Data Structure Assignment 7

ID: E14066282	**Name:** 溫梓傑	**Department:** ME 110

○ Result Screenshots

```
Wun@DESKTOP-1MSRHTB MINGW64 /c/2020-NCKU_DS/HW7_AOE_graph/code (master)
$ gcc -std=c11 ./*.c -o hw7

Wun@DESKTOP-1MSRHTB MINGW64 /c/2020-NCKU_DS/HW7_AOE_graph/code (master)
$ ./hw7.exe < input0_windows.txt > ans0_windows.txt
egde_num: 11
Normal:
v_0  : 0   --> v_1  , a_0  , 6   --> v_2  , a_1  , 4   --> v_3  , a_2  , 5
v_1  : 1   --> v_4  , a_3  , 1
v_2  : 1   --> v_4  , a_4  , 1
v_3  : 1   --> v_5  , a_5  , 2
v_4  : 2   --> v_6  , a_6  , 9   --> v_7  , a_7  , 7
v_5  : 1   --> v_7  , a_8  , 4
v_6  : 1   --> v_8  , a_9  , 2
v_7  : 2   --> v_8  , a_10 , 4
v_8  : 2

Invert:
v_0  : 3
v_1  : 1   --> v_0  , a_0  , 6
v_2  : 1   --> v_0  , a_1  , 4
v_3  : 1   --> v_0  , a_2  , 5
v_4  : 2   --> v_1  , a_3  , 1   --> v_2  , a_4  , 1
v_5  : 1   --> v_3  , a_5  , 2
v_6  : 1   --> v_4  , a_6  , 9
v_7  : 1   --> v_4  , a_7  , 7   --> v_5  , a_8  , 4
v_8  : 0   --> v_6  , a_9  , 2   --> v_7  , a_10 , 4
Calculate early: g->v_start = 0
delete 0 node.
Calculate late: g_inv->v_start = 8
delete 0 node.
Vertex Record: [e] [l]
v_0         :   0   0
v_1         :   6   6
v_2         :   4   6
v_3         :   5   8
v_4         :   7   7
v_5         :   7  10
v_6         :  16  16
v_7         :  14  14
v_8         :  18  18
delete 0 node.
delete 0 node.
Edge Record: (e) (l)
a_0         :   0   0
a_1         :   0   2
a_2         :   0   3
a_3         :   6   6
a_4         :   4   6
a_5         :   5   8
a_6         :   7   7
a_7         :   7   7
a_8         :   7  10
a_9         :  16  16
a_10        :  14  14
Critical edges: #6
0 3 6 7 9 10

Wun@DESKTOP-1MSRHTB MINGW64 /c/2020-NCKU_DS/HW7_AOE_graph/code (master)
$ diff ../output0_windows.txt ./ans0_windows.txt
```

Figure 1 Screenshot of command line

ans0_windows.txt

HW7_AOE_graph > code > ans0_windows.txt

```
1   0 0 0
2   1 0 2
3   2 0 3
4   3 6 6
5   4 4 6
6   5 5 8
7   6 7 7
8   7 7 7
9   8 7 10
10  9 16 16
11  10 14 14
12  0 3 6 7 9 10
```

Figure 2 ans_output0_windows.txt

○ Program Architecture

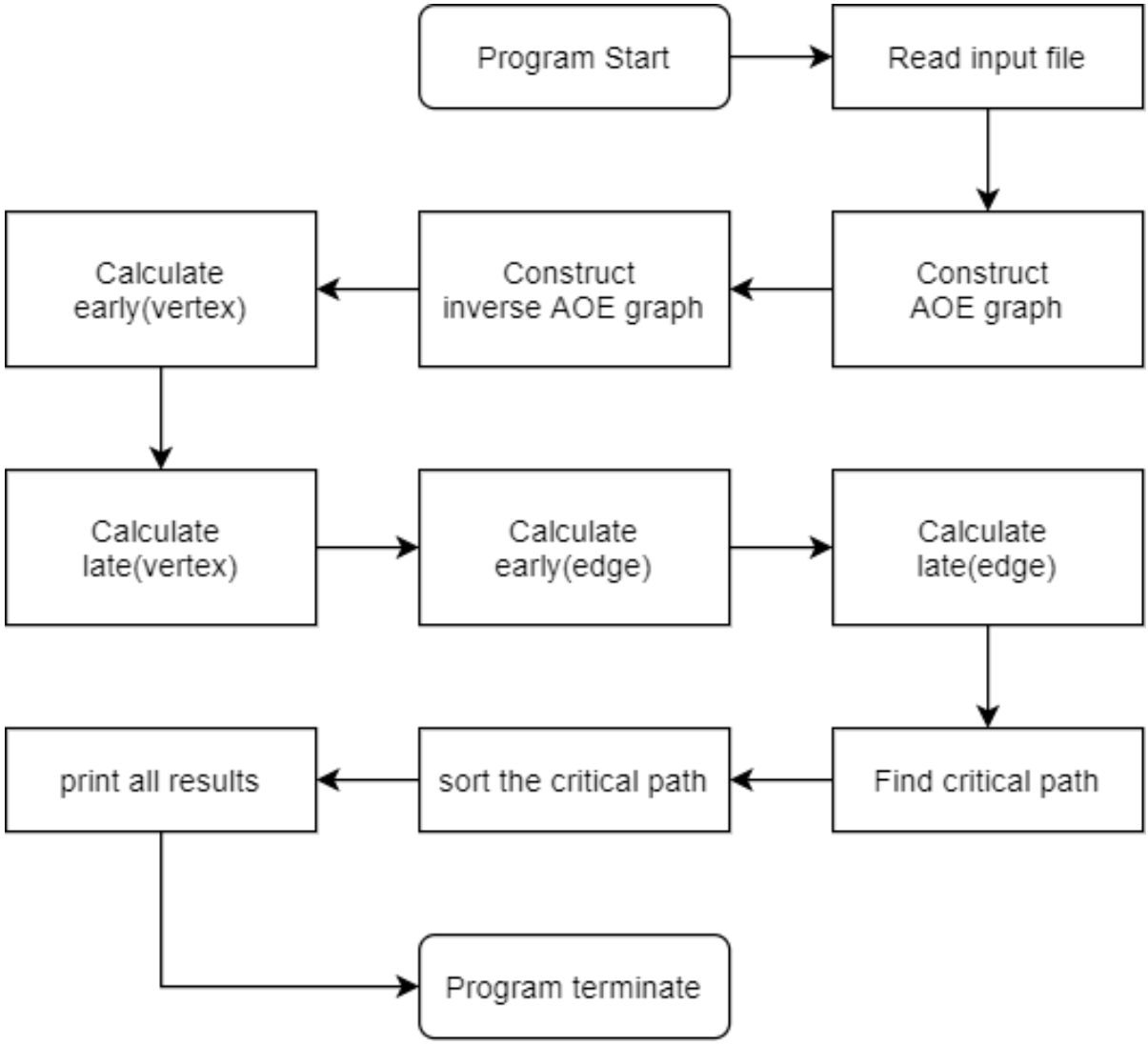

Figure 3 Flow chart of hw7

○ Program Functions

- **graph.h**

• Data Structure of graph, vheadnode, gnode

```
typedef struct gnode
{
    int vertex;
    int weight;
    int edge_code; // A -> B: B.edge_code = E(A,B)
    struct gnode *link;
} gnode;

typedef struct vheadnode
{
    int count;
    gnode *link;
} vheadnode;

typedef struct graph
{
    vheadnode *vhead;
    int v_num;
    int v_start; // default = 0
} graph;
```

```
graph *create_aoe(int v_num)
```

Construct a graph object.

• Parameters

`v_num`

The size of vheadnode array.

• Return Value

Returns the new pointer the graph.

- If construction fails, returns NULL.


```
gnode *create_gnode(int v_end, int weight, int edge_code)
```

A gnode represents a weighted-edge and the succeeding vertex.

Construct a gnode object.

- **📐Parameters**

`v_end`

The vertex code which is the end of edge.

`weight`

The weight of the edge.

`edge_code`

The code of the edge which is corresponding to the input.

- **⮌Return Value**

Return the new pointer of gnode.

```
int find_vstart(graph *g)
```

Find the start vertex of the graph.

- **📐Parameters**

`g`

The graph to be found.

- **⮌Return Value**

The start code of the vertex of the graph.

```
graph *construct_aoe(int **g_in, int e_num)
```

Construct an AOE graph object.

- **📐Parameters**

`g_in`

The description about the graph. (e_code, v_start, v_end, weight)

`e_num`

The number of the edge.

- **⮌Return Value**

Return the pointer of the AOE graph.

```
graph *construct_aoe_inv(int **g_in, int e_num)
```

Construct an inverse AOE graph object.

- **📐Parameters**

`g_in`

The description of edge about the graph. (e_code, v_start, v_end, weight)

`e_num`

The number of the edge.

- **⮌Return Value**

Return the pointer of the inverse AOE graph.

```
void cal_edge_rec(int *e_rec, int v_rec[100], graph *g, int mode)
```

Calculate the early edge or the late edge.

- **📐Parameters**

`e_rec`

The record of the edges.

`v_rec`

The record of the vertex.

`graph`

The AOE graph to be analyzed.

`mode`

0 represent the calculation of early.

1 represent the calculation of late.

- **⮌Return Value**

None.

- **Program Design**

本次作業使用 Adjacency lists 來儲存圖形，其資料結構請參考上面敘述，基本上這次作業完全依照講義上的方法來實現，使用 Topological order 來計算所有 vertex 的 early 與 late。

而在計算所有 edge 的 early 與 late 時，使用廣度優先的造訪方法來依序存取 edge 與 vertex，再套用課本公式：

$$early(i) = earliest[k]$$

$$late(i) = latest[l] - duration\ of\ activity\ a_i$$

- **Operating System**

Windows 10

- **Compiler**

gcc.exe (MinGW.org GCC Build-20200227-1) 9.2.0

- **Compile**

```
gcc -std=c11 ./*.c -o hw7
```

- **Run**

```
./hw7.exe < input.txt > output.txt
```