5K

Contributor: 1337c0d3r

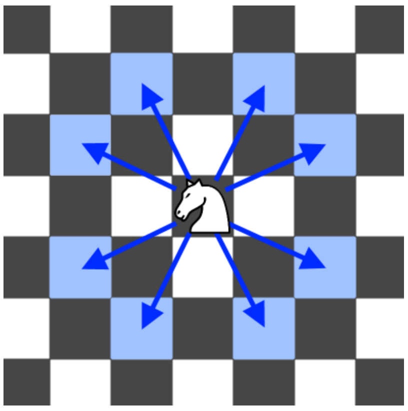
Related Topics: Dynamic Programming

Similar Questions

Out of Boundary Paths

On an `NxN` chessboard, a knight starts at the `r`-th row and `c`-th column and attempts to make exactly `K` moves. The rows and columns are 0 indexed, so the top-left square is `(0, 0)`, and the bottom-right square is `(N-1, N-1)`.

A chess knight has 8 possible moves it can make, as illustrated below. Each move is two squares in a cardinal direction, then one square in an orthogonal direction.



Each time the knight is to move, it chooses one of eight possible moves uniformly at random (even if the piece would go off the chessboard) and moves there.

The knight continues moving until it has made exactly `K` moves or has moved off the chessboard. Return the probability that the knight remains on the board after it has stopped moving.

Example:

```
Input: 3, 2, 0, 0
Output: 0.0625
Explanation: There are two moves (to (1,2), (2,1)) that will keep the knight on
the board.
```

```
From each of those positions, there are also two moves that will keep the knight
on the board.
The total probability the knight stays on the board is 0.0625.
```

Note:

- `N` will be between 1 and 25.
- `K` will be between 0 and 100.
- The knight always initially starts on the board.

#solution : let dp[r][c] be the probability moves for r,c after k steps
for each valid moves mv in set {{1,2},{1,-2},{-1,2},{-1,-2},{2,1},{2,-1},{-2,1},{-2,-1}};
dp[r][c] = dp[r][c] + dp[r+mv][c+mv]

```
//c++
#include<iostream>
#include<stdio.h>
#include<vector>
#include<math.h>
using namespace std;

bool isInBoard(int x,int y, int r,int c)
{
	return x>=0 && x<r && y>=0 && y<c;
}

double knightProbability(int N, int K, int r, int c) {
	vector<vector<int>> moves = {{1,2},{1,-2},{-1,2},{-1,-2},{2,1},{2,-1},{-
2,1},{-2,-1}};
	vector<vector<double>> dp(N,vector<double>(N,1));
	for(int k=0;k<K;k++)
	{
		vector<vector<double>> dp1(N,vector<double>(N,0));
		for(int i=0;i<N;i++)
		{
			for(int j=0;j<N;j++)
			{
				for(auto elem:moves)
				{
					int xx = i+elem[0];
					int yy = j+elem[1];
					if(isInBoard(xx,yy,N,N))
					{
						dp1[i][j] += dp[xx][yy];
					}
				}
			}
		}
		dp=dp1;
	}

	double ans = dp[r][c]/(pow(8,K));
	return ans;
}

int main(int argc,char *argv[])
{
	double ans = knightProbability(8,30,6,4);
```

```
    cout << ans << endl;
    return 0;
}
```