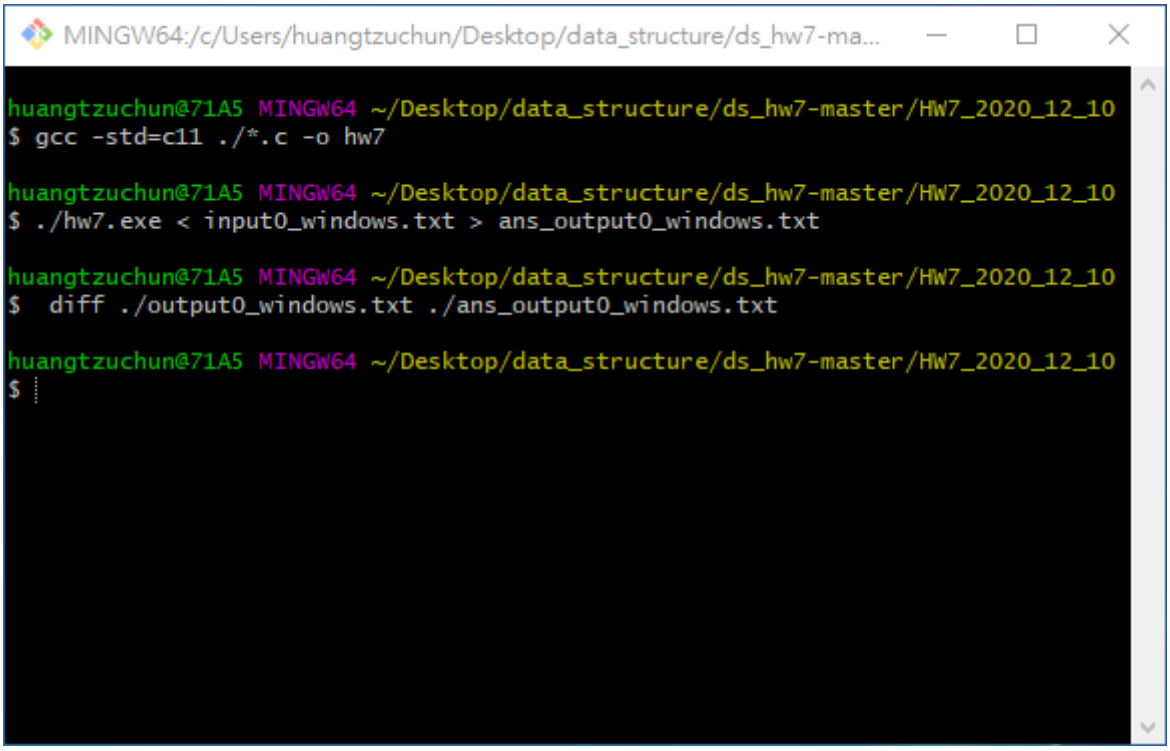
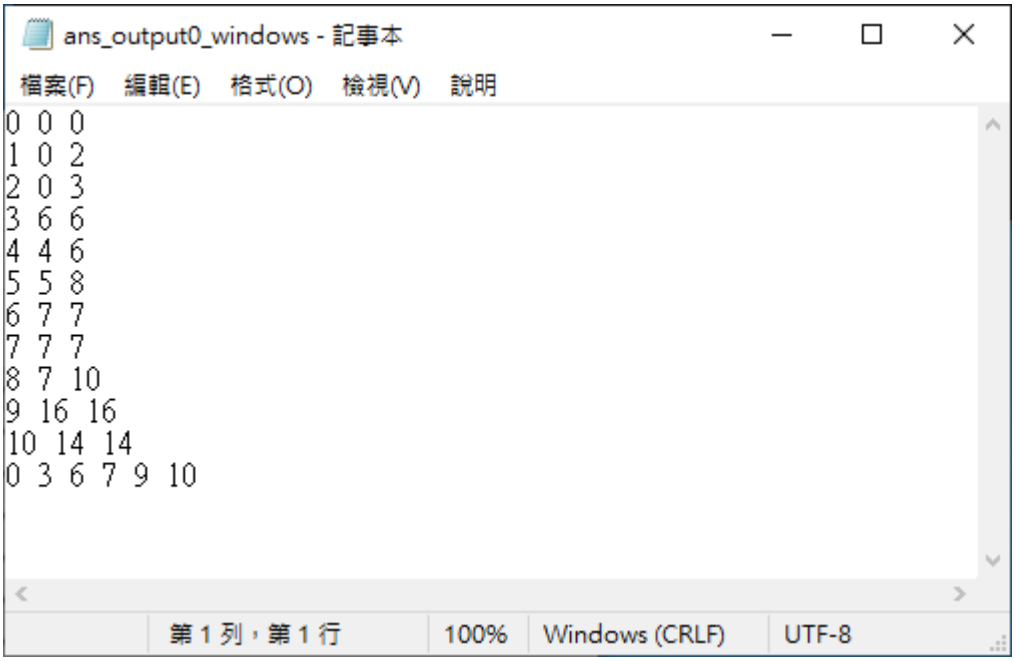
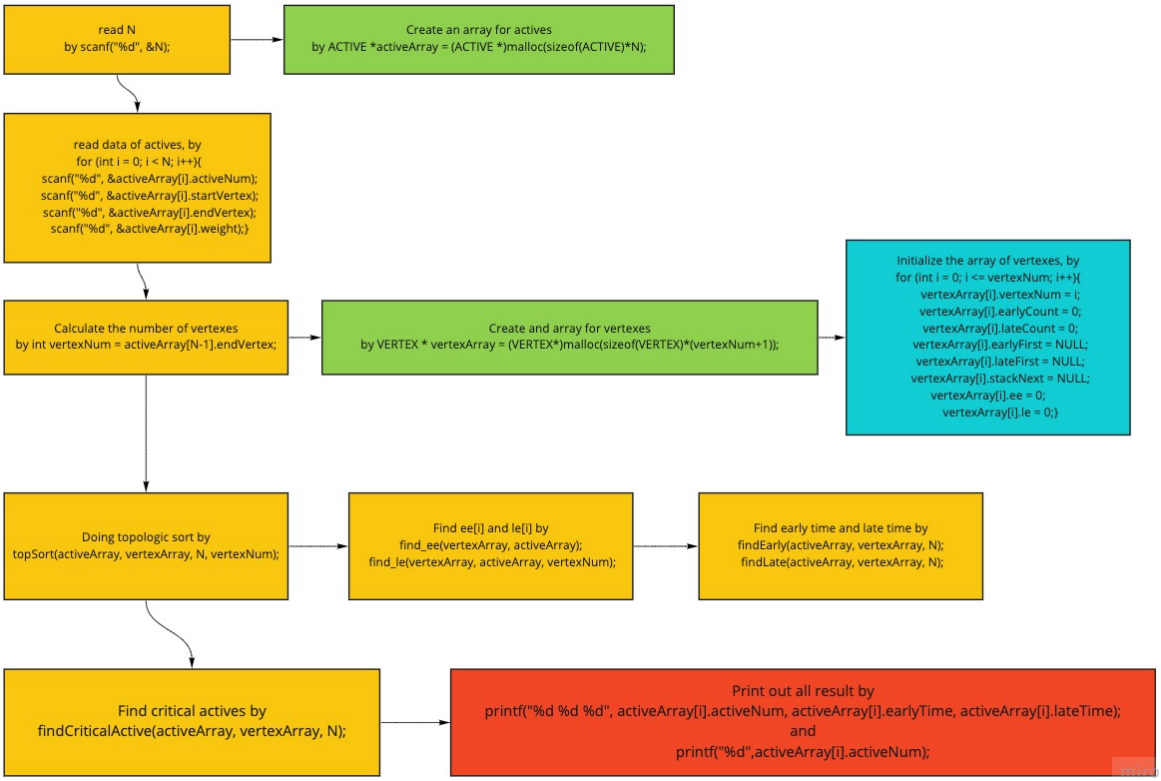
E14084117_黃子峻

Result screenshot





Program structure



Self defined type and structure

`ELEMENT` : Using in the topSort, for record link and weight data (early or late). It also apply in the stack.

```
typedef struct node ELEMENT;
struct node
{
    int nodeNum;
    union
    {
        ELEMENT *earlyNext;
        ELEMENT *lateNext;
        ELEMENT *stackNext;
    }next;
    union
    {
        int earlyDuration;
        int lateDuration;
    }duration;
};
```

`ACTIVE` : For create the array have size of N, which contain the fallowing data.

```
typedef struct active ACTIVE;
struct active
{
    int activeNum;
    int startVertex;
    int endVertex;
    int weight;
    int earlyTime;
    int lateTime;
    bool criticalActive;
};

ACTIVE *activeArray = (ACTIVE *)malloc(sizeof(ACTIVE)*N);
```

`VERTEX` : For create the array have size of vertexNum+1, which contain the fallowing data.

```
typedef struct vertex VERTEX;
struct vertex
{
    int vertexNum;
    int earlyCount;
    int lateCount;
    int ee;
    int le;
    ELEMENT *earlyFirst;
    ELEMENT *lateFirst;
};
VERTEX * vertexArray = (VERTEX*)malloc(sizeof(VERTEX)*(vertexNum+1));
```

Program functions

`void topSort(ACTIVE* activeArray, VERTEX* vertexArray, int N, int vertexNum)`

> The function to doing topologic sort.

Parameters

- `ACTIVE* activeArray` :The pointer point to the first location of `ACTIVE` type array which contain all the actives.

- `VERTEX* vertexArray` : The pointer point to the first location of `VERTEX` type array which contain all the vertexes.
- `int N` : The number of activities.
- `int vertexNum` : The number of vertexes.

Return values

- No return value (void)

`void stackAdd(ELEMENT **topPtr, int addNum)`

> The function for adding an ELEMENT to the stack.

Parameters

- `ELEMENT **topPtr` : The pointer point to the `ELEMENT *stack_top` for modify it's value.
- `int addNum` : The number for create an new `ELEMENT` and add to the stack.

Return values

- No return value (void)

`int stackPop(ELEMENT **topPtr)`

> The function for delete an ELEMENT from the stack.

Parameters

- `ELEMENT **topPtr` : The pointer point to the `ELEMENT *stack_top` for modify it's value.

Return values

- No return value (void)

`void find_ee(VERTEX* vertexArray, ACTIVE *activeArray)`

> The function to find ee[i] of all vertex.

Parameters

- `ACTIVE* activeArray` :The pointer point to the first location of `ACTIVE` type array which contain all the actives.
- `VERTEX* vertexArray` : The pointer point to the first location of `VERTEX` type array which contain all the vertexes.

Return values

- No return value (void)

`void find_le(VERTEX* vertexArray, ACTIVE *activeArray, int vertexNum)`

> The function to find le[i] of all vertex.

Parameters

- `ACTIVE* activeArray` :The pointer point to the first location of `ACTIVE` type array which contain all the actives.
- `VERTEX* vertexArray` : The pointer point to the first location of `VERTEX` type array which contain all the vertexes.
- `int vertexNum` : The number of vertexes.

Return values

- No return value (void)

`void findCriticalActive(ACTIVE *activeArray, VERTEX *vertexArray, int N)`

> The function for label all critical actives.

Parameters

- `ACTIVE* activeArray` :The pointer point to the first location of `ACTIVE` type array which contain all the actives.
- `VERTEX* vertexArray` : The pointer point to the first location of `VERTEX` type array which contain all the vertexes.
- `int N` : The number of activities.

Return values

- No return value (void)

`void findEarly(ACTIVE *activeArray, VERTEX *vertexArray, int N)`

> Functions to find the early time for all actives.

Parameters

- `ACTIVE* activeArray` :The pointer point to the first location of `ACTIVE` type array which contain all the actives.
- `VERTEX* vertexArray` : The pointer point to the first location of `VERTEX` type array which contain all the vertexes.
- `int N` : The number of activities.

Return values

- No return value (void)

`void findLate(ACTIVE *activeArray, VERTEX *vertexArray, int N)`

> Functions to find the late time for all actives.

Parameters

- `ACTIVE* activeArray` :The pointer point to the first location of `ACTIVE` type array which contain all the actives.
- `VERTEX* vertexArray` : The pointer point to the first location of `VERTEX` type array which contain all the vertexes.
- `int N` : The number of activities.

Return values

- No return value (void)