HW4\_Extraction of Connected Components

1. Extraction of Connected Components(連通分量)

連通分量是指圖片中的一組相鄰的黑色像素，它們形成一個區塊，也就是說若兩個點之間存在一條路徑，則這兩個點屬於同一個連通分量。

提取連通分量是在圖片處理中一個非常常見的操作，可以用來對圖片中的物體進行計數或辨識，通常在提取連通分量時，我們會使用演算法來找出圖片中所有連通分量的像素位置，例如使用深度優先搜尋(Depth-First Search)或廣度優先搜尋(Breadth-First Search)來實踐連通分量的提取。

1. 深度優先搜尋演算法(DFS)

DFS的核心精神就是先遇到的頂點就先Visiting，並且以遇到的頂點做為新的搜尋起點，直到所有相連的頂點都被探索過。

在提取連通分量的問題中，DFS可以透過搜尋所有相鄰的像素，來尋找與起始位置有相連的所有像素，透過不斷遞迴DFS函數，可以把圖片中所有連通分量都提取出來。

具體來說，首先將起始位置的像素標記為已拜訪，並把它加入到連通分量的陣列中，然後搜尋與起始位置相鄰的所有像素，如果該像素值為0(黑色)且沒有被拜訪過，就對該像素再使用DFS函數，進一步搜尋與它相連的像素，這樣就能夠找出整張圖片中所有與起始位置相連的像素。

1. 將起始位置的像素標記為已拜訪，並將它加入道連通分量的陣列中
2. 搜尋起始位置的相鄰像素，如果該像素值為0且沒有被拜訪過，則調用DFS函數搜尋與該像素相連的像素
3. 對於每個相鄰像素，重複步驟1和2，直到找出整個連通分量。

三、程式實作

首先，我們進入draw\_img函數，為我們畫一張大小為(100,300)的全白(像素值為255)的灰階圖片，並使用puText在這張全白的灰階圖片上寫字，寫完字後回傳。



圖 一 程式起始點

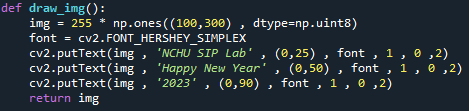


圖 二 draw\_img函數內容



圖 三 輸出的圖片

第二步我們將圖三的圖片傳入ConnectedComponents函數，函數會先讀取圖片的row和col，並且製作一張與圖片大小相同、全為False的bool圖片visited，用於紀錄是否拜訪過，再建立一個空陣列components用於儲存所有的連通分量，之後用雙層迴圈讀取圖片中每一個像素值，若像素值尚未被拜訪過且該像素值為黑色(0)，就建立一個空陣列component，用於儲存此像素的連通分量，之後將圖片、此像素的位置以及visited、component傳入DFS函數，開始針對此像素值進行DFS搜尋，搜尋後的結果再append進components陣列。



圖 四 呼叫ConnectedComponents函數

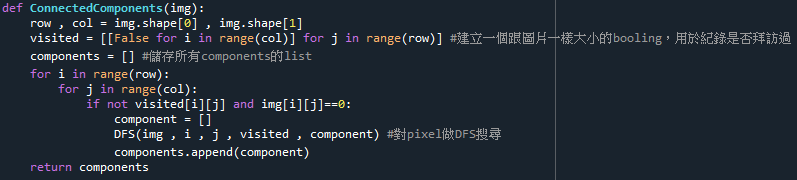


圖 五 ConnectedComponents函數內容

進入DFS函數後，要再確認一次像素值是否為0以及是否沒有拜訪過，因為剛剛我們所確認的是原始點，原始點往外擴散的其他點也同樣會繼續當作原始點，所以要再確認一次。確認過後，我們將該點設為True，表示我們拜訪過這次的pixel，之後不再拜訪，之後將該點位置append進component陣列內，代表此原始點的連通分量，之後使用遞迴的方式搜尋該點的上下左右四個像素點，全部的遞迴完成後即可得到此次的連通分量。

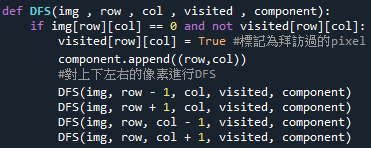


圖 六 DFS函數內容

待每一個像素點都順過一輪後，就能提取我們的連通分量，components內有幾個component代表此圖有幾個區塊。得到結果後，我們再將結果傳入print\_table函數，將我們的結果印出來。



圖 七 呼叫print\_table

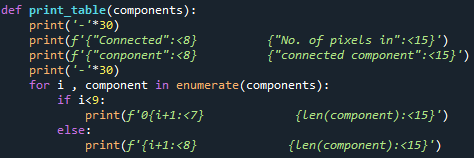


圖 八 print\_table內容

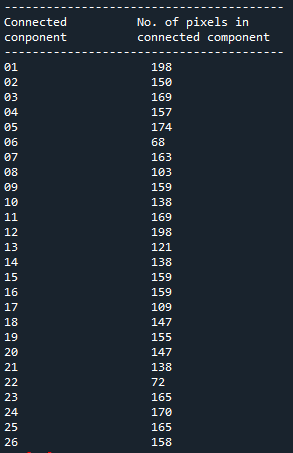


圖 九 Extraction of Connected Components結果

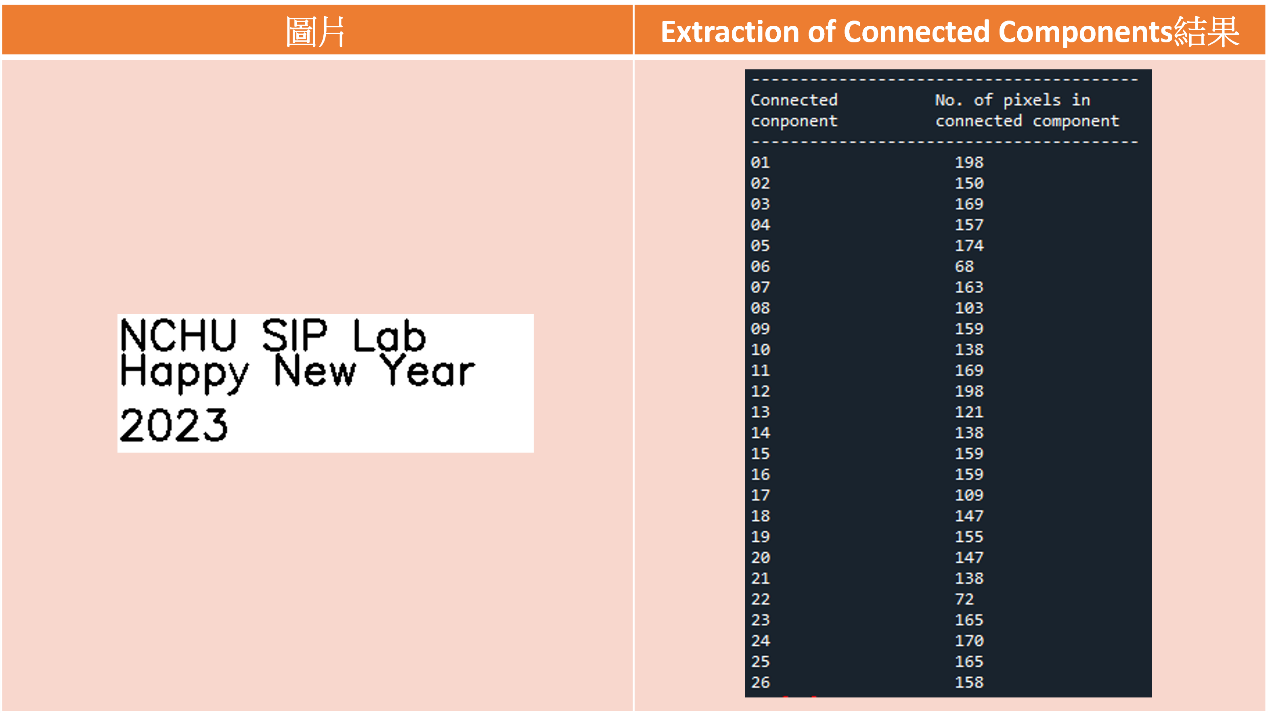


圖 十 對照圖

從圖十來看，可以發現確實原始圖片有26個區塊，且第6塊區塊(I)及第22塊區塊(r)的pixel值較少，這是合理的，因此程式邏輯無誤。

完整程式碼如下：

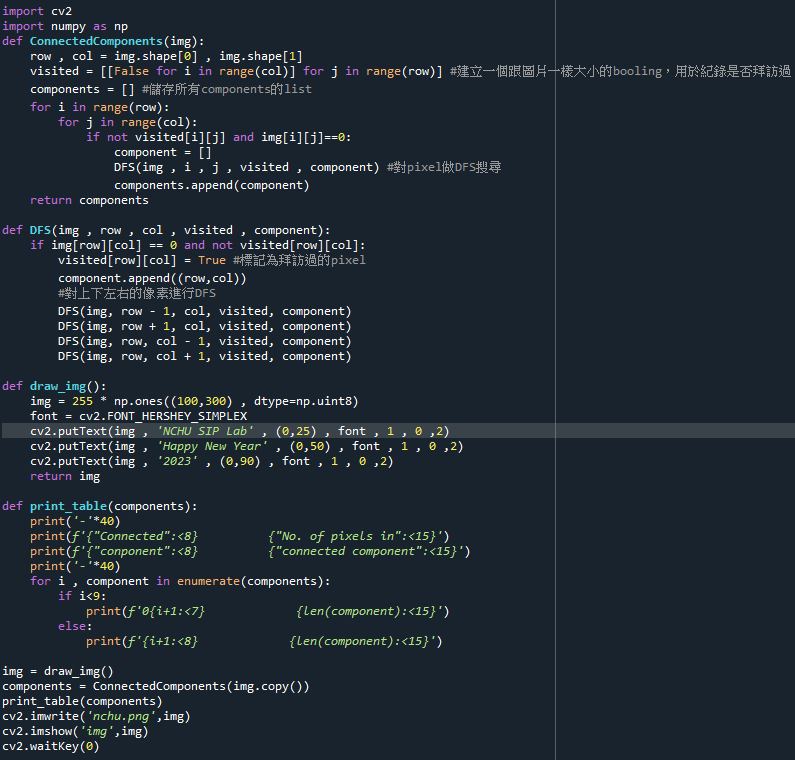


圖 十一 完整程式碼

四、心得

從這次作業當中可以學到何謂Connected\_Components，以及了解到其中的應用，也因此學到了DFS的應用，可以說是一舉兩得，此次作業雖然比前三次作業的程式碼都要來的短，但觀念也不少，需要蒐集許多資料才能寫出來。