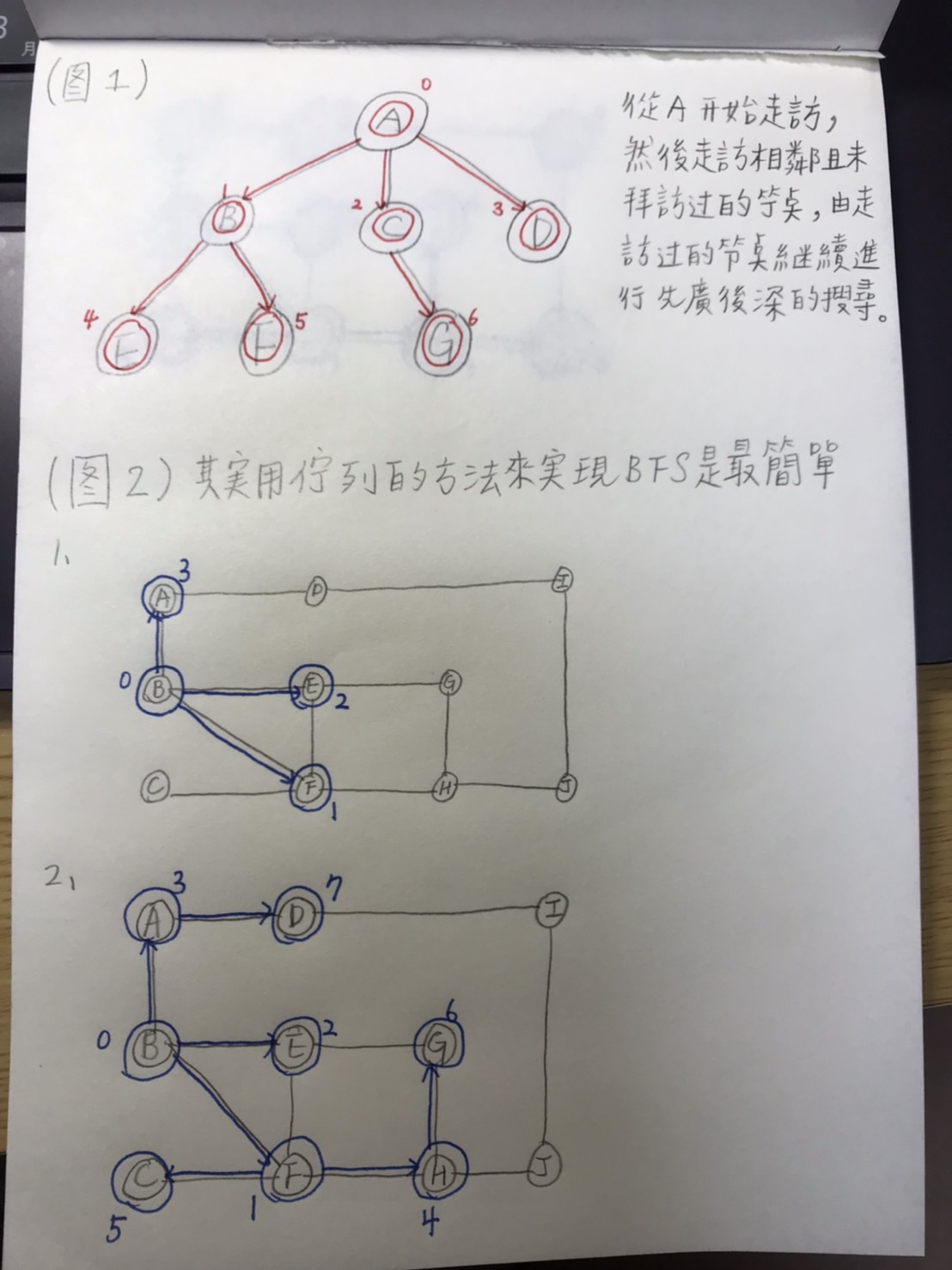
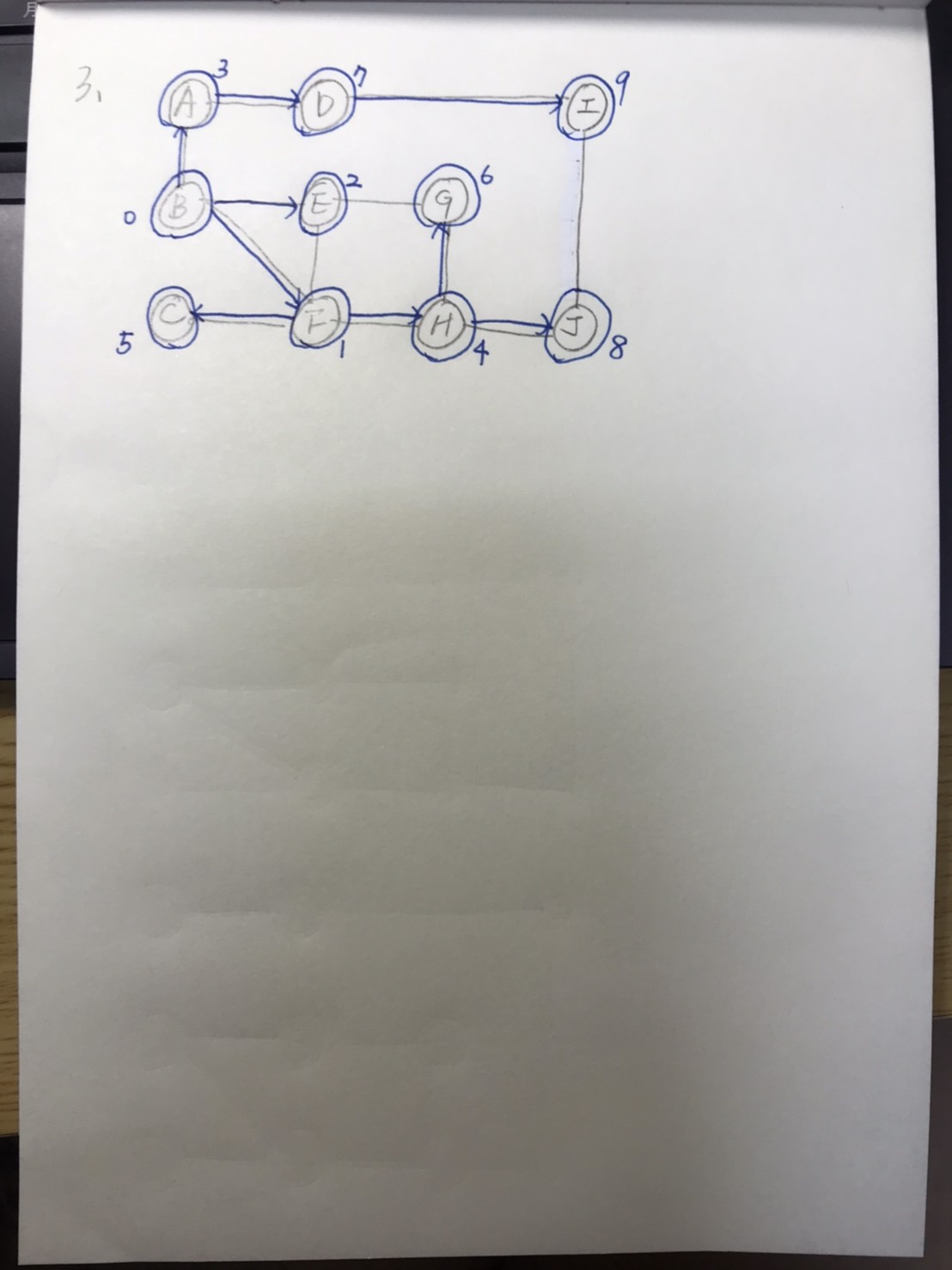
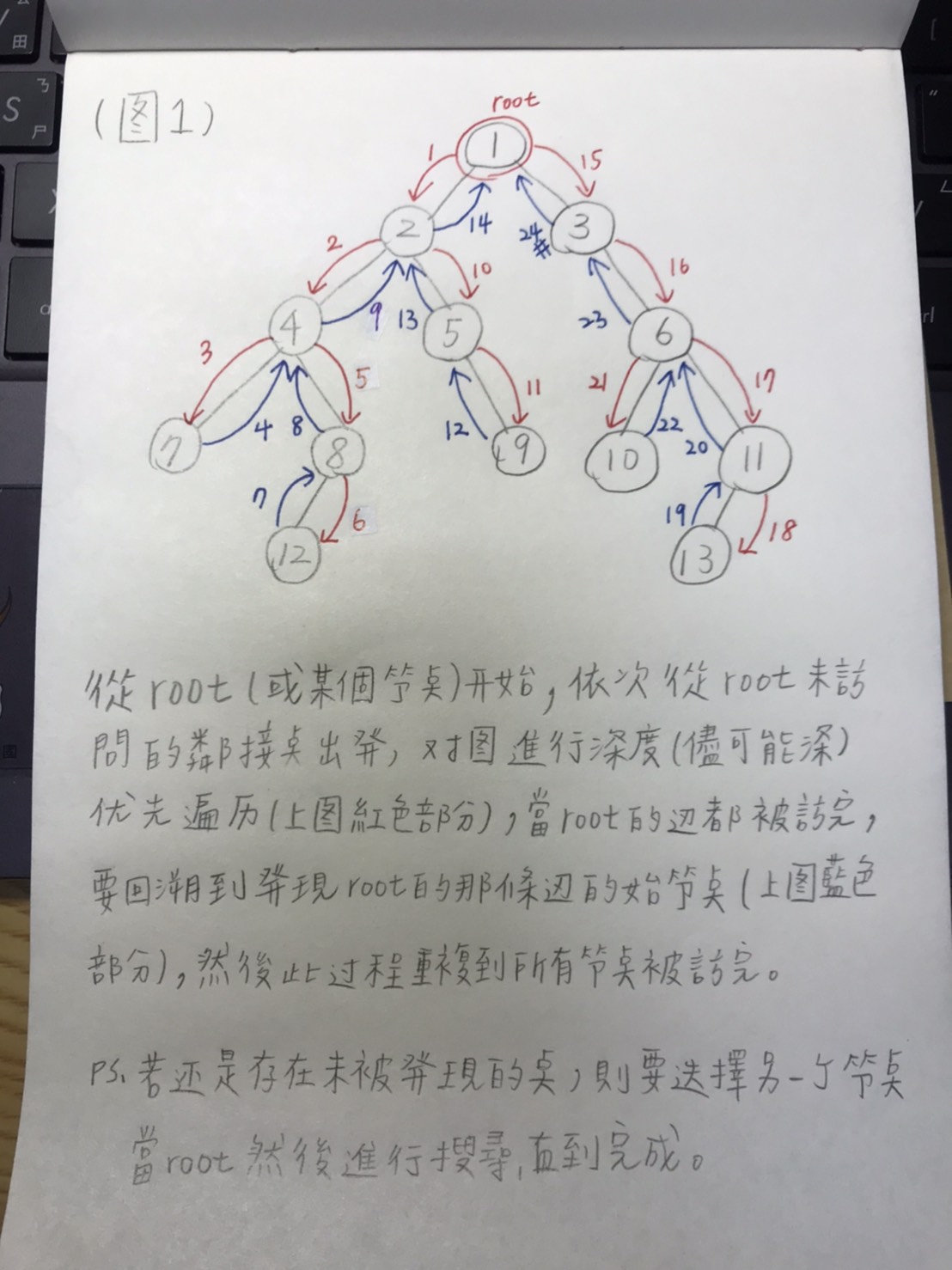
BFS與DFS流程圖、程式碼學習歷程與BFS與DFS原理與比較

【BFS流程圖】





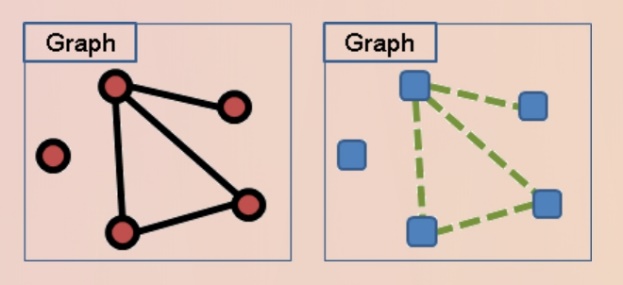
【DFS流程圖】



【BFS與DFS原理與比較】

* BFS原理

Breadth-First-Search 廣度優先搜尋演算法，又可以稱為寬度優先搜尋法或是橫向修先搜尋，是一種圖形搜尋演算法。講Breadth-First-Search之前要先了解甚麼是圖形搜尋演算法，這樣對BFS的原理才好理解。

圖形搜尋演算法([Graph Traversal](http://www.csie.ntnu.edu.tw/~u91029/Graph.html#5))，雖然中文翻作圖，但不是我們平常所想像的那種圖，而是由數個點和數個邊所構成的一張圖，點與點之間是由邊連接，表示點之間是有關係的。(像下圖)

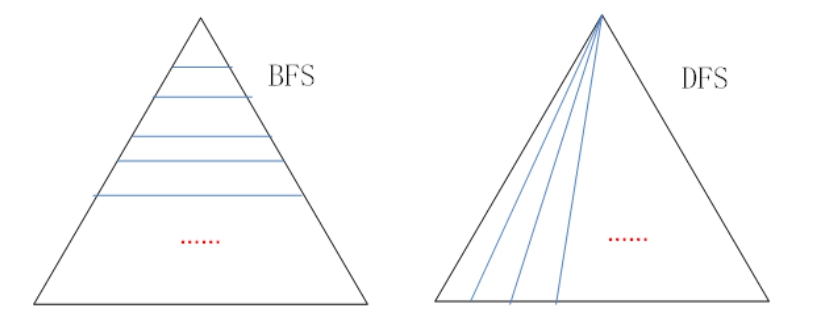
BFS類似於二交叉樹的層序遍歷演算法，其實他的基本原理很簡單，是依靠沿著樹(圖)的寬度遍歷樹的節點，首先我們挑一個起始節點v，然後從v開始，依序訪問v的鄰近還未被訪問的點w1,w2,w3…wn，然後在從w1,w2,w3…wn開始訪問鄰近還未被訪問的點。然後再從這些訪問過的點去接著訪問那些訪問過的點的鄰近點，就這樣一直下去，直到每個點都被訪問完。BFS就是把同一深度(level)的節點走訪完，再繼續向下一個深度搜尋，直到找到目的節點或遍尋全部節點。(參照我上面的流程圖)。BFS屬於盲目搜索(uninformed search)是利用佇列(Queue)來處理，通常以迴圈的方式呈現。

* DFS原理

Depth-First-Search深度優先搜尋演算法，是一種遍歷樹或圖形搜尋的演算法。DFS的原理其實也很簡單，是依靠沿著樹(圖)的深度遍歷樹的節點，儘可能深的搜尋樹的分支。當起始節點v的所在邊都己被探尋過，要回溯到發現節點v的那條邊的起始節點。這個過程會一直進行到已發現從源節點可達的所有節點為止。如果還存在未被發現的節點，則選擇其中一個作為源節點並重複以上過程，整個行程反覆進行直到所有節點都被訪問完為止。屬於盲目搜尋。(參照我上面DFS流程圖)

* BFS與DFS與比較

BFS和DFS最大的差異就是他們走訪的方式不同，BFS是用同階層然後寬度在深度，而DFS是一直往深度的方向走訪，沒東西後才回朔。



* 時間複雜度：兩個都是O(|V|+|E|)， v為圖的頂點數，E為邊數。
* 空間複雜度：BFS是O(V)，DFS是O(logV)。
* 效能：能用DFS解決的問題，都能用BFS，不過DFS比BFS佳。
* 占比：BFS通常用佇列方式所以占比會比較大，也就是占記憶體空間較大，而DFS比較小。
* 解決的問題：DFS一般是解決連通性問題，BFS一般是解決最短路徑問題。

REF:

1. 吾居吾束

<http://simonsays-tw.com/web/DFS-BFS/BreadthFirstSearch.html>

1. ITREAD(BFS)

<https://www.itread01.com/content/1541841612.html>

1. [**Jason Chen's Blog**](https://jason-chen-1992.weebly.com/)

<https://jason-chen-1992.weebly.com/home/-graph-searching-methods>

1. ITREAD(DFS)

<https://www.itread01.com/content/1546369025.html>

1. <http://www.csie.ntnu.edu.tw/~u91029/Graph.html#5>
2. <https://zh.wikipedia.org/wiki/%E6%B7%B1%E5%BA%A6%E4%BC%98%E5%85%88%E6%90%9C%E7%B4%A2>
3. <https://zh.wikipedia.org/wiki/%E5%B9%BF%E5%BA%A6%E4%BC%98%E5%85%88%E6%90%9C%E7%B4%A2#%E6%99%82%E9%96%93%E8%A4%87%E9%9B%9C%E5%BA%A6>
4. <https://super9.space/archives/1562>
5. <https://www.itread01.com/content/1502472128.html>