Final project

- Digital system design

104070040 科管 19 林聖亞

一、 設計構想

1.Topological sort

在圖論中，拓撲排序（Topological Sorting）是一個有向無環圖（DAG, Directed Acyclic Graph）的所有頂點的線性序列。該序列必須滿足下面兩個條件：

(1)每個頂點出現且只出現一次。

(2)若存在一條從頂點 A 到頂點 B 的路徑，那麼在序列中頂點 A 出現在頂點 B 的前面。

有向無環圖（DAG）才有拓撲排序，非 DAG 圖就沒有拓樸排序。

2.軟體分析

分析軟體程式碼，判斷適合放入硬體執行的部分，考慮硬體的優勢是在處理重複運算下加速較明顯，所以選擇加速運算部分。

(1)重複加法

```
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        indeg[i]=indeg[i]+a[j][i];
```

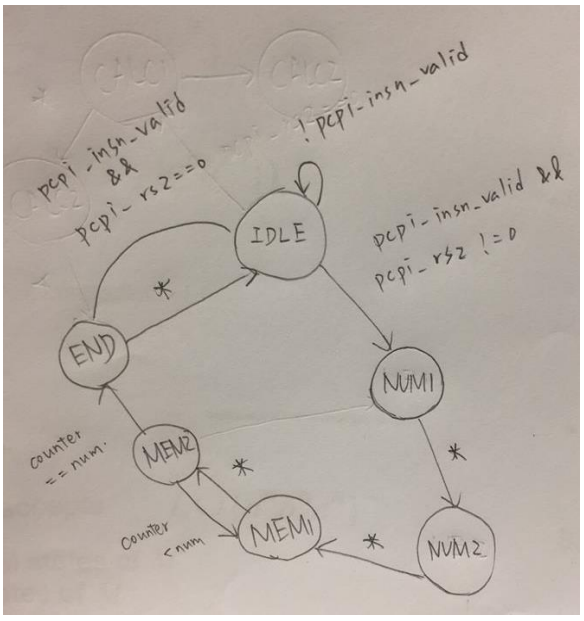
(2)重複減法

```
for(i=0;i<n;i++){
    if(a[k][i]==1)
        indeg[i]--;
}
```

因為測資假設在能做拓樸排序的前提下，所以沒有特別去寫 cycle 的偵測。

3.硬體設計

根據軟體丟入的參數範圍去判斷該執行那個 state transition cycle，

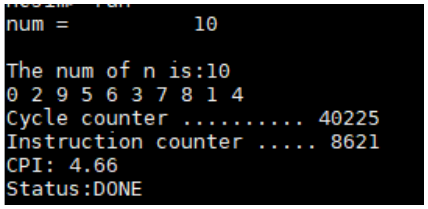


二、 結果分析

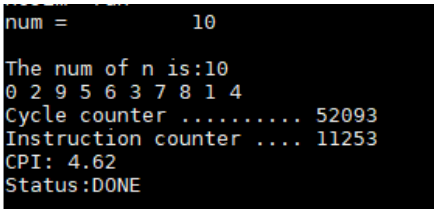
因為沒有寫自動生成 input 的軟體，所以以小範圍測資為主，但 testbench 裡寫的 memory 最大值是 5000，所以最多只能跑 70 多個 vertex 的測資。

Case1

硬體加速後

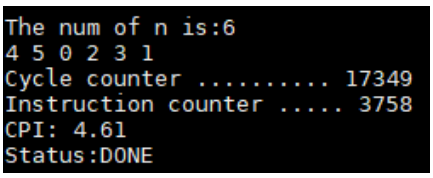


純軟體

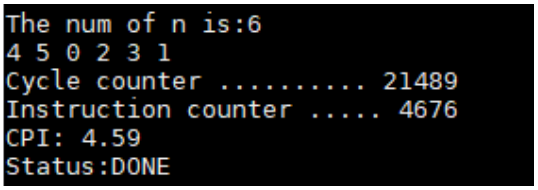


Case2

硬體加速後



純軟體



結果上來看硬體是真的有加速，測資愈大可以看出兩者差距愈大，然後我嘗試有將 indeg--的部分改回軟體執行，發現單純將 indeg--放入硬體並沒有比較快，推測可能是因為軟硬體頻繁交換，執行的時間反而會更長。

三、 瓶頸

這次卡最久的地方一樣是吃資料的部分，這次讀檔的部分研究了許久，把 memory 那邊的 code 又再 trace 了一次，畢竟是之前做 knn 沒有用過的部分，在這個部分上就花了比較多的時間。

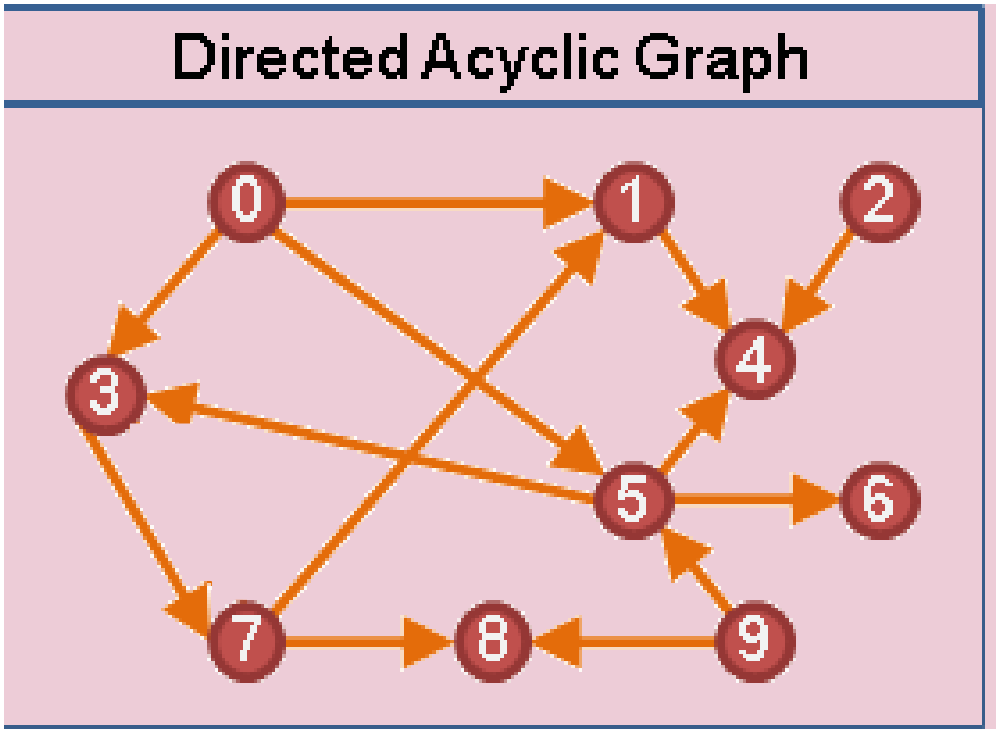
四、 問題與處理---軟體 bug

1.測資格式

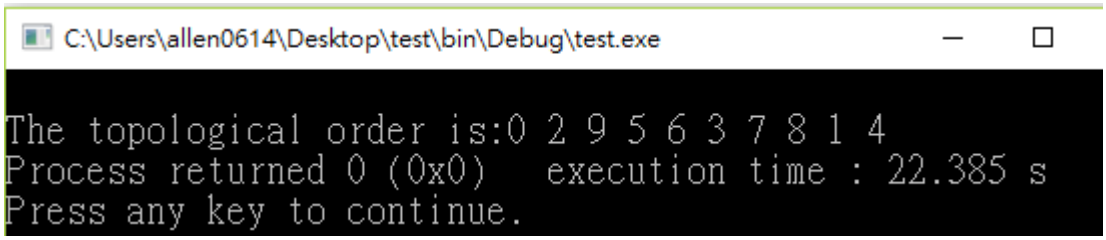
```
4
0 1 1 0
0 0 0 1
0 0 0 1
0 0 0 0
```

	Node 0	Node 1	Node 2	Node 3	Total outdeg
Node 0	0	1	1	0	2
Node 1	0	0	0	1	1
Node 2	0	0	0	1	1
Node 3	0	0	0	0	0
Total indeg	0	1	1	2	

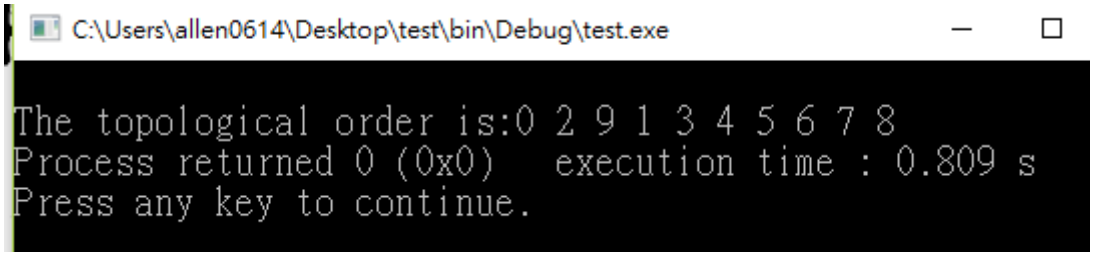
2.case



(1)預期執行結果



(2)實際執行結果



3.bug

```
while(count<n){
    for(k=0;k<n;k++){
        if((indeg[k]==0) && (flag[k]==0)){
            printf("%d ",(k+1));
            flag [k]=1;
        }

        for(i=0;i<n;i++){
            if(a[i][k]==1)
                indeg[k]--;
        }
    }

    count++;
}
```

這是網站上的程式碼，從 0 開始，偵測 indeg 是否為 0，若為 0 則將其標記，然後檢測 matrix 裡 column 的值，若為 1 則歸 0，用 graph 去想，就是將所有指向這個 vertex 的 edge 移除掉，因為是 for 迴圈，所以會從 vertex 0 到 vertex n-1 依序走一次，只有在處理初始 indeg 為 0 的狀況下是對的，當 count++，因為所有 indeg 都被歸 0，所以接下來只會順序印出，但是之後 vertex 之間其實還是有相對關係的存在，因此產生了 bug。

4.debug

```
while(count<n){
    for(k=0;k<n;k++){
        if((indeg[k]==0) && (flag[k]==0)){
            printf("%d ",(k));
            flag [k]=1;
            for(i=0;i<n;i++){
                if(a[k][i]==1)
                    indeg[i]--;
            }
        }
    }
    count++;
}
```

對軟體程式碼做了修改，indeg 歸 0 的部分放到 if 裡面，代表的意思是當我偵測到 indeg 為 0 時，將這個 vertex 與所有指出的 edge 移除掉，因為 indeg 為 0 所以沒有指向這個 vertex 的 edge，若是沒有 cycle 的話，一定會找的到 indeg 為 0 的 vertex，移除掉的話也會產生新的 indeg 為 0 的 vertex。

5.原版與新版導致的硬體差異

我的硬體初始設計裡面沒有存 indeg 的值，若是要存的話得開個很大的 reg，設計上覺得很麻煩，所以 indeg 值必須由外面傳入，所以在前提是 call 一次硬體傳入兩個參數的狀況下，原版中(圖 1)傳入 indeg[k]與 k 就能處理掉，但在新版(圖 2)要處理掉這個 for 迴圈，需要的是 indeg[i]與 k，但又因為 i 是在 for 迴圈中所賦予的，所以必須在 for 迴圈中，這樣除非我在硬體存放每個 indeg 值，要不然沒辦法把新的 for 迴圈丟進去硬體做。

圖 1

```
for(i=0;i<n;i++){
    if(a[i][k]==1)
        indeg[k]--;
}
```

圖 2

```
for(i=0;i<n;i++){
    if(a[k][i]==1)
        indeg[i]--;
}
```

五、 思考

從 GCD 開始接觸 picorv32，到這次才算真的了解它，從吃檔案到得到想要的資料，不過可能是我太晚開始動工，到最後全部做完時才發現軟體 bug，已經來不及在重新架構我的硬體部分，只能就現有架構去做修正，因此加速效果其實沒有達到最好，不論是對現有的硬體架構的優化或是新增新的硬體部分都是可以繼續再做的，如果行有餘力的話，還能把 cycle detector 什麼的丟到硬體去做，讓程式碼更完整，或是去做測資自動生成與檢核，其實還蠻多東西可以繼續做，但無奈鄰近期末時間壓力大，沒辦法將其做的更好，只能完成基本功能符合題目基本要求而已。