7-6 擴張樹

定義

是指 n 個頂點的相連圖形，經由前面所介紹演算法拜訪的結果，會得到**用最少的邊來連結所有的頂點**，且**不會形成循環迴路**。並且任何兩個頂點之間的**路徑是唯一**的。因此，這種可連結所有頂點且路徑唯一的樹狀結構，就稱為「**擴張樹**」**(Spanning Tree)**。

應用

興建的道路：以最少的道路，來連通所有的鄉鎮。

特性

假設 $G = (V, E)$ 是一個圖形，而 $S = (V, T)$ 是 G 的擴張樹。其中 T 是追蹤時所拜訪過的邊，而以 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：

- **特性一：$E = T + K$**

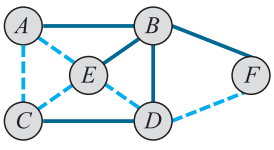


圖 7-7 擴張樹

其中：E 代表圖形的各邊。

實線 T 代表圖形追蹤時所拜訪過的邊。

虛線 K 代表圖形追蹤後未被拜訪過的邊。

- **特性二：加入未被拜訪過的 K 邊於擴張樹 S 中，將會造成循環迴路。**

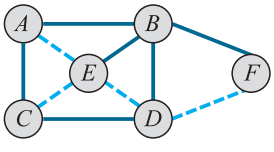


圖 7-8 不是擴張樹

例如在上圖中，我們再加入一條虛線 *AC* 這一個邊到擴張樹 *S*，此時就**產生循環現象**，**循環迴路 (*A* → *B* → *C* → *D*)**。

動畫圖解

參看隨書光碟。

單元評量

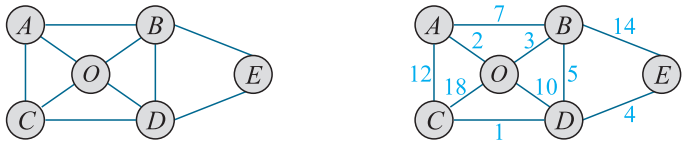
1. 假設有 *n* 個頂點的相連圖形，利用最少的邊來連結所有頂點，且不會形成迴路，任何兩個頂點之間的路徑唯一，這種可連結所有頂點且路徑唯一的樹狀結構，稱為：
   (A) 堆積樹　(B) 二元樹　(C) 二元搜尋樹　(D) 擴張樹
2. 關於擴張樹的敘述，下列何者有誤？
   (A) 擴張樹可以用來判斷該圖是否為連通
   (B) 擴張樹中的任兩個頂點間都是相連的，也就是存在一條路徑可通
   (C) 擴張樹不會形成迴路現象
   (D) 擴張樹所連的路徑一定是最短

7-7 最小成本擴張樹

定義

擴張樹在實際的應用上不只是找出頂點和邊而已，如果一個相連圖形的邊加上**權重值**，來代表邊的**成本、距離或關係強度**時，則我們希望所產生的擴張樹之所有邊的**權重值加總為最小**，具有這樣性質的擴張樹稱為**最小成本擴張樹 (Minimum-Cost Spanning Tree)**。

概念圖

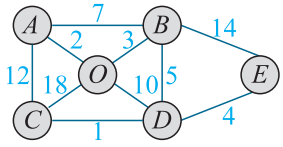


(a) 各處室的邏輯架構圖　　(b) 計算機中心到各處室的實際距離，以權重值來表示

圖 7-9 最小成本擴張樹

實例

假設我們現在想要設計某一大學的校務行政網路系統，並且以計算機中心 (也就是以英文字 *O*) 為出發點，到各處室 (*A*～*E*) 的實際距離，是以相連圖形上的邊之權重值來表示。請追蹤此相連圖形可能的擴張樹。



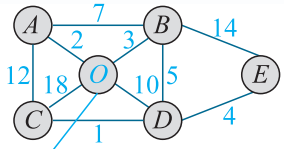
解答

在上圖的相連圖形中，其追蹤擴張樹可能有以下三**種方法**：

第一種方法

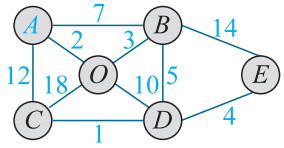
利用**深度優先追蹤法**，得到權重值總和為 28，其追蹤步驟如下：

- **步驟一：**以頂點 *O* 為從發點，所以，先輸出頂點 *O*，接著與此一頂點的所有相鄰並且未拜訪過的頂點 *D*, *C*, *B*, *A* 放入「堆疊」中，此時，頂點 *O* 就會被輸出到「已被拜訪頂點區」中。如下圖所示。



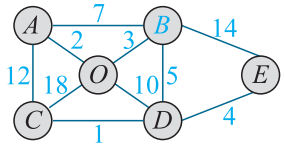
已被拜訪頂點區	堆疊內的元素
O	A B C D

- **步驟二：**拜訪「堆疊」頂端的頂點 *A*，由於與頂點 *A* 相鄰頂點都已被拜訪，此時，頂點 *A* 就會被輸出到「已被拜訪頂點區」中。如下圖所示。



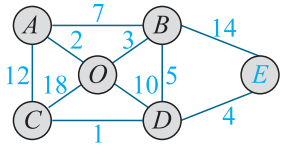
已被拜訪頂點區	堆疊內的元素
OA	A B C D

- **步驟三：**拜訪「堆疊」頂端的頂點 *B*，再將與頂點 *B* 相鄰且未拜訪過的頂點 *E* 放入堆疊中，此時，頂點 *B* 就會被輸出到「已被拜訪頂點區」中。如下圖所示。



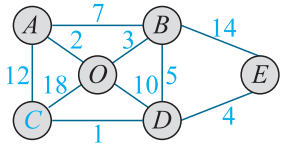
已被拜訪頂點區	堆疊內的元素
OAB	B C D

- **步驟四：**拜訪「堆疊」頂端的頂點 E，由於與頂點 E 相鄰頂點都已被拜訪，此時，頂點 E 就會被輸出到「已被拜訪頂點區」中。如下圖所示。



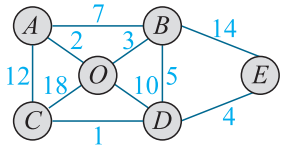
已被拜訪頂點區	堆疊內的元素
$OABE$ ←	E C D

- **步驟五：**拜訪「堆疊」頂端的頂點 C，由於與頂點 C 相鄰頂點都已被拜訪，此時，頂點 C 就會被輸出到「已被拜訪頂點區」中。如下圖所示。

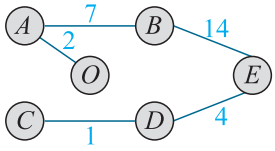


已被拜訪頂點區	堆疊內的元素
$OABEC$ ←	C D

- **步驟六：**最後，再拜訪「堆疊」頂端的頂點 D，此時，頂點 D 就會被輸出到「已被拜訪頂點區」中。即可完成深度優先追蹤法，並且得到權重值總和為 28。如下圖所示。



深度優先度



權重值總和為：1 + 2 + 4 + 7 + 14 = 28

第二種方法

利用**廣度優先追蹤法**，得到權重值總和為 37，其追蹤步驟如下：

- **步驟一：**以頂點 *O* 為從發點，所以，先輸出 *O*，再將相鄰的頂點 *A*, *B*, *C*, *D* 加入佇列中。

A	*B*	*C*	*D*		

- **步驟二：**輸出頂點 *A*。

B	*C*	*D*			

- **步驟三：**輸出頂點 *B*，再將與頂點 *B* 相鄰且未拜訪過的頂點 *E* 加入佇列中。

C	*D*	*E*			

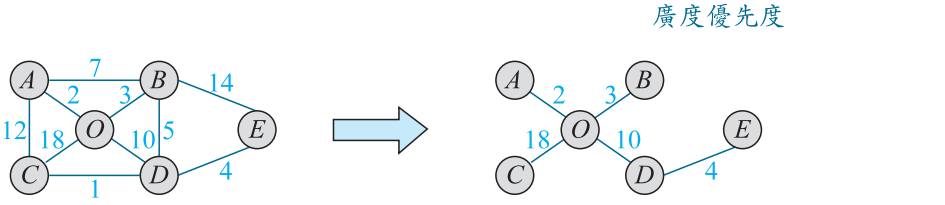
- **步驟四：**輸出頂點 *C*。

D	*E*				

- **步驟五：**輸出頂點 *D*。

E					

- **步驟六：**輸出最後的頂點 *E*，即可完成廣度優先追蹤法，並且得到權重值總和為 37。如下圖所示。

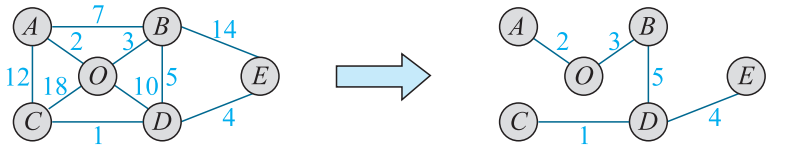


第三種方法

利用**最小成本擴張樹追蹤法**，得到權重值總和為 15，其追蹤步驟如下：

- **步驟一：**將所有邊的權重值先由小到大排序。
- **步驟二：**首先，找出最小權重值的邊 *CD*，其權重值為 1。接下來，再找出第 2 小權重值的邊 *AO*，其權重值為 2，再找出第 3 小權重值的邊 *BO*，其權重值為 3，接下來，再找出第 4 小權重值的邊 *DE*，其權重值為 4，最後，再找出第 5 小權重值的邊 *BD*，其權重值為 5。

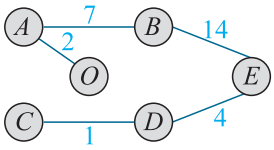
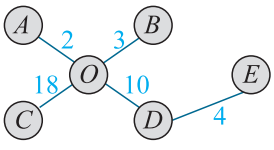
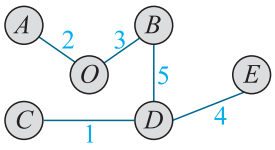
由於圖中有 6 個頂點，而此時最小成本擴張樹中已有 5 個邊，因此，可得最後結果如下圖所示。



最小成本擴張樹

權重值總和為：1 + 2 + 3 + 4 + 5 = 15

綜合以上三種方法，可清楚得知，利用最小成本擴張樹追蹤法，其成本最小。如下表所示：

深度優先追蹤法	廣度優先追蹤法	最小成本擴張樹追蹤法
		
權重值總和為： 1 + 2 + 4 + 7 + 14 = 28	權重值總和為： 2 + 3 + 4 + 10 + 18 = 37	權重值總和為： 1 + 2 + 3 + 4 + 5 = 15

基本上，「最小成本擴張樹追蹤法」的演算法有兩種：

1. Kruskal 演算法。
2. Prims 演算法。

參看隨書光碟。

單元評量

1. 那個演算法能找出最小擴張樹？

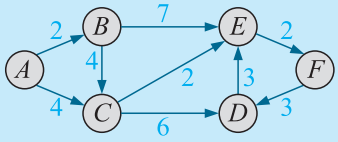
(A) Kruskal 演算法　　(B) 深度優先追蹤演算法

(C) 廣度優先追蹤演算法　(D) 雜湊法

2. 在下圖中，A 至頂點 F 的最短路徑為：

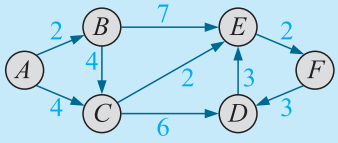
(A) $A \rightarrow C \rightarrow D \rightarrow E \rightarrow F$　(B) $A \rightarrow B \rightarrow E \rightarrow F$

(C) $A \rightarrow B \rightarrow C \rightarrow D \rightarrow F$　(D) $A \rightarrow B \rightarrow C \rightarrow E \rightarrow F$



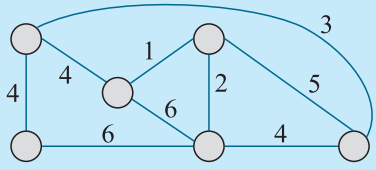
3. 在下圖中，A 至頂點 F 的最短距離為：

(A) 3　(B) 8　(C) 10　(D) 11



4. 請問下圖最少成本擴張樹的成本總和為何？

(A) 12　(B) 14　(C) 15　(D) 16



7-7.1 Kruskal 演算法

由前一個單元可以清楚得知，利用最小成本擴張樹追蹤法，其成本最小。因此，我們就來介紹「**最小成本擴張樹追蹤法**」的第一種演算法，也就是 **Kruskal 演算法**。

作法

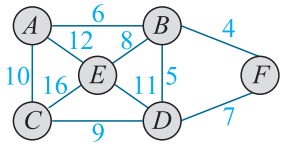
每次挑選一個權重值最小的邊，加入 T 中，並形成最小成本擴張樹，但**不可以形成迴圈，直到數量達 $n-1$ 個邊為止**。此種演算法是根據各邊的加權值大小，再由小到大排序後，再選取要加入 T 的邊。

演算法

1. 邊的權重值先**由小到大排序**。
2. 從所有未走訪的邊中**取出最小權重值的邊**，記錄此邊已走訪，檢查是否形成迴路。
   (1) **形成迴路**，此邊不能加入 MST 中，回到步驟 2。
   (2) **未形成迴路**，此邊加入 MST 中，如果邊數已達 $(n-1)$ 條則到步驟 3，否則回到步驟 2。
3. Kruskal 演算法可以找出 MST，結束。

實例

假設我們現在有一個具有六個頂點的相連圖形，請利用 Kruskal 演算法來求出此圖形的最小成本擴張樹。

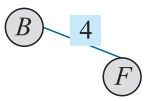


解答

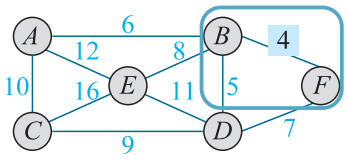
- **步驟一：**將所有邊的權重值先由小到大排序。

起始頂點	終止頂點	權重值 (成本)	由小到大排序
B	F	4	1
B	D	5	2
A	B	6	3
D	F	7	4
B	E	8	5
C	D	9	6
A	C	10	7
D	E	11	8
A	E	12	9
C	E	16	10

- **步驟二：**選擇最小權重值的邊當作最小成本擴張樹的起點。



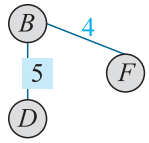
解析過程



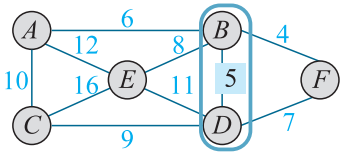
起始頂點	終止頂點	權重值(成本)	由小到大排序
B	*F*	4	1
B	*D*	5	2
A	*B*	6	3
D	*F*	7	4
B	*E*	8	5
C	*D*	9	6
A	*C*	10	7
D	*E*	11	8
A	*E*	12	9
C	*E*	16	10

最小權重值

- **步驟三：**依照步驟 1 的表格之權重值大小，加入第 2 小的權重值於最小成本擴張樹中。



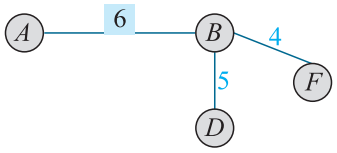
解析過程



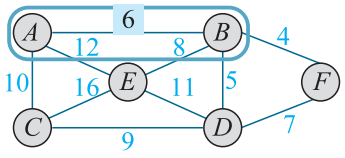
起始頂點	終止頂點	權重值(成本)	由小到大排序
B	*F*	4	1
B	*D*	5	2
A	*B*	6	3
D	*F*	7	4
B	*E*	8	5
C	*D*	9	6
A	*C*	10	7
D	*E*	11	8
A	*E*	12	9
C	*E*	16	10

第 2 小的權重值

- **步驟四：**依照步驟 1 的表格之權重值大小，加入第 3 小的權重值於最小成本擴張樹中。



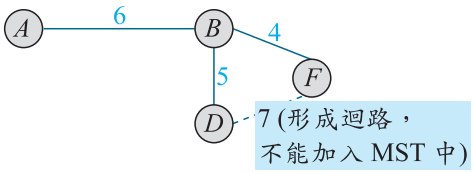
解析過程



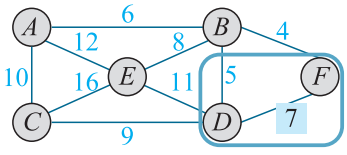
起始頂點	終止頂點	權重值(成本)	由小到大排序
B	*F*	4	1
B	*D*	5	2
A	*B*	6	3
D	*F*	7	4
B	*E*	8	5
C	*D*	9	6
A	*C*	10	7
D	*E*	11	8
A	*E*	12	9
C	*E*	16	10

第 3 小的權重值

- **步驟五：**依照步驟 1 的表格之權重值大小，加入第 4 小的權重值於最小成本擴張樹中，但是形成迴路，不能加入 MST 中，所以直接跳過。



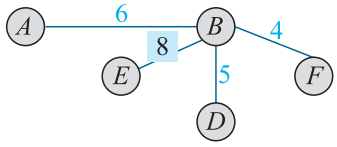
解析過程



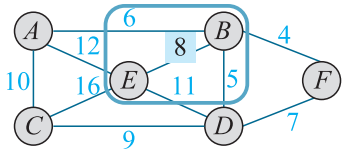
起始頂點	終止頂點	權重值(成本)	由小到大排序
B	*F*	4	1
B	*D*	5	2
A	*B*	6	3
D	*F*	7	4
B	*E*	8	5
C	*D*	9	6
A	*C*	10	7
D	*E*	11	8
A	*E*	12	9
C	*E*	16	10

第 4 小的權重值

- **步驟六：**依照步驟 1 的表格之權重值大小，加入第 5 小的權重值於最小成本擴張樹中。



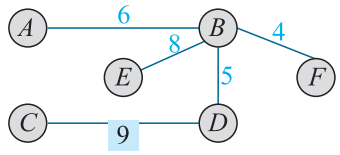
解析過程



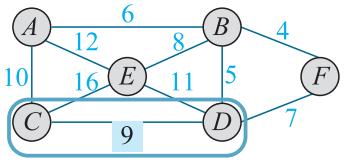
起始頂點	終止頂點	權重值(成本)	由小到大排序
B	*F*	4	1
B	*D*	5	2
A	*B*	6	3
D	*F*	7	4
B	*E*	8	5
C	*D*	9	6
A	*C*	10	7
D	*E*	11	8
A	*E*	12	9
C	*E*	16	10

第 5 小的權重值

- **步驟七：**依照步驟 1 的表格之權重值大小，加入第 6 小的權重值於最小成本擴張樹中。由於圖中有 6 個頂點，而此時最小成本擴張樹中已有 5 個邊，因此，最後結果如下圖所示。



解析過程



起始頂點	終止頂點	權重值(成本)	由小到大排序
B	*F*	4	1
B	*D*	5	2
A	*B*	6	3
D	*F*	7	4
B	*E*	8	5
C	*D*	9	6
A	*C*	10	7
D	*E*	11	8
A	*E*	12	9
C	*E*	16	10

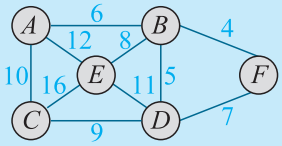
第 6 小的權重值

動畫圖解

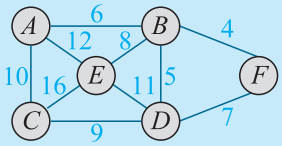
參看隨書光碟。

單元評量

1. 在下圖中，以 Kruskal 演算法求最小成本擴張樹，選擇邊的順序可為：
(A) 456789 (B) 45678 (C) 45689 (D) 4567



2. 在下圖中，以 Kruskal 演算法求最小成本擴張樹，總成本為：
(A) 39 (B) 32 (C) 31 (D) 47



3. 使用 Kruskal 演算法求最小成本擴張樹的成本總和時；若樹的節點共有 6 個，則最小花費擴張樹的邊會有幾個？
(A) 4 (B) 5 (C) 6 (D) 7 (E) 不一定

4. 何種演算法是每次挑選一個權重最小的邊，加入 T 中，並形成最小成本擴張樹，但不可形成迴圈，直到數量達 $n-1$ 個邊為止？
(A) Prims 演算法 (B) Kruskal 演算法 (C) Dijkstra's 演算法 (D) 以上皆非

7-7.2 Prims 演算法

定義

假設有一個圖形 $G=(V, E)$，其中 $V=\{1, 2, 3, \cdots, n\}$，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 U 集合等於 V 集合為止。

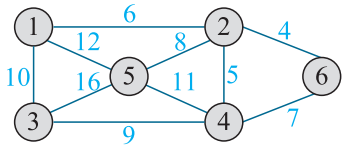
演算法

1. 選出某一節點 U 作為出發點。

2. 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。
3. 重複加入新節點，直到 $n-1$ 條邊為止。(其中 n 為節點數)

實例

假設我們現在有一個具有六個頂點的相連圖形，請利用 Prims 演算法求出此圖形的最小成本擴張樹。

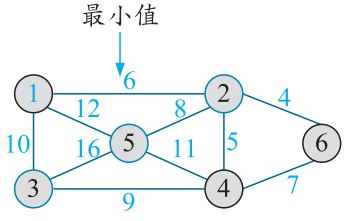


解答

- **步驟一：**設定 U 及 V 是圖形中頂點的集合，假設 U 集合的起始點為 1，而 V 集合就是全部的頂點 (1 到 6)。

$U=\{1\}$　　$V=\{1,2,3,4,5,6\}$

- **步驟二：**從頂點 1 出發，與頂點 1 相連且尚未被選取的頂點有頂點 2、頂點 3、頂點 5，因此，我們找到頂點 2，其邊為最小 6，所以，將頂點 2 加入 U 集合中。



$U=\{1\}$

10　12

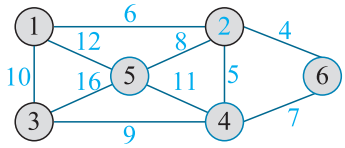
6

$V-U=\{2,3,4,5,6\}$

$U=\{1,2\}$

$V-U=\{3,4,5,6\}$

- **步驟三：**再從頂點 2 出發，與頂點 2 相連且尚未被選取的頂點有頂點 4、頂點 5、頂點 6，因此，我們找到頂點 6，其邊為最小 4，所以將頂點 6 加入 U 集合中。



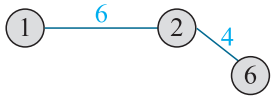
$U = \{1, 2\}$

5　8　4

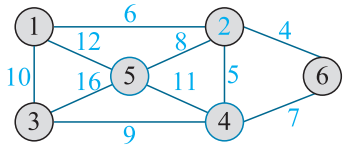
$V - U = \{3, 4, 5, 6\}$

$U = \{1, 2, 6\}$

$V - U = \{3, 4, 5\}$



- **步驟四：**接下來，一樣從頂點 2 出發，與頂點 2 相連且尚未被選取的頂點有頂點 4、頂點 5，因此，我們找到頂點 4，其邊為最小 5，所以將頂點 4 加入 U 集合中。



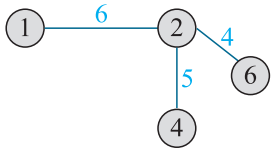
$U = \{1, 2, 6\}$

5　8

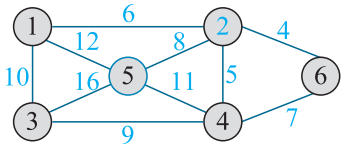
$V - U = \{3, 4, 5\}$

$U = \{1, 2, 4, 6\}$

$V - U = \{3, 5\}$



- **步驟五：**接下來，一樣從頂點 2 出發，與頂點 2 相連且尚未被選取的頂點有頂點 5，因此，我們找到頂點 5，其邊為 8，將頂點 5 加入 U 集合中。



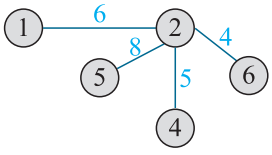
$U = \{1, 2, 4, 6\}$

8

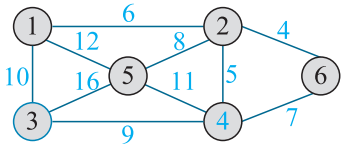
$V - U = \{3, 5\}$

$U = \{1, 2, 4, 5, 6\}$

$V - U = \{3\}$



- **步驟六：**接下來，再從頂點 4 出發，與頂點 4 相連且尚未被選取的頂點有頂點 3，因此，我們找到頂點 3，其邊為 9，將頂點 3 加入 U 集合中。



$U = \{1, 2, 4, 5, 6\}$

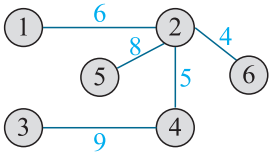
9

$V - U = \{3\}$

$U = \{1, 2, 3, 4, 5, 6\}$

$V - U = \{\ \}$

$V = \{1, 2, 3, 4, 5, 6\}$



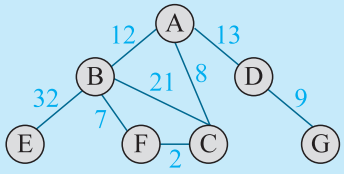
- **步驟七：**最後，將頂點 3 加入 U 集合，此時集合 U 等於集合 V，動作結束。

動畫圖解

參看隨書光碟。

單元評量

1. 在下圖中，請利用 Prims 演算法求出最小成本擴張樹，選擇邊的順序可為：
(A) 8, 2, 7, 9, 13, 32 (B) 8, 2, 7, 32, 9, 13 (C) 8, 2, 7, 13, 32, 9 (D) 8, 2, 7, 13, 9, 32



7-8 最短路徑

單點到其他各頂點之最短路徑問題的典型應用是由甲城市 (頂點) 到乙城市 (頂點) 之間的距離 (權重值)，計算由甲城市出發，經由多重交通網路的計算，到達乙城市的最短路徑，此種問題往往有多條可行走的路徑，因此，我們必須從這多條路徑中選擇最短路徑。

定義

最短路徑 (Shortest Path) 問題是目前圖形結構中常見的**典型問題之一**。因為圖形中某頂點到達各頂點的路徑不是唯一，如果要從眾多的路徑中找出路徑最短者，則稱為**最短路徑問題**。一般而言，常用的方法是利用 **Dijkstra's Algorithm 求得**。

形式

1. 單點到其他各頂點之最短路徑。
2. 各個節點之間之最短路徑。

演算法

- **步驟一**

  $D[I] = A[F, I]$ $(I = 1, \cdots, N)$

  $S = \{F\}$

  $V = \{1, 2, 3, \cdots, N\}$

  其中：D 為 N 個位置的陣列，用來儲存某一頂點到其他頂點的最短距離。

  F 表示由某一起始點開始。

  $A[F, I]$ 是表示 F 點到 I 點的距離。

  V 是網路中所有頂點的集合。

  S 也是頂點的集合。

- **步驟二**

  從 $V-S$ 集合中找一頂點 t 使得 $D[t]$ 是最小值，並將 t 放入 S 集合，一直到 $V-S$ 是空集合為止。

- **步驟三**

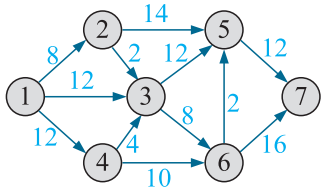
  根據下面的公式調整 D 陣列中的值。

$$D[I] = \min(D[I], D[t] + A[t, I])$$

  此處 I 是指 t 的相鄰各頂點。繼續回到步驟 2 執行。

實例

假設我們現在有一個具有 7 個頂點的圖形結構，請利用「最短路徑」的演算法來求出下圖中，頂點 1 到各點的最短距離。



解答

- **步驟一：** $F = 1$; $S = \{ 1 \}$; $V = \{1, 2, 3, 4, 5, 6, 7\}$

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	12	12	∞	∞	∞

(1) 假設我們從起始頂點 1 開始，頂點 1 到頂點 2 的距離為 8，我們就可以把陣列 $D[2]$ 寫成 8 ，而 $D[3]$、$D[4]$ 的值也是由頂點 1 到頂點 3 和頂點 4 的距離，但是由於頂點 1 無法由直接到達頂點 5、頂點 6 及頂點 7，因此，我們把 $D[5]$、$D[6]$、$D[7]$ 的值設定為 ∞。

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	12	12	∞	∞	∞

(2) 上面的陣列中，頂點 1 到頂點 2 的距離最短 (即 $D[2]$)，因此，將頂點 2 加入 S 的集合中，$S = \{1, 2\}$, $V-S = \{3, 4, 5, 6, 7\}$。

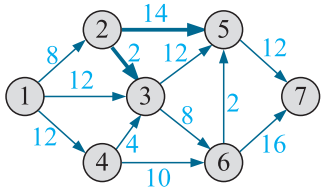
距離最短

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	12	12	∞	∞	∞

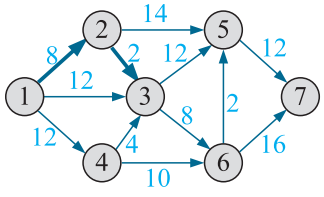
原始狀態：$S = \{1\}$, $V - S = \{2, 3, 4, 5, 6, 7\}$

目前狀態：$S = \{1, 2\}$, $V - S = \{3, 4, 5, 6, 7\}$

(3) 頂點 2 的相鄰頂點為 3, 5 ，則：



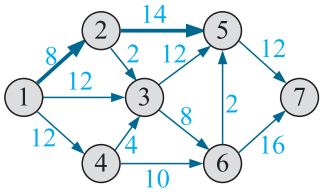
$D[3] = \min(D[3], D[2] + A[2, 3]) = \min(12, 8 + 2) = 10$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	12 → 10	12	∞	∞	∞

此時，頂點 1 到頂點 3 可透過頂點 2，其距離就變成 10。

$$D[5] = \min(D[5], D[2] + A[2, 5]) = \min(\infty, 8 + 14) = 22$$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	12	12	$\infty \rightarrow 22$	∞	∞

此時，頂點 1 到頂點 5 可透過頂點 2，其距離就變成 22。陣列中的內容如下：

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	∞	∞

- **步驟二：**

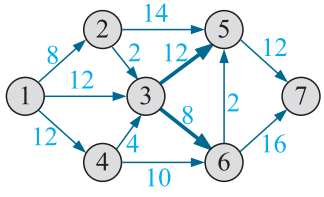
(1) 上面的陣列中，頂點 1 到頂點 3 的距離最短 (即 $D[3]$)，因此，將頂點 3 加入 S 的集合中，因此，$S = \{1, 2, 3\}$, $V - S = \{4, 5, 6, 7\}$。

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	∞	∞

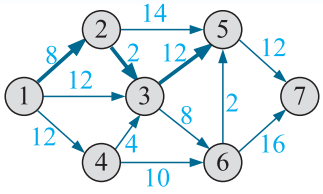
上次狀態：$S = \{1, 2\}$, $V - S = \{3, 4, 5, 6, 7\}$

目前狀態：$S = \{1, 2, 3\}$, $V - S = \{4, 5, 6, 7\}$

(2) 頂點 3 的相鄰頂點為 5, 6 ，則：



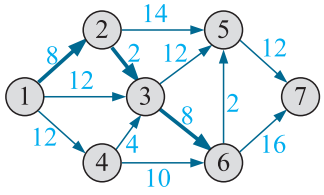
$$D[5] = \min(D[5], D[3] + A[3, 5]) = \min(22, 10 + 12) = 22$$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	∞	∞

此時，頂點 3 的相鄰頂點為頂點 5 與頂點 6 ，因此，頂點 1 到頂點 5 可透過頂點 2 與頂點 3，其距離就變成 22。

$$D[6] = \min(D[6], D[3] + A[3, 6]) = \min(\infty, 10 + 8) = 18$$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	∞ → 18	∞

此時，頂點 1 到頂點 6 可透過頂點 2 與頂點 3，其距離就變成 18。陣列中的內容如下：

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	18	∞

- **步驟三：**

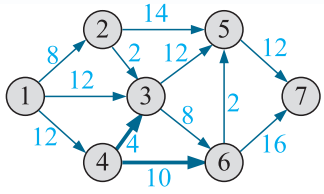
(1) 上面的陣列中，頂點 1 到頂點 4 的距離最短 (即 $D[4]$)，因此，將頂點 4 加入 S 的集合中，因此，$S = \{1, 2, 3, 4\}$, $V - S = \{5, 6, 7\}$。

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	18	∞

上次狀態：$S = \{1, 2, 3\}$, $V - S = \{4, 5, 6, 7\}$

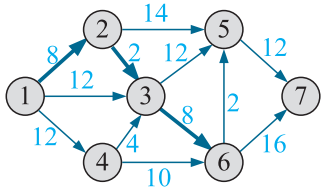
目前狀態：$S = \{1, 2, 3, 4\}$, $V - S = \{5, 6, 7\}$

(2) 頂點 4 的相鄰頂點為 3, 6 ，則：



$D[3] = \min(D[3], D[4] + A[4, 3]) = \min(10, 12 + 4) = 10$

$D[6] = \min(D[6], D[4] + A[4, 6]) = \min(18, 12 + 10) = 18$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	∞ → 18	∞

此時，頂點 1 到頂點 3，其距離就變成 10。頂點 1 到頂點 6，可透過頂點 2 與頂點 3，其距離就變成 18。

所以陣列內容如下：

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	18	∞

步驟四：

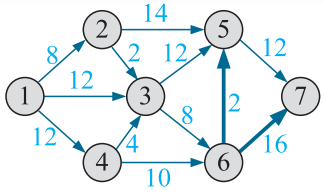
(1) 上面的陣列中，頂點 1 到頂點 6 的距離最短 (即 $D[6]$)，因此，將頂點 6 加入 S 的集合中，因此，$S = \{1, 2, 3, 4, 6\}$, $V - S = \{5, 7\}$

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22	18	∞

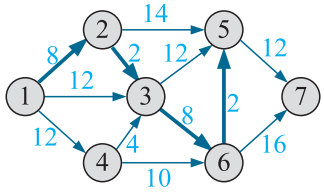
上次狀態：$S = \{1, 2, 3, 4\}$, $V - S = \{5, 6, 7\}$

目前狀態：$S = \{1, 2, 3, 4, 6\}$, $V - S = \{5, 7\}$

(2) 頂點 6 的相鄰頂點為 5, 7，則：



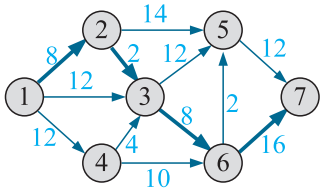
$D[5] = \min(D[5], D[6] + A[6, 5]) = \min(22, 18 + 2) = 20$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	22 → 20	18	∞

此時，頂點 6 的相鄰頂點為頂點 5 與頂點 7 ，因此，頂點 1 到頂點 5，可透過頂點 2 與頂點 3，其距離就變成 20。

$D[7] = \min(D[7], D[6] + A[6, 7]) = \min(\infty, 18 + 16) = 34$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	20	18	34

此時，頂點 1 到頂點 7，可透過頂點 2、頂點 3 及頂點 6，其距離就變成 34。陣列內容變為：

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	20	18	34

步驟五：

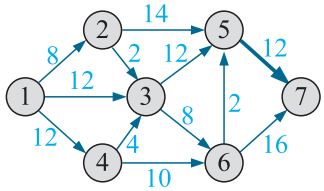
(1) 上面的陣列中，頂點 1 到頂點 5 的距離最短 (即 $D[5]$)，因此，將頂點 5 加入 S 的集合中，因此，$S = \{1, 2, 3, 4, 5, 6\}$, $V-S = \{7\}$。

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	20	18	34

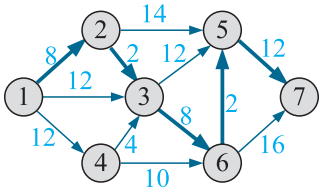
上次狀態：$S = \{1, 2, 3, 4, 6\}$, $V - S = \{5, 7\}$

目前狀態：$S = \{1, 2, 3, 4, 5, 6\}$, $V - S = \{7\}$

(2) 頂點 5 的相鄰頂點為 7，則：



$D[7] = \min(D[7], D[5] + A[5, 7]) = \min(34, 20 + 12) = 32$



$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	20	18	34 → 32

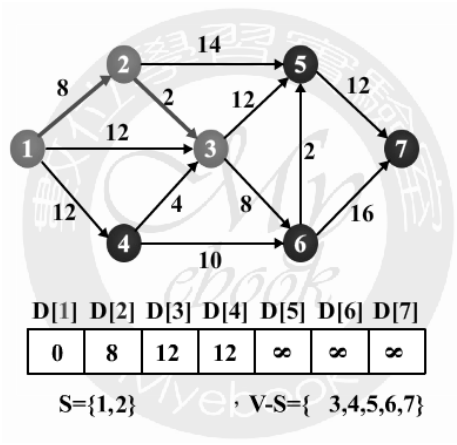
此時，頂點 5 的相鄰頂點為頂點 7 ，因此，頂點 1 到頂點 7，可透過頂點 2、頂點 3、頂點 6 及頂點 5，其距離就變成 32。

由於頂點 7 為最終頂點，將其加入 S 集合，此時，$S = \{1, 2, 3, 4, 5, 6, 7\}$ 因此，$V-S = \{\}$ ，最後陣列內容為：

$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$D[7]$
0	8	10	12	20	18	32

此陣列表示由頂點 1 到任一頂點的最短距離。

動畫圖解



7-9 拓樸排序

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，即是整個計畫的完成。因此，如果我們將「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**。

定義

在「頂點工作網路」中，若 V_i 工作 是 V_j 工作的前行者，則在線性的排列中，V_i 工作一定要在 V_j 工作的前面完成，此種特性稱之為「**拓樸排序**」**(Topological Sort)**。

演算法

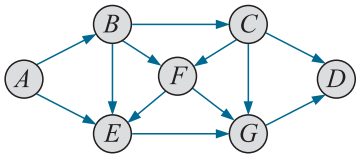
1. 在 AOV 網路中任意挑選一個沒有前行者的頂點。
2. 輸出此頂點，並將該頂點所連接的邊刪除。重複步驟 (1) 及步驟 (2) ，一直到全部的頂點皆輸出為止。

應用

這種事件應用的例子很多，例如：大專院校的選課資訊系統**先修或擋修等限制**、政府部門的**公文有層層呈報**和**會簽的流程**、**資訊系統開發**及**專案管理**等。

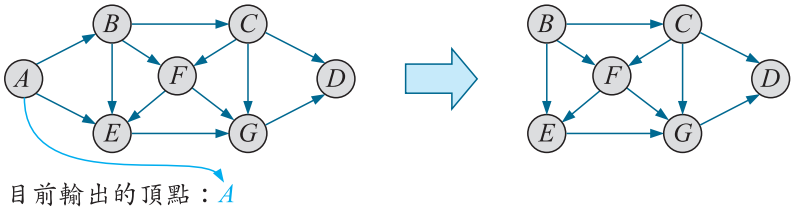
實例

假設我們現在有一個具有 7 個頂點的工作網路圖，請求出此工作網路圖的「拓樸排序」。

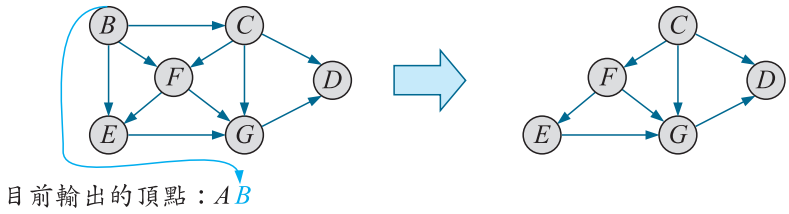


解答

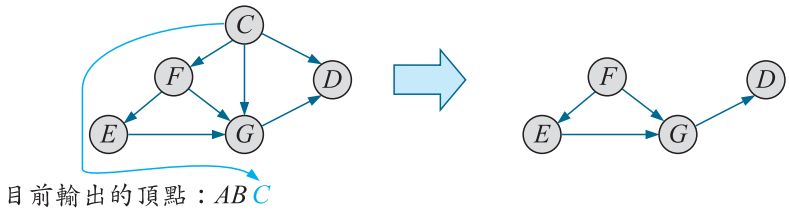
1. 輸出 *A* ，並刪除 <*A*, *B*> 與 <*A*, *E*> 兩個邊。



2. 輸出 *B* ，並刪除 <*B*, *C*>、<*B*, *E*> 及 <*B*, *F*> 三個邊。



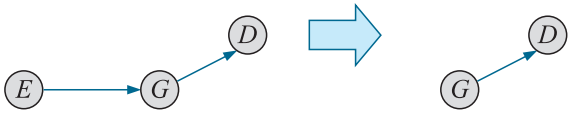
3. 輸出 *C* ，並刪除 <*C*, *D*>、<*C*, *F*> 及 <*C*, *G*> 三個邊。



4. 輸出 *F*，並刪除 <*F*, *E*> 與 <*F*, *G*> 兩個邊。

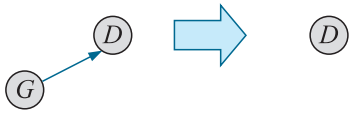
目前輸出的頂點：*ABC F*

5. 輸出 *E* ，並刪除 <*E*, *G*>。



目前輸出的頂點：*ABCF E*

6. 輸出 *G* ，並刪除 <*G*, *D*>。



目前輸出的頂點：*ABCFE G*

7. 最後，再輸出頂點 *D*，其所得的資料輸出順序為頂點 *A*, *B*, *C*, *F*, *E*, *G*, *D*，以上的輸出順序即為拓樸排序。

目前輸出的頂點：*ABCFEG D*

動畫圖解

參看隨書光碟。

單元評量

1. 若在 AOV 網路中，V_i 是 V_j 的前行者，則在線性的排列中，V_i 一定在 V_j 的前面，此種特性稱為何種排序方式？

(A) 基數排序 (B) 優先排序 (C) 堆積排序 (D) 拓樸排序

2. 有關於 AOV 網路之拓樸排序實際的應用，下列那一種情況適合？

(A) 大專院校的選課資訊系統先修或擋修等限制

(B) 政府部門的公文有層層呈報和會簽的流程

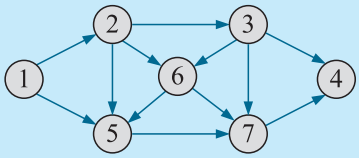
(C) 資訊系統開發

(D) 以上皆是

3. 下圖 AOV 網路之拓樸排序為何？

(A) 1, 2, 3, 5, 6, 7, 4 (B) 1, 2, 3, 6, 5, 7, 4

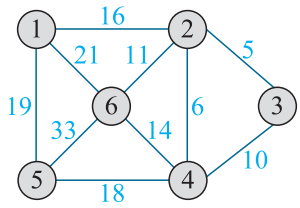
(C) 1, 2, 3, 7, 5, 6, 4 (D) 1, 2, 3, 6, 7, 5, 4



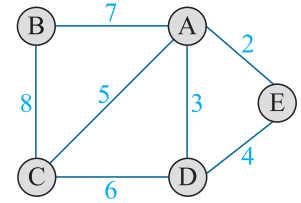
課後評量 EXERCISES

題庫來源：研究所及高普考

A-1. 下面圖形利用 Kruskal 演算法求出最小擴張樹。



A-2. 下面圖形利用 Kruskal 演算法求出最小擴張樹。



A-3. 試就下列兩小題的有向圖形列出其拓樸序列。