資料結構與進階程式設計（107-2）
作業七

作業設計：楊其恆
國立臺灣大學資訊管理學系

繳交作業時，請將第一題的答案以中文或英文作答後，以 PDF 檔上傳到 NTU COOL；***不接受紙本繳交***；第三至四題請至 PDOGS（`http://pdogs.ntu.im/judge/`）上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳）。這份作業的截止時間是 ***2019 年 6 月 3 日星期一凌晨 1:00***。在你開始前，請閱讀課本的第 15、16 章[1] 不接受遲交。

第一題

(15 分，每題 3 分) 是非題，請回答`True` 或`False`

(a) 一個高度為 h 的 n-ary tree 之 node 數量必定小於 n^h

(b) 將 n 個整數使用 tree sort algorithm 進行排序，可以將 sorting 的複雜度降到 $O(n)$

(c) 一個高度為 h 的 binary search tree，若總共有 h 個 node，則 inorder traversal 的結果與 postorder traversal 的結果必定相同。

(d) 在高度為 h 的 binary search tree 之中，若刪除位於第 i 層的節點，可能會影響第 $i+1$ 至第 h 層的結構。

(e) Binary tree 的左子樹必定也是一個 Binary tree。

第二題

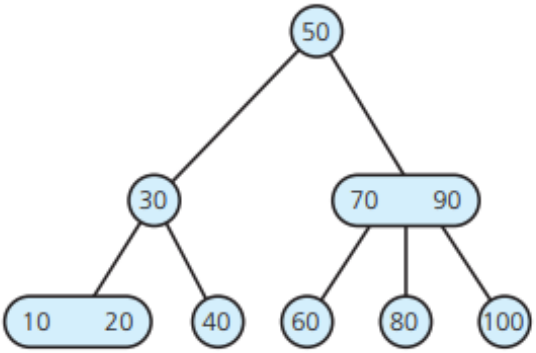
（25 分）Biary search tree 是在搜尋時很常使用的資料結構之一，可以大幅減少搜尋的時間。請自行至網路上搜尋 “2-3 tree”，並回答以下數題。

(a) 請說明一般的 binary search tree 在移除 root node 時應如何操作。（5 分）

(b) 請說明 2-3 tree 在 tree node 的部分與一般的 binary search tree 有何不同。（5 分）

(d) 在以下的 2-3 tree 之中，尋找 80 這個整數需要經過哪些 node，以及會需要與哪些整數進行比較？（5 分）

[1]課本是 Carrano and Henry 著的 *Data Abstraction and Problem Solving with C++: Walls and Mirrors* 第六版。



(c) 請說明為什麼我們希望一個 binary search tree 可以達到平衡，以及 2-3 tree 在插入新的元素時，有什麼機制可以在這方面相較於 binary search tree 有較好的表現，。（10 分）

第三題

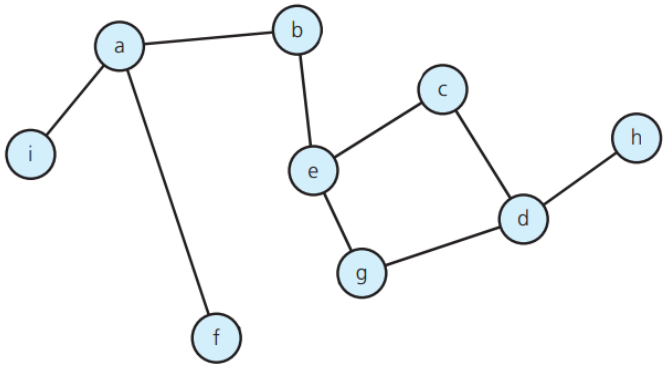
（20 分）給定一個以 adjacency matrix 表示的無向圖，這個圖如果任二個 node 之間只有一條路徑，則這個圖是一個 tree。在本題中請根據輸入的無向圖，判斷其是否為一個 tree，如果是的話，請印出這個 tree 可能的最小高度。

輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。每個檔案會有 $n+1$ 行，第一行會有一個整數 n，第二行到第 n+1 行之中，每行會有 n 個整數 0 或 1，分別以空白字元分隔，代表這個圖的 adjacency matrix。亦即第 $i+1$ 行的第 j 個整數代表這個圖的第 i 與第 j 個 node 之間是否有邊連接。其中，$2 \leq n \leq 1000$

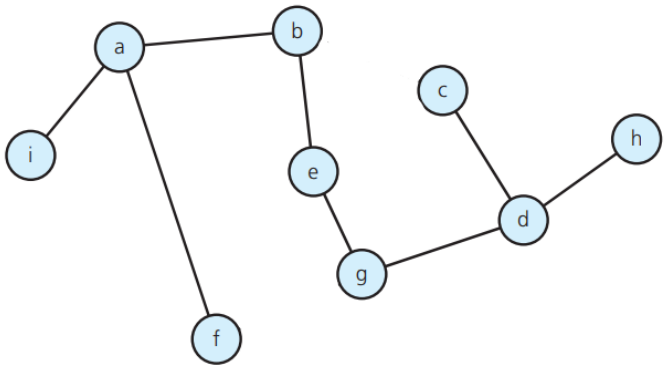
讀入上述資料後，請判斷這個圖是否為一個 tree，若是請印出一個 1，一個空白字元，以及這個 tree 的最小可能高度。如果不是一個 tree，請印出一個 0。

下圖所示為一個不是 tree 的 graph，由圖中可以發現，由 node e 至 node d 之間存在二個可能的路徑，可以透過 c 或透過 g。



然後經過刪除上圖中 node c 與 e 之間的路徑後，便只剩一條路徑可以連接 e 與 d。而在此圖之中，

可以藉由選定 node e 為 root，使的其高度為 4，若是選擇其他的 node 作為 root，則其高度必定會大於之（例如選擇 node i 會使高度為 7）。



舉例來說，如果輸入為

```
9
0 1 0 0 0 1 0 0 1
1 0 0 0 1 0 0 0 0
0 0 0 1 1 0 0 0 0
0 0 1 0 0 0 1 1 0
0 1 1 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0 0
```

則輸出應該為

```
0
```

如果輸入為

```
9
0 1 0 0 0 1 0 0 1
1 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0
0 0 1 0 0 0 1 1 0
0 1 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0 0
```

則輸出應該為

```
1 4
```

你上傳的原始碼裡應該包含什麼

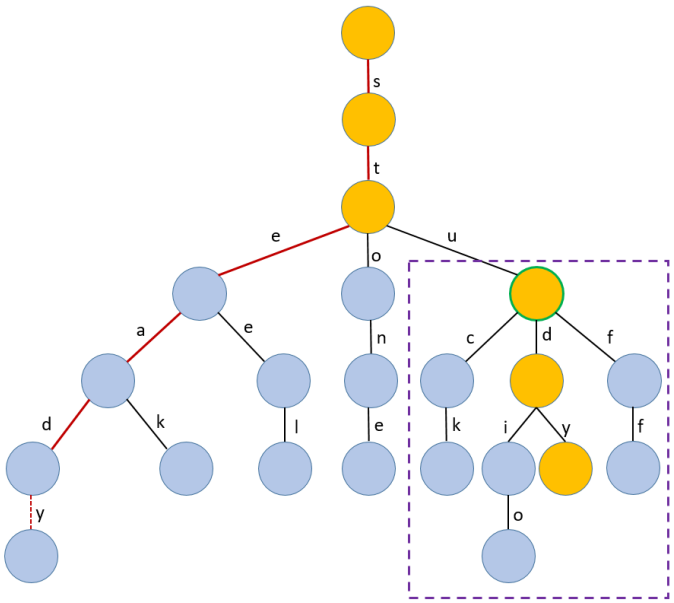
你的.cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。

評分原則

這一題的分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。

第四題

(40 分) Prefix tree 是一種建立字典時常用的資料結構，屬於一種多元樹（n-ary tree）。如下圖所示，有相同前輟的字會被儲存在同一個子樹之中。也因為如此，如果要在這個樹之中尋找一個詞（word），只需要根據它的字（letter），一一向下尋找即可。例如若需要尋找 study 這個詞，便會經過下圖中黃色的 node。對於一個具有 n 個詞的 prefix tree，尋找一個詞的複雜度平均而言可以落在 $O(\log n)$。



同理，如果要在這個樹之中加入新的詞，就從這個詞的第一個字開始，一一尋找在對應的層級中是否已經具有代表這個字的 child，若有則沿著這個 arc 往下走一層到下一個 node，若沒有則在現在所在的 node 新增一個 child，代表這個字。對這個詞的每一個字都進行這個過程後，在代表最後一個字的 node 進行標記，代表到這個 node 為止的字可以組成一個詞。例如若在這個樹之中加入 steady 這個字，就會經過圖中紅色的邊，並且於最左方的 lead node 新增一個 child（圖中虛線處）。

在本題中，我們將會給定多個詞，以及一些前輟，請將這些詞放入一個 prefix tree 之中，並將包含這些前輟的詞尋找出來。例如若要在上圖的樹中尋找包含 stu 這個前輟的所有詞，只需要尋找代表 stu 的這個 node（圖中綠框者），並對其 subtree（圖中紫框處）進行 preorder traversal，即可依照字典順序取得包含這個前輟的所有詞。

輸入輸出格式

系統會提供 20 組的測試資料，每組測試資料裝在一個檔案裡。在每個檔案之中，會有 3 行。第一行會有 2 個整數 m 及 n，二者以空白字元隔開，分別代表詞的總數，以及要搜尋的前輟總數。第二行包含 m 個字串 s_1 至 s_m，字串之間以空白字元分隔。第三行包含 n 個字串 w_1 至 w_n，字串之間以空白字元分隔。所有字串都不會重複，且只會包含英文大小寫字母。其中 $1 \leq m \leq 10^8$，$1 \leq n \leq 1000$，且任一字串 (word) 的長度都不會超過 10000。

讀入以上資料後，請在第 i 行印出在 s_1 到 s_m 之中，字首包含 w_i 的那些字串。字串之間以一個空白字元分隔，印出時請依照字典順序印出。

舉例來說，如果輸入是

```
8 2
steel steak stuff study stone studio stuck stead
stu ste
```

則輸出應該是

```
stuck studio study stuff
stead steak steel
```

你上傳的原始碼裡應該包含什麼

你的.cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。

評分原則

這一題的分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。在本題中助教會打開你的程式碼檢查，，你應該要實作一遍 tree 的資料結構，並使用 prefix tree 進行搜尋。如果你沒有這樣做，本題將**不予給分**，且你很可能會無法通過 PDOGS 的時間限制。