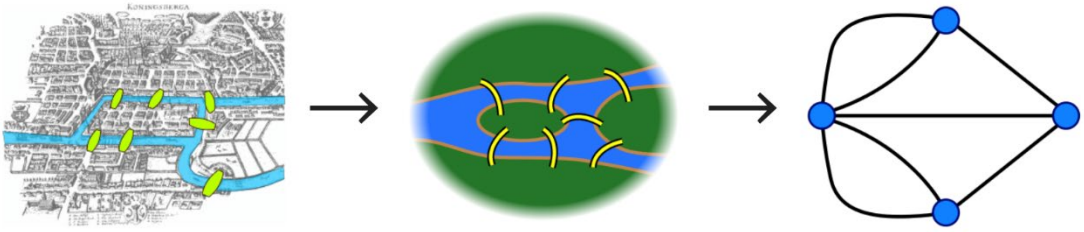
Graph Basics

1. 當我們把問題先「圖像化」再「抽象化」之後，可以將問題以下圖方式呈現。



而在將問題抽象化後，所謂的「七橋問題」，其實就是「一筆畫」

我們是否能找到一個方式只使用一筆就走完所有的線，且不重複呢？

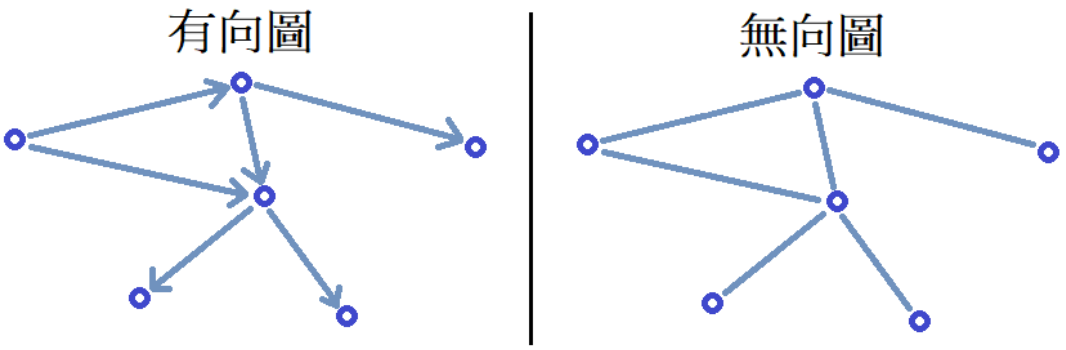
每條邊只訪問一次的情況下

給定一個連通圖，if 超過兩個以上有奇數通路的頂點，此圖便無法一筆走完。

if 兩個奇數通路頂點，最少走完此圖的路徑為個數的一半。

圖是很多節點 V 和邊 E 的集合，即可以表示為有序對 G=(V, E)。
V 是節點（vertex）E 是邊（edge）。**

圖片分為有向圖與無向圖



「圖形」可以用來表達**物體之間的關係**，所以圖形還蠻常用在計算機網路上面。我們可以幫每一個 Vertex 取一個代號，然後也可以幫每一個 edge 加上一個權重 (weights) 用來表示連線的 Cost，如此一來就可以很清楚的表達一個網路拓樸

並且我們就可以應用圖形演算法來解決一些問題

例如 找 兩點間成本最低的路徑 。

遍尋圖形的演算法

I. **深度優先搜尋法 Depth-first Search**

II. **廣度優先搜尋法 Breadth-first Search**

兩個最經典的圖形搜尋演算法有兩種「深度優先搜尋法」跟「廣度優先搜尋法」。

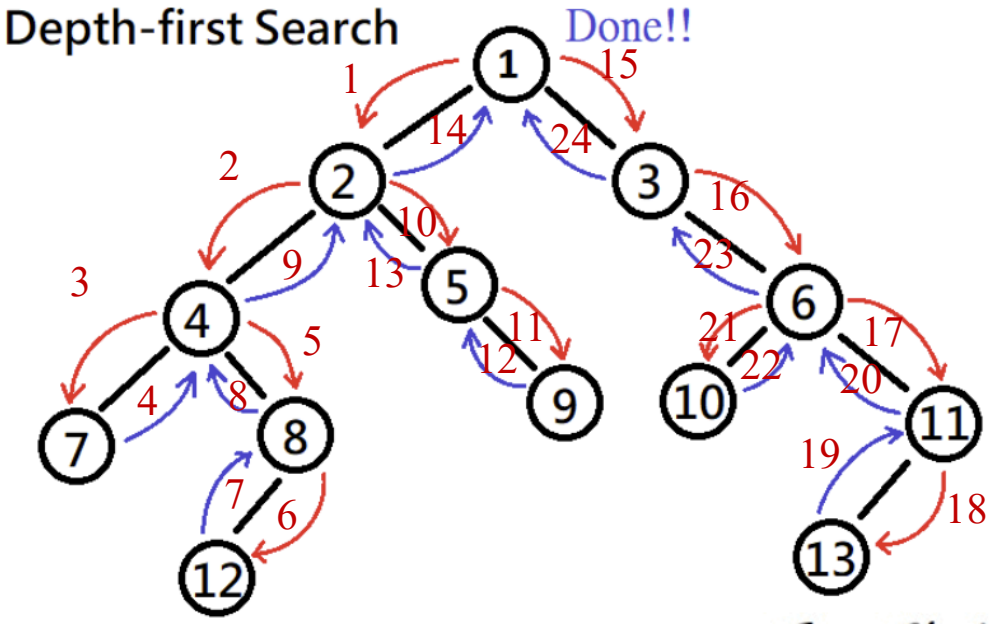
- **深度優先搜尋法 Depth-first Search**

  - 深度優先搜尋法，是一種用來遍尋圖形的演算法。
  - 資料結構常常使用到的「樹」其實是可以被視為是一種圖的。

以樹為例深度優先搜尋法的做法是：

由 root (或圖的某一點當成根) 開始探尋，先探尋邊 (edge) 上未搜尋的一節點 (vertex or node) ，並盡可能深的搜索，直到該節點的所有邊上節點都已探尋；就回 (backtracking) 到前一個節點，重覆探尋未搜尋的節點，直到找到目的節點或遍尋全部節點。

平均時間複雜度 $O(b^m)$
空間複雜度 O(bm)
b >> 分支係數　　m >> 圖的最大深度
最佳解 >> NO　　完全性 >> YES



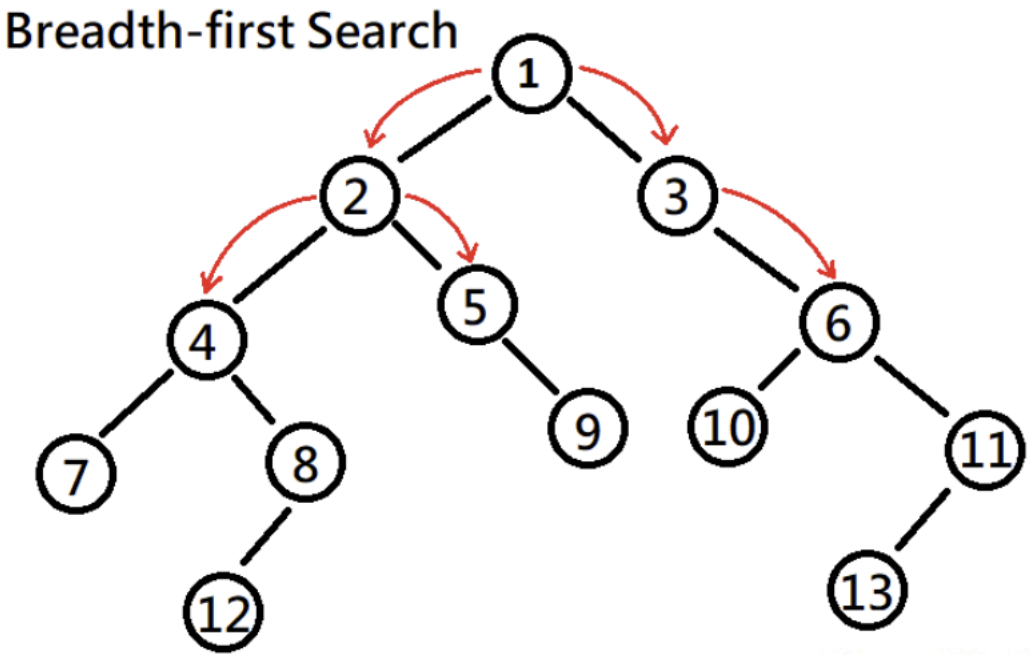
******節點的選擇是使用 LIFO 的方式管理，所以可以用 Stack 結構******

- **廣度優先搜尋法 Breadth-first Search**

➢ 廣度優先搜尋法，也是一種用來遍尋圖形的演算法。
➢ 也可用資料結構常常使用到的「樹」來解釋。

基本上這個演算法在實作的時候，就是靠 Queue 來完成。
看圖解釋的話 >>>
當 (1) 走訪到 (2) 的時候會先將 (2) 存在 queue 裡面，然後看看 (1) 還有沒有子節點；發現還有子節點 (3) 然後走訪到 (3) ，接著一樣將 (3) 存到 queue 再回頭看看 (1) 有沒有子節點；發現沒有了，於是就從 queue 裡面提取一個節點出來當作起點周而復始，直到 queue 裡面完全都沒有節點後結束。

平均時間複雜度 **O(|V| + |E|) = O(b^d)**
空間複雜度 **O(|V|) = O(b^d)**
最佳解 **>> YES** 完全性 **>> YES**



******節點的選擇是使用 FIFO 的方式管理，所以可以用 Queue 結構******

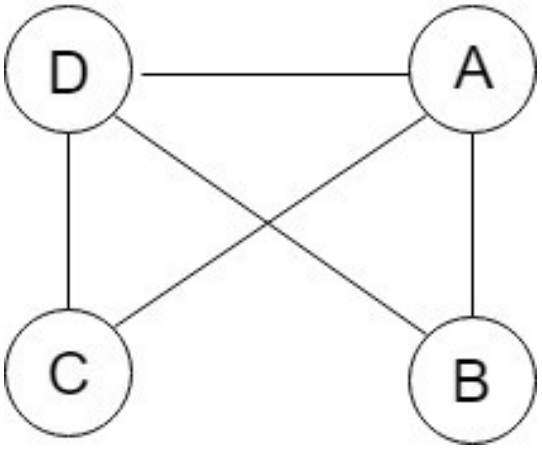
圖形的表示法

- 鄰接矩陣 Adjacency matrix
- 鄰接串列 Adjacency list

鄰接矩陣 Adjacency matrix

- 含有 n 個頂點的圖形，其 Adjacency matrix 的大小就會是 n 列 x n 行。
- matrix 中的元素只會是 0 或 1

若以下列無向邊為例的話



它的 Adjacency matrix 就會是下圖

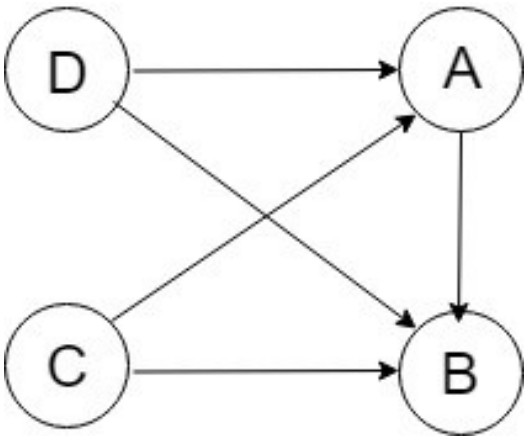
	A(0)	B(1)	C(2)	D(3)
A(0)	0	1	1	1
B(1)	1	0	0	1
C(2)	1	0	0	1
D(3)	1	1	1	0

4x4

上圖可整理出幾個重點！

- 矩陣的 1，可代表兩個頂點間有連線。
- 因自己不可能自成迴路，所以主對角線上的元素都會是 0。
- 無向圖的 Adjacency matrix 一定會是對稱矩陣。
- 將每一列的數值相加就是該頂點的分支度。
  EX： 第 0 列數值相加為 0+1+1+1 = 3，所以 A 點分支度為 3。

若以下列有向邊為例的話



它的 Adjacency matrix 就會是下圖

	A(0)	B(1)	C(2)	D(3)
A(0)	0	1	0	0
B(1)	0	0	0	0
C(2)	1	1	0	0
D(3)	1	1	0	0

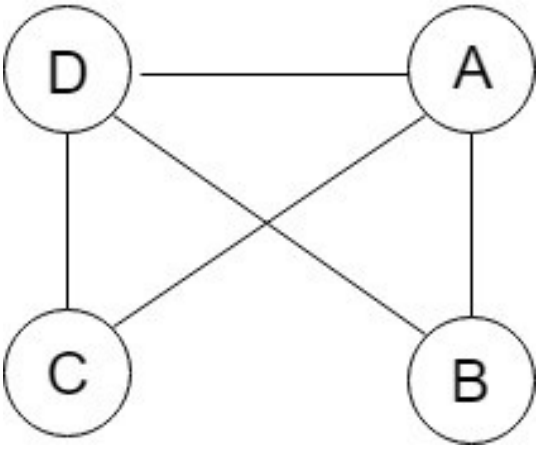
上圖可整理出幾個重點！

- ✧ 矩陣中，第 0 列有 1 個非零數，表示 A 點出分支度為 1。
- ✧ 矩陣中，第 0 行有 2 個非零數，表示 A 點入分支度為 2。
- ✧ 有向圖的 Adjacency matrix 不一定為對稱矩陣。

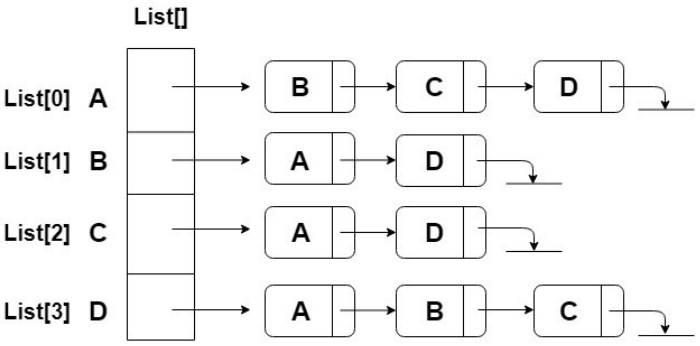
鄰接串列 Adjacency list

- 利用串列的方式將相鄰的頂點串起來。
- 可以節省記憶體空間。

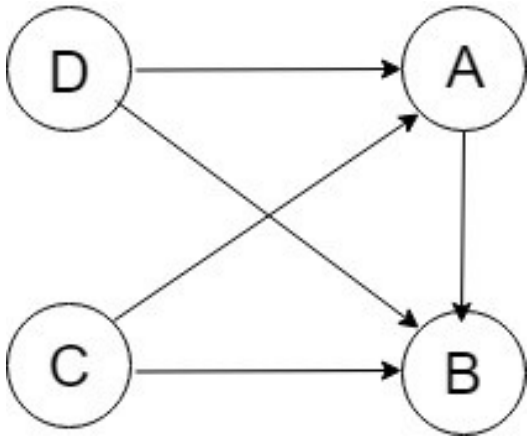
  若以下列無向邊為例的話



它的 Adjacency List 就會是下圖



若以下列有向邊為例的話



它的 Adjacency List 就會是下圖