

[Dashboard](#) / [My courses](#) / [Graph Theory-HK3-0405](#) / [Tuần 8 - Thứ tự topo & Ứng dụng](#) / [001. Thứ tự topo \(chiều rộng\)](#)

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, ... %)

```

1 #include <stdio.h>
2 #define MAX_N 100
3
4 // _____ LIST _____
5 typedef struct
6 {
7     int data[MAX_N]; // mảng bao gồm các phần tử của danh sách
8     int size;        // độ dài của danh sách
9 } List;
10
11 void make_null_list(List *pL)
12 {
13     pL->size = 0;
14     // (*L).size=0;
15 }
16
17 // Thêm một phần tử mới vào cuối ds
18 void push_back(List *pL, int x)
19 {
20
21     // pL->data[pL->size] = x;
22     // pL->size++;
23     pL->data[pL->size++] = x;
24 }
25
26
27 // Trả về phần tử ở vị trí p
28 int element_at(List *pL, int i)

```

```

29 {
30     return pL->data[i-1];
31 }
32
33 void copy_list(List *pS1, List *pS2) {
34     make_null_list(pS1);
35     for (int i = 1; i <= pS2->size; i++)
36         push_back(pS1, element_at(pS2, i));
37 }
38
39 // _____QUEUE_____
40
41 typedef struct
42 {
43     int data[MAX_N];
44     int front, rear;
45 } Queue;
46
47 // khởi tạo hàng đợi
48 void make_null_queue(Queue *pQ)
49 {
50     pQ->front = 0;
51     pQ->rear = -1;
52 }
53
54 // đưa ptu vào cuối hàng đợi
55 void enqueue(Queue *pQ, int u)
56 {
57     pQ->rear++;
58     pQ->data[pQ->rear] = u;
59 }
60
61 int front(Queue *pQ)
62 {
63     return pQ->data[pQ->front];
64 }
65
66 void dequeue_queue(Queue *pQ)
67 {
68     pQ->front++;
69 }
70
71 int empty_queue(Queue *pQ)
72 {
73     return pQ->front > pQ->rear;
74 }
75
76 // _____Graph_____
77
78 typedef struct
79 {
80     int n, m;
81     int A[MAX_N][MAX_N];
82 } Graph;
83
84 void init_graph(Graph *pG, int n)
85 {
86     pG->n = n;
87     pG->m = 0;
88     for (int u = 1; u <= n; u++)
89         for (int v = 1; v <= n; v++)
90             pG->A[u][v] = 0;
91 }
92
93 void add_edge(Graph *pG, int u, int v)
94 {
95     pG->A[u][v] += 1;
96     // pG->A[v][u] = 1;
97     pG->m++;
98 }
99
100 // _____TOPO_____
101
102 void topo_sort(Graph *pG, List *pL)
103 {
104     int d[MAX_N]; // lưu bậc vào của đỉnh u
105
106     // tính bậc vào của đỉnh u
107     for (int u = 1; u <= pG->n; u++)
108     {
109         d[u] = 0;
110         for (int x = 1; x <= pG->n; x++)
111             if (pG->A[x][u] != 0) // bậc vào của u
112                 d[u]++;
113     }
114
115     Queue Q;

```

```

116 // làm rỗng hàng đợi
117 make_null_queue(&Q);
118
119 // đưa các đỉnh có d[u] = 0 vào hàng đợi
120 for (int u = 1; u <= pG->n; u++)
121 {
122     if (d[u] == 0)
123         enqueue(&Q, u);
124 }
125
126 // làm rỗng danh sách
127 make_null_list(pL);
128
129 // vòng lặp chính, lặp đến khi Q rỗng thì dừng
130 while (!empty_queue(&Q))
131 {
132     int u = front(&Q); // lấy đỉnh đầu tiên trong Q => gọi nó là đỉnh u
133     dequeue_queue(&Q);
134     push_back(pL, u); // đưa u vào cuối danh sách
135
136     // xóa đỉnh u <=> giảm bậc vào của các đỉnh kề v của u
137     for (int v = 1; v <= pG->n; v++)
138     {
139         if (pG->A[u][v] != 0)
140         {
141             d[v]--;
142             if (d[v] == 0)
143                 enqueue(&Q, v);
144         }
145     }
146 }
147 }
148
149 int main() {
150     Graph G;
151     int n, m, u, v, e;
152     scanf("%d%d", &n, &m);
153     init_graph(&G, n);
154
155     for (e = 0; e < m; e++) {
156         scanf("%d%d", &u, &v);
157         add_edge(&G, u, v);
158     }
159
160     List L;
161     topo_sort(&G, &L);
162     for (int i = 1; i <= L.size; i++)
163         printf("%d ", element_at(&L, i));
164
165     return 0;
166 }
167
168

```

	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):

```

1  #include <stdio.h>
2
3  #define MAX_N 100
4  typedef struct {
5      int n, m;
6      int A[MAX_N][MAX_N];
7  } Graph;
8
9
10 void init_graph(Graph *pG, int n) {
11     pG->n = n;
12     pG->m = 0;
13     for (int u = 1; u <= n; u++)
14         for (int v = 1; v <= n; v++)
15             pG->A[u][v] = 0;
16 }
17
18 void add_edge(Graph *pG, int u, int v) {
19     pG->A[u][v] += 1;
20 }
21
22

```



Correct

Marks for this submission: 1.00/1.00.

[◀ Tự học - Quản lý dự án \(ngẫu nhiên\)](#)

Jump to...

[001b. Thứ tự topo \(chiều rộng hoặc chiều sâu\) ▶](#)