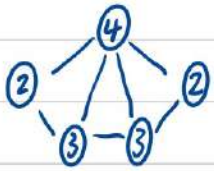
10827135 王力元

柯尼斯堡七橋問題



每個橋只造訪一次

超過兩個奇頂點 → 不存在

n個奇頂點 → 需要 $\frac{n}{2}$ 筆畫出

Graphs of As ADT

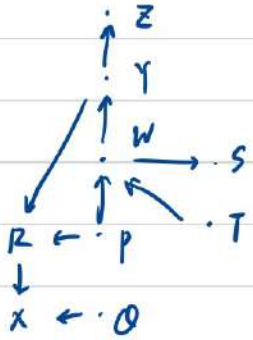
陣列可能不會有內容

有有向邊和無向邊

可以有走訪的動作:

getNumVertices

getNumEdges



Traverse(): O(|V|²)

Adjacency 陣列

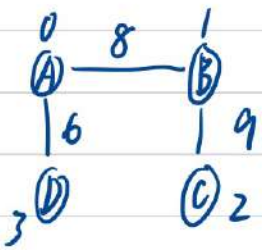
Adjacency 串列

Adjacency Matrix

Matrix[i][j]

1 → i到j有邊

0 → i到j無邊



0 P → R → W

1 Q → X

2 R → X

3 S → T

4 T → W

5 W → S → Y

6 X →

7 Y → R → Z

8 Z →

Graph 的表示法

Adjancency Matrix

Supports operation 1 efficiently

List

Support operation 2

Requires less space than Matrix

Mapping from vertex labels to array

A B C D

0 1 2 3

[0] (1,0): 1

[1] (2,0): 1

[2] (3,0): 0

[3] (3,0): 0

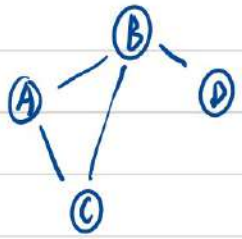
[4] (3,1): 1

[5] (3,2): 0

[10C] → (u,v) = ?

	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	0
D	0	1	0	0

呈現對稱



深度優先走訪 DFS

Last visit, first explored

會先一直往下走訪，到最底才上來

recursive DFS　　易用 stack 實做

標記已走訪的;

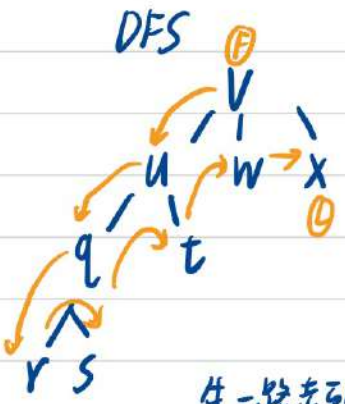
for (未走訪的點的陣列)

recursive DFS

廣度優先走訪 BFS

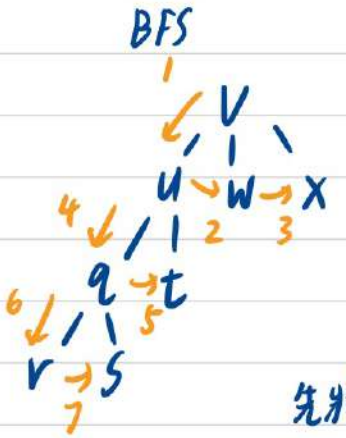
first visited, first explored

易用 Queue 實做



先一路走到底

再往旁、上.



先將同層走訪完

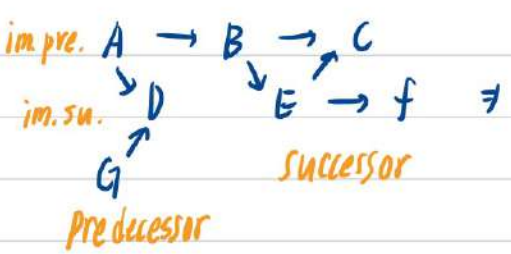
再往下

拓樸排序

有向無環圖的頂點組成的序列

1. 序列中包含每一個頂點且只出現一次.
2. 若A排在B前, 則不存在B到A之路徑

滿足以上條件稱該圖之拓樸排序, 拓樸排序並不唯一

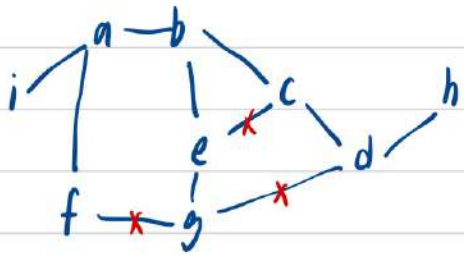


(a) a g d b e c f

(b) a b g d e f c

生成樹 spanning tree

無向圖 A 的生成樹，就是碰到 A 的全部頂點
且邊數最少的圖

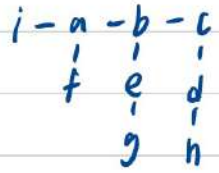


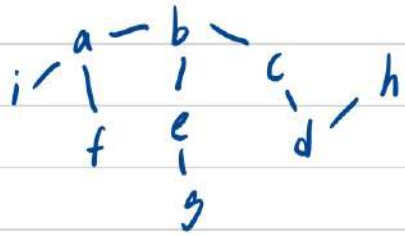
將 cycle 消除來保持 spanning tree

Spanning tree 的權重，就是樹上自除邊的權重總和
Minimum Spanning Tree，權重最小的生成樹，不唯一

2 nodes → $2^{2-2} = 1$
3 nodes → $3^{3-2} = 3$
4 nodes → $4^{4-2} = 16$

Prufer sequence

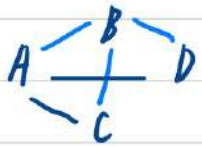




DFS in iterative form

A → B → C → D
B → A → C → D
C → A → B
D → A → B

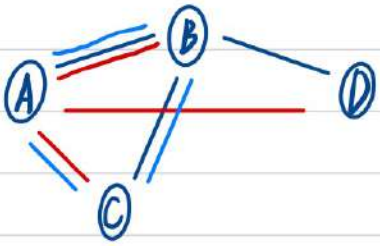
AB BC BD



最小生成樹

一個權重最小的無向圖生成樹

並不唯一!



DFS

BFS

MST

① 先找到一個從已走訪的 v 到未走訪的 u 權重最小的邊

② 標記 u 為已走訪

③ 將 (v, u) 加入 M.S.T.

④ 重覆以上直至完成

有許多種演算法來找到 M.S.T.

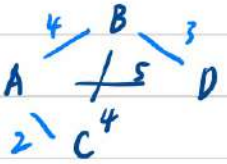
Prim 演算法

A → C2 → B4 → D5

B → D3 → A4 → C4

C → A2 → B4

D → B3 → A5

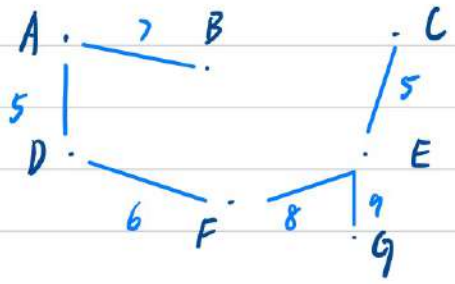


Prim 演算法每一步都會給樹添加一個邊

```
Prim Algorithm()
    while(count < |V|-1)
        (v, u) v至u之最小權重邊
        標記u為已走訪
        加入(v,u)至M.S.T.
        count++;
```

Kruskal 演算法

依照權重順序（小至大）加入生成樹中，但如果加入該邊會形成就不加
直到樹中有 V-1 條邊 即為最小生成樹



Sollin 演算法

找到 V 在 T 內和 U 在 T 外的權重最小邊
合併二樹然後重複直到剩一顆樹.

最短路徑整理

確定起點的：Dijkstra 演算法
確定終點的：當作是確定起點的但將路徑皆反轉
確定起終點：兩點最短路徑
全域最短：Floyd-Warshall 演算法

Dijkstra's 演算法

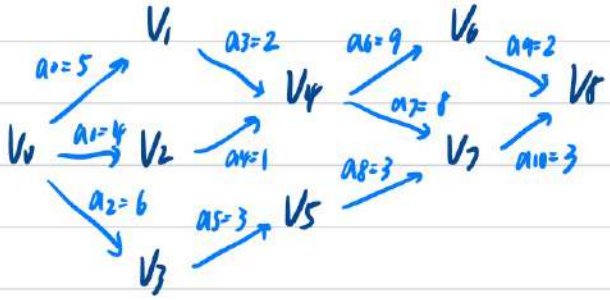
拓樸排列會產生沒有 cycle 的線性順序
Spanning tree 是無向圖
樹是沒有 cycle 的無向圖

AOE 網

表示活動的網

有權重的有向圖 (no cycles)

在計算 performance 或處理流程費用



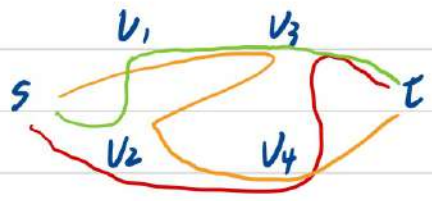
路徑最長的路叫關鍵路徑

找到關鍵路徑的方法:

1. 輸入e條弧(j,k)建立AOE網
2. 從源点出發,按拓撲順序求各頂点最早發生時间
3. 從匯点出發,按逆拓撲順序求其餘頂点發生時间.
4. 根據各頂点ve和vl值,若某弧滿足條件e(s)=l(s),則為关鍵活动

最大流問題

涉及單一源点、單匯点中最大的一條



最大流最小割定理

Ford-Fulkerson 演算法

計算網路流的貪心算法

1. 先為每一個边初始化 $C_f(u,v)$
2. 找到一個s到t的路徑 $P \ni C_f(u,v)>0 \ \forall (u,v) \in P$
3. $C_f(P) = \min\{C_f(u,v) : (u,v) \in P\}$
4. $C_f(u,v) = C_f(u,v) - C_f(P)$
   $C_f(v,u) = C_f(v,u) + C_f(P)$

Edmond-Karp 演算法

1. 初始化每一個邊 $cf(u,v)$
2. 以 heuristic 找到 s 至 t 的路徑 P
3. $Cp(P) = \min\{Cf(u,v) : (u,v) \in P\}$;
4. For each edge $(u,v) \in P$
   - $Cf(u,v) = Cf(u,v) - Cf(P)$
   - $Cf(u,v) = Cf(u,v) + Cf(P)$

Heuristic - 1. max-capacity 優先
- 2. 廣度優先

次要儲存

當要處理大量 data，待排序的 data 不可以一次存入記憶體（內存）
先存在硬碟待先前 data 處理好再讀入

Sequential Access（循序存取）連續、讀取較快
Direct Access（random access）可以存取特定位置、well organized.

delay		size
短	暫存器	小
↑	快取	
	記憶體	↓
長	硬碟	大

外部排序 External sort

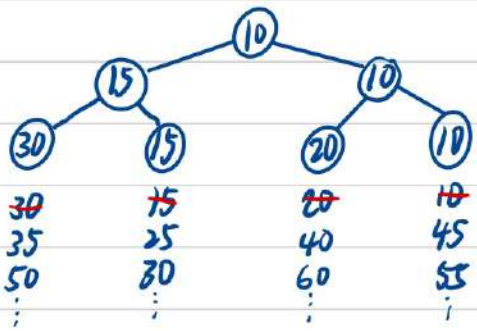
k-way Merge

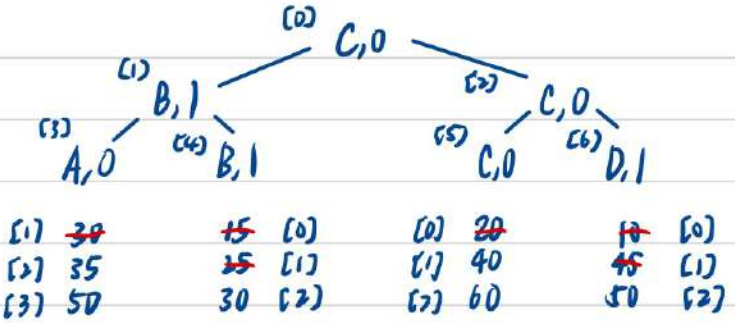
2-way Merge
64 runs → 32, 16, 8, 4, 2, 1 → $\log_2 64 = 6$ 回合
8 runs → 4, 2, 1 → $\log_2 8 = 3$ 回合

∋ k-way merge need $\log_k m$ 回合
越高的 order 可以減少 I/O Time（因為不須太多回合）

K-way merge : Selection Tree

可以用來減少比較回合數來找到最小值





B樹索引

B樹可以自己平衡，從2-3樹延伸而來

Balanced m-way tree = B-tree order m

Given the order m and tree height h, the num. of keys $N \le m^h - 1$

ex. B-tree of order 20, height 3 $\Rightarrow N \le 20^3 - 1$

✓ [0] C,0
✓ [1] B,1
[2] C,0
[3] A,0
✓ [4] B,1
[5] C,0
[6] D,1

其它變體：B^+ tree、B^* tree.