6. 圖形概論：
   (1) 七橋問題：四塊陸地，用七座橋將他們連接，在所有橋都只能走一遍的前提下，如何才能把這個地方所有的橋都走遍？(Y or N)
   (2) 尤拉圖：H={V，E}，degree：每一點上連到的邊個數。Total degree = 2

   * total E。(每個邊都必定只會連到兩個點(貢獻兩次 degree))
   Vertex type：odd or even degrees
   **不可能有奇數個 degree 為奇數的 vertex
   0 or 2 vertex with odd degree 才會有解，剩下的狀況都不會有解(odd degrees 代表最後一次經過 vertex 一定是進)
   ⇨ Solve 七橋：No

   (3) 基本術語：
   (Un)deirective graph (digraph)
   Adjacent vertices：相鄰的 vertex
   Edge = 邊
   Path = 路徑
   Cycle = 終點等於起點
   Connected graph
   Complete graph

   Strong connected graph
   Weight graph：權重

   可用儲存圖的方法：matrix(容易浪費空間) or (pointer or vector), linked list() or sequential represent(修改麻煩)

   (4) Traversaal
   深度 or 寬度優先。

   (5) 延伸：spanning tree

7. 圖形應用：

(1) Topological sort(拓樸)：

條件：Acyclic diagraph(DAG)(1)有向圖(2)沒有 cycle

類型：AOE、AOV(activity on vertex)

如果將每個點看作任務，而邊表示一個任務受到上一個任務的約束，那麼拓樸排序就是找到一個有效的任務順序的方法。

(2) 實作：

{

(1)找到 No-successor(no-predecessor) vertex

(2)將 vertex 加入 list 中

(3)將 vertex 以及其所連接的 edge 刪除

(4)重複(1)到(3)直到 graph 中無資料

}

Or {可延伸 DFS 來做}

*可配合原本存好的資料(有紀錄連接 vertex 數量)，每執行完一 vertex，便做刪除及更新一次資料紀錄的 vertex 數量的動作，至資料結束。

(3) spanning tree： 可大略看到整張圖的狀況

通訊網路中很常看到其應用(資料傳遞)，ex：CISCO (STP：spanning tree protocal)

特性：

Connected(讓資料傳遞自由、穩定，不會因為某個點掛掉就收不到資料

Acyclic(防止資料做無謂的傳遞，進而浪費頻寬

在上述兩條件間做平衡

可以用 DFS or BFS 做出 spanning tree：+count

*isomorphic：同構

*有 n 個點時，必有(n-1)個邊，可能的 spanning 會有 n^(n-2)種

*pruffer sequencr：

壓縮資料的一種方法，可用作壓縮及解壓縮樹狀(圖)結構等等資料，大製作法為將末端節點拿掉，並記錄其父節點的 label，便可將 n 個節點的資料存成 n-2 的長度

(4) Minimum spanning tree： 最小權重的生成樹，答案不一定唯一

*類似概念：steiner tree(指定點來做為路徑)、K-spanning tree(指定幾個點)

(5) Algorithm：

Prim：以某一節點(共 n 個節點)出發，在與其相連並且未被連接的節點中，選權重最小的邊並加入新節點，重複動作直到增加 n-1 個邊為止。

Kruskal：以權重最小邊開始，每次挑選權重最小的邊，並且檢查如果加入到樹中是否會出現環(會就不加入)，重複直到樹中有(n-1)個邊。

Sollin：上述兩種方法的綜合，但犧牲空間，讓每一次的回合都能夠為各個集合找到一個新的花費最小的邊。

(6) 最短路徑：

Dijkstra：以動態規劃將大路徑拆分成小路徑的策略，節此找出所求之最短路徑。(最小路徑+最小路徑=總共的最短路徑)，很常應用於地圖的路線規劃，但上面還有很多不聰明的地方可加以改善。

Floyd-Warshall：和 Dijkstra 相似，但目標改為圖中任意兩點的最短距離，透過多次的中轉路徑長度來和初始的路徑比較，找出最短所求。

A：一個較聰明，較精準的方法(有設定目的地和終點)，過程中利用預估值，用預測的方式避開多餘的計算，藉此提高效率，雖然不保證結果是最短路徑。

8. 圖形問題：

(1) AOE：與 AOV 網的區別在於不僅關心各個 node 的先後順序，同時也關心從起點到終點的最短時間，因此複雜度較高。

(2) 分析：

最早發生時間：從前往後，算出最早可開始的時間
最晚發生時間：從後往前，算出最遲要開始的時間
=>最晚-最早，進而找出關鍵節點、關鍵路徑。

可延伸拓譜排序的方式來計算。

(3) 最大流量：

給定一張圖，以及給定一個起點與一個終點，所有可能的 Flow 當中，流量最大(min-cut)者。

Residual Networks：記錄圖上的 edge 還有多少「剩餘的容量」可以讓 flow 流過。

Ford-Fulkerson Algorithm：

每找到一條可從起點到終點的路徑，並且以路徑中節點的嘴小容量為限制，就發送一個流，持續動作直到沒有可用路徑，可以用 DFS 做延伸。

Edmonds-Karp Algorithm： *Heuristic(符合直覺，但不一定最好的方法。)

和 Ford-Fulkerson 相似，但在找路徑時，以最小容量邊節點為優先路徑，可以用 BFS 做延伸。

Eulerian graph：

一筆畫問題。可利用 DFS 找到可行路徑。

TSP on the web：

Greedy Algorithm：任選一個點出發，跑過條件下所有成果，挑選最符合要求的答案，速度快但答案不一定是最好。

Brute-force Algorithm：暴力解題，好寫，但沒效率。

Branch and bound Algorithm：在找到可行路徑後，將路徑長度設定上限，因此加快程式效率，為上兩者折衷。

(4) 上色問題：

Vertex Coloring：

一張圖每個點塗上顏色，鄰點不同色。

sequential ordering algorithm：

建立一張色表配色 BFS，來系統配色。

Welsh-Powell algorithm：

Degree 大的 vertex 優先上色。

*可能因為 vertex 的 degree 相同的狀況，或是 vertex 編號的狀況，導致不能保證是最少顏色的狀況發生。

*有一些特別的圖形可以直接算出有幾個顏色。Ex：Wheel、Cycle、Complete。

(5) Connected Graph：

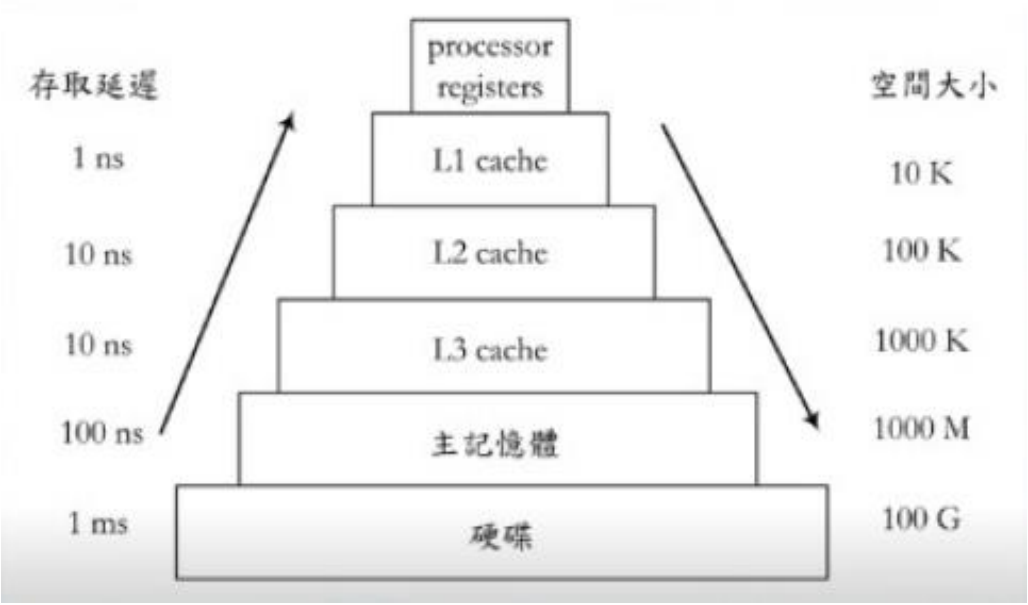
Articulation point：關節點，拿掉會導致圖 disconnected。

如果圖中有這類結構存在，則是個不安全的網絡。

判斷：被檢查點的鄰點是否有機會可走回被檢查點的上一步，

可利用 DFS 之 back-track 性質實施。

9. 次要儲存體：



(1) 通常速度較快的記憶體成本會較高昂，反之亦然，因此我們會希望利用資料結構來讓成本較高昂的地方做較重要的計算，藉此提高硬碟使用效率。

(2) CPU time VS. I/O time(seek(讀寫頭移動的時間) + latency(磁軌旋轉速度，固定) + transfer(讀取資料的速度，有關於電腦的性能))。

(3)外部排序(external sort)：

當資料量大 or 資料存在外部記憶體，為了盡量減少 I/O 讀取次數，或是因內存空間(buffer size)不足，進而產生此法。

*K way Merge：可改變一個 block 存的資料量 or 一次合併的 block 數量來提高速率

(4)Index：索引，在一開始建好索引可幫助後面讀取資料的方便。

讀取：

1. 固定空間大小
2. 在資料前表示好要用的空間大小(file header)
3. 在資料中加間隔符號
4. 花費較多的空間，讓資料存取時就記錄好要存取的位置，與上述三中方法比較的好處是「較容易與不同的程式做資料交換」
5. 將每筆資料存成一個獨立的 file(inderx)，優點是後續的資料維護會方便很多。

刪除：

1. 刪除後將其他資料做搬移的動作。

2. 將某個指定的資料存到被刪除資料的位置。
3. 不做搬移和讀取的動作，只是把位置的 index 紀錄為空白並記錄資料位移量 offset。

B tree：

延伸至 2-3 tree，可自行定義 node 裡要放幾個 key，key 用來存資料的位置，這樣在找資料就可以類似 binary search tree 向下搜尋，並且打開特定檔案並找到特定位置，進而讀取需要之資料，可應用於讀寫較大數據時，減少內存負擔。

B* tree：

每一個節點內 key 的數量少於 2/3 時，就會強制合併，進而提升空間使用率(樹高較矮)。

B+ tree：

樹葉的部分是拿來存資料的，並且每個樹葉 node 是有指標相連的，而除數底以外(上面的)node 是用來存索引的，適合用來處理查找範圍資料時的狀況。

Hash 也可結合 bucket 概念做索引，存入位置，當要查找資料時在依 hash 中對應位置找到檔案中所需資料的位置並讀取。

R tree：

R=(region)，當資料需要以大於一個的欄位做存取索引時(例如：座標)，就可用到這樣的結構。