Problem 688: Knight Probability in Chessboard

Problem Information

Difficulty: Medium
Acceptance Rate: 56.81%
Paid Only: No
Tags: Dynamic Programming

Problem Description

On an `n x n` chessboard, a knight starts at the cell `(row, column)` and attempts to make exactly `k` moves. The rows and columns are **0-indexed** , so the top-left cell is `(0, 0)`, and the bottom-right cell is `(n - 1, n - 1)`.

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

![](https://assets.leetcode.com/uploads/2018/10/12/knight.png)

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 1:

Input: n = 3, k = 2, row = 0, column = 0 **Output:** 0.06250 **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.

Example 2:

Input: n = 1, k = 0, row = 0, column = 0 **Output:** 1.00000

Constraints:

* `1 <= n <= 25` * `0 <= k <= 100` * `0 <= row, column <= n - 1`

Code Snippets

C++:

```
class Solution {
public:
double knightProbability(int n, int k, int row, int column) {

}
};
```

Java:

```
class Solution {
public double knightProbability(int n, int k, int row, int column) {

}
}
```

Python3:

```
class Solution:
def knightProbability(self, n: int, k: int, row: int, column: int) -> float:
```