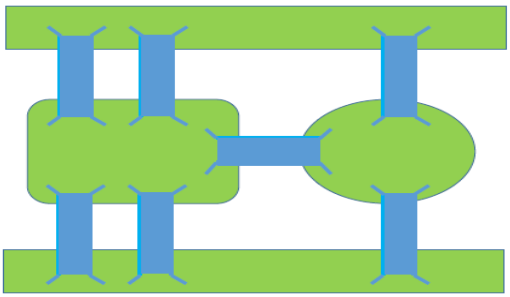
# 九、圖

## 一、圖 (Graph)



☆圖形(Graph)的理論是起源於西元 1736 年，有一位數學家尤拉（Eular）為了解決「肯尼茲堡七橋問題(Koenigshberg Seven Bridge Problem)，而想出的一種資料結構。

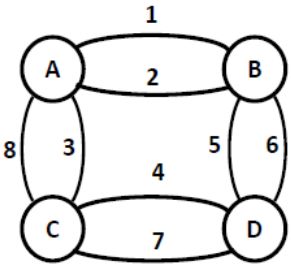
☆肯尼茲堡橋梁問題

①尤拉找出的規則：如果每一個頂點的分支度皆為偶數，才能從某頂點出發，經過每個邊一次，再回到出發的頂點

②依尤拉找出規則：肯尼茲堡橋梁問題的解，因為其有奇數頂點，故無解

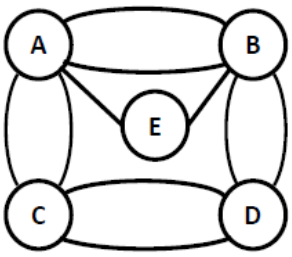
1. 尤拉循環(Eulerian cycle)：由任一頂點出發，經過所有的邊，且只能經過一次，最後回到出發頂點的路徑(條件：所有頂點的分支度必須為偶數)

**(A,B)→(B,A)→(A,C)→(C,D)→(D,B)→(B,D)→(D,C)→(C,A)**




2. 尤拉鏈(Eulerian chain)：從圖形中的任一頂點出發，經過所有的邊，而且只能經過一次，最後不一定要回到原出發頂點的路徑(條件：允許有兩個頂點的分支度為奇數，但剩下所有頂點的分支度必須為偶數)

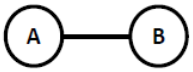
**(A,B)→(B,A)→(A,E)→(E,B)→(B,D)→(D,C)→(C,A)→(A,C)→(C,D)→(D,B)**




3. 圖(Graph)：圖是由頂點(Vertices)和邊(Edges)所組成，以 G=(E,V)來表示；其中 V 為所有頂點的集合，E 為所有邊的集合

①無向圖(Undirected Graph)：

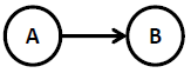
❶邊(Edges)無方向性 ❷邊(V1,V2)與邊(V2,V1)是相同的




②有向圖(Directed Graph)：

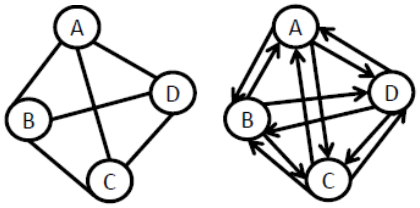
❶邊(Edges)有方向性 ❷邊(V1,V2)與邊(V2,V1)是不同的

❸<V1,V2>其中 V1 為頭(head)，V2 為尾(tail)，方向：從 V1 指向 V2



③非圖結構：有自身迴路(Self Loop)與有重邊(Multi Edge)

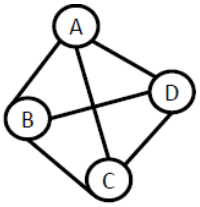
4. 完整圖(Complete Graph)：




①在無向圖中，若有 n 個頂點，並恰好有 n(n-1)/2 個邊，則稱為完整圖

②在有向圖中，若有 n 個頂點，並恰好有 n(n-1)個邊，則稱為完整圖

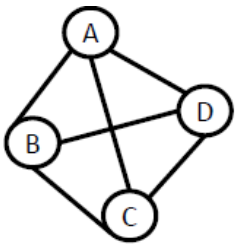
5. 路徑(Path)：在圖 G 中，相異兩點間所經過的邊稱為路徑




A,C間的路徑：
(A,B),(B,C)
(A,C)
(A,D),(D,B),(B,C)
(A,D),(D,C)

①簡單路徑(Simple Path)：指除了起點(第一個節點)與終點(最後一個節點)外的所有節點都不相同的路徑

②循環路徑(cycle)：如果起點及終點為同一點的簡單路徑




A到C的簡單路徑：
(A,C)
(A,B),(B,C)
(A,D),(D,B),(B,C)

循環路徑：
(A,B),(B,D),(D,A)
(A,B),(B,C),(C,D),(D,A)
(A,B),(B,C),(C,A)

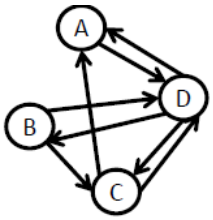
6. 子圖(Sub-graph)：若G'=(V',E')是G=(V,E)的子圖，則V'⊆V與E'⊆E

7. 連通(connected)：在無向圖中，若頂點 $V_i$ 到頂點 $V_k$ 間路徑存在，則稱 $V_i$ 與 $V_k$ 是連通的

8. 連通圖(Connected Graph)：如果圖形 G 中，任兩個頂點均為連通，則此圖形稱為連通圖，否則稱為非連通圖

9. 連通單元(Connected Component)：圖中最大的聯通子圖(Maximal Connected Subgraph)

10. 緊密連通(Strongly Connected)：有向圖中，任一頂點<$V_i$, $V_k$>之間都有一條從 $V_i$ 到 $V_k$ 的路徑，並且也有一條從 $V_k$ 到 $V_i$ 的路徑




11. 相鄰(Adjacent)：

- 無向圖G=(V,E)
  - u,v∈V, (u,v)∈E, 其中u,v代表相異兩個頂點
  - 稱頂點u與頂點v相鄰。
- 有向圖G=(V,E)
  - u,v∈V, <u,v>∈E, 其中u,v代表相異兩個頂點
  - 稱u相鄰到v (u adjacents to v)
  - 稱v相鄰至u (v adjacents from u)

12. 路徑長度(Path Length)：在圖中，相異兩點間所經過的邊的數量

13. 分支度(Degree)：

①無向圖 G=(V,E)：頂點 u 的分支度=附著於 u 的邊的總數。

②有向圖 G=(V,E)：

a. 入分支度(in-degree)：指某頂點 v 擁有「箭頭」的邊數。

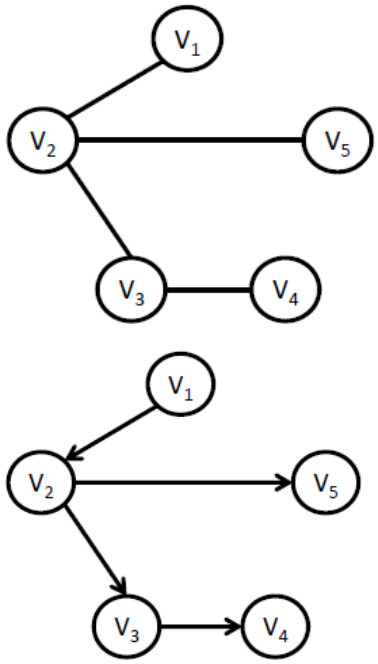
b. 出分支度(out-degree)：指某頂點 v 擁有「尾」的邊數。

## 二、圖(Graph)的表示法

1. 相鄰矩陣(Adjacency Matrix)：若圖 G=(V,E)是具有 n 個頂點的圖形，則使用一個 nxn 的矩陣 A={$a_{ij}$},1≦i≦n,1≦j≦n，若頂點 $V_i$ 與 $V_j$ 相鄰(adjacent)，$a_{ij}$的值為 1，否則 $a_{ij}$的值為 0

※範例講解

**⊙請利用相鄰矩陣表示下圖(上:無向圖、下:有向圖)**







2. 相鄰串列(Adjacency Lists)：假設圖 G=(V,E)包含 n 個頂點(n≦1)時，則可以使用 n 個鏈結串列來存放此圖，每個鏈結串列代表一個頂點及其相鄰的頂點

☆特性：

❶每個頂點使用一個串列

❷對於無向圖，若有 n 個頂點 e 條邊，則共需 n 個串列首節點及 2*e 個節點

❸對於有向圖，若有 n 個頂點 e 條邊，則共需 n 個串列首節點及 e 個節點

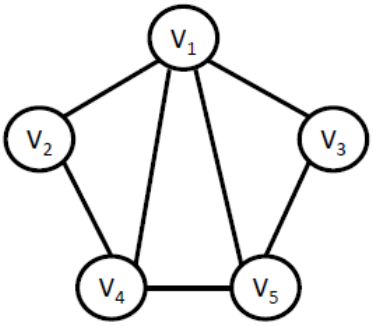
❹在相鄰串列中，計算所有頂點分支度所需的時間複雜度為 O(n+e)

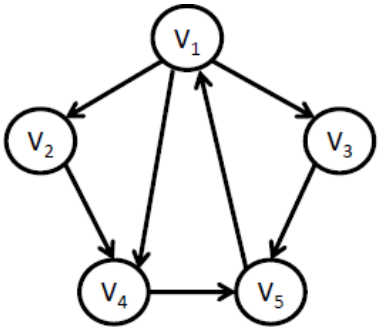
☆節點結構：

| 頂點 (Vertex) | 鏈結 (Link) |
|---|---|

※範例講解

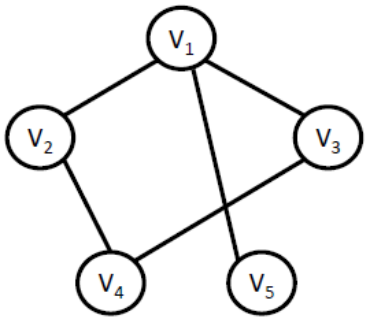
**⊙請利用相鄰串列表示下圖(上:無向圖、下:有向圖)**

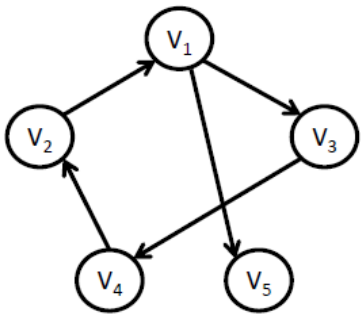



※學生練習

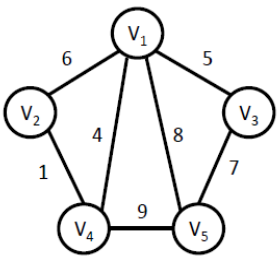
**⊙請利用相鄰串列表示下圖(上:無向圖、下:有向圖)**

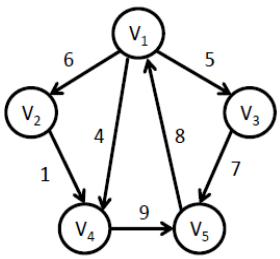



## 三、加權圖

1. 定義：圖的每個邊，都給予一個權重值(weight)

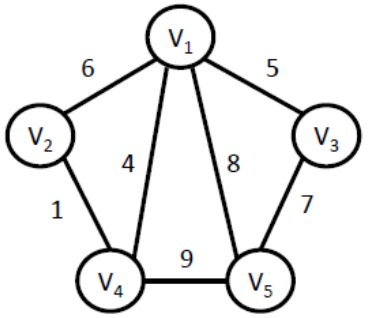


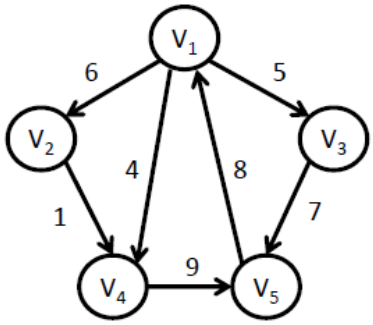



2. 以相鄰矩陣表示

※範例講解

**⊙請利用相鄰矩陣表示下圖(上:無向圖、下:有向圖)**

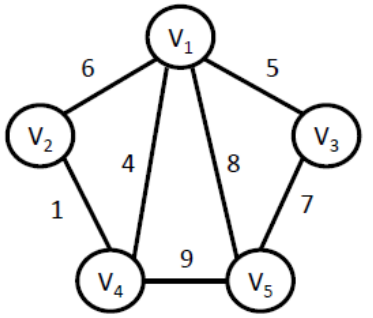


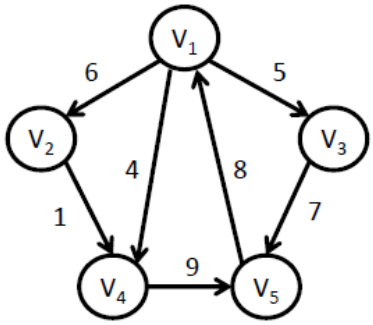




3. 以相鄰串列表示

※範例講解

**⊙請利用相鄰串列表示下圖(上:無向圖、下:有向圖)**



# 四、圖的走訪

1.圖的走訪形式：深度優先搜尋法(Depth-First-Search, DFS)、廣度優先搜尋法(Breadth-First-Search, BFS)

①深度優先搜尋法(Depth-First-Search, DFS)

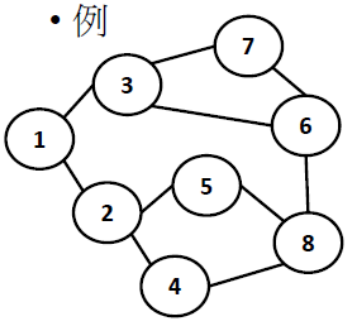
a.利用堆疊處理

b.從圖的某一頂點開始，走訪過的頂點會被標示已拜訪過的記號，接著拜訪此頂點的所有相鄰且尚未拜訪之頂點中的任一個頂點，並標示已拜訪過的記號，再以該點為新的起點繼續進行先深後廣的搜尋。

※範例講解

**⊙請利用深度優先走訪下圖**

• 例




②廣度優先搜尋法(Breadth-First-Search, BFS)

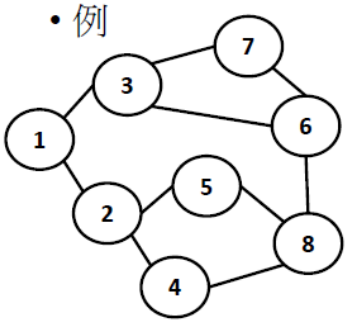
a.先廣後深搜尋法是以佇列(Queue)及遞迴技巧來走訪

b.先廣後深的方式是從圖形的某一頂點開始走訪，被拜訪過的頂點就會被標示已拜訪過的記號。接著走訪此一頂點的所有相鄰並且尚未拜訪過的頂點中的任一頂點，並標示已拜訪過的記號，再以該點為新的起點繼續進行先廣後深的搜尋

※範例講解

**⊙請利用廣度優先走訪下圖**

• 例



# 五、擴張樹(Spanning Tree)

1.定義：對一個有 n 個頂點的相連圖形，經由 DFS 或 BFS 走訪的結果，會得到用最少的邊來連結所有的頂點，且不會形成迴路，這樣的子圖是一種樹狀結構，也就是任何兩頂點之間的路徑唯一。這種可連結所有頂點且路徑唯一的樹狀結構稱為擴張樹(spanningtree)或稱為生成樹、擴展樹

2.令 G=(V,E)是一個圖，而 S=(V,T)是 G 的擴張樹。其中 T 是追蹤時所拜訪過的邊，而 K 表示追蹤後所未被拜訪過的邊。此時擴張樹具有下列幾點特性：

①E=T+K

②V 中的任何兩頂點 V1,V2，在 S 中有唯一的邊

③加入 K 中任何一個邊於 S 中，會造成循環

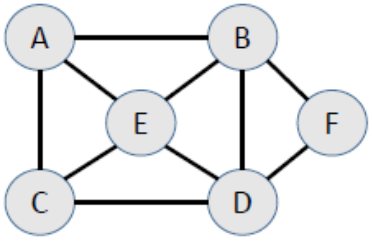
☆特性：

❶擴張樹可用來判斷該圖是否為連通圖

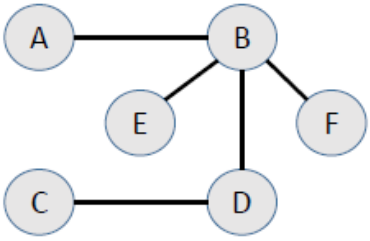
❷擴張樹中的任何兩個頂點間都是相連的，也就是存在一條路徑可通

❸擴張樹不會形成迴路現象

❹若某圖存在有擴張樹，此樹非唯一




圖




擴張樹

3.最小成本擴張樹(Minimum Cost Spanning Tree)：如果一個相連圖(connected graph)的邊(edge)具有權重值(weight)，其代表成本、距離、或關係強度時。當此圖所產生的擴張樹之所有邊的權重值加總為最小，此擴張樹稱為最小成本擴張樹

①Kruskal 演算法

a.邊的權重值先由小到大排序

b.從所有未走訪的邊中，選取權重值最小的邊，記錄此邊已走訪，檢查是否形成迴路

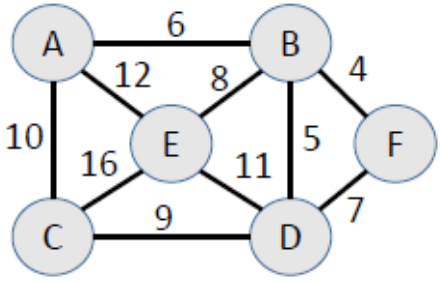
❶形成迴路，此邊不能加入最小成本擴張樹(MST)中，回到步驟 b.

❷未形成迴路，此邊加入最小成本擴張樹(MST)中，如果邊數已達 n-1 條，則到步驟 c.，否則回到步驟 b.

c. Kruskal 可以找出 MST，結束

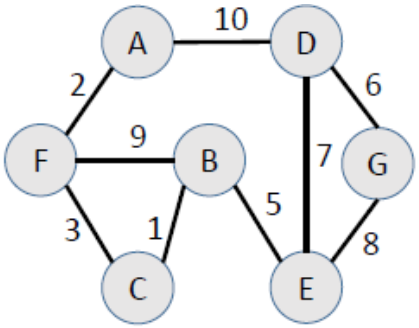
※範例講解

**⊙使用 Kruskal 演算法求下圖的最小成本擴張樹**




※學生練習

**⊙使用 Kruskal 演算法求下圖的最小成本擴張樹**




②Prime 演算法:假設有個圖 G=(V,E),其中 V={1,2,…,n},且最初設定 U={1},U,V 是兩個頂點的集合,並且每次會產生一個邊。亦即從 U-V 集合中找一個頂點 V,能與 U 集合中的某個頂點形成最小成本的邊,把這一頂點 V 加入 U 集合,繼續此步驟,直到集合 U 等於 V 集合為止

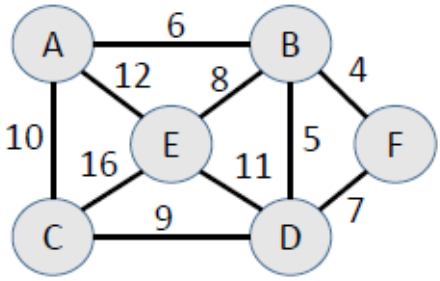
a. 選出某一節點 U 出發點

b. 從與 U 節點相連且尚未被選取的節點中,選擇權重最小的邊,加入新節點

c. 重複加入新節點,直到 n-1 條邊為止(n 為節點數)

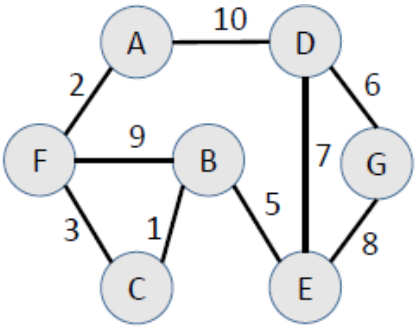
※範例講解

**⊙使用 Prime 演算法求下圖的最小成本擴張樹**



※學生練習

**⊙使用 Prime 演算法求下圖的最小成本擴張樹**




# 六、拓樸排序

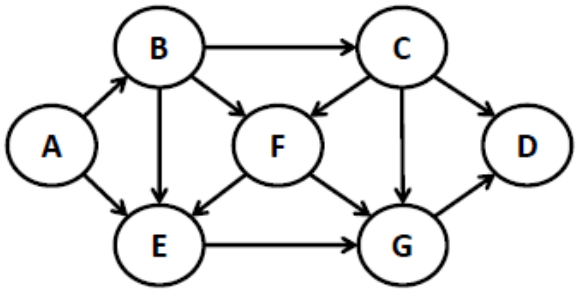
☆前言：

- 所謂“工作”(Activity)，是指將一個計畫分成數個子計畫，而每一個子計畫完成時，即是將整個計畫完成，這個就稱為工作
- 因此,如將“工作”稱為工作網路上的“頂點”，而工作與工作之間的連線，代表著工作的優先順序時，稱為工作網路上的“邊”
- 因此，這種以頂點來代表工作項目的網路稱為頂點工作網路(Activity On Vertex Network)，簡稱為 AOV 網路

1.定義：若在 AOV 網路中，頂點 Vi 是頂點 Vj 的前行者，則在線性的排列中，Vi 一定在 Vj 的前面，此種特性稱之為拓樸排序(Topological Sort)

※範例講解

**⊙求下面 AOV 網路之拓樸排序**




※學生練習

**⊙求下面 AOV 網路之拓樸排序**