Started on	Tuesday, 1 July 2025, 4:20 PM
State	Finished
Completed on	Tuesday, 1 July 2025, 4:21 PM
Time taken	30 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Viết chương trình đọc vào một *đồ thị có hướng không chu trình* G. Áp dụng thuật toán sắp xếp topo theo phương pháp duyệt theo chiều rộng để sắp xếp các đỉnh của G. In các đỉnh ra màn hình theo thứ tự topo.

Đầu vào (Input)

Dữ liệu đầu vào được nhập từ bàn phím với định dạng:

- Dòng đầu tiên chứa 2 số nguyên n và m, tương ứng là số đỉnh và số cung.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v mô tả cung (u, v).

Đầu ra (Output)

- In các đỉnh ra màn hình theo thứ tự topo. In các đỉnh trên một dòng, cách nhau 1 khoảng trắng.

For example:

Input	Result
3 2 1 3 3 2	1 3 2
7 10 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7 6 7	1 2 4 6 5 3 7
7 12 1 2 1 3 2 4 2 5 2 6 3 2 3 5 3 6 4 7 5 7 6 4 6 5	1 3 2 6 4 5 7

Answer: (penalty regime: 10, 20, ... %)

```c
#include <stdio.h>
#define MAX_N 100

//__________LIST___________
typedef struct
{
    int data[MAX_N]; // mảng bao gồm các phần tử của danh sách
    int size;            // độ dài của danh sách
} List;

void make_null_list(List *pL)
{
    pL->size = 0;
    // (*L).size=0;
}

// Thêm một phần tử mới vào cuối ds
void push_back(List *pL, int x)
{

        // pL->data[pL->size] = x;
        // pL->size++;
        pL->data[pL->size++] = x;

}

// Trả về phần tử ở vị trí p
int element_at(List *pL, int i)
```

```
{
    return pL->data[i-1];
}

void copy_list(List *pS1, List *pS2) {
    make_null_list(pS1);
    for (int i = 1; i <= pS2->size; i++)
        push_back(pS1, element_at(pS2, i));
}

//_________________QUEUE_____________________

typedef struct
{
    int data[MAX_N];
    int front, rear;
} Queue;

// khoi tao hang doi
void make_null_queue(Queue *pQ)
{
    pQ->front = 0;
    pQ->rear = -1;
}

// dua ptu vao cuoi hang doi
void enqueue(Queue *pQ, int u)
{
    pQ->rear++;
    pQ->data[pQ->rear] = u;
}

int front(Queue *pQ)
{
    return pQ->data[pQ->front];
}

void dequeue_queue(Queue *pQ)
{
    pQ->front++;
}

int empty_queue(Queue *pQ)
{
    return pQ->front > pQ->rear;
}

//_________________Graph_________________

typedef struct
{
    int n, m;
    int A[MAX_N][MAX_N];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
        for (int v = 1; v <= n; v++)
            pG->A[u][v] = 0;
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] += 1;
    // pG->A[v][u] = 1;
    pG->m++;
}

//_____________________TOPO_____________________________

void topo_sort(Graph *pG, List *pL)
{
    int d[MAX_N]; // luu bac vao cua dinh u

    // tính bậc vào của đỉnh u
    for (int u = 1; u <= pG->n; u++)
    {
        d[u] = 0;
        for (int x = 1; x <= pG->n; x++)
            if (pG->A[x][u] != 0) // bậc vào của u
                d[u]++;
    }

    Queue Q;
```

```
    // làm rỗng hàng đợi
    make_null_queue(&Q);

    // đưa các đỉnh có d[u] = 0 vào hàng đợi
    for (int u = 1; u <= pG->n; u++)
    {
        if (d[u] == 0)
            enqueue(&Q, u);
    }

    // làm rỗng danh sách
    make_null_list(pL);

    // vòng lặp chính, lặp đến khi Q rỗng thi dừng
    while (!empty_queue(&Q))
    {
        int u = front(&Q); // lấy đỉnh đầu tiên trong Q => gọi nó là đỉnh u
        dequeue_queue(&Q);
        push_back(pL, u);  // đưa u vào cuối danh sách

        // xóa đỉnh u <=> giảm bậc vào của các đỉnh kề v của u
        for (int v = 1; v <= pG->n; v++)
        {
            if (pG->A[u][v] != 0)
            {
                d[v]--;
                if (d[v] == 0)
                    enqueue(&Q, v);
            }
        }
    }
}

int main() {
    Graph G;
    int n, m, u, v, e;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    List L;
    topo_sort(&G, &L);
    for (int i = 1; i <= L.size; i++)
        printf("%d ", element_at(&L, i));

    return 0;
}
```

	Input	Expected	Got	
✔	3 2 1 3 3 2	1 3 2	1 3 2	✔
✔	7 10 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7 6 7	1 2 4 6 5 3 7	1 2 4 6 5 3 7	✔

	Input	Expected	Got	
✔	7 12 1 2 1 3 2 4 2 5 2 6 3 2 3 5 3 6 4 7 5 7 6 4 6 5	1 3 2 6 4 5 7	1 3 2 6 4 5 7	✔
✔	7 10 3 5 4 7 7 1 6 2 4 1 3 6 5 1 2 1 6 4 3 4	3 5 6 2 4 7 1	3 5 6 2 4 7 1	✔
✔	9 12 1 8 6 5 8 9 3 8 7 5 2 8 7 6 4 8 8 5 5 9 2 5 1 5	1 2 3 4 7 8 6 5 9	1 2 3 4 7 8 6 5 9	✔
✔	8 11 3 8 5 7 4 8 6 5 6 4 4 5 1 5 6 3 6 7 1 2 5 8	1 6 2 3 4 5 7 8	1 6 2 3 4 5 7 8	✔

Passed all tests! ✔

Question author's solution (C):

```c
#include <stdio.h>

#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} Graph;


void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
        for (int v = 1; v <= n; v++)
            pG->A[u][v] = 0;
}

void add_edge(Graph *pG, int u, int v) {
    pG->A[u][v] += 1;
}


```

Correct

Marks for this submission: 1.00/1.00.

◄ Tự học - Quản lý dự án (ngẫu nhiên)

Jump to...

001b. Thứ tự topo (chiều rộng hoặc chiều sâu) ►