

Project 4: Kruskal's algorithm

(1)Implementation:

T(選出來的邊)=空集合

While((T 包含的邊少於 n-1 個)&&(E 不是空的))

 從 E 中選一個花費最低的邊(v,w)

 從 E 中刪除(v,w)

 If((v,w)不會在 T 中產生迴路)=>把(v,w)加到 T

 Else 忽略(v,w)

```
edge { int a[3];
```

```
struct edge* next;
```

```
}; //member in T,E
```

```
vertice { int v[2]; //v[0]=ver_num;v[1]=set_num;
```

```
int access_num; //to find start vertice
```

```
}; //vertice
```

*處理字串:

(1)若讀入' ':新增一個 edge node 且把它跟之前的 edge linked list 接起來。

(2)若讀入',':將 buffer 裡的字轉整數存入 edge 的 a[0]或 a[1](vertice),清空 buffer 和 b_index。再檢查 vertice array 裡有無此 vertice,有則不做事,無則插入 vertice array。A_index++(下個 number)。

(3)若讀入')': 將 buffer 裡的字轉整數存入 edge 的 a[2](weight),清空 buffer 和 a_index,b_index。

(4)否則(數字): 插入 buffer。

*找最小邊:

先設 edge_root 的邊為最小邊(low),然後 run 過整個 edge list,若比 low 小=>low 等於該邊。

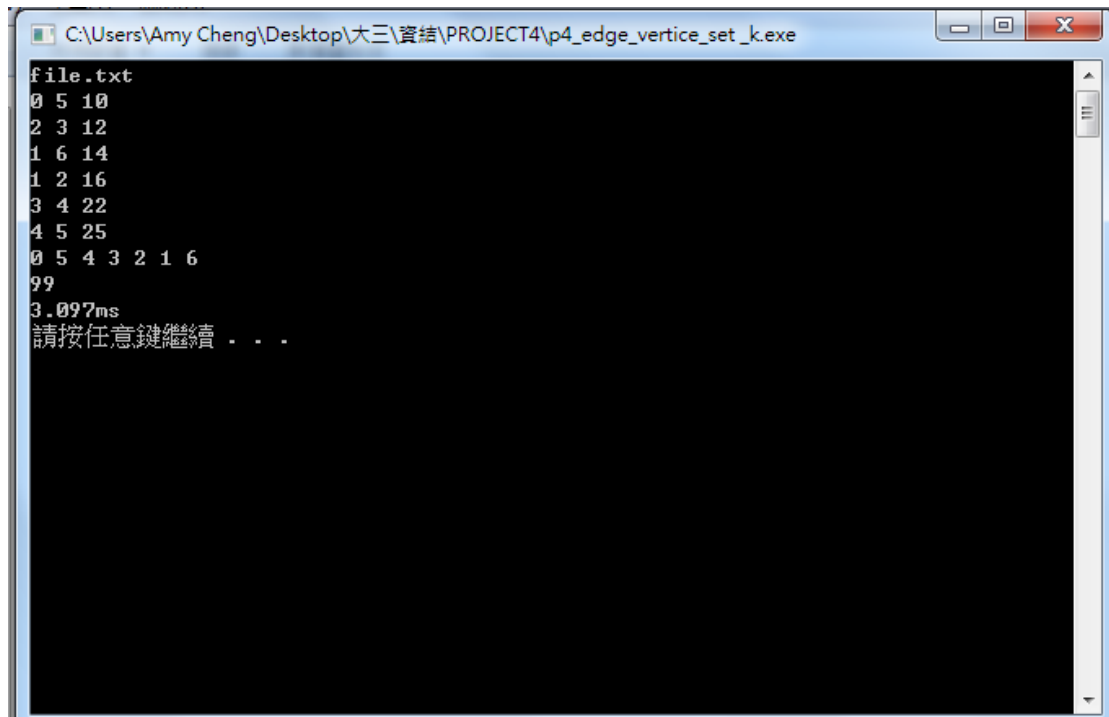
*判斷有無產生 cycle:

一開始每個頂點是不同 set(v.set_num)。若選了邊(v1,v2)則將 v1,v2 改為同 set。當挑最小邊時碰到 v1.set=v2.set 則會產迴圈所已放棄,否則就可以加入 T。

*path:

先找到起始頂點,也就是只有被用到一次的頂點(v.access_num=1)。然後把頂點 push 到 stack 中以紀錄 path 後,再去尋找用到這個頂點的 edge,不斷循環。

(2)Records:



```
C:\Users\Amy Cheng\Desktop\大三\資結\PROJECT4\p4_edge_vertice_set_k.exe
file.txt
0 5 10
2 3 12
1 6 14
1 2 16
3 4 22
4 5 25
0 5 4 3 2 1 6
99
3.097ms
請按任意鍵繼續 . . .
```