Problem 3： 資料結構—樹

子題 1：是否爲樹。(程式執行限制時間: 2 秒)

在資料結構中，樹狀結構是可以用來描述有分支的結構，包含 1 個或多個節點。其存在一個特殊的節點，稱爲根節點(root)，可連結若干子樹，也可以沒有子樹；從任一節點到根節點，都只有唯一的節點不重複路徑。例如 F 到 A 的路徑爲 F→B→A，其路徑長度爲 2，此路徑 F 到 A，中間所經過的節點集合爲{B}。

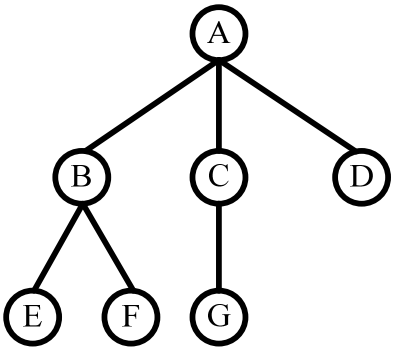


圖 3.1.1

在圖 3.1.1 中，有編號的圓形代表節點，A 爲根節點，B、C 及 D 均爲 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總節點數減 1，在樹上任意添加一條邊，就會產生迴圈；在樹上任意刪除一條邊，一顆樹就裂成兩棵樹(森林)。沒有迴圈的圖，就是樹或森林。若爲無根樹則是任一節點皆可爲根節點。

專有名詞介紹：

(1) 無父節點的節點爲根節點(Root)，如 A。
(2) 父節點 (Parent)：一節點的上層節點爲父節點，如 B 的父節點爲 A，如 G 的父節點爲 C。
(3) 子節點 (children)：一節點的下層節點爲子節點，如 B 的子節點有 E 及 F；C 的子節點有 G。
(4) 兄弟節點 (siblings)：有共同父節點的節點稱爲兄弟節點，如 B、C、D 互爲兄弟節點。
(5) 分支度 (degree)：一個節點的子樹個數稱爲其分支度，如 A 的分支度爲 3；B 的分支度爲 2；C 的分支度爲 1；E 的分支度爲 0。
(6) 樹葉節點(terminal node)：無子節點的節點，如 D、E、F、G。
(7) 內部節點 (non-terminal node)：樹葉以外的節點均爲內部節點，如 A、B、C。
(8) 階層或階度 (level)：A 爲階層 1；B、C、D 爲階層 2；E、F、G 爲階層 3。
(9) 高度 (height)：樹的最大階度，例如圖 3.1.1，因最大階度階度爲 3，則其樹的高度爲 3。

在圖 3.1.2 中的圖，則不是樹，因爲 B、E、F 三節點形成迴圈。

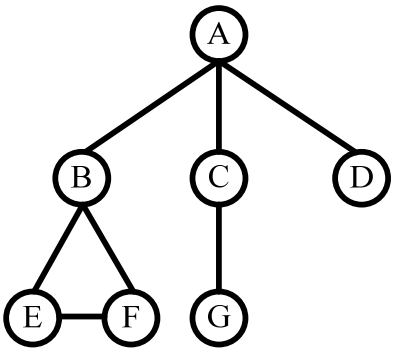


圖 3.1.2

寫一個程式，讀入一圖形的資料，然後回答該圖是否爲樹，在測試檔中，節點的編號不一定是連續的號碼。如果檢測的圖形是樹，則輸出 T，若該圖不是樹則輸出 F。

輸入說明：

第一列的數字 *n* 代表共有幾組資料要測試，$2 \leq n \leq 5$。

第二列起每一行代表一組測試資料。每組測試資料代表一圖形，內容爲邊的資料。每個邊以 2 個整數i, j表示，$0 \leq i, j \leq 20$ and $i \neq j$，其中 *i* 和 *j* 爲節點的編號，代表從 *i* 節點和 *j* 節點有邊相連，每組測試資料，同一列中，每個邊的資料以空白(　)隔開，而空白不限定一個，$|i, j|$爲邊的個數，$2 \leq |i, j| \leq 20$。

輸出說明：

每組測試資料輸出一列。輸出每組測試資料是否爲樹。若該圖是樹，則輸出 T；若該圖不是樹，則輸出 F。

輸入檔案 1：【檔名：in1.txt】

4

6,8　5,3　5,2　6,4　5,6　1,2　2,0

8,1　1,3　6,2　8,10　7,5　1,4　7,8　7,6　8,0

3,8　6,8　6,4　5,3　5,6　8,2　2,0

1,0　4,3　1,2

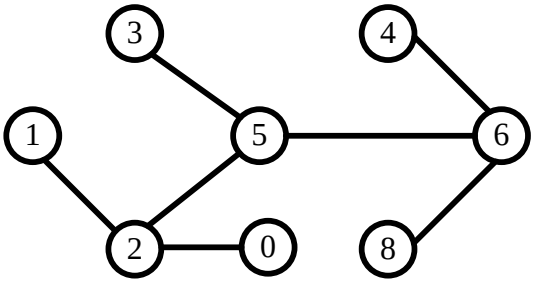


圖 3.1.3.1(**in1.txt)**

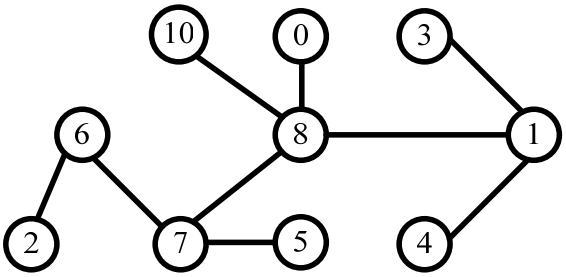


圖 3.1.3.2(**in1.txt)**

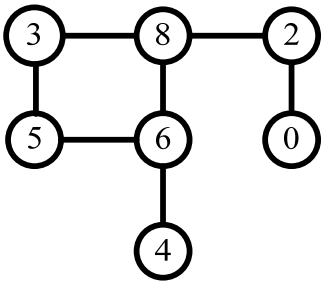


圖 3.1.3.3(**in1.txt)**

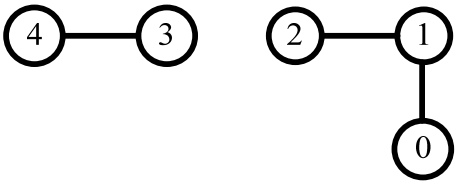


圖 3.1.3.4**(in1.txt)**

輸入檔案 2：【檔名：in2.txt】

3

4,3　2,3　2,1　1,0

1,2　2,3　4,0

1,2　2,3　3,1　4,5　5,0

輸出範例：【檔名：out.txt】

T

T

F

F

T

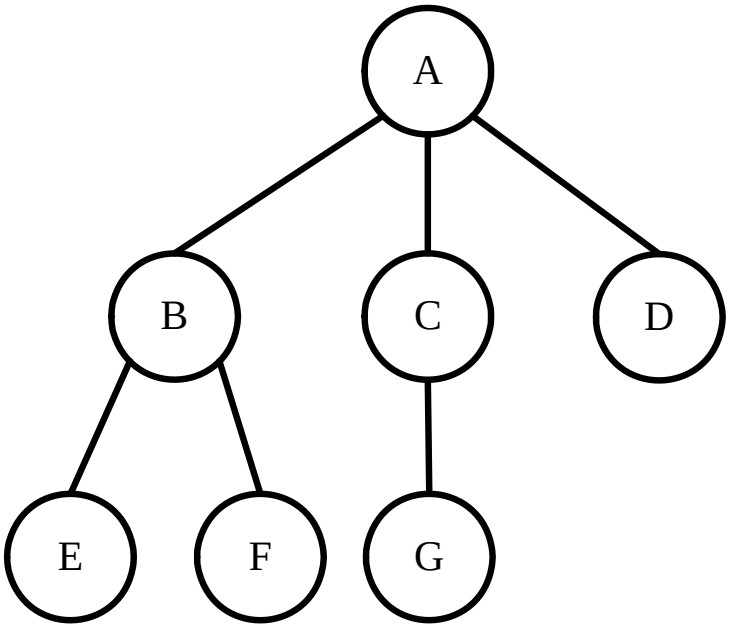
F

F

Problem 2：

子題 1：(程式執行限制時間: 2 秒)

在資料結構中，樹狀結構的定義：可以用來描述有分支的結構，節點個數是一或一個以上的有限集合，其特質如下：存在一個特殊的節點，稱爲樹根(root)，其餘的節點分爲 n>=0 個互斥集合，包括 T1 、T2 、 … 、Tn，每一個子集合也是一棵樹，每個子集合稱爲子樹，從任一節點到根，都只有唯一的路徑，例如 F 到 A 的路徑爲 F→B→A。在樹上的任二節點，路徑爲唯一。如下圖：



圖一

內有編號的圓形代表節點，A 爲根節點、B、C 及 D 均爲 A 的子節點。樹在各節點之間不可以有迴圈，樹上所有節點之間都相連通，在樹上任意添加一條邊，就會產生迴圈(環)，在樹上任意刪除一條邊，一顆樹就裂成兩棵樹(森林)。邊數等於點數減一。沒有迴圈(環)的圖，就是樹或森林。

專有名詞介紹：

(1) 存在一個節點稱爲根節點(Root)(無父節點的節點)，如 A。

(2) 父節點 (Parent)：每一節點的上層節點爲父節點，如 B 的父節點爲 A，如的父節點爲 C。

(3) 子節點 (children)：每一節點的下層節點爲子節點，如 B 的子節點有 E 及 F。C 的子節點有 G。

(4) 兄弟節點 (siblings)：有共同父節點的節點稱爲兄弟節點，如 B、C、D 之父節點爲 A。

(5) 分支度 (degree)：子樹的個數稱爲該節點之分支度，如 A 的分支度爲 3，B 的分支度爲 2，C 的分支度爲 1，E 的分支度爲 0。

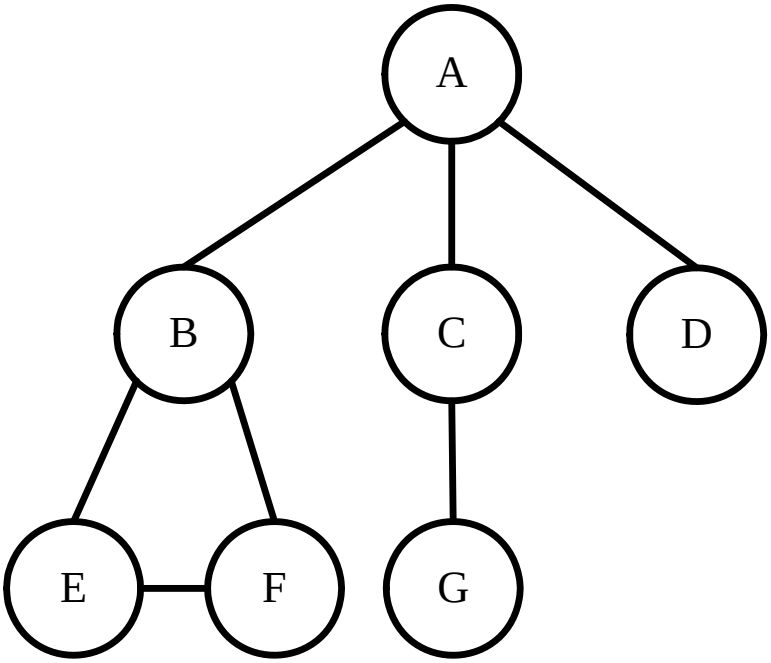
(6) 樹葉節點或終端節點 (terminal node)：無子節點的節點，如 D、E、F、G。

(7) 內部節點或非終端節點(non-terminal node)：樹葉以外的節點均爲內部節點或非終端節點，如 A、B、C。

(8) 階層或階度 (level)：A 爲階層 1，B、C、D 爲階層 2，E、F、G 爲階層 3。

(9) 高度 (height)：樹的最大階度，例如圖一，樹階度爲 3。

例如在圖二中，則不是樹，因爲 BEF 三節點形成迴圈。



圖二

寫一個程式，讀入一樹狀結構的資料，然後回答某節點到根節點的路徑長度及所經過的節點。

輸入說明：

第一列的數字 n, m；n 代表有幾組資料要測試，而 n 的值介於 1 和 10 之間。m 代表某節點。第二列以後則是每一組測試資料。每組測試資料代表樹狀結構的圖形，第一個值爲節點的個數，之後內容爲邊的資料，每組測試資料都是以 0 爲根節點。每個邊以 2 個整數 i,j 表示，0<= i,j <=30，此 2 整數爲節點的編號，代表從 i 節點和 j 節點有邊相連，邊的資料依節點編號順序描述其父節點，i 的值會 1,2,…,遞增，以 0 爲根節點的樹狀結構，j 的值爲 i 的父節點。

輸出說明：

每組測試資料輸出一列。輸出爲測試資料某節點到根節點的路徑長度及路徑所經過的節點。

輸入檔案 1：【檔名：in1.txt】

2, 4

7　1,0　2,1　3,1　4,3　5,3　6,1

5　1,4　2,0　3,2　4,2

輸入檔案 2：【檔名：in2.txt】

1, 4

6　1,3　2,4　3,5　4,5　5,0

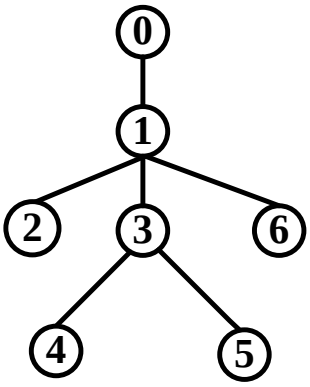
輸出範例：【檔名：out.txt】

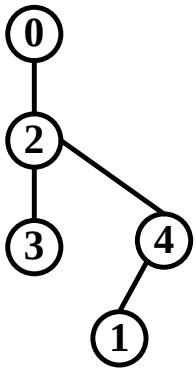
路徑長度爲 4:　4 -> 3 -> 1 -> 0

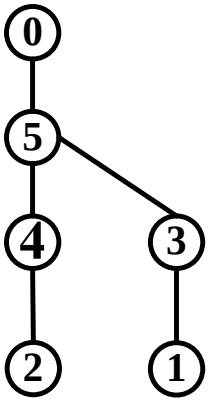
路徑長度爲 3:　4 -> 2 -> 0

路徑長度爲 3:　4 -> 5 -> 0

三組輸入的資料所對應到的樹狀結構。







Problem 2： 資料結構—樹

子題 1：樹的高度。(程式執行限制時間: 2 秒) 10 分

在資料結構中，樹狀結構是可以用來描述有分支的結構，其節點個數是一或一個以上的有限集合。其存在一個特殊的節點，稱為根節點(root)，可連結若干子樹，也可以沒有子樹，從任一節點到根節點，都只有唯一的路徑，例如 F 到 A 的路徑為 F→B→A，其路徑長度為 3。

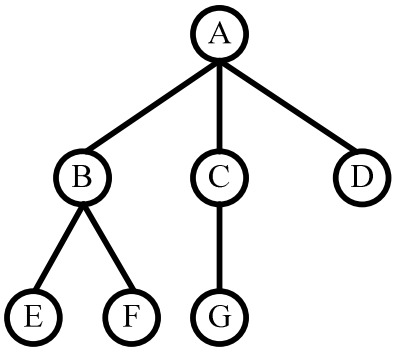


圖 2.1.1

在如圖 2.1.1 中，有編號的圓形代表節點，A 為根節點，B、C 及 D 均為 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總點節點減一，在樹上任意添加一條邊，就會產生迴圈；在樹上任意刪除一條邊，一顆樹就裂成兩棵樹(森林)。沒有迴圈的圖，就是樹或森林。

專有名詞介紹：

(1) 無父節點的節點為根節點(Root)，如 A。
(2) 父節點 (Parent)：一節點的上層節點為父節點，如 B 的父節點為 A，如 G 的父節點為 C。
(3) 子節點 (children)：一節點的下層節點為子節點，如 B 的子節點有 E 及 F；C 的子節點有 G。
(4) 兄弟節點 (siblings)：有共同父節點的節點稱為兄弟節點，如 B、C、D 互為兄弟節點。
(5) 分支度 (degree)：一個節點子樹的個數稱為其分支度，如 A 的分支度為 3；B 的分支度為 2：C 的分支度為 1；E 的分支度為 0。
(6) 樹葉節點(terminal node)：無子節點的節點，如 D、E、F、G。
(7) 內部節點 (non-terminal node)：樹葉以外的節點均為內部節點，如 A、B、C。
(8) 階層或階度 (level)：A 為階層 1；B、C、D 為階層 2；E、F、G 為階層 3。
(9) 高度 (height)：樹的最大階度，例如圖 2.1.1，因最大階度階度為 3，則其樹的高度為 3。

在圖 2.1.2 中的圖，則不是樹，因為 B、E、F 三節點形成迴圈。

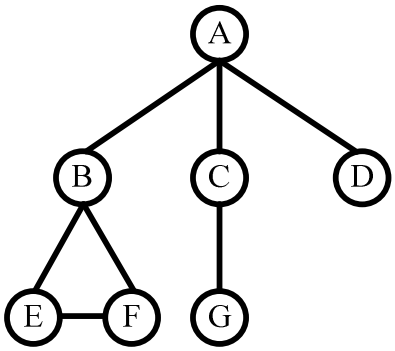


圖 2.1.2

寫一個程式，讀入一樹狀結構的資料，然後回答每組測試資料中，所有節點到根節點路徑長度最大的值。

輸入說明：

第一列的數字 *n* 代表共有幾組資料要測試，而 *n* 的值介於 1 和 5 之間。

第二列起則是每一組測試資料。每組測試資料代表一個樹狀結構，每組測試資料中的第一列值為節點的個數 *m*(>=2)，之後的 *m*-1 列的內容為邊的資料。每組測試資料都是以 0 代表根節點。每個邊以 2 個整數 *i*,*j* 表示，0<= *i*,*j* <=20，其為節點的編號，代表從 *i* 節點和 *j* 節點有一個邊相連，節點 *j* 為節點 *i* 的父節點，在測試檔中，邊的資料依節點編號順序描述，即 *i* 的值會 1,2,…, *m*-1 遞增，每組測試資料皆以 0 為根節點。在一行空行之後為下一組的測試資料。

輸出說明：

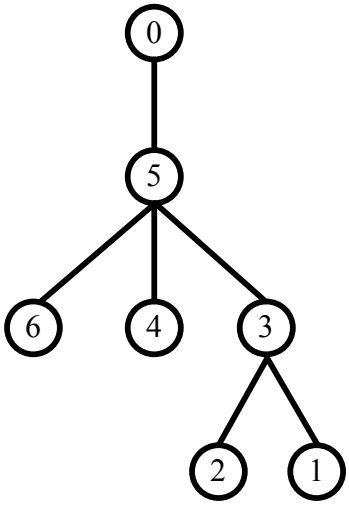
每組測試資料輸出一列，在測試資料中，計算所有節點到根節點 0 的路徑長度，輸出每組測試資料中路徑長度最大的值。

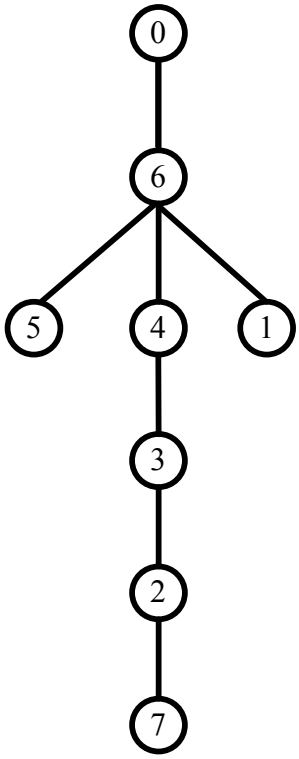
輸入檔案 1：【檔名：in1.txt】

```
2
7
1,3
2,3
3,5
4,5
5,0
6,5

8
1,6
2,3
3,4
4,6
5,6
6,0
7,2
```

二組輸入的資料所對應到的樹狀結構。





輸入檔案 2：【檔名：in2.txt】

3
4
1,2
2,3
3,0

4
1,0
2,0
3,0

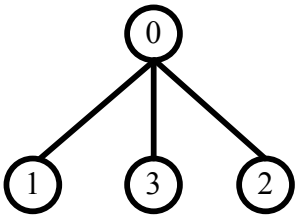
2
1,0

輸出範例：【檔名：out.txt】

4
6

4
2
2

三組輸入的資料所對應到的樹狀結構。



子題 2：樹和樹葉節點。(程式執行限制時間: 3 秒) 14%

寫一個程式，讀入一圖形的資料，然後回答該圖是否為樹，在測試檔中，節點的編號不一定是連續的號碼。樹葉節點為無子節點的節點，若以節點 0 為圖形之根節點，在圖 2.2.1.1 中的節點 1、3、4 及 8 是樹葉節點；在圖 2.2.1.2 中的節點 2、3、4、5 及 10 是樹葉節點；在圖 2.2.2.1 中的節點 4 為樹葉節點。我們假設如果檢測的圖形是樹，其根節點為節點 0，若測試圖是樹則輸出樹葉節點的個數，若該圖不是樹則輸出 F。

輸入說明：

第一列的數字 *n* 代表有幾組資料要測試，而 *n* 的值介於 1 和 5 之間。

第二列起每一行代表一組測試資料。每組測試資料代表一圖形，內容為邊的資料。每個邊以 2 個整數 *i,j* 表示，0<= *i,j* <=20，其中 *i* 和 *j* 為節點的編號，代表從 *i* 節點和 *j* 節點有邊相連，同一列中，每個邊的資料以空白(　)隔開，而空白不限定一個。測試資料中的 0, 0 代表此組輸入資料結束，它不代表一個邊，而是一個稱為節點 0 的節點，如果該圖是一棵樹，節點 0 為圖形之根節點。

輸出說明：

每組測試資料輸出一列。輸出每組測試資料是否為樹。若該圖是樹，則輸出樹葉節點的個數，但根節點不累計為樹葉節點；若該圖不是樹，則輸出 F。（輸出字母為大寫，選手請注意。）

輸入檔案 1：【檔名：in1.txt】

5

6,8 5,3 5,2 6,4 5,6 1,2 2,0 0,0

8,1 1,3 6,2 8,10 7,5 1,4 7,8 7,6 8,0 0,0

3,8 6,8 6,4 5,3 5,6 8,2 2,0 0,0

0,0

1,0 0,0

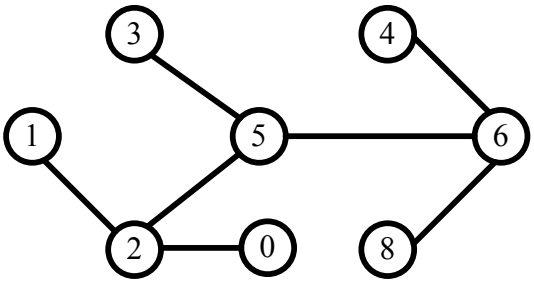


圖 2.2.1.1(**in1.txt)**

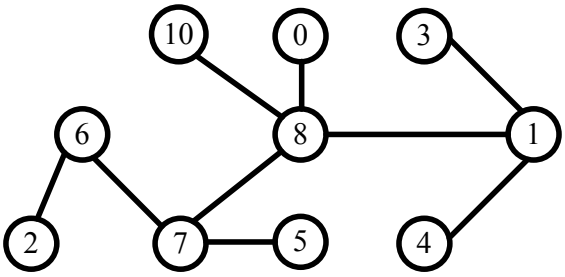


圖 2.2.1.2(**in1.txt)**

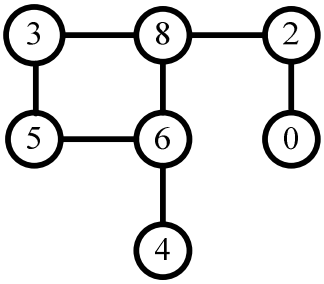


圖 2.2.1.3(**in1.txt)**

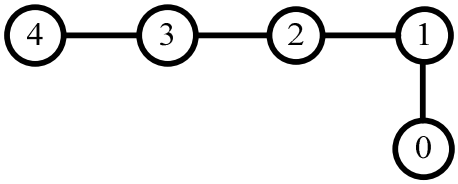


圖 2.2.2.1(**in2.txt)**

輸入檔案 2：【檔名：in2.txt】

4

4,3 2,3 2,1 1,0 0,0

1,1 0,0

1,2 2,3 4,0 0,0

1,2 2,3 3,1 4,5 5,0 0,0

輸出範例：【檔名：out.txt】

4

5

F

0

1

1

F

F

F

Problem 2： 資料結構—樹

子題 1：樹葉節點到根節點之路徑。(程式執行限制時間: 2 秒)

在資料結構中，樹狀結構是可以用來描述有分支的結構，其節點個數是一或一個以上的有限集合。其存在一個特殊的節點，稱爲根節點(root)，可連結若干子樹，也可以沒有子樹，從任一節點到根節點，都只有唯一的路徑，例如 F 到 A 的路徑爲 F→B→A，其路徑長度爲 2，此路徑 F 到 A，中間所經過的節點集合爲{B}。

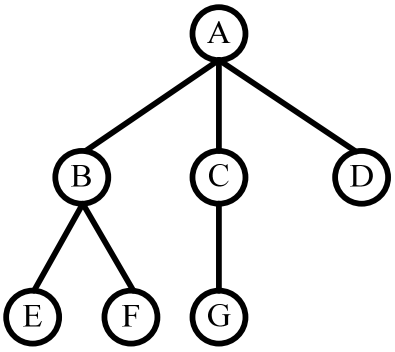


圖 2.1.1

在如圖 2.1.1 中，有編號的圓形代表節點，A 爲根節點，B、C 及 D 均爲 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總點節點減一，在樹上任意添加一條邊，就會產生迴圈；在樹上任意刪除一條邊，一顆樹就裂成兩棵樹(森林)。沒有迴圈的圖，就是樹或森林。若爲無根樹則是任一節點皆可爲根節點。

專有名詞介紹：

(1) 無父節點的節點爲根節點(Root)，如 A。
(2) 父節點 (Parent)：一節點的上層節點爲父節點，如 B 的父節點爲 A，如 G 的父節點爲 C。
(3) 子節點 (children)：一節點的下層節點爲子節點，如 B 的子節點有 E 及 F；C 的子節點有 G。
(4) 兄弟節點 (siblings)：有共同父節點的節點稱爲兄弟節點，如 B、C、D 互爲兄弟節點。
(5) 分支度 (degree)：一個節點子樹的個數稱爲其分支度，如 A 的分支度爲 3；B 的分支度爲 2；C 的分支度爲 1；E 的分支度爲 0。
(6) 樹葉節點(terminal node)：無子節點的節點，如 D、E、F、G。
(7) 內部節點 (non-terminal node)：樹葉以外的節點均爲內部節點，如 A、B、C。
(8) 階層或階度 (level)：A 爲階層 1；B、C、D 爲階層 2；E、F、G 爲階層 3。
(9) 高度 (height)：樹的最大階度，例如圖 2.1.1，因最大階度階度爲 3，則其樹的高度爲 3。

在圖 2.1.2 中的圖，則不是樹，因爲 B、E、F 三節點形成迴圈。

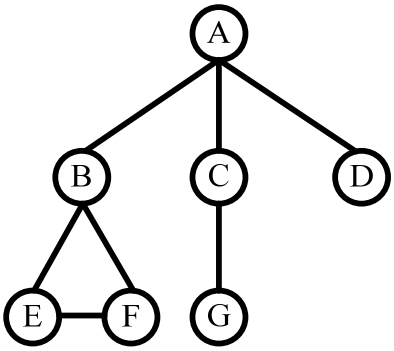


圖 2.1.2

寫一個程式，讀入一樹狀結構的資料，然後回答每組測試資料中，所有樹葉節點到根節點之路徑，路徑中間所經過的節點集合。

輸入說明：

第一列的數字 *n* 代表共有幾組資料要測試，而 *n* 的值介於 1 和 5 之間。

第二列起則是每一組測試資料。每組測試資料代表一個樹狀結構，每組測試資料中的第一列值爲節點的個數 *m*(>=2)，之後的 *m* 列的內容爲邊的資料。每個邊以 2 個整數 *i*,*j* 表示，0<= *i*,*j* <=80，其爲節點的編號，代表從 *i* 節點和 *j* 節點有一個邊相連，節點 *j* 爲節點 *i* 的父節點，若 *j* 爲 99，則 *j* 爲這組測試資料的根節點，邊的資料依節點編號順序描述，即 *i* 的值會 0,1,2,…, *m*-1 遞增。在一行空行之後爲下一組的測試資料。

輸出說明：

依每組測試資料找出所有樹葉節點，列出每個樹葉節點到根節點之路徑，中間所經過的節點集合，中間以”,”隔開，依路徑經過順序先後列出。若節點集合爲空集合則輸出 N。

每組測試資料之後輸出換行。

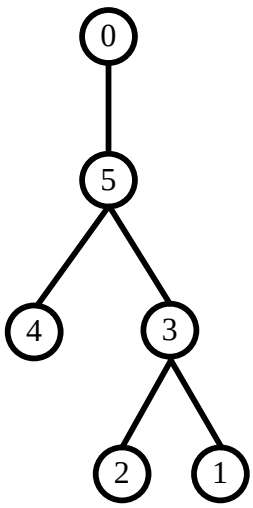
輸入檔案 1：【檔名：in1.txt】

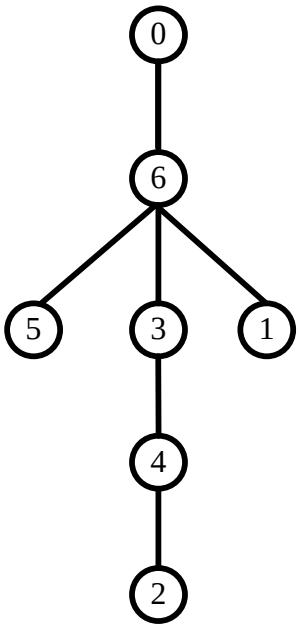
3
6
0,99
1,3
2,3
3,5
4,5
5,0

7
0,99
1,6
2,4
3,6
4,3
5,6
6,0

4
0,99
1,2
2,3
3,0

三組輸入的資料所對應到的樹狀結構。





輸入檔案 2：【檔名：in2.txt】

2
4
0,99
1,0
2,0
3,0

2
0,1
1,99

輸出範例：【檔名：out.txt】

1:{3, 5}
2:{3, 5}
4:{5}

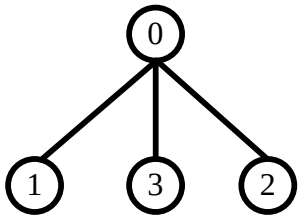
1:{6}
2:{4, 3, 6}
5:{6}

1:{2, 3}

1:N
2:N
3:N

1:N

二組輸入的資料所對應到的樹狀結構。



子題 2：樹、內部節點和重複使用的邊。(程式執行限制時間: 3 秒)

寫一個程式，讀入一圖形的資料，然後回答該圖是否爲樹，在測試檔中，節點的編號爲連續的號碼。爲無根樹，任一節點皆可爲根節點。在圖 2.2.1.1 中的節點 2、4 及 5 是內部節點；在圖 2.2.1.2 中的節點 1、3 及 6 是內部節點；在圖 2.2.2.1(a)中的節點 2、4 及 5；在圖 2.2.2.1(b)中的節點 1、3 及 6 爲內部節點。若測試圖是樹則輸出內部節點，若同組測試資料有重複使用的邊(例如圖 2.2.2.1(a)(b)中 6-4 和 4-6)，在印出”；” 之後，印出重複使用的邊。若該圖不是樹則輸出 F。在圖 2.2.2.2(a)中的節點 2、4 及 5 爲內部節點；在圖 2.2.2.1(b) 該圖不是樹，則輸出 F；若同組測試資料中，若有測試圖不是樹則這組測試資料不考慮重複使用邊的問題。

輸入說明：

第一列的數字 *n* 代表有幾組資料要測試，而 *n* 的值介於 1 和 5 之間。

第二列起爲測試資料。每組測試資料代表二個圖形，內容爲邊的資料。每個邊以 2 個整數 *i,j* 表示，1<= *i,j* <=80，其中 *i* 和 *j* 爲節點的編號，代表從 *i* 節點和 *j* 節點有邊相連，每一列對應到一個圖形測試資料，每個邊的資料以空白(　)隔開，而空白不限定一個。每組測試資料以空行分開。

輸出說明：

每組測試資料輸出二列，之後輸出換行。輸出每一列對應到一個圖形測試資料，判斷是否爲樹。若該圖是樹，則列出每顆樹的內部節點；若同組測試資料有重複使用的邊，在印出”；” 之後，印出重複使用的邊。若該圖不是樹，則輸出 F（輸出字母爲大寫，選手請注意）。若同組測試資料中，若有測試圖不是樹則這組測試資料不考慮重複使用邊的問題。

輸入檔案 1：【檔名：in1.txt】

1

1-5 2-3 2-4 4-5 4-6

1-3 1-4 1-6 2-6 3-5

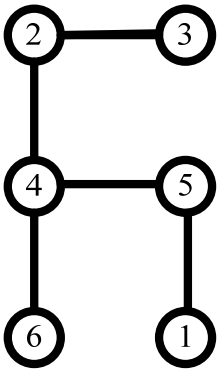


圖 2.2.1.1

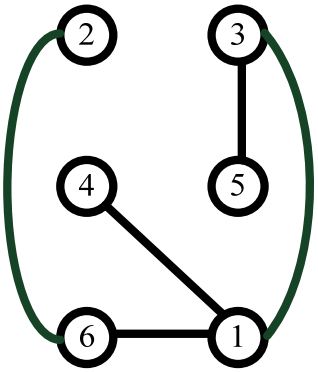


圖 2.2.1.2

輸入檔案 2：【檔名：in2.txt】

2

1-5 2-3 2-4 4-5 6-4

1-3 1-6 4-6 2-6 3-5

1-5 2-3 2-4 4-5 4-6

2-5 1-3 2-6 4-1 5-6

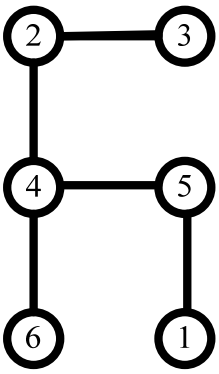


圖 2.2.2.1(a)

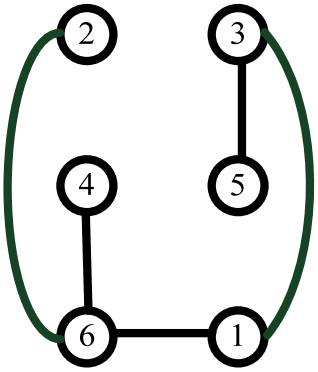


圖 2.2.2.1 (b)

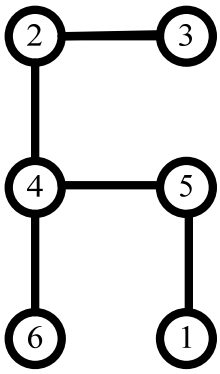


圖 2.2.2.2(a)

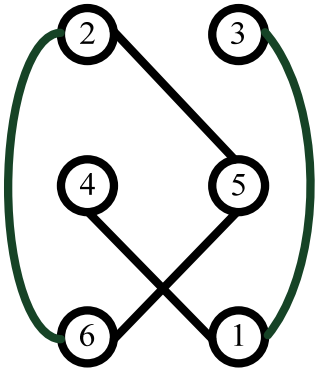


圖 2.2.2.2(b)

輸出範例：【檔名：out.txt】

2, 4, 5

1, 3, 6

2, 4, 5; 6-4

1, 3, 6; 4-6

2, 4, 5

F

Problem 3： 資料結構—樹

子題 1：樹葉節點或內部節點到根節點之路徑。(程式執行限制時間: 2 秒) 12 分

在資料結構中，樹狀結構是可以用來描述有分支的結構，包含 1 個或多個節點。其存在一個特殊的節點，稱爲根節點(root)，可連結若干子樹，也可以沒有子樹；從任一節點到根節點，都只有唯一的節點不重複路徑。例如 F 到 A 的路徑爲 F→B→A，其路徑長度爲 2，此路徑 F 到 A，中間所經過的節點集合爲{B}。

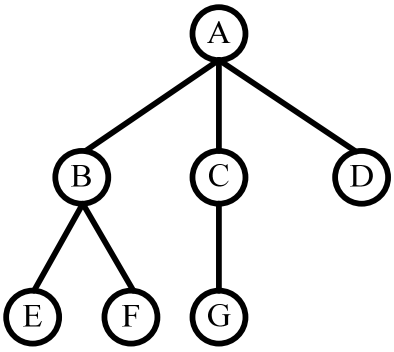


圖 3.1.1

在圖 3.1.1 中，有編號的圓形代表節點，A 爲根節點，B、C 及 D 均爲 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總節點數減 1，在樹上任意添加一條邊，就會產生迴圈；在樹上任意刪除一條邊，一顆樹就裂成兩棵樹(森林)。沒有迴圈的圖，就是樹或森林。若爲無根樹則是任一節點皆可爲根節點。

專有名詞介紹：

(1) 無父節點的節點爲根節點(Root)，如 A。
(2) 父節點 (Parent)：一節點的上層節點爲父節點，如 B 的父節點爲 A，如 G 的父節點爲 C。
(3) 子節點 (children)：一節點的下層節點爲子節點，如 B 的子節點有 E 及 F；C 的子節點有 G。
(4) 兄弟節點 (siblings)：有共同父節點的節點稱爲兄弟節點，如 B、C、D 互爲兄弟節點。
(5) 分支度 (degree)：一個節點的子樹個數稱爲其分支度，如 A 的分支度爲 3；B 的分支度爲 2；C 的分支度爲 1；E 的分支度爲 0。
(6) 樹葉節點(terminal node)：無子節點的節點，如 D、E、F、G。
(7) 內部節點 (non-terminal node)：樹葉以外的節點均爲內部節點，如 A、B、C。
(8) 階層或階度 (level)：A 爲階層 1；B、C、D 爲階層 2；E、F、G 爲階層 3。
(9) 高度 (height)：樹的最大階度，例如圖 3.1.1，因最大階度階度爲 3，則其樹的高度爲 3。

在圖 3.1.2 中的圖，則不是樹，因爲 B、E、F 三節點形成迴圈。

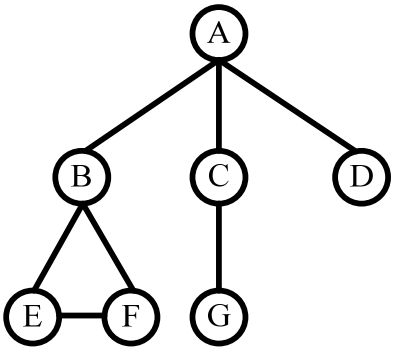


圖 3.1.2

寫一個程式，讀入一樹狀結構的資料，依每組測試資料找出不包含根節點的所有**節點**，到根節點之路徑，各路徑長度爲中間所經過的節點形成的集合之元素個數加 1。若中間所經過的節點集合爲空集合則路徑長度爲 1。請計算每組測試資料中，每個節點到根節點之路徑長度爲某 *k* 值的節點個數。

輸入說明：

第一列的數字 *n* 代表共有幾組資料要測試，$2 \le n \le 5$。第二列起則是每一組測試資料。

每組測試資料代表一個樹狀結構，每組測試資料中的第一列值爲 *m,k*；$m(\ge 2)$爲節點的個數；$k(0 < k < m)$爲路徑長度，之後的 *m* 列的內容爲邊的資料。每個邊以 2 個整數 *i,j* 表示，$0 \le i, j \le 80$，*i,j* 爲節點的編號，代表 *i* 節點和 *j* 節點相連的一個邊，且節點 *j* 爲節點 *i* 的父節點。若 *j* 爲 99，則 *i* 爲這組測試資料的根節點。輸入資料中，邊的資料依節點編號由小到大依序描述，即 *i* 的值會 0,1,2,…, *m*-1 遞增。在一行空行之後爲下一組的測試資料。

輸出說明：

計算每組測試資料中，不包含根節點的所有節點到根節點之路徑長度，計算路徑長度爲某 *k* 值的節點個數，每組測試資料之後輸出換行。

在 in1.txt 檔案中，第一組測試資料的第一列值m, k爲$7, 3$，代表共有 7 個節點，且尋找的路徑長度 *k* 爲 3。節點到根節點之路徑長度爲 3 的節點集合爲{1,2,6}，集合元素個數爲 3；第二組測試資料中，第一列值$m, k(8, 2)$，*k* 爲 2，節點到根節點之路徑長度爲 2 的節點集合爲{1,3,5,7}，集合元素個數爲 4；第三組測試資料中，第一列值$m, k(4, 3)$，*k* 爲 3，節點到根節點之路徑長度爲 3 的節點集合爲{0}，集合元素個數爲 1。其最後輸出結果分別爲 3、 4 和 1。

在 in2.txt 檔案的第一組測試資料中，第一列值$m, k(4, 1)$，*k* 爲 1，節點到根節點之路徑長度爲 1 的節點集合爲{1,2,3}，集合元素個數爲 3；第二組測試資料中，第一列值$m, k(3, 2)$，*k* 爲 2，節點到根節點之路徑長度爲 2，其節點集合爲空集合，集合元素個數爲 0。其最後輸出結果分別爲 3 和 0。

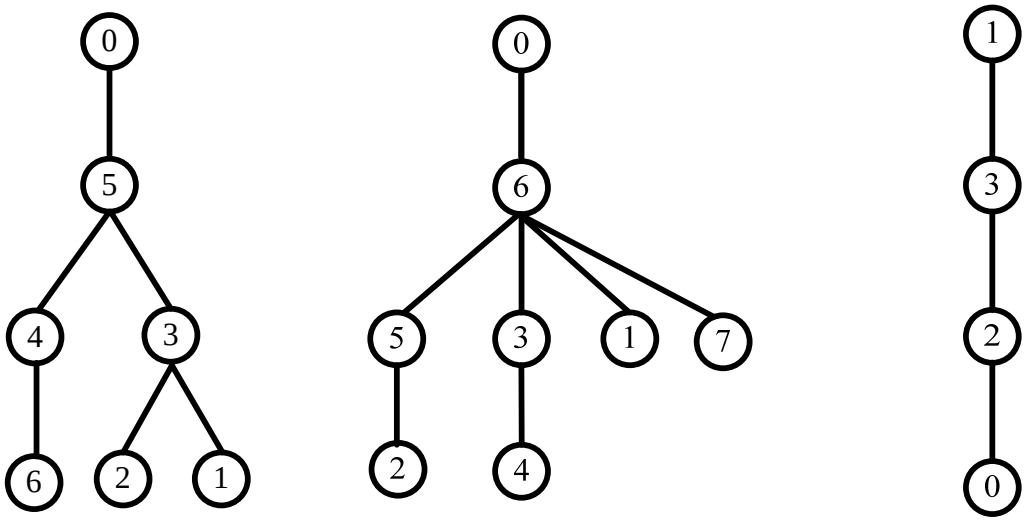
輸入檔案 1：【檔名：in1.txt】

```
3
7,3
0,99
1,3
2,3
3,5
4,5
5,0
6,4

8,2
0,99
1,6
2,5
3,6
4,3
5,6
6,0
7,6

4,3
0,2
1,99
2,3
3,1
```

三組輸入的資料所對應到的樹狀結構。由左而右，分別代表第 1、2 及 3 組資料。



輸入檔案 2：【檔名：in2.txt】

2
4,1
0,99
1,0
2,0
3,0

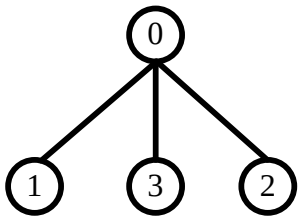
3,2
0,99
1,0
2,0

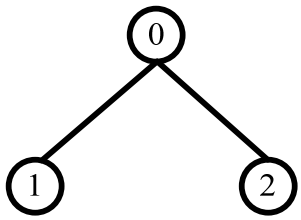
輸出範例：【檔名：out.txt】

3
4
1

3
0

二組輸入的資料所對應到的樹狀結構。由左而右，分別代表第 1 及 2 組資料。





子題 2：樹、內部節點最多只能當 1 次和沒有重複使用的邊。

(程式執行限制時間: 3 秒)　18 分

寫一個程式，讀入一組圖形的資料，每組測試資料中，每列代表一個圖形。請判斷該組測試資料，每列的圖形是否**都爲樹；**且該組測試資料中，同一節點只能在其中一圖形爲**內部節點；**且該組測試資料中，**各圖的邊均不重複使用**。在測試檔中，節點的編號爲連續的號碼，同一組圖形的資料節點相同。本題爲無根樹，任一節點皆可爲根節點。

在圖 3.2.1.1(a)中的節點 2、4 及 5 是內部節點；在圖 3.2.1.1(b)中的節點 1、3 及 6 是內部節點；在圖 3.2.1.1(a)與(b)中，這二列圖形**都爲樹；**節點 1、2、3、4、5 及 6 在這組測試資料中都只在一個圖形中作爲**內部節點(內部節點最多只能當 1 次)**；並且**沒有重複使用的邊**。則這組測試資料結果輸出爲 T。

在圖 3.2.2.1(a)中的節點 2、4 及 5 及在圖 3.2.2.1(b)中的節點 1、3 及 6 爲內部節點，這二列圖形都爲樹。若同組測試資料中，有重複使用的邊，例如圖 3.2.2.1(a) 與(b)中 6-4 和 4-6 視爲相同的邊，在這組測試資料中，因爲，**有重複使用的邊**，則這組測試資料結果輸出爲 F。

在圖 3.2.2.2(a)中，這一列圖形爲樹；節點 2、4 及 5 爲內部節點。在圖 3.2.2.2(b) 該圖不是樹；在這組測試資料中，因爲，存在任一圖形**不是樹**，則這組測試資料結果輸出爲 F。

在圖 3.2.2.3(a)中的節點 2、4 及 5 及在圖 3.2.2.3(b)中的節點 1、2 及 6 爲內部節點，這二列圖形都爲樹。若同組測試資料中，有節點重複當內部節點，例如圖 3.2.2.3(a) 與(b)中節點 2 在這二圖形中爲內部節點。在這組測試資料中，因爲，在某棵樹 3.2.2.3(a)中節點 2，爲內部節點，在其他樹棵樹 3.2.2.3(b)中節點 2 也是內部節點，**有節點重複當內部節點**，則這組測試資料結果輸出爲 F。

輸入說明：

第一列的數字 *n* 代表有幾組資料要測試，$1 \le n \le 5$。第二列起爲測試資料。

每組測試資料中，第一列的數字 *k* 代表 *k* 個圖形($2 \le k \le 10$)，接下來 *k* 列的內容爲 *k* 個圖形邊的資料。每個邊以 2 個整數 *i,j* 表示，$1 \le i, j \le 60$，其中 *i* 和 *j* 爲節點的編號，代表從 *i* 節點和 *j* 節點有邊相連。每一列的圖形邊的資料，對應到同一個圖形測試資料，不同邊的資料以空白(　)隔開，而空白不限定一個，節點的個數比邊的個數多 1。每組測試資料以空行分開。

輸出說明：輸出字母爲大寫，選手請注意。

每組測試資料輸出一列，輸出每一列對應到一組的測試資料。同一組的測試資料中，同一組的 *k* 個圖形中，必須同時滿足下列三個條件，才能輸出 T：

1. 每一個圖形都是樹；
2. 每個節點在這組測試資料中都只當過內部節點 1 次或 0 次；
3. 同組測試資料中沒有重複使用的邊。

若同一組的 *k* 個圖形，下列三個情況，存在其中一個或一個以上，則輸出 F：

1.在 *k* 個的圖形中存在一個或一個以上的圖形不是樹。

2.若同一組的測試資料中，在某棵樹中，其內部節點在其他樹中也是內部節點。

3.若同一組測試資料中，有重複使用的邊。

輸入檔案 1：【檔名：in1.txt】

3

2

1-5 2-3 2-4 4-5 4-6

1-3 1-4 1-6 2-6 3-5

4

1-2 1-3 1-4 1-5 5-6 5-7 5-8

2-3 2-4 2-5 2-6 6-1 6-7 6-8

3-4 3-7 3-5 3-6 7-8 7-1 7-2

4-8 4-5 4-6 4-7 8-1 8-2 8-3

6

1-2 1-3 1-4 1-5 1-6 1-7 7-8 7-9 7-10 7-11 7-12

1-8 2-3 2-4 2-5 2-6 2-7 2-8 8-9 8-10 8-11 8-12

1-9 2-9 3-4 3-5 3-6 3-7 3-8 3-9 9-10 9-11 9-12

1-10 2-10 3-10 4-5 4-6 4-7 4-8 4-9 4-10 10-11 10-12

1-11 2-11 3-11 4-11 5-6 5-7 5-8 5-9 5-10 5-11 11-12

1-12 2-12 3-12 4-12 5-12 6-7 6-8 6-9 6-10 6-11 6-12

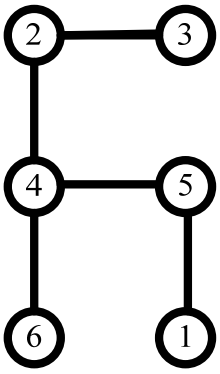


圖 3.2.1.1(a)

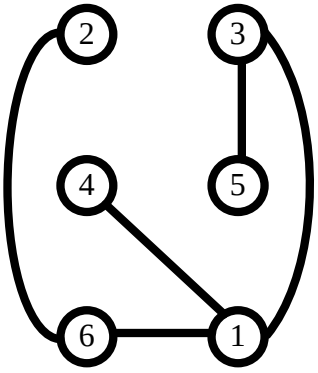


圖 3.2.1.1(b)

輸入檔案 2：【檔名：in2.txt】

3

2

1-5 2-3 2-4 4-5 6-4

1-3 1-6 4-6 2-6 3-5

2

1-5 2-3 2-4 4-5 4-6

2-5 1-3 2-6 4-1 5-6

2

1-5 2-3 2-4 4-5 4-6

2-5 1-3 2-6 4-1 3-6

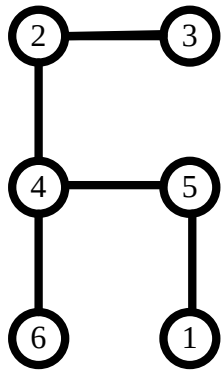


圖 3.2.2.1(a)

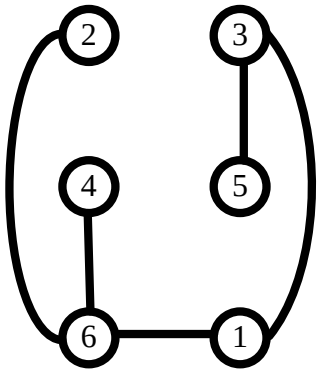


圖 3.2.2.1 (b)

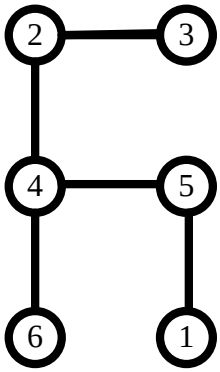


圖 3.2.2.2(a)

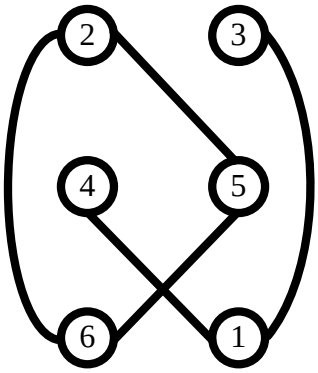


圖 3.2.2.2(b)

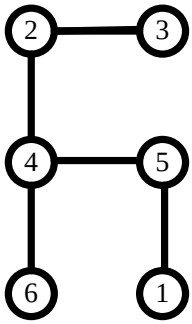


圖 3.2.2.3(a)

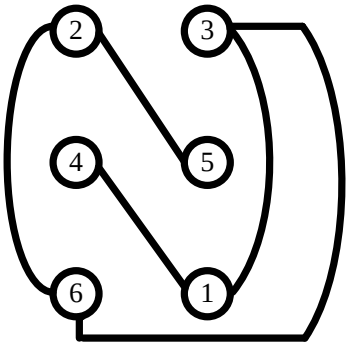


圖 3.2.2.3 (b)

輸出範例：【檔名：out.txt】

T

T

T

F

F

F

子題 2：列出所有樹的某節點到根節點之路徑長度。(程式執行限制時間: 2 秒) 15 分

在資料結構中，樹狀結構是可以用來描述有分支的結構，其包含 1 個或多個節點。每棵樹存在一個特殊的節點，稱為根節點(root)，可連結若干子樹，也可以沒有子樹；從任一節點到根節點，都只有唯一一條的節點不重複路徑。例如 F 到 A 的路徑為 F→B→A，其路徑長度為 2，此路徑 F 到 A，A 為此樹之根節點，路徑中間所經過的節點集合為{B}。

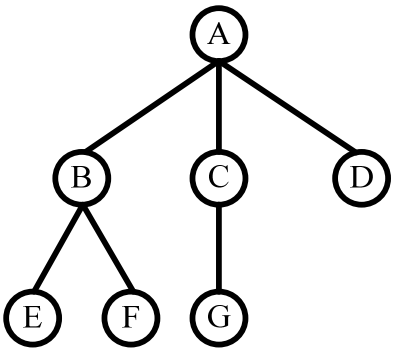


圖 4.2.1

在圖 4.2.1 中，有編號的圓形代表節點，A 為根節點，B、C 及 D 均為 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總節點數減 1，在樹上任意添加一條邊，就會產生迴圈；在樹上任意刪除一條邊，一顆樹就裂成兩棵樹(森林)。沒有迴圈的圖，就是樹或森林。若為無根樹則是任一節點皆可為根節點。

寫一個程式，讀入多棵樹狀結構的資料，依每組測試資料算出某節點到根節點之路徑長度，需分別算出各棵樹的結果(各路徑長度為中間所經過的節點形成的集合之元素個數加 1，若中間所經過的節點集合為空集合則路徑長度為 1)。

輸入說明：

第一列的數字 ***n*** 代表共有幾組資料要測試，$2 \le n \le 5$。第二列起則是每一組測試資料。每組測試資料代表一個樹狀結構，每組測試資料中的第一列值為m, k, v，
$3 \le m \le 256, 2 \le k \le 8, 0 \le v \le 255$；$m$為節點的個數、$k$為樹的個數、$v$為某節點($v$不會是根節點)。之後的 ***m*** 列的內容為k棵樹k個邊的資料，每列的內容有$k+1$個值，第一個值為節點 *i*；之後k個值依序代表節點 *i* 在第 0 棵樹到第 *k*-1 棵樹中的父節點。同一列中，**每個父節點的資料以一個或多個空白隔開**。若父節點編號為 999，則 *i* 為這組測試資料的根節點。在測試資料中，所有樹的根節點編號均相同。且在測試資料中，內容依節點編號依序描述，即節點 *i* 的值為 0,1,2,…, *m*-1 遞增。接下來為下一組的測試資料。

輸出說明：

計算每組測試資料中，分別算出各棵樹某個節點到根節點之路徑長度，輸出的數字和數字之間需以逗號隔開，每組測試資料輸出一列。

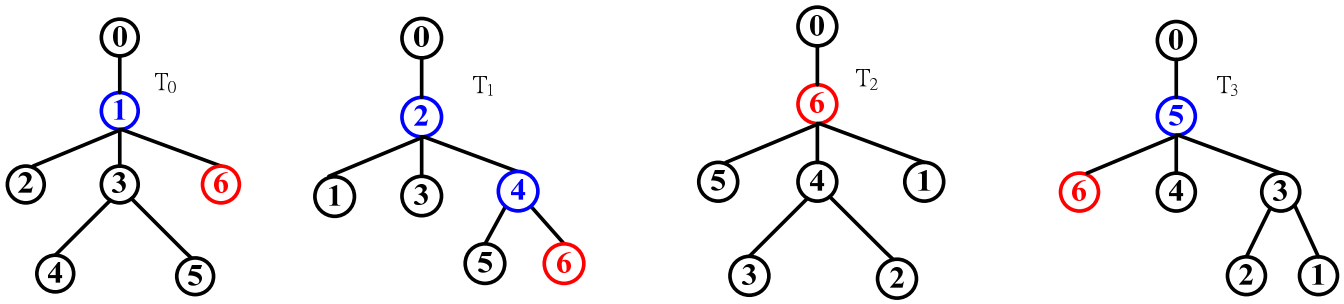
在 in1.txt 檔案中，第一組測試資料的第一列值$m, k, v = 7, 4, 6$，代表共有 7 個節點、4 棵樹，程式應該算出節點 *v*，即節點 6，到不同樹的根節點之路徑長度。

在 *m* 列資料中，第一列的資料：0　999　999　999　999。每列的內容為$k+1$個值，第一個值為節點 *i*，之後k個值分別代表節點 *i* 在第 0~3 棵樹中的父節點。因為父節點為 999，則節點 *i*=0 為這組測試資料的根節點，所以這組測試資料，根節點為 0。

在 *m* 列資料中，第二列的資料：1　0　2　6　3。第一個值 1，節點 *i* 為 1，之後k個值分別代表節點 *i* 在第 0~3 棵樹中的父節點，依序為 0　2　6　3。

在 *m* 列資料中，第三列的資料：2　1　0　4　3。第一個值 2 節點 *i* 為 2；之後k個值分別代表節點 *i* 在第 0~3 棵樹中的父節點，依序為 1　0　4　3。

在 *m* 列資料中，第四列的資料：3　1　2　4　5。第一個值 3 節點 *i* 為 3；之後k個值分別代表節點 *i* 在第 0~3 棵樹中的父節點，依序為 1　2　4　5。

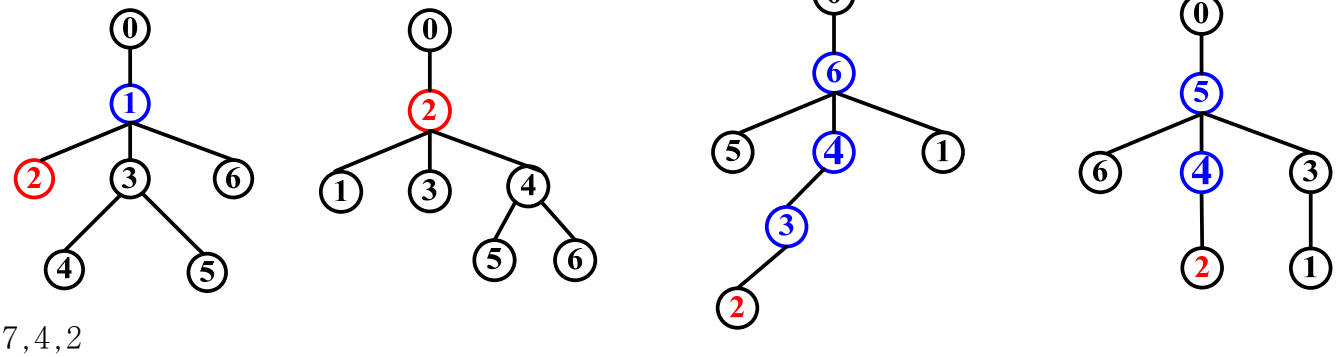


在這 4 棵樹中，節點 6 到根節點 0 路徑，中間所經過的節點集合分為{1}、{4,2}、{}和{5}，路徑長度為中間所經過的節點形成的集合之元素個數加 1；所以這組測試資料輸出為 2,3,1,2。

```
7,4,6
0   999 999 999 999
1   0   2   6   3
2   1   0   4   3
3   1   2   4   5
4   3   2   6   5
5   3   4   6   0
6   1   4   0   5
```

第二組測試資料的第一列值$m,k,v=7,4,2$，代表共有 7 個節點、4 棵樹，在這 4 棵樹中，分別算出節點 2 到根節點 0 的路徑長度。

在這 4 棵樹中，節點 2 到根節點 0 路徑，中間所經過的節點集合分為{1}、{}、{3,4,6}和{4,5}，所以，這組測試資料輸出為 2,1,4,3。

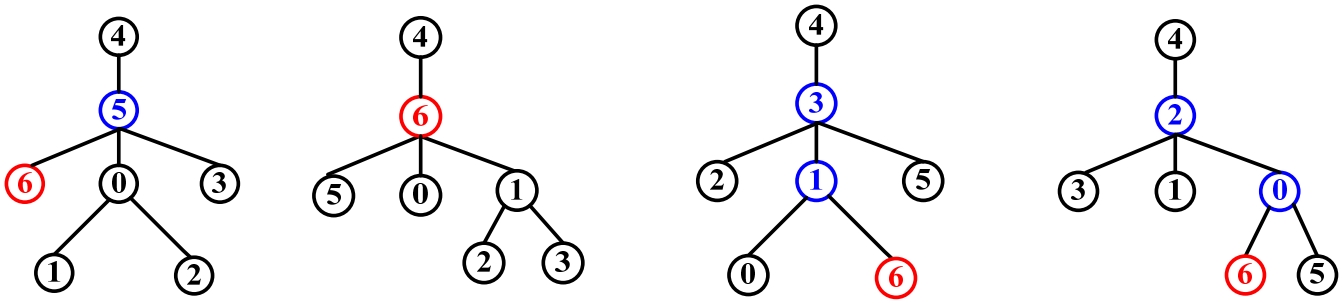


7,4,2

0	999	999	999	999
1	0	2	6	3
2	1	0	3	4
3	1	2	4	5
4	3	2	6	5
5	3	4	6	0
6	1	4	0	5

第三組測試資料的第一列值$m,k,v=7,4,6$，代表共有 7 個節點、4 棵樹，在這 4 棵樹中，分別算出節點 6 到根節點 4 的路徑長度。

在這 4 棵樹中，節點 6 到根節點 4 路徑，中間所經過的節點集合分為{5}、{}、{1,3}和{0,2}，所以，這組測試資料輸出為 2,1,3,3。

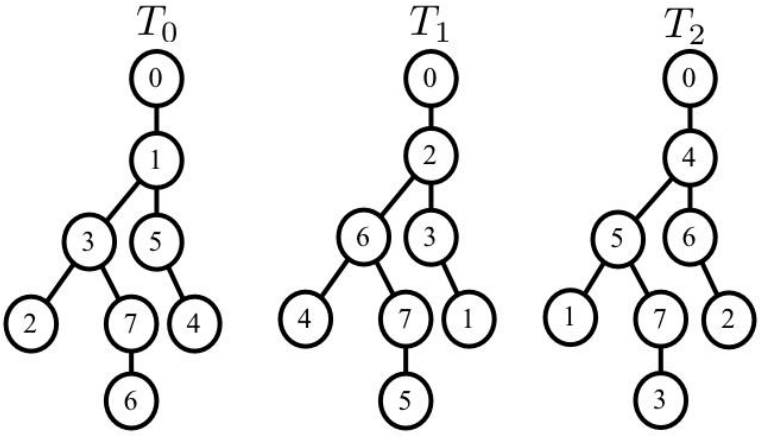


7,4,6

0	5	6	1	2
1	0	6	3	2
2	0	1	3	4
3	5	1	4	2
4	999	999	999	999
5	4	6	3	0
6	5	4	1	0

第四組測試資料的第一列值$m, k, v = 8, 3, 1$，代表共有 8 個節點、3 棵樹，在這 3 棵樹中，分別算出節點 1 到根節點 0 的路徑長度。

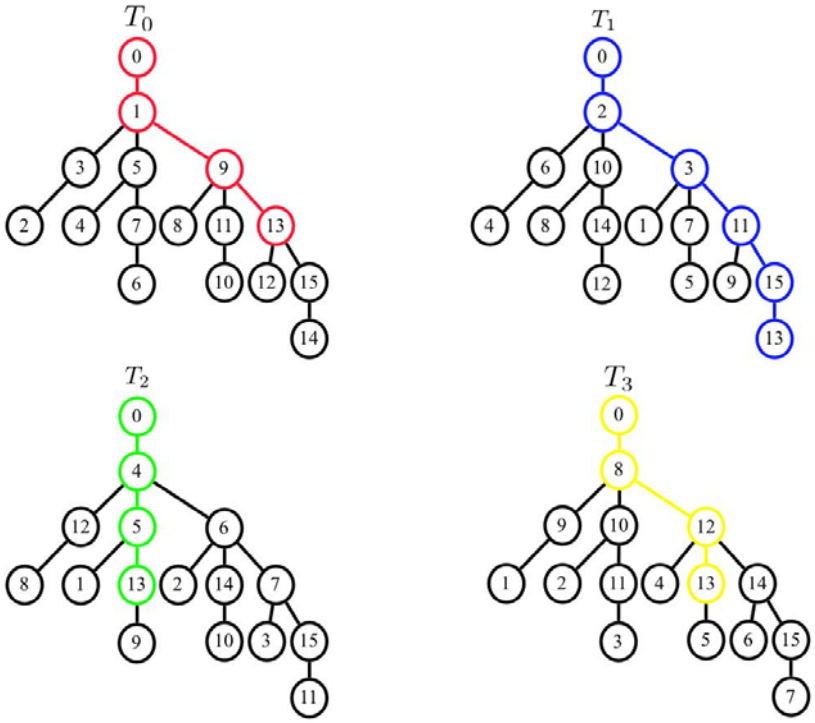
在這 3 棵樹中，節點 1 到根節點 0 路徑，中間所經過的節點集合分為{}、{3,2}和{5,4}，所以，這組測試資料輸出為 1,3,3。




```
8,3,1
0    999  999  999
1    0    3    5
2    3    0    6
3    1    2    7
4    5    6    0
5    1    7    4
6    7    2    4
7    3    6    5
```

在 in2.txt 檔案中，第一組測試資料的第一列值$m, k, v = 16, 4, 13$，代表共有 16 個節點、4 棵樹，在這 4 棵樹中，分別算出節點 13 到根節點 0 的路徑長度。

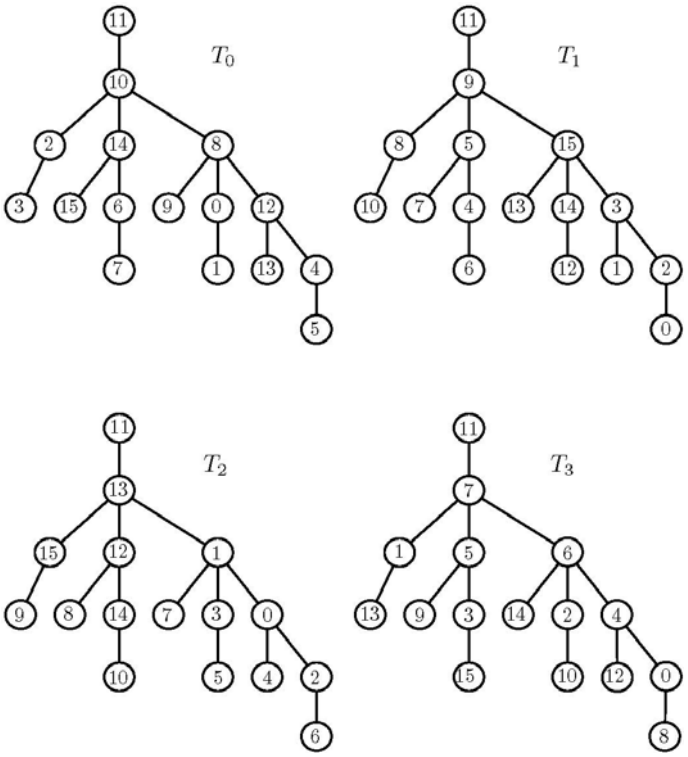
在這 4 棵樹中，節點 13 到根節點 0 路徑，中間所經過的節點集合分為{9,1}、{15,11,3,2}、{5,4}和{12,8}，所以，這組測試資料輸出為 3,5,3,3。




```
16,4,13
0    999  999  999  999
1    0    3    5    9
2    3    0    6    10
3    1    2    7    11
4    5    6    0    12
5    1    7    4    13
6    7    2    4    14
7    5    3    6    15
8    9    10   12   0
9    1    11   13   8
10   11   2    14   8
11   9    3    15   10
12   13   14   4    8
13   9    15   5    12
14   15   10   6    12
15   13   11   7    14
```

第二組測試資料的第一列值$m, k, v = 16, 4, 15$，代表共有 16 個節點、4 棵樹，在這 4 棵樹中，分別算出節點 15 到根節點 11 的路徑長度。

在這 4 棵樹中，節點 15 到根節點 11 路徑，中間所經過的節點集合分為{14,10}、{9}、{13}和{3,5,7}，所以，這組測試資料輸出為 3,2,2,4。



16,4,15
0 8 2 1 4
1 0 3 13 7
2 10 3 0 6
3 2 15 1 5
4 12 5 0 6
5 4 9 3 7
6 14 4 2 7
7 6 5 1 11
8 10 9 12 0
9 8 11 15 5
10 11 8 14 2
11 999 999 999 999
12 8 14 13 4
13 12 15 11 1
14 10 15 12 6
15 14 9 13 3

輸入檔案 1：【檔名：in1.txt】

```
4
7,4,6
0    999 999 999 999
1    0    2    6    3
2    1    0    4    3
3    1    2    4    5
4    3    2    6    5
5    3    4    6    0
6    1    4    0    5
7,4,2
0    999 999 999 999
1    0    2    6    3
2    1    0    3    4
3    1    2    4    5
4    3    2    6    5
5    3    4    6    0
6    1    4    0    5
7,4,6
0    5    6    1    2
1    0    6    3    2
2    0    1    3    4
3    5    1    4    2
4    999 999 999 999
5    4    6    3    0
6    5    4    1    0
8,3,1
0    999 999 999
1    0    3    5
2    3    0    6
3    1    2    7
4    5    6    0
5    1    7    4
6    7    2    4
7    3    6    5
```

輸入檔案 2：【檔名：in2.txt】

```
2
16,4,13
0    999 999 999 999
1    0   3   5   9
2    3   0   6   10
3    1   2   7   11
4    5   6   0   12
5    1   7   4   13
6    7   2   4   14
7    5   3   6   15
8    9   10  12  0
9    1   11  13  8
10   11  2   14  8
11   9   3   15  10
12   13  14  4   8
13   9   15  5   12
14   15  10  6   12
15   13  11  7   14
16,4,15
0    8   2   1   4
1    0   3   13  7
2    10  3   0   6
3    2   15  1   5
4    12  5   0   6
5    4   9   3   7
6    14  4   2   7
7    6   5   1   11
8    10  9   12  0
9    8   11  15  5
10   11  8   14  2
11   999 999 999 999
12   8   14  13  4
13   12  15  11  1
14   10  15  12  6
15   14  9   13  3
```

輸出範例：【檔名：**out.txt**】

2,3,1,2
2,1,4,3
2,1,3,3
1,3,3

3,5,3,3
3,2,2,4

子題 2：二元搜尋樹的路徑長度最長的值。 **(程式執行限制時間: 2 秒)**

在資料結構中，每棵樹存在一個特殊的節點，稱為根節點(root)，可連結若干子樹，也可以沒有子樹；從任一節點到根節點，都只有唯一一條的節點不重複路徑。例如 F 到 A 的路徑為 F→B→A，其路徑長度為 2。

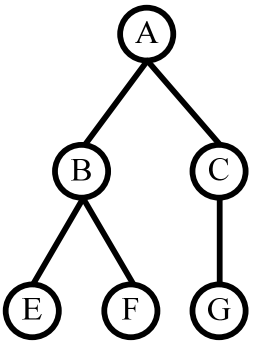


圖 4.2.1

由使用者輸入 *n* 筆資料，建立一個二元搜尋樹(Binary Search Tree)，在這二元搜尋樹算出每個節點到根節點路徑長度，**輸出這棵二元搜尋樹路徑長度最長的值**。

二元搜尋樹(Binary Search Tree)定義:
二元搜尋樹是一種二元樹，它可以為空，若不為空，則必須要滿足以下條件：
1.若左子樹不為空，則左子樹的鍵值均須要小於樹根的鍵值。
2.若右子樹不為空，則右子樹的鍵值均須要大於樹根的鍵值。
3.左子樹與右子樹必須也要保持二元搜尋樹。

輸入說明：

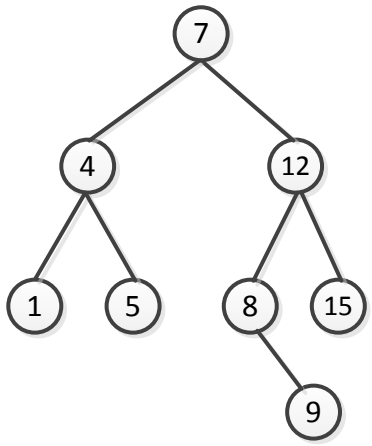
第一列的數字 *n* 代表這組測試資料有幾個節點，$2 \le n \le 30$。第二列起則為這組測試各節點編號，節點編號為一整數$0 \le N \le 1000$。用測試資料以二元搜尋樹方式建樹。

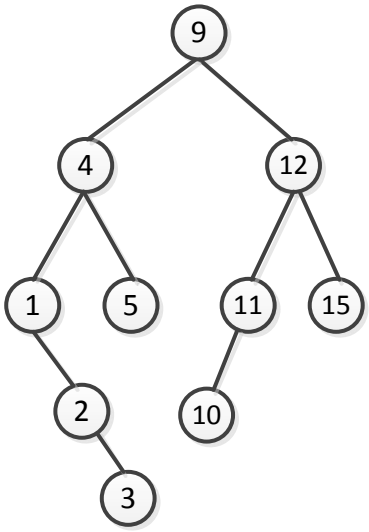
輸出說明：

在測試資料中所建二元搜尋樹，在二元搜尋樹中算出每個節點到根節點路徑長度，輸出這棵二元搜尋樹路徑長度最長的值，這組測試資料輸出一列。

輸入檔案 1：【檔名：in1.txt】

8
7
4
1
5
12
8
9
15





in1.txt **in2.txt**

輸入檔案 2：【檔名：in2.txt】

9
4
1
5
12
11
10
15
2
3

輸出範例：【檔名：out.txt】

3

4

子題 2：中序表示法。 **(程式執行限制時間: 2 秒)**

由使用者輸入 *n* 筆資料，建立一個二元搜尋樹(Binary Search Tree)，在這二元搜尋樹，輸出這棵二元搜尋樹之中序表示法。

二元搜尋樹(Binary Search Tree)定義:

二元搜尋樹是一種二元樹，它可以為空，若不為空，則必須要滿足以下條件：

1.若左子樹不為空，則左子樹的鍵值均須要小於樹根的鍵值。

2.若右子樹不為空，則右子樹的鍵值均須要大於樹根的鍵值。

3.左子樹與右子樹必須也要保持二元搜尋樹。

二元樹的走訪（Traversal of Tree）

對於一個二元樹，我們有三種最常用的方法可以走過這棵樹所有的節點。

1. 前序表示法（pre-order）：根節點 -> 左子樹 -> 右子樹
2. 中序表示法（in-order）：左子樹 -> 根節點 -> 右子樹
3. 後序表示法（post-order）：左子樹 -> 右子樹 -> 根節點

輸入說明：

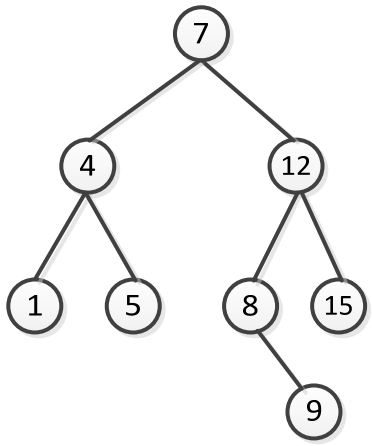
第一列的數字 *n* 代表這組測試資料有幾個節點，$2 \leq n \leq 30$。第二列起則為這組測試各節點編號，節點編號為一整數$0 \leq N \leq 100$。用測試資料以二元搜尋樹方式建樹。

輸出說明：

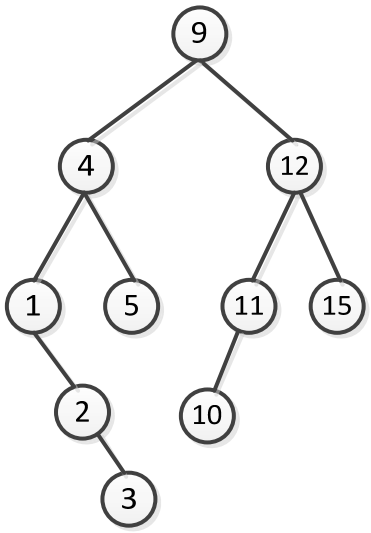
在測試資料中所建二元搜尋樹，輸出這棵二元搜尋樹之中序表示法，這組測試資料輸出一列。

輸入檔案 1：【檔名：in1.txt】

8
7
4
1
5
12
8
9
15



in1.txt



in2.txt

輸入檔案 2：【檔名：in2.txt】

10
9
4
1
5
12
11
10
15
2
3

輸出範例：【檔名：out.txt】

1,4,5,7,8,9,12,15

1,2,3,4,5,9,10,11,12,15

Problem 3： 資料結構—樹

子題 1：是否為堆積樹(Heap tree)或二元搜尋樹(Binary search tree)**。(程式執行限制時間: 2 秒) 14 分**

在資料結構中，樹狀結構是可以用來描述有分支的結構，包含 1 個或多個節點。其存在一個特殊的節點，稱為根節點(root)，可連結若干子樹，也可以沒有子樹；從任一節點到根節點，都只有唯一的節點不重複路徑。

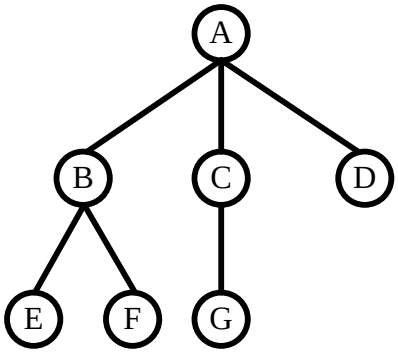


圖 3.1.1

在圖 3.1.1 中，有編號的圓形代表節點，A 為根節點，B、C 及 D 均為 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總節點數減 1，在樹上任意添加一條邊，就會產生迴圈。

專有名詞介紹：

(1) 無父節點的節點為根節點(Root)，如 A。
(2) 父節點 (Parent)：一節點的上層節點為父節點，如 B 的父節點為 A，如 G 的父節點為 C。
(3) 子節點 (Children)：一節點的下層節點為子節點，如 B 的子節點有 E 及 F；C 的子節點有 G。
(4) 兄弟節點 (Siblings)：有共同父節點的節點稱為兄弟節點，如 B、C、D 互為兄弟節點。
(5) 分支度 (Degree)：一個節點的子樹個數稱為其分支度，如 A 的分支度為 3；B 的分支度為 2；C 的分支度為 1；E 的分支度為 0。
(6) 樹葉節點(Terminal node)：無子節點的節點，如 D、E、F、G。
(7) 內部節點 (Non-terminal node)：樹葉以外的節點均為內部節點，如 A、B、C。
(8) 階層或階度 (Level)：A 為階層 1；B、C、D 為階層 2；E、F、G 為階層 3。
(9) 高度 (Height)：樹的最大階度，例如圖 3.1.1，因最大階度階度為 3，則其樹的高度為 3。

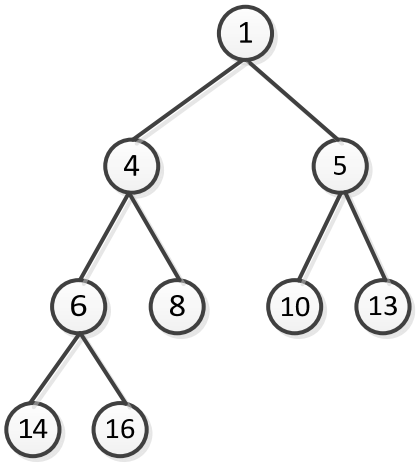
堆積樹(Heap tree)是一個二元樹，每個父節點最多只有兩個子節點，堆積樹的父節點若小於子節點，則稱之為最小堆積（Min heap tree），父節點若大於子節點，則稱之為最大堆積（Max heap tree），而同一層的子節點則無需理會其大小關係。

最小堆積樹(Min heap tree)

指每一個節點的鍵值必須小於它的子節點的鍵值。其特性如下：

1. 每一棵 Min heap tree 是一棵「完整二元樹」(Complete Binary Tree)。
2. 樹根的鍵值小於左子樹與右子樹的鍵值。
3. 其左子樹與右子樹亦是 Min heap tree。如下圖所示：

將下圖的堆積樹轉換為一維陣列之後如下所示：{1,4,5,6,8,10,13,14,16}

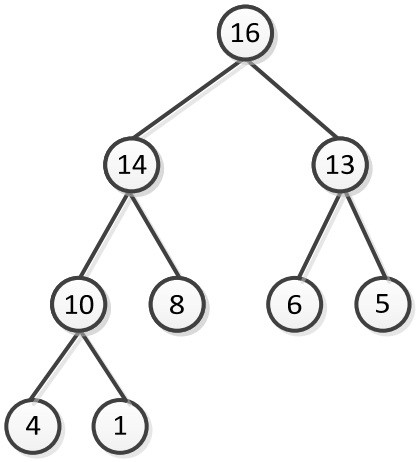


最大堆積樹(Max heap tree)

指每一個節點的鍵值必須大於它的子節點的鍵值。其特性如下：

1. 每一棵 Max heap tree 是一棵「完整二元樹」(Complete Binary Tree)。
2. 樹根的鍵值大於左子樹與右子樹的鍵值。
3. 其左子樹與右子樹亦是 Max heap tree。如下圖所示：

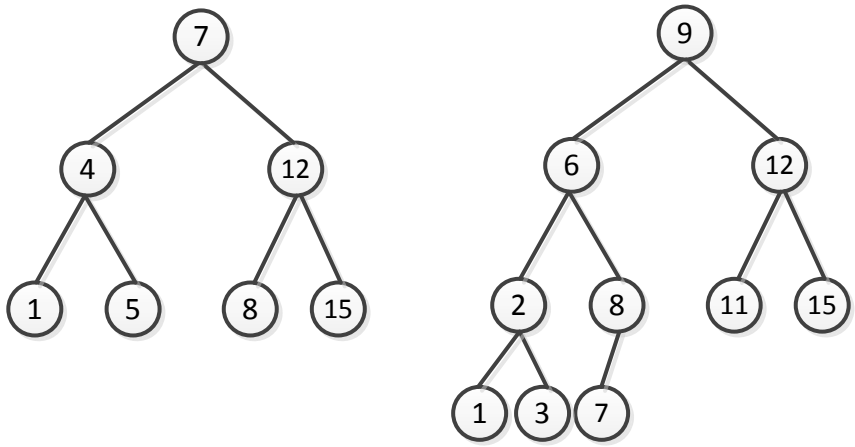
將下圖的堆積樹轉換為一維陣列之後如下所示：{16,14,13,10,8,6,5,4,1}



而在一棵二元樹中，除最後一層外，若其餘層都是滿的，並且最後一層或者是滿的，或者是在右邊缺少連續若干節點，則此二元樹為「完整二元樹」(Complete Binary Tree)。

二元搜尋樹(Binary search tree)定義:
二元搜尋樹是一種二元樹，它可以為空，若不為空，則必須要滿足以下條件：
1.若左子樹不為空，則左子樹的鍵值均須要小於樹根的鍵值。
2.若右子樹不為空，則右子樹的鍵值均須要大於樹根的鍵值。
3.左子樹與右子樹必須也要保持二元搜尋樹。

將下圖的二元搜尋樹(「完整二元樹」(Complete Binary Tree))轉換為一維陣列之後如下所示：{7,4,12,1,5,8,15}和{9,6,12,2,8,11,15,1,3,7}。



寫一個程式，讀入一資料，資料內容為「完整二元樹」(Complete Binary Tree)，然後回答該資料是否為堆積樹(Heap tree)或二元搜尋樹(Binary search tree)。

輸入說明：
第一列的數字 *n* 代表共有幾組資料要測試，$2 \le n \le 8$。
第二列起每一列代表一組測試資料。每組測試資料代表一「完整二元樹」(Complete Binary Tree)。測試資料為多個不同的數字x_i，$0 \le x_i \le 65535$，$4 \le |x_i| \le 64$，中間用逗號隔開。

輸出說明：
每組測試資料輸出一列。輸出每組測試資料是否為堆積樹(Heap tree)或二元搜尋樹(Binary search tree)。若該資料是堆積樹(Heap tree)，則輸出 H；若該資料是二元搜尋樹(Binary search tree)，則輸出 B；若不是堆積樹(Heap tree)同時也不是二元搜尋樹(Binary search tree)，則輸出 F。不會有同時為堆積樹(Heap tree)和二元搜尋樹(Binary search tree)的測試資料。

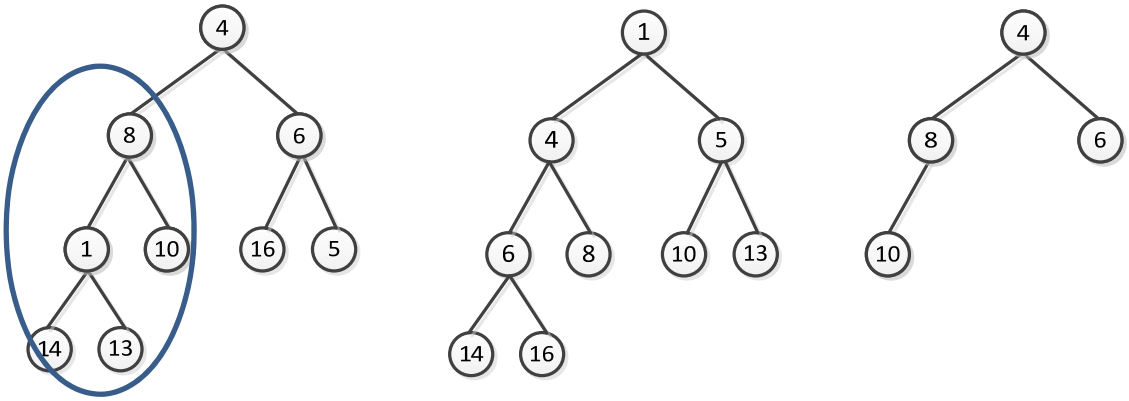
輸入檔案 1：【檔名：in1.txt】

3

4,8,6,1,10,16,5,14,13

1,4,5,6,8,10,13,14,16

4,8,6,10



輸入檔案 2：【檔名：in2.txt】

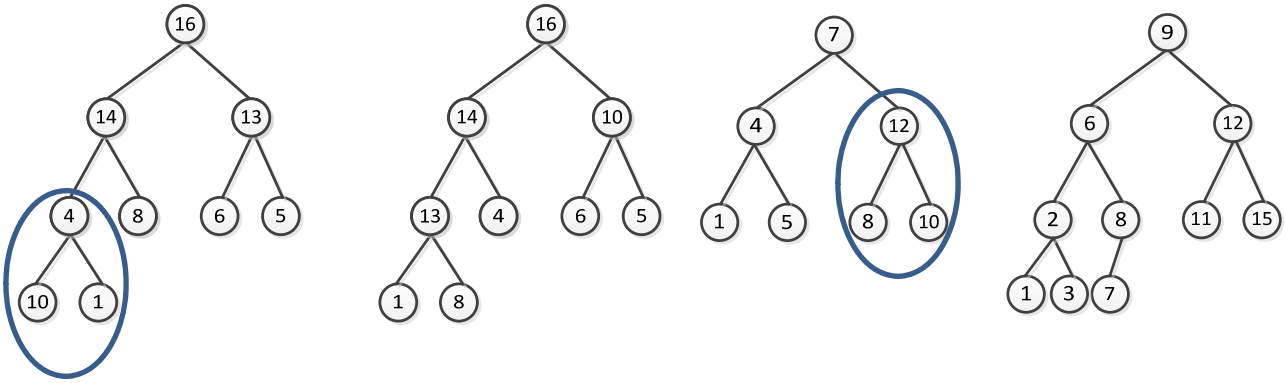
4

16,14,13,4,8,6,5,10,1

16,14,10,13,4,6,5,1,8

7,4,12,1,5,8,10

9,6,12,2,8,11,15,1,3,7



輸出範例：【檔名：out.txt】

F

H

H

F

H

F

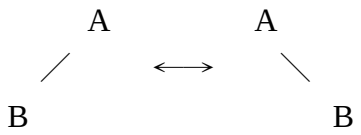
B

子題 2：後序表示法（post-order）。 **(程式執行限制時間: 2 秒) 15 分**

二元樹的定義：

1.樹不可以為空集合，亦即至少必須有一個根節點，但二元樹卻可以是空集合。

2.樹的兄弟節點位置次序並非固定，但二元樹是固定的。也就是下面是相同的樹，但卻不是相同的二元樹。



在二元樹的運用上，常常需要找出所有的節點資料，這個過程稱為樹的拜訪或追蹤。依拜訪追蹤的次序可分成下列三種：前序表示法（pre-order）、中序表示法（in-order）及後序表示法（post-order）。

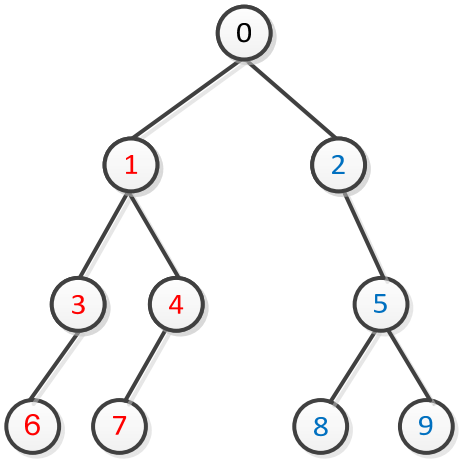
二元樹的走訪（Traversal of Tree）

對於一個二元樹，我們有三種最常用的方法可以走過這棵樹所有的節點。

1. 前序表示法（pre-order）：根節點 -> 左子樹 -> 右子樹
2. 中序表示法（in-order）：左子樹 -> 根節點 -> 右子樹
3. 後序表示法（post-order）：左子樹 -> 右子樹 -> 根節點

有兩種序，就有機會還原出唯一的一棵二元樹。比方說，知道中序表示法（in-order）和前序表示法（pre-order），可以求出原本的二元樹。

在前序表示法（pre-order）之中，最左邊的元素就是 root ；在中序表示法（in-order） 之中，root 的兩邊分別為左子樹和右子樹 —— 利用 root 便可區分左子樹和右子樹。子樹也是樹，可以用相同手法繼續分割，最後便可求出整棵二元樹的架構。



中序表示法（in-order） : 6,3,1,7,4,0,2,8,5,9

前序表示法（pre-order）: 0,1,3,6,4,7,2,5,8,9

輸入說明：

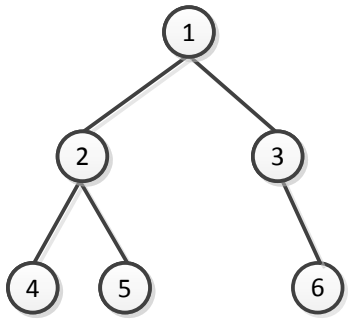
第一列的數字 *n* 代表有幾組資料要測試，$2 \le n \le 8$，第二列起為每組的測試資料，之後每二列為每組的測試資料。每組測試資料的第一列為中序表示法（in-order）；每組測試資料的第二列為前序表示法(pre-order)，各節點編號不會相同。測試資料為多個數字x_i，$0 \le x_i \le 65535$，$4 \le |x_i| \le 64$，中間用逗號隔開。用測試資料找出整棵二元樹的架構。

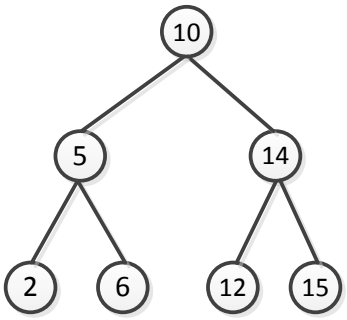
輸出說明：

在測試資料中所建二元樹，輸出這棵二元樹之後序表示法（post-order），每組測試資料輸出一列。

輸入檔案 1：【檔名：in1.txt】

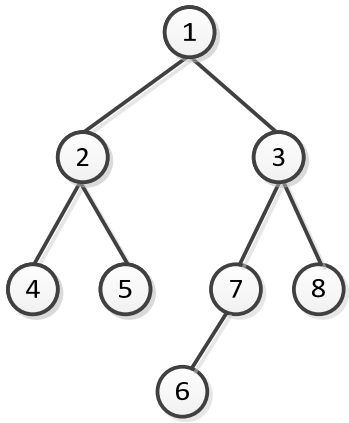
2
4,2,5,1,3,6
1,2,4,5,3,6
2,5,6,10,12,14,15
10,5,2,6,14,12,15

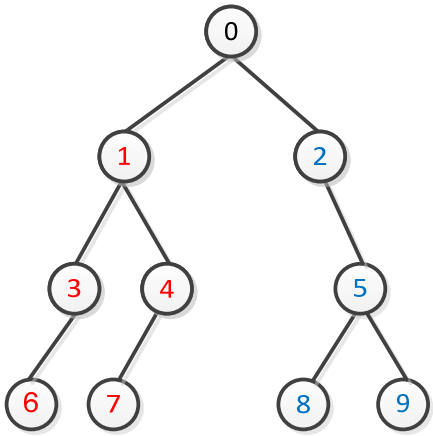




輸入檔案 2：【檔名：in2.txt】

2
4,2,5,1,6,7,3,8
1,2,4,5,3,7,6,8
6,3,1,7,4,0,2,8,5,9
0,1,3,6,4,7,2,5,8,9





輸出範例：【檔名：out.txt】

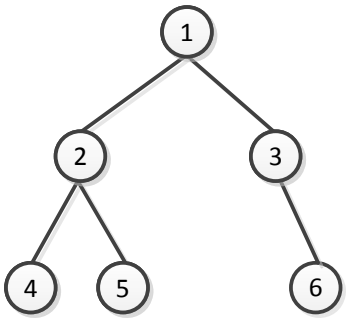
4,5,2,6,3,1

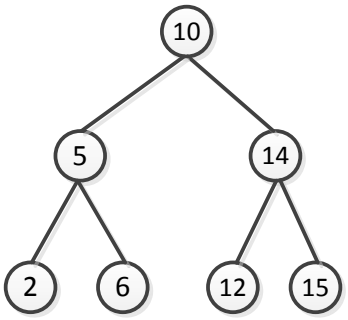
2,6,5,12,15,14,10

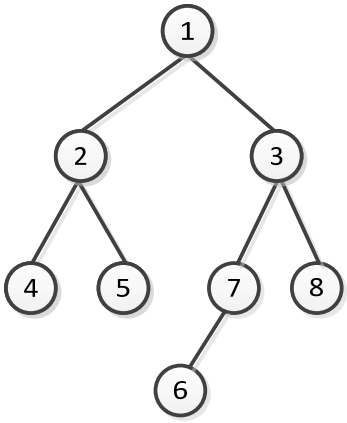
4,5,2,6,7,8,3,1

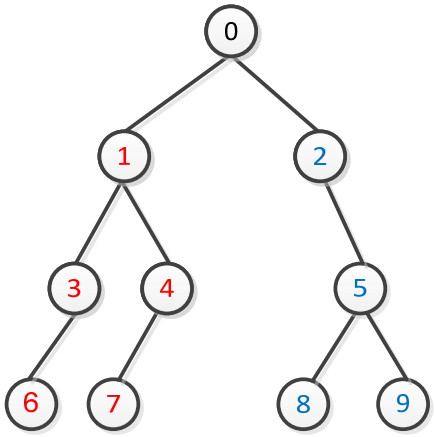
6,3,7,4,1,8,9,5,2,0

方便選手比對結果，把上一頁的圖，這也放相同的一份。









Problem 4：資料結構—樹

子題 1：樹。(程式執行限制時間: 2 秒)

在資料結構中，樹狀結構是可以用來描述有分支的結構，包含 1 個或多個節點。其存在一個特殊的節點，稱為根節點(root)，可連結若干子樹，也可以沒有子樹；從任一節點到根節點，都只有唯一的節點不重複路徑。

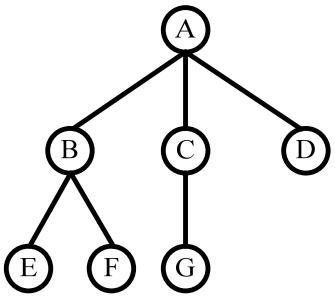


圖 4.1.1

在圖 4.1.1 中，有編號的圓形代表節點，A 為根節點，B、C 及 D 均為 A 的子節點，各節點之間不會有迴圈，且所有節點之間都有一個或多個邊相連通。任一樹狀結構的總邊數等於其總節點數減 1，在樹上任意添加一條邊，就會產生迴圈。

專有名詞介紹：

(1) 無父節點的節點為根節點(Root)，如 A。
(2) 父節點 (Parent)：一節點的上層節點為父節點，如 B 的父節點為 A，如 G 的父節點為 C。
(3) 子節點 (Children)：一節點的下層節點為子節點，如 B 的子節點有 E 及 F；C 的子節點有 G。
(4) 兄弟節點 (Siblings)：有共同父節點的節點稱為兄弟節點，如 B、C、D 互為兄弟節點。
(5) 分支度 (Degree)：一個節點的子樹個數稱為其分支度，如 A 的分支度為 3；B 的分支度為 2；C 的分支度為 1；E 的分支度為 0。
(6) 樹葉節點(Terminal node)：無子節點的節點，如 D、E、F、G。
(7) 內部節點 (Non-terminal node)：樹葉以外的節點均為內部節點，如 A、B、C。
(8) 階層或階度 (Level)：A 為階層 1；B、C、D 為階層 2；E、F、G 為階層 3。
(9) 高度 (Height)：樹的最大階度，例如圖 4.1.1，因最大階度階度為 3，則其樹的高度為 3。

寫一個程式，讀入一無向圖的資料，一棵樹是一個無向圖且沒有迴圈。在測試檔中，節點的編號不一定是連續的號碼。如果檢測的圖形有迴圈，則輸出造成迴圈的邊；若該圖是樹，則輸出 T；沒有迴圈又不是樹，則輸出 F。

輸入說明：

第一列的數字 ***n*** 代表共有幾組資料要測試，$2 \le n \le 5$。

第二列起每一行代表一組測試資料。每組測試資料代表一圖形，內容為邊的資料。每個邊以 2 個整數i, j表示，$0 \le i, j \le 20$ and $i \ne j$，其中 ***i*** 和 ***j*** 為節點的編號，代表從 ***i*** 節點和 ***j*** 節點有邊相連，每組測試資料，同一列中，每個邊的資料以空白(　)隔開，而空白不限定一個，$|i, j|$為邊的個數，$2 \le |i, j| \le 20$。

輸出說明：

每組測試資料輸出一列。測試資料的每個邊依序加入圖形，輸出造成迴圈的邊，測試資料拿掉這些造成迴圈的邊之後為一棵樹。如果檢測的圖形有迴圈，則輸出造成迴圈的邊；若該圖是樹，則輸出 T；沒有迴圈又不是樹，則輸出 F。

輸入檔案 1：【檔名：in1.txt】

4

5,8 5,3 5,2 5,4 5,6 1,2 2,0

8,1 1,3 6,2 8,10 7,5 1,4 7,8 8,6 8,0

3,8 6,8 6,4 0,6 8,2 2,0 5,3

1,0 4,3 1,2

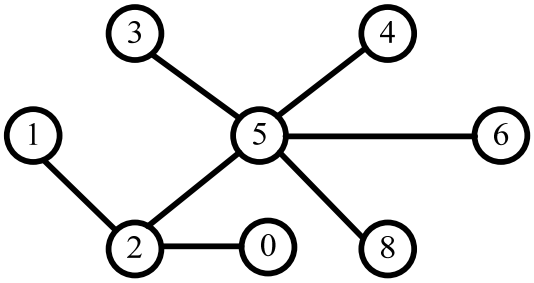


圖 4.1.1.1(**in1.txt)**

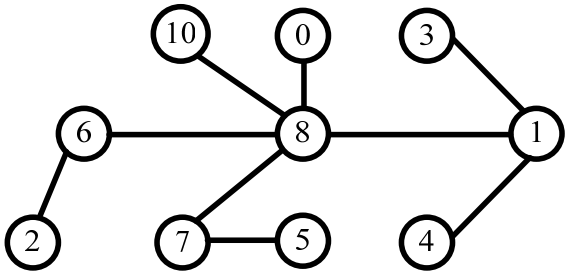


圖 4.1.1.2 (**in1.txt)**

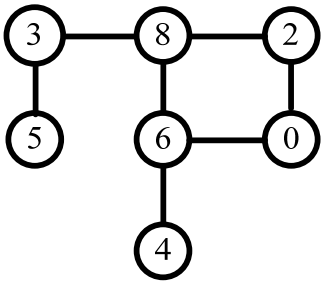


圖 4.1.1.3 (**in1.txt)**

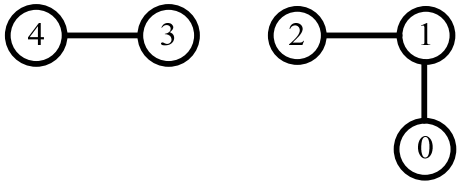


圖 4.1.1.4 (**in1.txt)**

輸入檔案 2：【檔名：in2.txt】

5

1,2 2,3 4,0

4,3 2,3 2,1 1,0

1,2 2,3 3,4 1,4 1,5

1,2 1,3 2,3

1,2 2,3 3,4 1,4 1,5 5,4

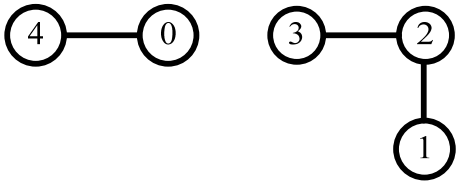


圖 4.1.2.1(**in2.txt)**

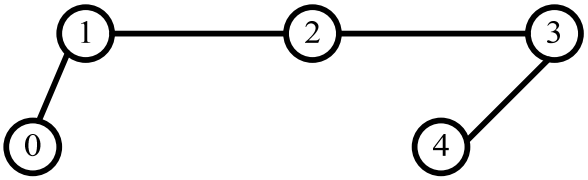


圖 4.1.2.2 (**in2.txt)**

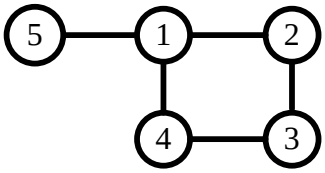


圖 4.1.2.3 (**in2.txt)**

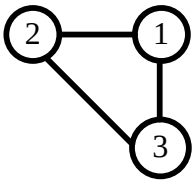


圖 4.1.2.4 **(in2.txt)**

輸出範例：【檔名：out.txt】

T

T

2,0

F

F

T

1,4

2,3

1,4 5,4

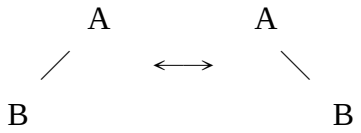
Problem 4：資料結構—樹

子題 1：輸出二元樹的後序拜訪的結果。(程式執行限制時間: 2 秒) **14 分**

二元樹的定義：

1.樹不可以為空集合，亦即至少必須有一個根節點，但二元樹卻可以是空集合。

2.樹的兄弟節點位置次序並非固定，但二元樹是固定的。也就是下面是相同的樹，但卻不是相同的二元樹。



在二元樹的運用上，常常需要找出所有的節點資料，這個過程稱為樹的拜訪或追蹤。依拜訪追蹤的次序可分成下列三種：前序 preorder、中序 inorder 及後序 postorder。

後序 postorder 定義：

拜訪根節點前，若有左子樹，先拜訪其左子樹的所有節點；若有右子樹，再拜訪其右子樹的所有節點，最後再拜訪根節點。

二元搜尋樹(Binary Search Tree)定義：

二元搜尋樹是一種二元樹，它可以為空集合，若不為空集合，則必須要滿足以下條件：

1.若左子樹不為空集合，則左子樹的鍵值均須要小於樹根的鍵值。

2.若右子樹不為空集合，則右子樹的鍵值均須要大於樹根的鍵值。

3.左子樹與右子樹必須也要保持二元搜尋樹。

由使用者輸入 *x* 筆資料，建立一個二元搜尋樹(Binary Search Tree)，輸出二元搜尋樹的後序拜訪的結果。

輸入說明：

第一列的數字 *n* 代表有幾組資料要測試，$1 \leq n \leq 5$，第二列起為每組的測試資料，之後每二列為每組的測試資料。每組測試資料的第一列是一個整數$3 \leq x \leq 20$，用來表示這組測試資料有幾個節點；每組測試資料的第二列為這組測試資料各節點編號，以“,”分隔各節點編號，編號為一整數$0 \leq N \leq 99$，各節點編號不會相同。用測試資料以二元搜尋樹方式建樹。

輸出說明：

在測試資料中所建二元搜尋樹，輸出二元搜尋樹的後序拜訪的結果，以“,”分隔各節點編號。

輸入檔案 1：【檔名：in1.txt】

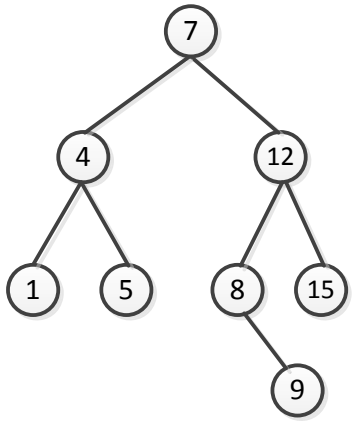
2

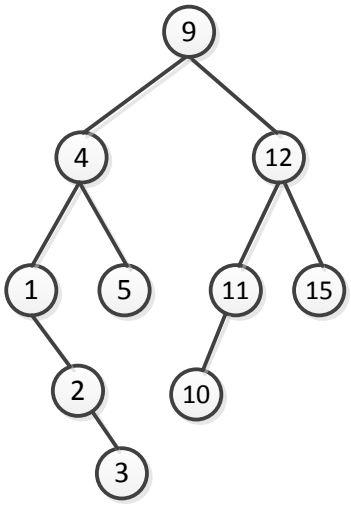
8

7,4,1,5,12,8,9,15

10

9,4,1,5,12,11,10,15,2,3





輸入檔案 2：【檔名：in2.txt】

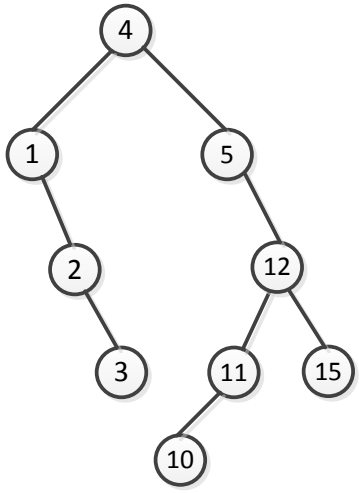
2

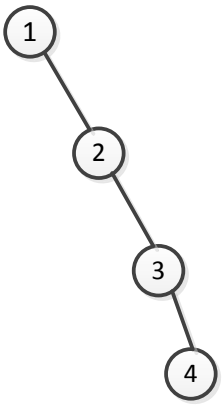
9

4,1,5,12,11,10,15,2,3

4

1,2,3,4





輸出範例：【檔名：out.txt】

1,5,4,9,8,15,12,7

3,2,1,5,4,10,11,15,12,9

3,2,1,10,11,15,12,5,4

4,3,2,1

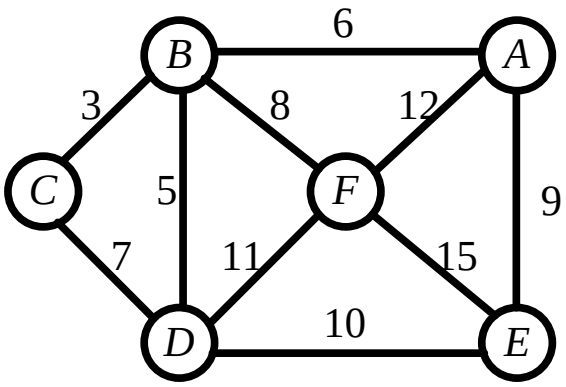
子題 2：最小成本生成樹。(程式執行限制時間: 2 秒) **16 分**

以有線電視電纜的架設為例，若只能沿著街道佈線，則以街道為邊，而路口為節點，其中必然有一最小成本生成樹能使佈線成本最低。

給定一個圖形中，有許多條邊(線)連結了所有的節點，這些邊都有一個數值，代表此邊的成本。我們可以去除圖形中的某些邊，使得剩下的邊能連結所有的節點，且邊的數量比節點的數量少 1，這些節點和留下的邊為一生成樹。一個圖形的生成樹有許多個，其中邊的總成本最低者為最小成本生成樹。最小成本生成樹不可以有循環(迴路)；最小成本生成樹不必是唯一的。

Kruskal 演算法：

假設節點數為n，Kruskal 演算法是將各邊先依成本(權重值)的大小由小到大排列，接著從成本(權重值)最低的邊開始加入最小成本生成樹, 如果加入的邊會造成循環(迴路)則捨棄不用，直到加了 $n-1$ 個邊為止。舉例說明如何以 Kruskal 演算法得到下圖中的最小成本生成樹 ：

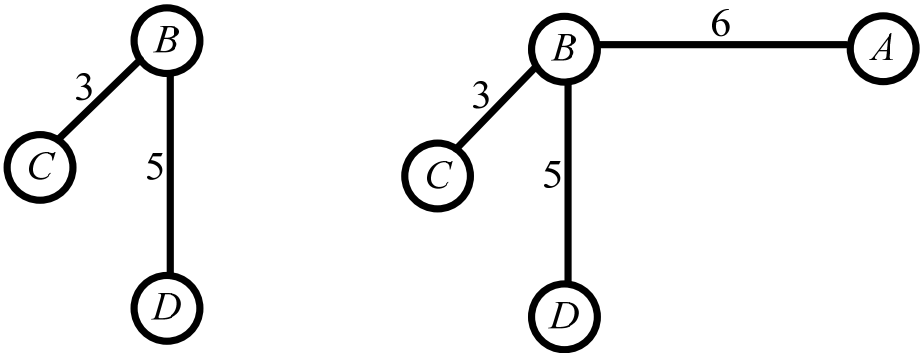


步驟一：將所有邊線的成本(權重值)列出並由小到大排序 ：

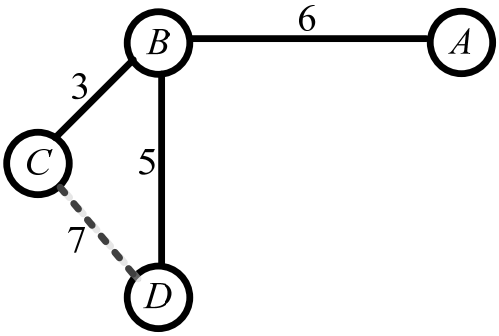
節點	節點	成本(權重值)
B	C	3
B	D	5
A	B	6
C	D	7
B	F	8
A	E	9
D	E	10
D	F	11
A	F	12
E	F	15

步驟二：選擇成本(權重值)最低的一條邊做為加入最小成本生成樹的起點，邊(B,C)->3。

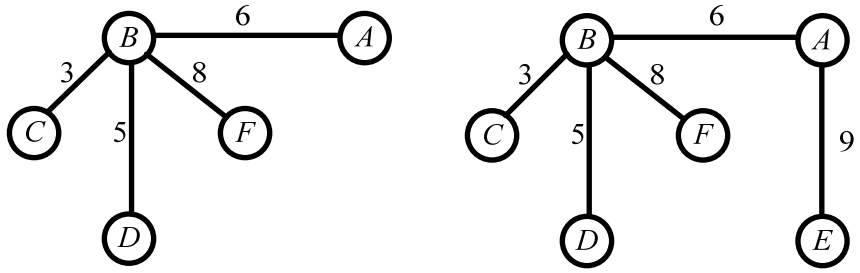
步驟三：依步驟一所建立表格，依序加入邊至最小成本生成樹。



步驟四：邊(C-D) 加入會形成迴路，所以捨棄不用。



重複步驟三和步驟四，直到加了 $n-1$ 個邊為止，完成圖 ：



最小成本生成樹的值為邊的成本之總合 3+5+6+8+9=31

輸入說明：

第一列的數字 *x* 代表共有幾組資料要測試，$2 \leq x \leq 5$。

第二列起每一列代表一組測試資料。每組測試資料代表一圖形，內容為邊的資料。每個邊以 i, j, k表示，其中 *i* 和 *j* 為節點的編號，為大寫英文字母(沒有順序)，代表從 *i* 節點和 *j* 節點有邊相連，k為邊的成本(正整數) $1 \leq N \leq 65535$，每個邊的資料以空白(　)隔開，而空白不限定一個，$|i, j|$為邊的個數，$3 \leq |i, j| \leq 20$。

輸出說明：

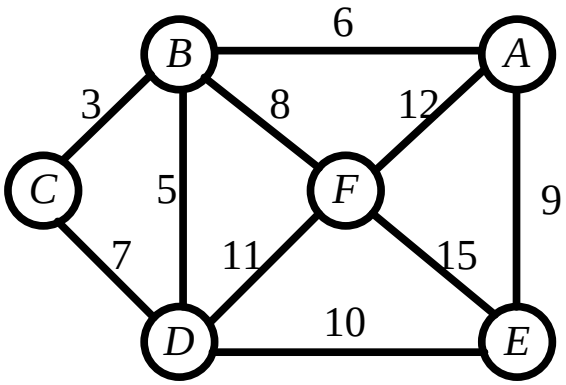
每組測試資料輸出一列。輸出每組測試資料最小成本生成樹的值。

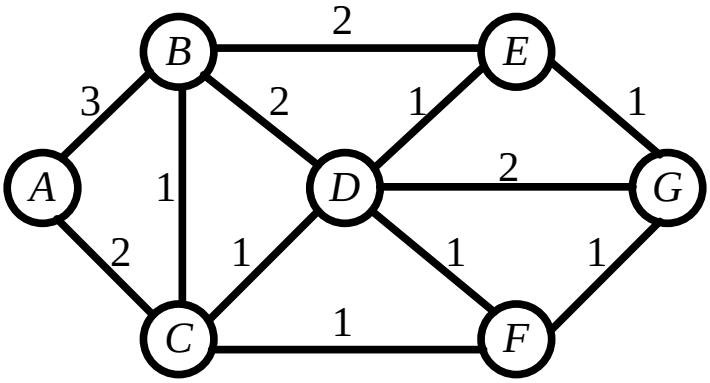
輸入檔案 1：【檔名：in1.txt】

2

A,B,6 A,E,9 B,C,3 B,D,5 C,D,7 B,F,8 D,E,10 D,F,11 A,F,12 E,F,15

A,B,3 A,C,2 B,C,1 B,D,2 C,D,1 B,E,2 C,F,1 D,E,1 D,F,1 D,G,2 E,G,1 F,G,1



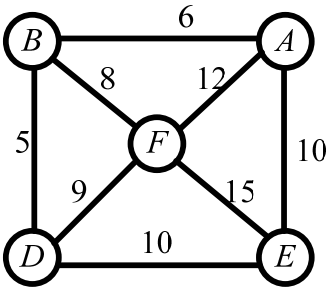


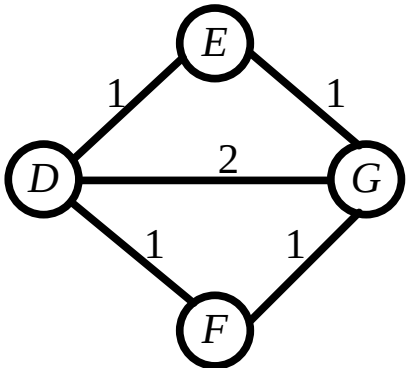
輸入檔案 2：【檔名：in2.txt】

2

B,A,6 B,F,8 B,D,5 D,E,10 D,F,9 A,F,12 A,E,10 E,F,15

D,E,1 D,G,2 D,F,1 E,G,1 F,G,1





輸出範例：【檔名：out.txt】

31

7

29

3