A knight is a chess piece which can legally move from a square \((i, j)\) on a chessboard (where \(i\) is the row index, and \(j\) is the column index) to any of the eight squares \((i - 1, j - 2), (i - 1, j + 2), (i + 1, j - 2), (i + 1, j + 2), (i - 2, j - 1), (i - 2, j + 1), (i + 2, j - 1), (i + 2, j + 1)\), as long as they are on the chessboard.

You are interested in finding knight’s tours of various \(n \times m\) gameboards. A knight’s tour is a sequence of squares from the gameboard so that each square appears exactly once in the sequence, and each square is a legal knight’s move from its previous square. A tour is closed if there is a legal knight’s move from the last square of the sequence back to the first square. Otherwise, the tour is open.

You will study a heuristic for this problem. A heuristic (without backtracking) will usually find a tour, but may occasionally fail to find a tour. Given a partial tour, you will try to lengthen the tour by adding the next possible square with lowest degree. A square is possible if it is not already in the partial tour, and it is a legal knight’s move from the last square in the partial tour. The degree of a square \((i, j)\) is the number of squares that are reachable using a single knight’s move, and which are not already in the partial tour. Notice that each time you add a new square to the tour, you must decrease the degrees of some squares. To begin the construction, you will pick some square as your initial square, making a partial tour of length one.

1. Use this heuristic to write a program to find knight’s tours. The inputs should be \(n\) and \(m\), the number of rows and columns on the board, and \((i, j)\), the starting square. (For simplicity, you will only need to do boards where \(n = m\).) The output should be an array such that the first square contains 1, the last square contains \(nm\), and the \(k^{th}\) square contains the number \(k\). If your program fails to find a tour, it should print out the partial tour it found and a message saying that it failed to find a full tour.

2. Test your program by using as starting square each of the 25 squares on a 5 × 5 gameboard.

3. Does your program always find a tour? Are the tours you found open or closed? How many different tours does your program find?

4. Run your program from 4 different initial squares on a 6 × 6 gameboard. Does your program always find a tour? Are the tours you found open or closed? How many different tours does your program find?

5. Run your program from 4 different initial squares on a 8 × 8 gameboard. Does your program always find a tour? Are the tours you found open or closed? How many different tours does your program find?