主要使用solveKTUtil()解決問題

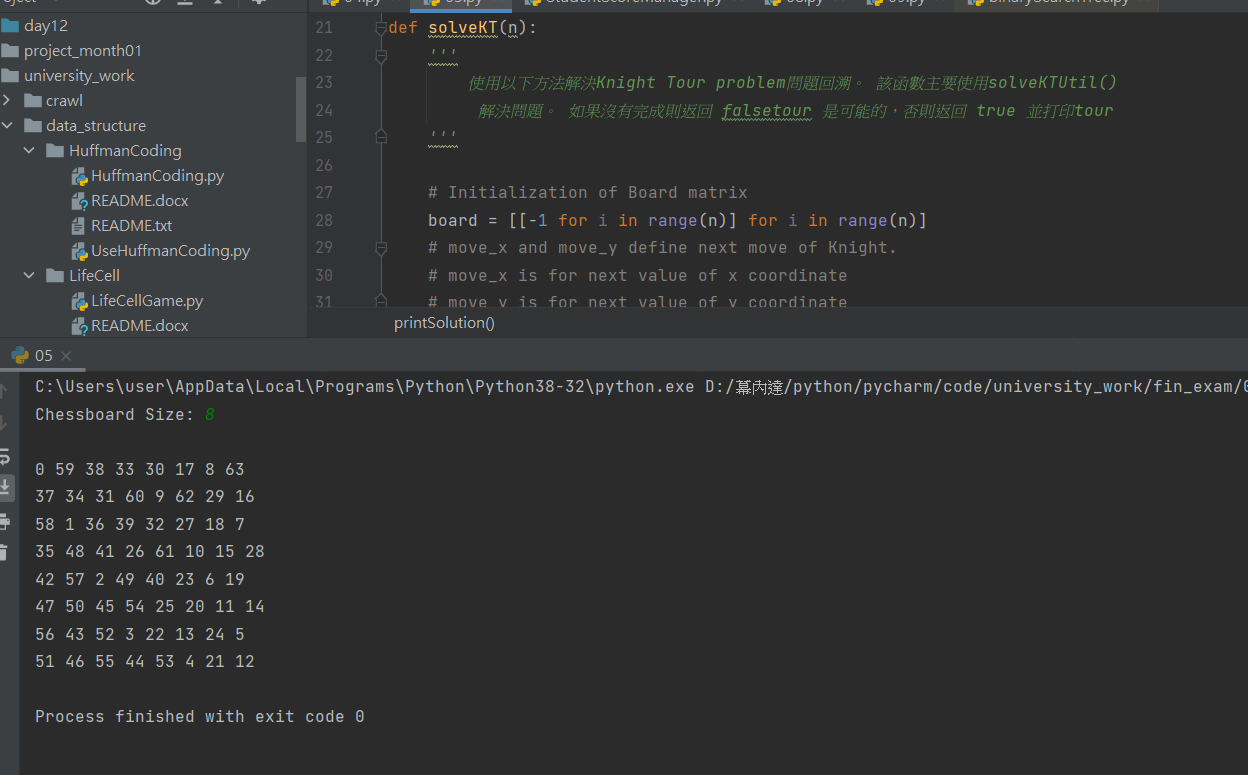
使用8個方向代表馬走的方向

move\_x = [2, 1, -1, -2, -2, -1, 1, 2]  
move\_y = [1, 2, 2, 1, -1, -2, -2, -1]

使用board = [[-1 for i in range(n)] for i in range(n)]

讓紀錄馬走路的行況

最後出來的結果(如果要跑的話要等一夏，因為計算蠻多的:



程式碼:

def isSafe(x, y, board):  
 *'''  
 A utility function to check if i,j are valid indexes  
 for N\*N chessboard  
 '''* if (x >= 0 and y >= 0 and x < n and y < n and board[x][y] == -1):  
 return True  
 return False  
  
  
def printSolution(n, board):  
 *'''  
 A utility function to print Chessboard matrix  
 '''* for i in range(n):  
 for j in range(n):  
 print(board[i][j], end=' ')  
 print()  
  
  
def solveKT(n):  
 *'''  
 使用以下方法解決Knight Tour problem問題回溯。 該函數主要使用solveKTUtil()  
 解決問題。 如果沒有完成則返回 falsetour 是可能的，否則返回 true 並打印tour  
 '''* # Initialization of Board matrix  
 board = [[-1 for i in range(n)] for i in range(n)]  
 # move\_x and move\_y define next move of Knight.  
 # move\_x is for next value of x coordinate  
 # move\_y is for next value of y coordinate  
 move\_x = [2, 1, -1, -2, -2, -1, 1, 2]  
 move\_y = [1, 2, 2, 1, -1, -2, -2, -1]  
  
 # Since the Knight is initially at the first block  
 board[0][0] = 0  
  
 # Step counter for knight's position  
 pos = 1  
  
 # Checking if solution exists or not  
 if (not solveKTUtil(n, board, 0, 0, move\_x, move\_y, pos)):  
 print("Solution does not exist")  
 else:  
 printSolution(n, board)  
  
  
def solveKTUtil(n, board, curr\_x, curr\_y, move\_x, move\_y, pos):  
 *'''  
 A recursive utility function to solve Knight Tour  
 problem  
 '''* if (pos == n \*\* 2):  
 return True  
  
 # Try all next moves from the current coordinate x, y  
 for i in range(8):  
 new\_x = curr\_x + move\_x[i]  
 new\_y = curr\_y + move\_y[i]  
 if (isSafe(new\_x, new\_y, board)):  
 board[new\_x][new\_y] = pos  
 if (solveKTUtil(n, board, new\_x, new\_y, move\_x, move\_y, pos + 1)):  
 return True  
  
 # Backtracking  
 board[new\_x][new\_y] = -1  
 return False  
  
  
n = int(input("Chessboard Size: "))  
# Function Call  
solveKT(n)