



# 基礎資料結構(二)

112-2 普台競程讀書會

2024.3.18

陳柏安



## 課程大綱

- 樹狀結構
- 堆積樹 Heap Tree
- 實用的內建工具 (C++ STL & Python)

---

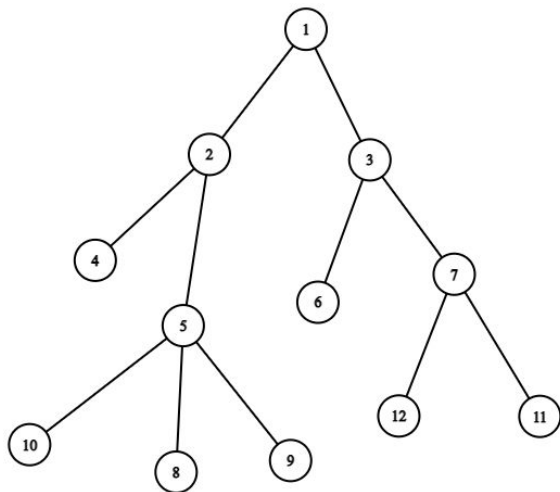
# 樹狀結構

# 樹狀結構是什麼??

請上 Slido 回答



## 資訊科學裡的樹是倒的 !!



# 樹的名詞解釋

---

# 樹的名詞解釋

## ❑ 根節點

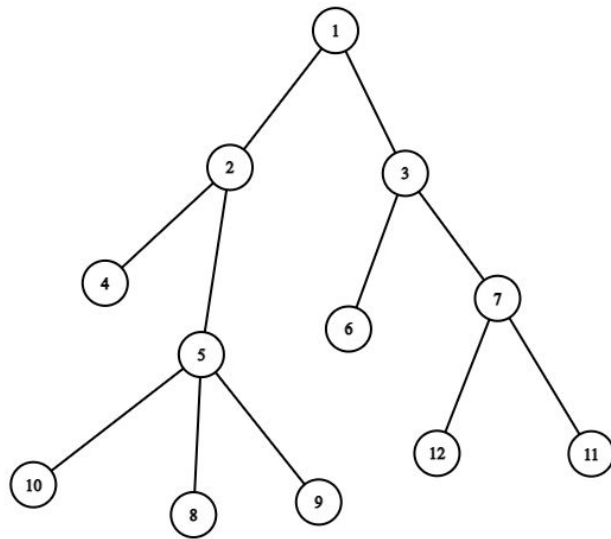
樹最上端的節點

## ❑ 父節點

樹中每個節點都可能有一個父節點，除了根節點。父節點直接指向它的子節點。

## ❑ 子節點

樹中每個節點都可能都有零個或多個子節點，這些子節點是由父節點生成的。



## 樹的名詞解釋

### ❑ 葉節點

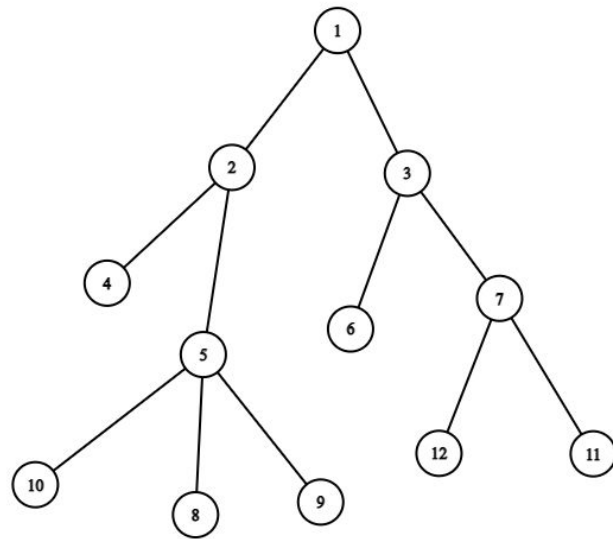
樹的終端節點

### ❑ 深度

節點  $i$  從根節點走的唯一路徑長

### ❑ 子樹

移除一個點之後,原樹會被拆成很多棵樹,稱為子樹。





# 樹的性質

---

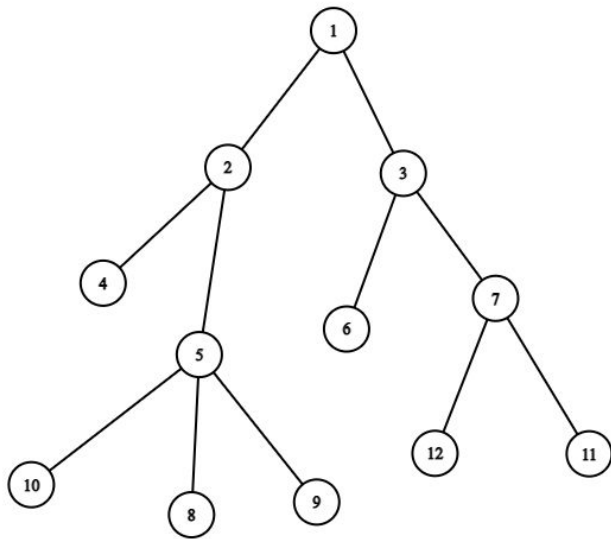
# 樹的性質

## ❑ 無向圖

所有邊皆無方向

## ❑ 無環性

沒有從一個節點出發經過若干邊又回到同一個節點的路徑。



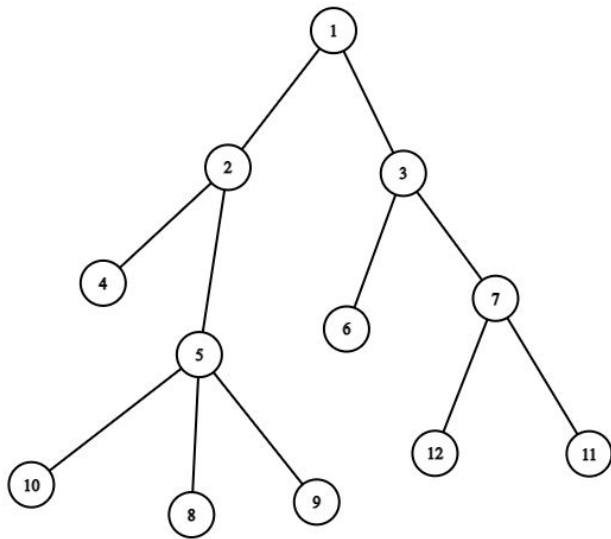
# 樹的性質

## □ 連通性

任一個節點一定存在唯一路徑  
可到另一個節點

## □ 節點與邊的數量關係

$V = E + 1$ ,  $V$  是節點數量,  $E$  是  
邊的數量



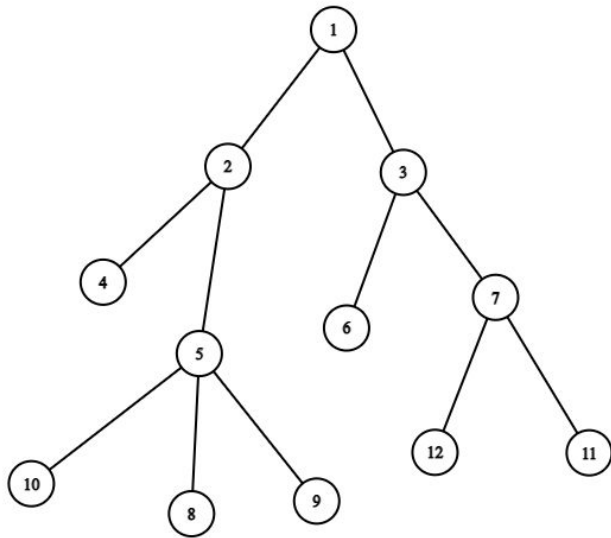
# 樹的性質

## □ 連通性

任一個節點一定存在唯一路徑  
可到另一個節點

## □ 節點與邊的數量關係

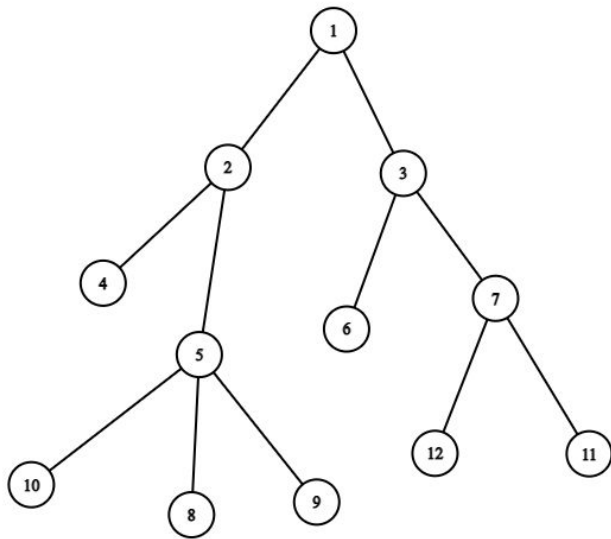
$V = E + 1$ ,  $V$  是節點數量,  $E$  是  
邊的數量



## 樹的性質

❑ 任意加上一條邊會形成環

❑ 去除一點會變不連通

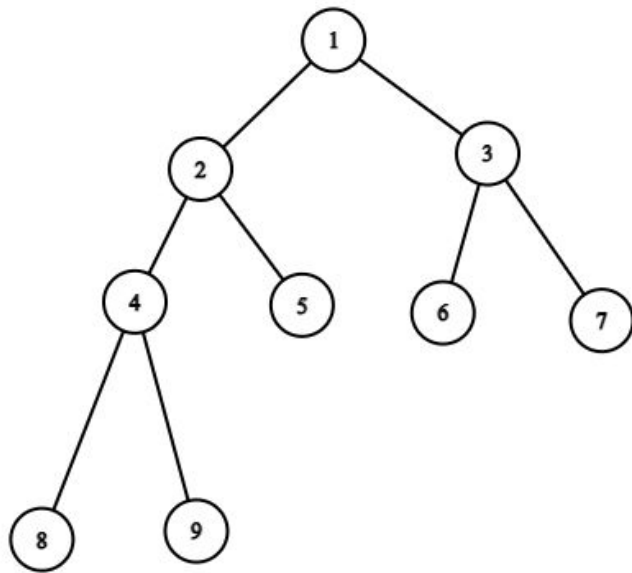


# 二元樹

---

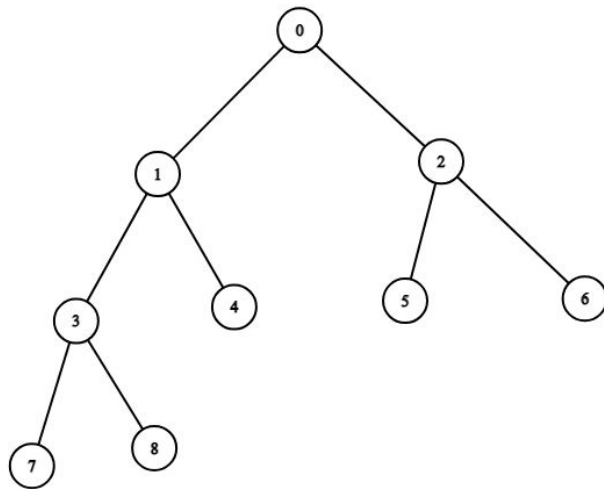
## 二元樹的定義

- ❑ 每個節點只會有兩個子節點 (葉節點除外)



## 二元樹的編號方式

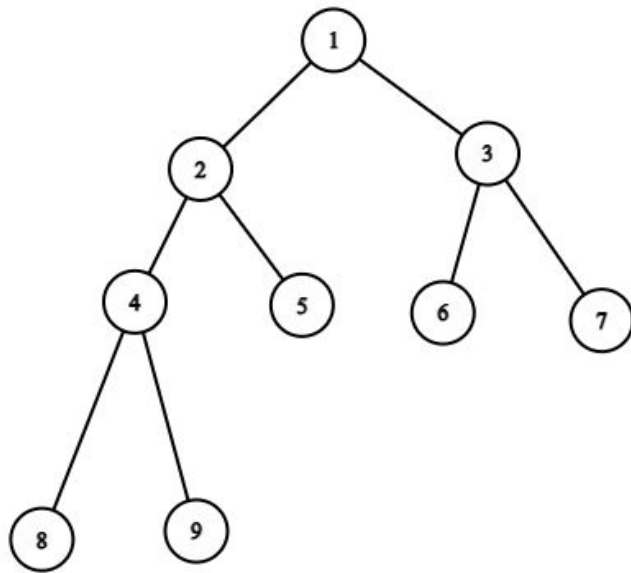
- ❑ 0 - base
- ❑ 根結點為 0





## 二元樹的編號方式

- ❑ 1 - base
- ❑ 根結點為 1



---

# 堆積樹 Heap Tree



## 堆積樹用途

積樹 (Heap Tree) 是一種可以用來維護資料「極值」，所以會有分最大堆積與最小堆積。



## 堆積樹支援的操作

- ❏ 插入元素 push
- ❏ 刪除極值 pop
- ❏ 查詢極值 top



## 還記的昨天第一堂課舉的例子嗎?

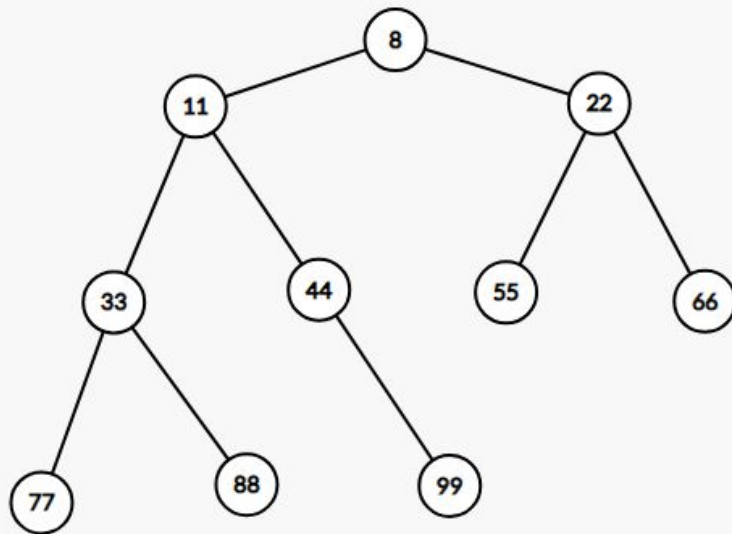
給一隨機數列  $a$ , 每次詢問數列當前最大值, 並且將其值取出。

- ❑ 這題是不是與堆積樹有似曾相似呢?
- ❑ 每次尋問就是 Heap 當中的 top 查詢
- ❑ 取出極值就是 Heap 當中的 pop 取出

# 堆積樹的性質 (課程皆以二元樹實作)

## 堆積性質


所有節點皆小於(或大於)其左右的子節點的元素



最小堆積樹

---

# 實用的內建工具



# C++ Standard Template Library, STL

詳細内容請至:

<https://hackmd.io/JnFDFgm6RvuSVmYYGIXSYQ?both>

<https://hackmd.io/@putailNF/Bk4TgvlAa>

```
#include<stack>
```

```
#include<queue>
```





## STL Stack

- ❑ 標頭檔
- ❑ 建立 Stack
- ❑ 堆入 Stack
- ❑ 取出 Stack上端元素
- ❑ 查詢 Stack上端元素

```
#include<stack>  
stack<資料型態> sk;  
sk.push(元素);  
sk.pop( );  
sk.top( );
```



## STL Queue

- ❑ 標頭檔
- ❑ 建立 Queue
- ❑ 堆入 Queue
- ❑ 取出 Queue 前端元素
- ❑ 查詢 Queue 前端元素

```
#include<queue>  
queue<資料型態> qu;  
qu.push(元素);  
qu.pop( );  
qu.front( );
```



## STL Priority queue (Heap)

- ❑ 標頭檔
- ❑ 建立 priority\_queue (預設最大堆積)
- ❑ 堆入 priority\_queue
- ❑ 取出 priority\_queue 上端元素
- ❑ 查詢 priority\_queue 上端元素
- ❑ 設定 priority\_queue 為最小堆積

```
#include<queue>
priority_queue<資料型態> pq;
pq.push(元素);
pq.pop();
pq.top();
priority_queue<資料型態, vector<資料型態>, greater<資料型態>>;
```