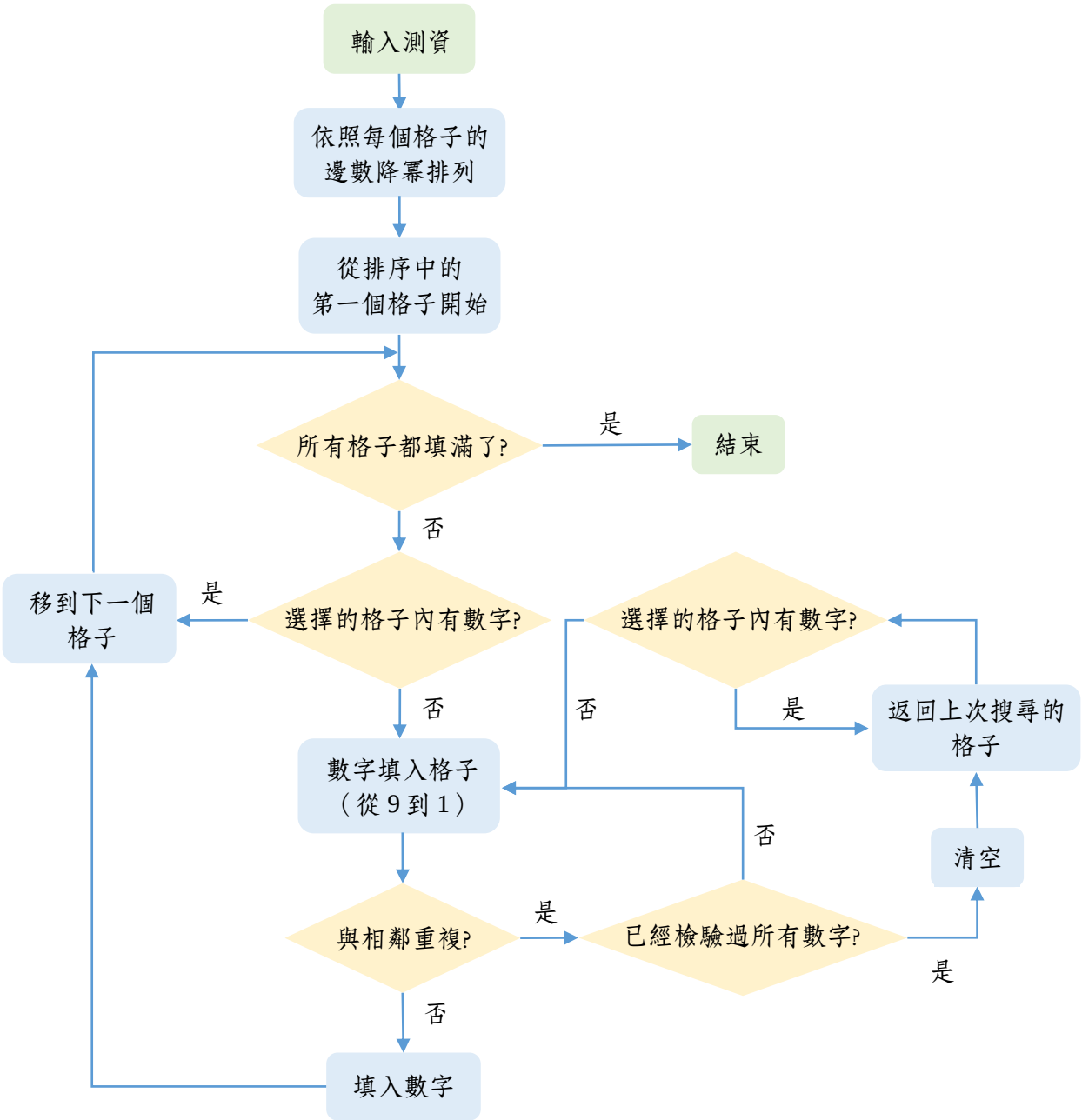
一、 題目：Sudoku (for groups)

（一）解決問題定義

給定一個數獨，在每個空格中填入 1~9 的數字，使 1~9 每個數字在每一行、每一列和每一宮都只出現一次。

（二）所使用之演算法



把每一個格子視為一個節點，把與其同行、同列、同宮之間（其他 20 個節點），視為有邊相連，所以不能填入相同的數字。

在解決問題前，我把問題分成三部分：

（1） 選擇下一個要填的格子：

在輸入測資後，紀錄對於每個格子相連的節點已經使用過的數字總數（重複不算），存進一個二維陣列，再依照總數降冪排列，得到下一個要填的格子位置的順序。

（2） 判斷可以填的數字：

在輸入測資時，宣告三個二維陣列，並存入每行、每列、每宮已經出現過的數字，例如在第 i 行已經出現過數字 j，則表示成 row[i][j]=1，後來在判斷格子是否能填某個數字時，就用這三個陣列判斷可以填入的數字。

（3） 選擇要填的數字：

格子能填入的數字可能多於一個，當填入的數字與後續的搜尋違背時，回溯到格子的上一層，繼續搜尋未被搜尋過的數字，這個過程反覆進行，直到數獨裡的所有格子都已經填入數字，代表搜尋完成。

（三）結果

因為題目要求用圖論的方法解數獨，所以我從圖論遍歷法中的深度搜尋著手，一開始我只想到從第一格開始從左到右、從上而下搜尋（圖一），後來想到用節點著色的方法，計算與每個節點相連的其他節點，已經使用過的顏色總數，降冪排列就是比較容易填入的格子順序，按照這個順序搜尋，得到比單純用暴力法解更好的結果（圖二）。

後來，我發現搜尋時，格子裡的數字如果逆著搜（從 9 到 1），程式的運行時間又會比正著搜（從 1 到 9）更少，這或許是因為測資的緣故。

（1）暴力法正搜：

#	Problem	Submitter	Time	Memory	Verdict	Execute	Length	Score	Submit Time
29822	350. 1.Sudoku (for groups)	0513311	184	3212	AC	C++	1396	100	2017-12-25 21:06:12

（2）著色正搜：

#	Problem	Submitter	Time	Memory	Verdict	Execute	Length	Score	Submit Time
34091	350. 1.Sudoku (for groups)	0513311	20	3280	AC	C++	2608	100	2018-01-07 20:25:55

（3）暴力法逆搜：

#	Problem	Submitter	Time	Memory	Verdict	Execute	Length	Score	Submit Time
32119	350. 1.Sudoku (for groups)	0513311	16	3284	AC	C++	1396	100	2018-01-01 22:54:34

（4）著色逆搜：

#	Problem	Submitter	Time	Memory	Verdict	Execute	Length	Score	Submit Time
38093	350. 1.Sudoku (for groups)	0513311	4	3244	AC	C++	2652	100	2018-01-18 18:16:43

在程式裡，我只用較直觀的基礎摒除法，假如使用高階的數獨解法，例如區塊摒除或鍊數刪減，或許會讓解題效率更好。

（四）團隊分工

0513311 資工 09 羅文慧

（五）程式建置環境

C++

（六）參考文獻

（1） DFS wiki：

https://zh.wikipedia.org/wiki/%E6%B7%B1%E5%BA%A6%E4%BC%98%E5%85%88%E6%90%9C%E7%B4%A2

（2） 數獨技巧：

https://baike.baidu.com/item/%E6%95%B0%E7%8B%AC%E6%8A%80%E5%B7%A7

二、 題目：Triangles and connected diameter problem (for individuals)

（一）解決問題定義

（1）Search for Triangles：

在無向圖中，找出所有由兩兩相鄰的三個點，組成的三角形個數。

（2）Search for Diameter：

在無向圖中，找出所有點與點間最短路徑，其中最長的距離。

（二）所使用之演算法

（1）Search for Triangles：

把 n 個節點之間的連接關係以矩陣 m1 呈現(m1[i][j]==0 代表點 i 到點 j 沒有長度為一的路徑， m1[i][j]==1 代表點 i 到點 j 有一條長度為一的路徑)，可以得到 m1 是一個 n×n 的對稱矩陣。

根據矩陣相乘的性質，可以得到一個 m2=m1×m1 的新矩陣，其中 m2[i][j]==0 代表點 i 到點 j 沒有長度為二的路徑， m2[i][j]==k 代表點 i 到點 j 有 k 條長度為二的路徑，依此類推，再求出一個 m3=m1×m2 的矩陣。

題目要求得到圖中所有三角形的個數，m3 是計算路徑長度為三的矩陣，三角形的個數＝m3 的每項 diagonal 相加，由於三角形每個頂點都會被計算到，所以要除以三，且有依照頂點順/逆時針的兩種計算方式，所以要再除以二，才是圖的三角形總數。

（2）Search for Diameter：

根據 D-Matrix 演算法，只要找出 m1 的 k 次方，使 s 除了 diagonal 之外全部不為零，就可以得知 k 為 diameter。

宣告整數變數 count 紀錄程式運行中 m1 的次方數，為了減少記憶體的使用，在之後的計算中，m1 矩陣保持不變，如果 count 是奇數，則把新求出的矩陣乘積存進 m2 矩陣；如果 count 是偶數，則把新求出的矩陣存進 m3 矩陣裡。

接著找出 m(count)中不為零且 s 為零的格子，再把他們以 count 取代，重複以上步驟，直到 s 所有格子(除了 diagonal)都不為零，那麼 count 就是圖的 diameter。

（三）自己想的演算法

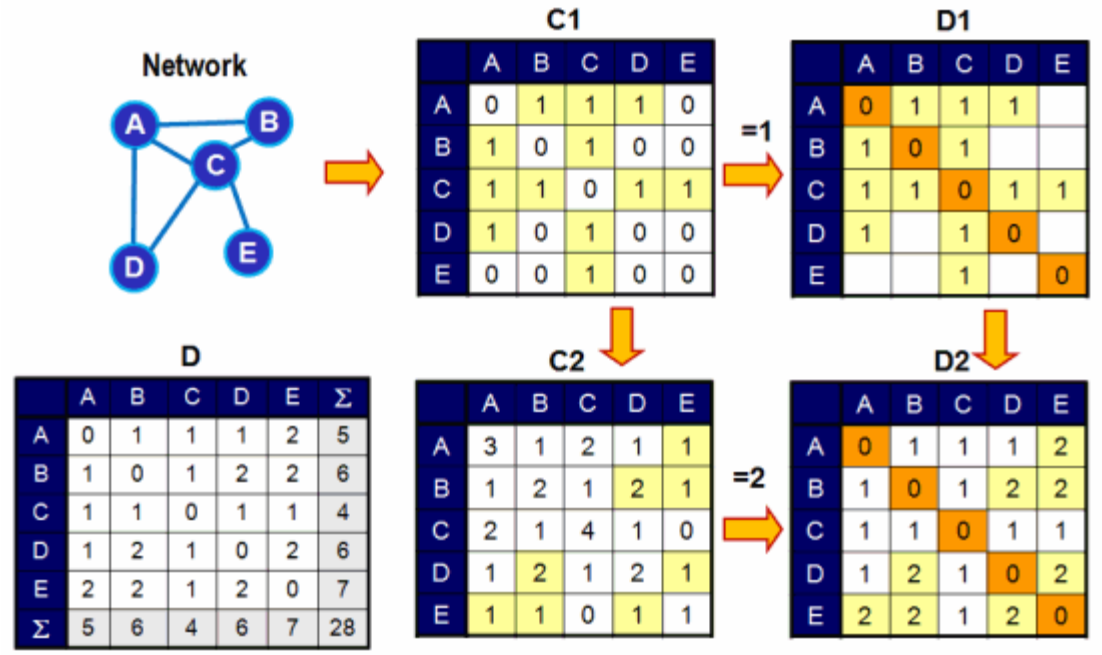
Shimbel Distance Matrix (D-Matrix)：

D-Matrix 紀錄了圖中每對節點間的最短路徑，其中最長的最短路徑（出現在 D-Matrix 的最大值）即是 diameter。

求出 D-Matrix（以 s[i][j]宣告）的步驟：

1. 矩陣 mk 是 m1 的 k 次方。
2. 紀錄在 mk 裡非零且在 s 裡為零的格子（不包含 diagonal 項）。
3. 以 k（次方數）取代 s 矩陣裡那些被標記的格子。
4. 重複步驟一到步驟三，直到 s 矩陣裡所有項（不包含 diagonal）都不等於零。

（如下圖）



	A	B	C	D	E	Σ
A	0	1	1	1	2	5
B	1	0	1	2	2	6
C	1	1	0	1	1	4
D	1	2	1	0	2	6
E	2	2	1	2	0	7
Σ	5	6	4	6	7	28

D-Matrix 的 diagonal 項永遠為零，因為在計算最短路徑時，環（cycle）是多餘的，如果把環的邊數算進路徑，則此路徑必然不是最短路徑。

（四）結果

剛開始看到題目時，本來想用 vector 來處理節點，後來發現用矩陣比較容易處理資料，於是上網找了許多線性代數的文章，發現了一些之前在學線性代數時，不曾在課本裡遇見的問題，與許多有趣的解法。

例如 D-Matrix 演算法，特色是用來計算網路的 diameter，在計算點 i 與點 j 的最短路徑時，並不是檢驗 i 到 j 所有的路徑，而是只檢驗 i 到 j 最短路徑的步數。這個演算法在處理大量的資料時，勢必減少許多計算時間。

（五）團隊分工

0513311 資工 09 羅文慧

（六）程式建置環境

C++

（七）參考文獻

（1）矩陣乘法 wiki：

https://zh.wikipedia.org/wiki/%E7%9F%A9%E9%99%A3%E4%B9%98%E6%B3%95

（2）Some interesting properties of adjacency matrices：

https://1stprinciples.wordpress.com/2008/03/30/some-interesting-properties-of-adjacency-matrices/

（3）TRIANGLE COUNTING IN LARGE NETWORKS：

http://www.d.umn.edu/math/Technical%20Reports/Technical%20Reports%202007-/TR%202012/yang.pdf

（4）Shimbel Distance Matrix (D-Matrix)：

https://people.hofstra.edu/geotrans/eng/methods/shimbelmatrix.html

（5）Transportation Network Analysis - Shimbel Index：

http://webspace.ship.edu/pgmarr/TransMeth/Lec%203-Shimbel%20Distance.pdf