<< **Python Data Structure** & **NumPy** Project >>

Knight's Tour Problem by "Dynamic" WARNSDORFF Algorithm

< 前 言 >

從圖論 (Graph Theory) 角度而言，Knight's Tour (騎士旅程問題) 是一個 Hamiltonian Path / Circle Problems (漢彌爾頓路徑 或 漢彌爾頓循環 的問題)。通常，以 Depth-first Search algorithm (深度優先搜尋演算法) 來求解在 8x8 西洋棋盤上的騎士旅程問題。求解過程中，當路徑搜尋遇到 dead-end 的問題時，會採用 backtracking (回溯) 方式解決該問題。從程式設計的角度，這將會利用 "遞迴" (Recursion) 演算法來協助實作 backtracking。

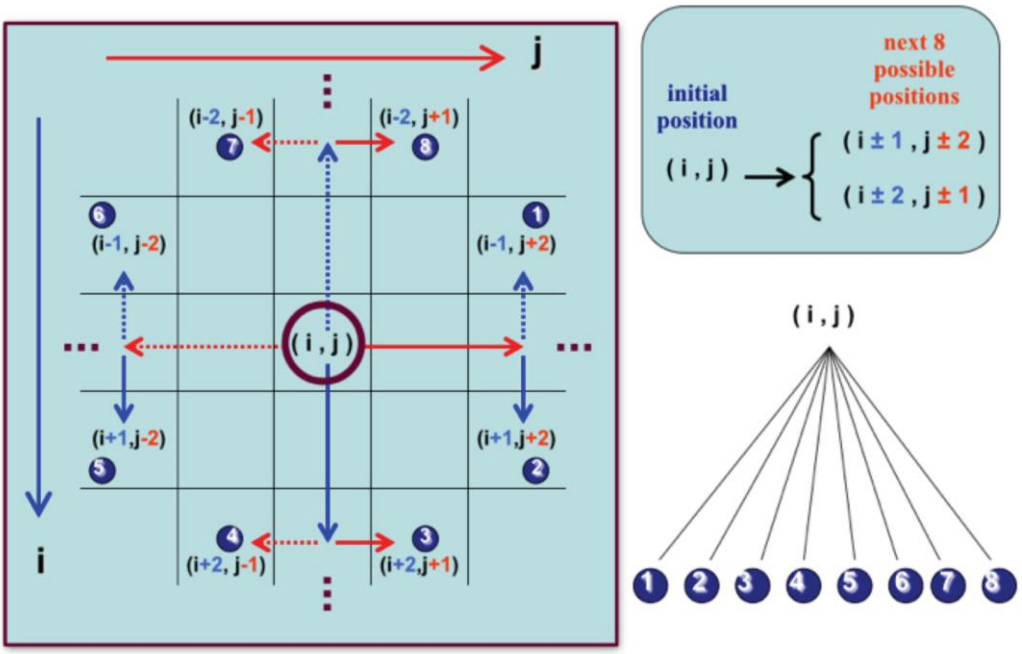
一般而言，上述的搜尋計算效能不高，因此，Warnsdorff (1823) 提出一套 "規則" (亦即 演算法)，有助於提升搜尋 Hamiltonian Path 的效能。雖然如此，"靜態" Warnsdorff rules 仍然無法避免搜尋時可能遇到 dead-end，亦即 必須 "回溯" (backtracking) 搜尋。

本專案旨在利用 "動態 (dynamic)" Warnsdorff 演算法，藉由動態更新各棋盤格點的 degree 值，來協助避開搜尋時遇到 dead-end 的問題；在無需回溯的情況下，快速求解騎士旅程問題。同時，將演算法擴增、求解 NxN 棋盤的搜尋問題 (其中，N > 8)。

[有關騎士旅程問題 — 簡述]

今有 8x8 的西洋棋棋盤，若將騎士的第一步放於棋盤中的任一位置，請利用 "動態" 的 Warnsdorff 演算法撰寫程式，找出騎士在 64 步內，依據：「每個方格只能走一次，不得重複！」方式，將棋盤全部方格走完。

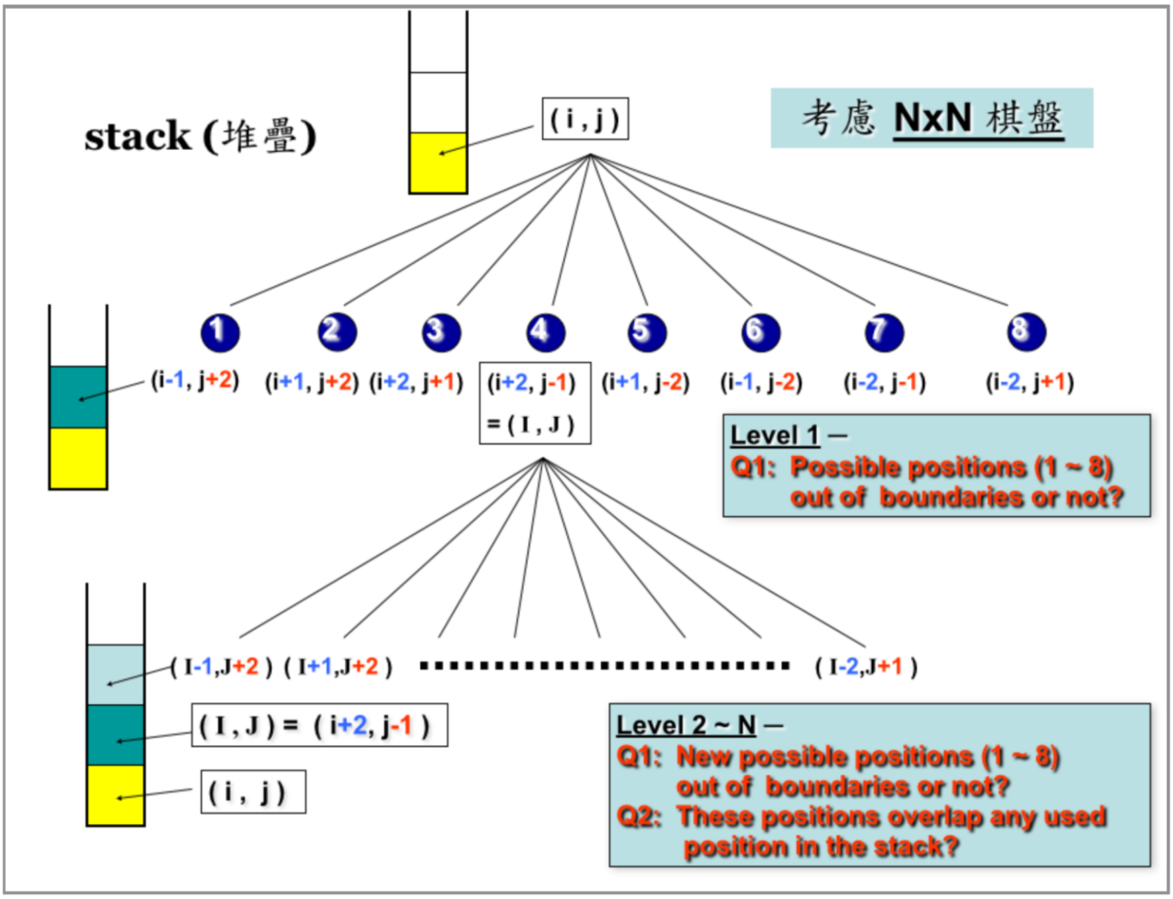
- **Depth-first Search (DFS) Algorithm with Backtracking**



initial position

next 8 possible positions

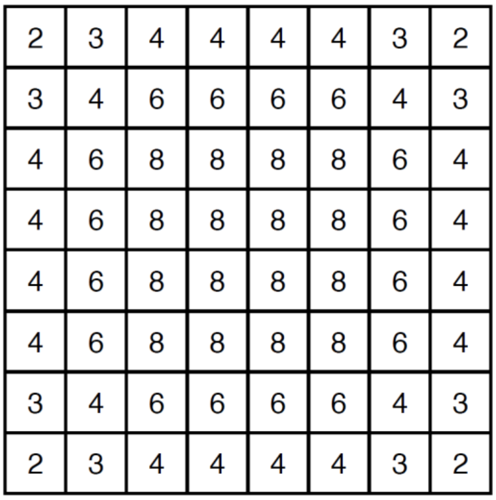
(i , j) → { (i ± 1 , j ± 2) (i ± 2 , j ± 1) }


stack (堆疊)
(i , j)
考慮 NxN 棋盤
1 2 3 4 5 6 7 8
(i-1, j+2) (i+1, j+2) (i+2, j+1) (i+2, j-1) = (I , J) (i+1, j-2) (i-1, j-2) (i-2, j-1) (i-2, j+1)
(I-1,J+2) (I+1,J+2) .... (I-2,J+1)
(I , J) = (i+2, j-1)
(i , j)

Level 1 –
Q1: Possible positions (1 ~ 8) out of boundaries or not?

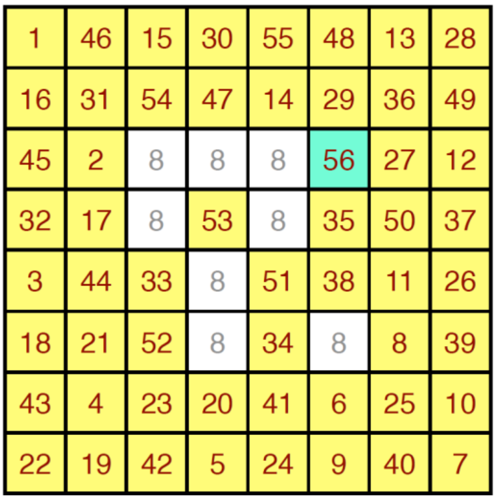
Level 2 ~ N –
Q1: New possible positions (1 ~ 8) out of boundaries or not?
Q2: These positions overlap any used position in the stack?

- **Warnsdorff Rules**



2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
3	4	6	6	6	6	4	3
2	3	4	4	4	4	3	2

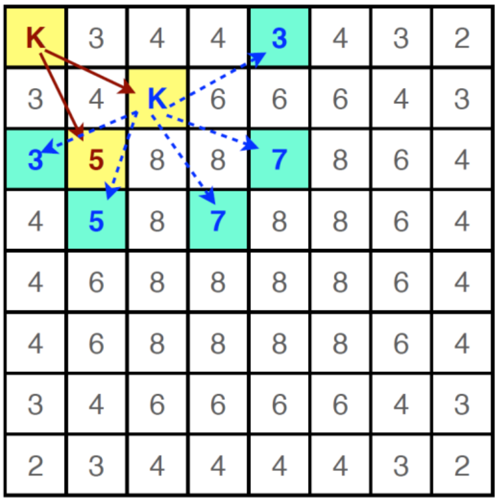
degree map (static)



1	46	15	30	55	48	13	28
16	31	54	47	14	29	36	49
45	2	8	8	8	56	27	12
32	17	8	53	8	35	50	37
3	44	33	8	51	38	11	26
18	21	52	8	34	8	8	39
43	4	23	20	41	6	25	10
22	19	42	5	24	9	40	7

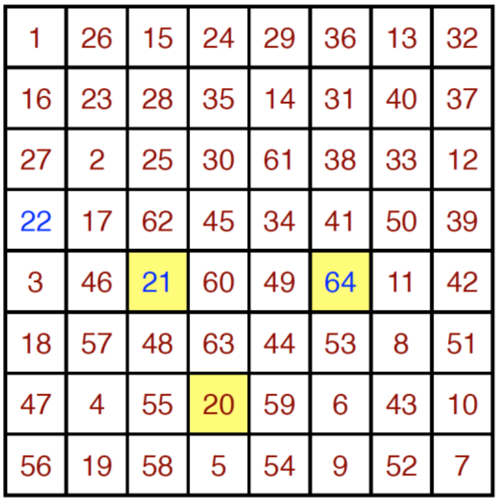
dead-end situation

(without backtracking)



K	3	4	4	3	4	3	2
3	4	K	6	6	6	4	3
3	5	8	8	7	8	6	4
4	5	8	7	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
3	4	6	6	6	6	4	3
2	3	4	4	4	4	3	2

degree map (dynamic)



1	26	15	24	29	36	13	32
16	23	28	35	14	31	40	37
27	2	25	30	61	38	33	12
22	17	62	45	34	41	50	39
3	46	21	60	49	64	11	42
18	57	48	63	44	53	8	51
47	4	55	20	59	6	43	10
56	19	58	5	54	9	52	7

Hamiltonian Path

(without backtracking)

REFERENCE

1. **"Knight's Tour"**, Wikipedia. https://en.wikipedia.org/wiki/Knight%27s_tour

2. D. Squirrel and P. Cull, **"A Warnsdorff-Rule Algorithm for Knight's Tours on Square Chessboards"**, PDF, 1996. http://math.oregonstate.edu/~math_reu/proceedings/REU_Proceedings/Proceedings1996/1996Squirrel.pdf

3. **"KNIGHT'S TOUR USING WARNSDORFF ALGORITHM (PYTHON RECIPE)"**, http://code.activestate.com/recipes/578382-knights-tour-using-warnsdorff-algorithm/

[ALGORITHM] : Knight's Tour by “Dynamic” WARNSDORFF Algorithm

< NOTE > : (1) Using Python’s data structures : `list, tuple, set, dictionary`, and ***functions*** to implement the algorithm;

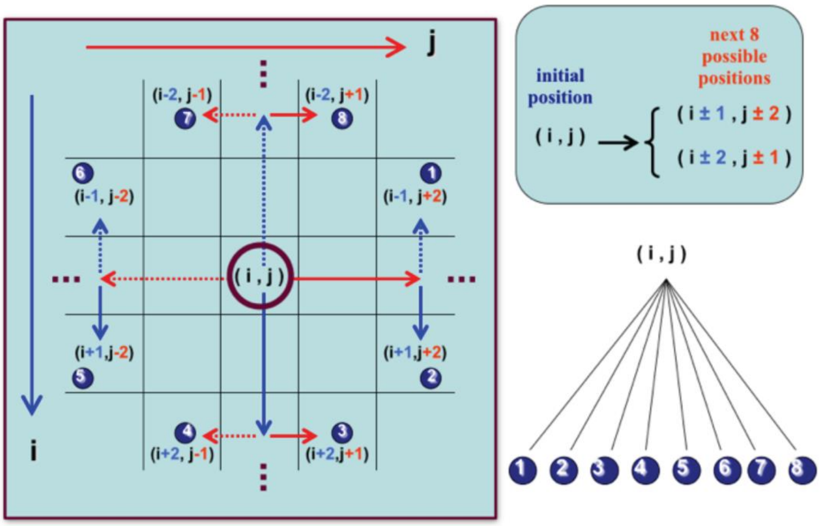
(2) Using **NumPy extension library** to implement the algorithm with Jupyter Notebook.

< Warnsdorff algorithm with dynamic degree-updating for 8x8 Chessboard >

STEP 1 : Creating a degree map for an 8x8 chessboard.

```
degree_map = [2,3,4,4,4,4,3,2,
              3,4,6,6,6,6,4,3,
              4,6,8,8,8,8,6,4,
              4,6,8,8,8,8,6,4,
              4,6,8,8,8,8,6,4,
              4,6,8,8,8,8,6,4,
              3,4,6,6,6,6,4,3,
              2,3,4,4,4,4,3,2]
```

STEP 2 : Creating the 8 possible moves.



STEP 3 : Initiating the **start position** of Knight.

[Note] : It could be chosen at ***any*** start position for the Knight.

STEP 4 : **Looping** for finding the Hamiltonian Path for Knight’s Tour.

(1) *Checking if the moves within the board boundaries or not.*

(2) *Finding the next position for Knight's movement.*

(3) *Updating the degree map and the new move.*

STEP 5 : **Print out** the Hamiltonian Path for Knight’s Tour; *for example*:

```
 1 26 15 24 29 36 13 32
16 23 28 35 14 31 40 37
27  2 25 30 61 38 33 12
22 17 62 45 34 41 50 39
 3 46 21 60 49 64 11 42
18 57 48 63 44 53  8 51
47  4 55 20 59  6 43 10
56 19 58  5 54  9 52  7
```

< NOTE > :

A Possible Solution with Python for the Warnsdorff algorithm above for an 8x8 Chessboard can be downloaded from the following address:

https://drive.google.com/file/d/1QwLkM8M9-kXYf95QyR5u53XWj7M9wXR-/view?usp=sharing

[Problem] : A Solution for Knight’s Tour with *N*x*N* Chessboard ($8 \leq N \leq 30$)

— Following the **STEPs** above to solve the Knight’s Tour problem for an *arbitrary* *N*x*N* chessboard, where $8 \leq N \leq 30$.

— **[NOTE] :** If *N* is an ***odd*** number, there exist the solutions ***only*** for Knight’s start positions at $(i,\ j)$ where the sum of *i* and *j is equal to an even number*.

[Requirement] : Your Python code for the Solution should be accomplished at most 50 statements (*if possible, make it less than 20 statements*).

Good luck !!