T28050: 骑士周游

dfs, http://cs101.openjudge.cn/practice/28050/

代码：

```python

import sys

def is\_valid\_move(x, y, board, n):

return 0 <= x < n and 0 <= y < n and board[x][y] == -1

def get\_degree(x, y, board, n, moves):

count = 0

for dx, dy in moves:

if is\_valid\_move(x + dx, y + dy, board, n):

count += 1

return count

def knights\_tour\_warnsdorff(n, sr, sc):

moves = [(2, 1), (1, 2), (-1, 2), (-2, 1),

(-2, -1), (-1, -2), (1, -2), (2, -1)]

board = [[-1 for \_ in range(n)] for \_ in range(n)]

board[sr][sc] = 0

def backtrack(x, y, move\_count):

if move\_count == n \* n:

return True

#罗列所有可能的路径并按照warnsdorff排序

next\_moves = []

for dx, dy in moves:

nx, ny = x + dx, y + dy

if is\_valid\_move(nx, ny, board, n):

degree = get\_degree(nx, ny, board, n, moves)

next\_moves.append((degree, nx, ny))

next\_moves.sort() # 按 Warnsdorff 规则选择最少可行移动的方向#1.减少出口少的子到后面无法被访问的可能性，从而减少回溯2.平衡选择：这种方法平衡了棋盘上各个区域的访问机会，防止骑士过早地被困在某个区域。

for \_, nx, ny in next\_moves:

board[nx][ny] = move\_count

if backtrack(nx, ny, move\_count + 1):

return True

board[nx][ny] = -1 # 回溯

return False

if backtrack(sr, sc, 1):

print("success")

else:

print("fail")

if \_\_name\_\_ == "\_\_main\_\_":

n = int(sys.stdin.readline().strip())

sr, sc = map(int, sys.stdin.readline().strip().split())

knights\_tour\_warnsdorff(n, sr, sc)