

[Dashboard](#) / [My courses](#) / [Graph Theory-HK3-0405](#) / [Tuần 6 - 7 - Đường đi ngắn nhất trên đồ thị](#)
/ [Bài tập 9 - Thuật toán Floyd - Warshall \(đường đi ngắn nhất giữa các cặp đỉnh\)](#)

Started on	Tuesday, 24 June 2025, 10:32 PM
State	Finished
Completed on	Tuesday, 24 June 2025, 10:32 PM
Time taken	26 secs
Marks	2.00/2.00
Grade	10.00 out of 10.00 (100%)

Question 1

Correct

Mark 1.00 out of 1.00

Viết chương trình đọc vào một **đơn đồ thị có hướng, có trọng số**. Áp dụng thuật toán [Floyd - Warshall](#) để tìm đường đi ngắn nhất giữa các cặp đỉnh.

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 ($0 < n < 100$; $0 < m < 500$).
- m dòng tiếp theo mỗi dòng chứa 3 số nguyên u, v, w mô tả cung (u, v) có trọng số w ($-100 < w < 100$).

Dữ liệu được đảm bảo không tồn tại chu trình âm.

Đầu ra (Output)

- In ra màn hình chiều dài đường đi ngắn nhất giữa các cặp đỉnh theo mẫu bên dưới. Nếu không có đường đi in ra **NO PATH**.

u -> v: chiều dài

- Liệt kê các cặp theo thứ tự tăng dần của u và v
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
3 3	1 -> 1: 0
1 2 9	1 -> 2: 9
2 3 4	1 -> 3: 4
1 3 4	2 -> 1: NO PATH
	2 -> 2: 0
	2 -> 3: 4
	3 -> 1: NO PATH
	3 -> 2: NO PATH
	3 -> 3: 0
3 4	1 -> 1: 0
1 2 9	1 -> 2: 9
2 3 4	1 -> 3: 4
1 3 4	2 -> 1: -3
2 1 -3	2 -> 2: 0
	2 -> 3: 1
	3 -> 1: NO PATH
	3 -> 2: NO PATH
	3 -> 3: 0

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

```

1 //Viết chương trình đọc vào một đơn đồ thị có hướng, có trọng số, Không có chu trình âm
2 //Áp dụng thuật toán Floyd - Warshall để tìm đường đi ngắn nhất giữa các cặp đỉnh.
3
4 #include <stdio.h>
5 #define MAX_N 105
6 #define INF 100000000
7 #define NO_EDGE -1
8
9
10 int pi[MAX_N][MAX_N]; //đường đi ngắn nhất u -> v
11 int next[MAX_N][MAX_N]; //đỉnh kế tiếp đỉnh u trên đường đi ngắn nhất từ u -> v
12 int path[MAX_N];
13
14 typedef struct{
15     int n,m;
16     int W[MAX_N][MAX_N];
17 }Graph;
18
19 //khởi tạo ma trận k/c
20 void init_graph (Graph *pG, int n){
21     pG->n = n;
22     pG->m = 0;
23     for (int u = 1; u <= n; u++){
24         for (int v = 1; v <= n; v++){
25             pG->W[u][v] = NO_EDGE;
26         }
27     }
28 }
29
30 void add_edge (Graph *pG, int u, int v, int w){
31     pG->W[u][v] = w;

```



```

32     pG->m++;
33 }
34
35 void FloydWarshall (Graph *pG){
36     // trong đoạn đường từ i -> j thì có 1 đỉnh trung gian ở giữa là k
37     int u, v, k;
38     for (u = 1; u <= pG->n; u++){
39         for (v = 1; v <= pG->n; v++){
40             pi[u][v] = INF;
41             next[u][v] = -1;
42         }
43     }
44
45     for (u = 1; u <= pG->n; u++){
46         pi[u][u] = 0;
47         next[u][u] = u;
48     }
49
50     for (u = 1; u <= pG->n; u++){
51         for (v = 1; v <= pG->n; v++){
52             if (pG->W[u][v] != NO_EDGE){
53                 pi[u][v] = pG->W[u][v]; //đi trực tiếp từ u -> v
54                 next[u][v] = v;
55             }
56         }
57     }
58
59     for (k = 1; k <= pG->n; k++){
60         for (u = 1; u <= pG->n; u++){
61             for (v = 1; v <= pG->n; v++){
62                 if (pi[u][k] + pi[k][v] < pi[u][v]){
63                     pi[u][v] = pi[u][k] + pi[k][v];
64                     next[u][v] = next[u][k];
65                 }
66             }
67         }
68     }
69 }
70
71 void print(Graph *pG) {
72     for (int u = 1; u <= pG->n; u++) {
73         for (int v = 1; v <= pG->n; v++) {
74             printf("%d -> %d: ", u, v);
75             if (next[u][v] == -1){
76                 printf("NO PATH\n");
77             }
78             else{
79                 printf("%d\n", pi[u][v]);
80             }
81         }
82     }
83 }
84
85 int main (){
86     int n,m,u,v,w;
87     Graph G;
88     scanf ("%d%d", &n, &m);
89     init_graph(&G, n);
90     G.m = m;
91
92     for (int e = 0; e < m; e++) {
93         scanf("%d%d%d", &u, &v, &w);
94         add_edge(&G, u, v, w);
95     }
96
97     FloydWarshall(&G);
98     print(&G);
99
100    return 0;
101 }
102
103
104
105
106

```

	Input	Expected	Got	
✓	3 3 1 2 9 2 3 4 1 3 4	1 -> 1: 0 1 -> 2: 9 1 -> 3: 4 2 -> 1: NO PATH 2 -> 2: 0 2 -> 3: 4 3 -> 1: NO PATH 3 -> 2: NO PATH 3 -> 3: 0	1 -> 1: 0 1 -> 2: 9 1 -> 3: 4 2 -> 1: NO PATH 2 -> 2: 0 2 -> 3: 4 3 -> 1: NO PATH 3 -> 2: NO PATH 3 -> 3: 0	✓
✓	3 4 1 2 9 2 3 4 1 3 4 2 1 -3	1 -> 1: 0 1 -> 2: 9 1 -> 3: 4 2 -> 1: -3 2 -> 2: 0 2 -> 3: 1 3 -> 1: NO PATH 3 -> 2: NO PATH 3 -> 3: 0	1 -> 1: 0 1 -> 2: 9 1 -> 3: 4 2 -> 1: -3 2 -> 2: 0 2 -> 3: 1 3 -> 1: NO PATH 3 -> 2: NO PATH 3 -> 3: 0	✓
✓	6 9 1 2 7 1 3 9 1 5 14 2 3 10 2 4 15 3 4 11 3 5 2 4 6 6 5 6 9	1 -> 1: 0 1 -> 2: 7 1 -> 3: 9 1 -> 4: 20 1 -> 5: 11 1 -> 6: 20 2 -> 1: NO PATH 2 -> 2: 0 2 -> 3: 10 2 -> 4: 15 2 -> 5: 12 2 -> 6: 21 3 -> 1: NO PATH 3 -> 2: NO PATH 3 -> 3: 0 3 -> 4: 11 3 -> 5: 2 3 -> 6: 11 4 -> 1: NO PATH 4 -> 2: NO PATH 4 -> 3: NO PATH 4 -> 4: 0 4 -> 5: NO PATH 4 -> 6: 6 5 -> 1: NO PATH 5 -> 2: NO PATH 5 -> 3: NO PATH 5 -> 4: NO PATH 5 -> 5: 0 5 -> 6: 9 6 -> 1: NO PATH 6 -> 2: NO PATH 6 -> 3: NO PATH 6 -> 4: NO PATH 6 -> 5: NO PATH 6 -> 6: 0	1 -> 1: 0 1 -> 2: 7 1 -> 3: 9 1 -> 4: 20 1 -> 5: 11 1 -> 6: 20 2 -> 1: NO PATH 2 -> 2: 0 2 -> 3: 10 2 -> 4: 15 2 -> 5: 12 2 -> 6: 21 3 -> 1: NO PATH 3 -> 2: NO PATH 3 -> 3: 0 3 -> 4: 11 3 -> 5: 2 3 -> 6: 11 4 -> 1: NO PATH 4 -> 2: NO PATH 4 -> 3: NO PATH 4 -> 4: 0 4 -> 5: NO PATH 4 -> 6: 6 5 -> 1: NO PATH 5 -> 2: NO PATH 5 -> 3: NO PATH 5 -> 4: NO PATH 5 -> 5: 0 5 -> 6: 9 6 -> 1: NO PATH 6 -> 2: NO PATH 6 -> 3: NO PATH 6 -> 4: NO PATH 6 -> 5: NO PATH 6 -> 6: 0	✓

	Input	Expected	Got	
✓	8 13	1 -> 1: 0	1 -> 1: 0	✓
	1 2 4	1 -> 2: 4	1 -> 2: 4	
	1 3 4	1 -> 3: 4	1 -> 3: 4	
	3 5 4	1 -> 4: 4	1 -> 4: 4	
	3 6 2	1 -> 5: 3	1 -> 5: 3	
	4 1 3	1 -> 6: 6	1 -> 6: 6	
	4 3 2	1 -> 7: 8	1 -> 7: 8	
	5 4 1	1 -> 8: 10	1 -> 8: 10	
	5 7 5	2 -> 1: NO PATH	2 -> 1: NO PATH	
	6 2 3	2 -> 2: 0	2 -> 2: 0	
	6 5 -3	2 -> 3: NO PATH	2 -> 3: NO PATH	
	7 6 2	2 -> 4: NO PATH	2 -> 4: NO PATH	
	7 8 2	2 -> 5: NO PATH	2 -> 5: NO PATH	
	8 5 -2	2 -> 6: NO PATH	2 -> 6: NO PATH	
		2 -> 7: NO PATH	2 -> 7: NO PATH	
		2 -> 8: NO PATH	2 -> 8: NO PATH	
		3 -> 1: 3	3 -> 1: 3	
		3 -> 2: 5	3 -> 2: 5	
		3 -> 3: 0	3 -> 3: 0	
		3 -> 4: 0	3 -> 4: 0	
		3 -> 5: -1	3 -> 5: -1	
		3 -> 6: 2	3 -> 6: 2	
		3 -> 7: 4	3 -> 7: 4	
		3 -> 8: 6	3 -> 8: 6	
		4 -> 1: 3	4 -> 1: 3	
		4 -> 2: 7	4 -> 2: 7	
		4 -> 3: 2	4 -> 3: 2	
		4 -> 4: 0	4 -> 4: 0	
		4 -> 5: 1	4 -> 5: 1	
		4 -> 6: 4	4 -> 6: 4	
		4 -> 7: 6	4 -> 7: 6	
		4 -> 8: 8	4 -> 8: 8	
		5 -> 1: 4	5 -> 1: 4	
		5 -> 2: 8	5 -> 2: 8	
		5 -> 3: 3	5 -> 3: 3	
		5 -> 4: 1	5 -> 4: 1	
		5 -> 5: 0	5 -> 5: 0	
		5 -> 6: 5	5 -> 6: 5	
		5 -> 7: 5	5 -> 7: 5	
		5 -> 8: 7	5 -> 8: 7	
		6 -> 1: 1	6 -> 1: 1	
		6 -> 2: 3	6 -> 2: 3	
		6 -> 3: 0	6 -> 3: 0	
		6 -> 4: -2	6 -> 4: -2	
		6 -> 5: -3	6 -> 5: -3	
		6 -> 6: 0	6 -> 6: 0	
		6 -> 7: 2	6 -> 7: 2	
		6 -> 8: 4	6 -> 8: 4	
		7 -> 1: 3	7 -> 1: 3	
		7 -> 2: 5	7 -> 2: 5	
		7 -> 3: 2	7 -> 3: 2	
		7 -> 4: 0	7 -> 4: 0	
		7 -> 5: -1	7 -> 5: -1	
		7 -> 6: 2	7 -> 6: 2	
		7 -> 7: 0	7 -> 7: 0	
		7 -> 8: 2	7 -> 8: 2	
		8 -> 1: 2	8 -> 1: 2	
		8 -> 2: 6	8 -> 2: 6	
		8 -> 3: 1	8 -> 3: 1	
		8 -> 4: -1	8 -> 4: -1	
		8 -> 5: -2	8 -> 5: -2	
		8 -> 6: 3	8 -> 6: 3	
		8 -> 7: 3	8 -> 7: 3	
		8 -> 8: 0	8 -> 8: 0	

Passed all tests! ✓

Question author's solution (C):

```

1 #include <stdio.h>
2
3 #define MAXM 500
4 #define MAXN 100
5 #define oo 999999
6 #define NO_EDGE -999999
7
8 typedef struct {
9     int n, m;
10     int L[MAXN][MAXN];

```



```
10     int w[NO_EDGE][NO_EDGE];
11 } Graph;
12
13 void init_graph(Graph *pG, int n) {
14     pG->n = n;
15     pG->m = 0;
16     for (int u = 1; u <= n; u++)
17         for (int v = 1; v <= n; v++)
18             pG->W[u][v] = NO_EDGE;
19 }
20
21 void add_edge(Graph *pG, int u, int v, int w) {
22     pG->W[u][v] = w;
```

Correct

Marks for this submission: 1.00/1.00.

Question 2

Correct

Mark 1.00 out of 1.00

Viết chương trình đọc vào một **đơn đồ thị có hướng, có trọng số**. Áp dụng thuật toán [Floyd - Warshall](#) để tìm đường đi ngắn nhất giữa các cặp đỉnh.

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 ($0 < n < 100$; $0 < m < 500$).
- m dòng tiếp theo mỗi dòng chứa 3 số nguyên u, v, w mô tả cung (u, v) có trọng số w ($-100 < w < 100$).

Dữ liệu được đảm bảo không tồn tại chu trình âm.

Đầu ra (Output)

- In ra màn hình đường đi từ ngắn nhất giữa các cặp đỉnh theo mẫu bên dưới. Nếu không có đường đi in ra **NO PATH**.

```
path(u, v): u -> x1 -> ... -> v
```

- Liệt kê các cặp theo thứ tự tăng dần của u và v
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
3 3 1 2 9 2 3 4 1 3 4	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(2, 1): NO PATH path(2, 2): 2 path(2, 3): 2 -> 3 path(3, 1): NO PATH path(3, 2): NO PATH path(3, 3): 3
3 4 1 2 9 2 3 4 1 3 4 2 1 -3	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(2, 1): 2 -> 1 path(2, 2): 2 path(2, 3): 2 -> 1 -> 3 path(3, 1): NO PATH path(3, 2): NO PATH path(3, 3): 3

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

```

1 #include <stdio.h>
2
3 #define MAXM 500
4 #define MAXN 100
5 #define oo 999999
6 #define NO_EDGE -999999
7
8 typedef struct {
9     int n, m;
10    int W[MAXN][MAXN];
11 } Graph;
12
13 void init_graph(Graph *pG, int n) {
14     pG->n = n;
15     pG->m = 0;
16     for (int u = 1; u <= n; u++)
17         for (int v = 1; v <= n; v++)
18             pG->W[u][v] = NO_EDGE;
19 }
20
21 void add_edge(Graph *pG, int u, int v, int w) {
22     pG->W[u][v] = w;
23     pG->m++;
24 }
25
26
27 #define oo 999999
28 int pi[MAXN][MAXN];
29 int next[MAXN][MAXN];
30 int FloydWarshall(Graph *pG) {
31     int u, v, k;
```

```

32     for (u = 1; u <= pG->n; u++)
33     for (v = 1; v <= pG->n; v++) {
34         pi[u][v] = oo;
35         next[u][v] = -1;
36     }
37
38     for (u = 1; u <= pG->n; u++)
39         pi[u][u] = 0;
40
41     for (u = 1; u <= pG->n; u++)
42     for (v = 1; v <= pG->n; v++)
43     if (pG->W[u][v] != NO_EDGE) {
44         pi[u][v] = pG->W[u][v]; //đi trực tiếp từ u -> v
45         next[u][v] = v;
46     }
47
48     for (k = 1; k <= pG->n; k++)
49     for (u = 1; u <= pG->n; u++) {
50         if (pi[u][k] == oo)
51             continue;
52         for (v = 1; v <= pG->n; v++) {
53             if (pi[k][v] == oo)
54                 continue;
55
56             if (pi[u][k] + pi[k][v] < pi[u][v]) {
57                 pi[u][v] = pi[u][k] + pi[k][v];
58                 next[u][v] = next[u][k];
59             }
60         }
61     }
62
63     //Kiểm tra chu trình âm (nếu cần thiết)
64     return 0; //no negative cycle
65 }
66
67
68 int main() {
69     Graph G;
70     int n, m;
71     scanf("%d%d", &n, &m);
72     init_graph(&G, n);
73
74     for (int e = 0; e < m; e++) {
75         int u, v, w;
76         scanf("%d%d%d", &u, &v, &w);
77         add_edge(&G, u, v, w);
78     }
79
80     FloydWarshall(&G);
81
82     for (int u = 1; u <= G.n; u++)
83     for (int v = 1; v <= G.n; v++)
84     if (pi[u][v] < oo) {
85         printf("path(%d, %d): %d", u, v, u);
86         int current = u;
87         while (current != v) {
88             current = next[current][v];
89             printf(" -> %d", current);
90         }
91         printf("\n");
92     } else
93         printf("path(%d, %d): NO PATH\n", u, v);
94
95     return 0;
96 }
97 }

```

	Input	Expected	Got	
✓	3 3	path(1, 1): 1	path(1, 1): 1	✓
	1 2 9	path(1, 2): 1 -> 2	path(1, 2): 1 -> 2	
	2 3 4	path(1, 3): 1 -> 3	path(1, 3): 1 -> 3	
	1 3 4	path(2, 1): NO PATH	path(2, 1): NO PATH	
		path(2, 2): 2	path(2, 2): 2	
		path(2, 3): 2 -> 3	path(2, 3): 2 -> 3	
		path(3, 1): NO PATH	path(3, 1): NO PATH	
		path(3, 2): NO PATH	path(3, 2): NO PATH	
		path(3, 3): 3	path(3, 3): 3	

	Input	Expected	Got	
✓	3 4 1 2 9 2 3 4 1 3 4 2 1 -3	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(2, 1): 2 -> 1 path(2, 2): 2 path(2, 3): 2 -> 1 -> 3 path(3, 1): NO PATH path(3, 2): NO PATH path(3, 3): 3	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(2, 1): 2 -> 1 path(2, 2): 2 path(2, 3): 2 -> 1 -> 3 path(3, 1): NO PATH path(3, 2): NO PATH path(3, 3): 3	✓
✓	6 9 1 2 7 1 3 9 1 5 14 2 3 10 2 4 15 3 4 11 3 5 2 4 6 6 5 6 9	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(1, 4): 1 -> 3 -> 4 path(1, 5): 1 -> 3 -> 5 path(1, 6): 1 -> 3 -> 5 -> 6 path(2, 1): NO PATH path(2, 2): 2 path(2, 3): 2 -> 3 path(2, 4): 2 -> 4 path(2, 5): 2 -> 3 -> 5 path(2, 6): 2 -> 4 -> 6 path(3, 1): NO PATH path(3, 2): NO PATH path(3, 3): 3 path(3, 4): 3 -> 4 path(3, 5): 3 -> 5 path(3, 6): 3 -> 5 -> 6 path(4, 1): NO PATH path(4, 2): NO PATH path(4, 3): NO PATH path(4, 4): 4 path(4, 5): NO PATH path(4, 6): 4 -> 6 path(5, 1): NO PATH path(5, 2): NO PATH path(5, 3): NO PATH path(5, 4): NO PATH path(5, 5): 5 path(5, 6): 5 -> 6 path(6, 1): NO PATH path(6, 2): NO PATH path(6, 3): NO PATH path(6, 4): NO PATH path(6, 5): NO PATH path(6, 6): 6	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(1, 4): 1 -> 3 -> 4 path(1, 5): 1 -> 3 -> 5 path(1, 6): 1 -> 3 -> 5 -> 6 path(2, 1): NO PATH path(2, 2): 2 path(2, 3): 2 -> 3 path(2, 4): 2 -> 4 path(2, 5): 2 -> 3 -> 5 path(2, 6): 2 -> 4 -> 6 path(3, 1): NO PATH path(3, 2): NO PATH path(3, 3): 3 path(3, 4): 3 -> 4 path(3, 5): 3 -> 5 path(3, 6): 3 -> 5 -> 6 path(4, 1): NO PATH path(4, 2): NO PATH path(4, 3): NO PATH path(4, 4): 4 path(4, 5): NO PATH path(4, 6): 4 -> 6 path(5, 1): NO PATH path(5, 2): NO PATH path(5, 3): NO PATH path(5, 4): NO PATH path(5, 5): 5 path(5, 6): 5 -> 6 path(6, 1): NO PATH path(6, 2): NO PATH path(6, 3): NO PATH path(6, 4): NO PATH path(6, 5): NO PATH path(6, 6): 6	✓

	Input	Expected	Got	
✓	8 13 1 2 4 1 3 4 3 5 4 3 6 2 4 1 3 4 3 2 5 4 1 5 7 5 6 2 3 6 5 -3 7 6 2 7 8 2 8 5 -2	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(1, 4): 1 -> 3 -> 6 -> 5 -> 4 path(1, 5): 1 -> 3 -> 6 -> 5 path(1, 6): 1 -> 3 -> 6 path(1, 7): 1 -> 3 -> 6 -> 5 -> 7 path(1, 8): 1 -> 3 -> 6 -> 5 -> 7 -> 8 path(2, 1): NO PATH path(2, 2): 2 path(2, 3): NO PATH path(2, 4): NO PATH path(2, 5): NO PATH path(2, 6): NO PATH path(2, 7): NO PATH path(2, 8): NO PATH path(3, 1): 3 -> 6 -> 5 -> 4 -> 1 path(3, 2): 3 -> 6 -> 2 path(3, 3): 3 path(3, 4): 3 -> 6 -> 5 -> 4 path(3, 5): 3 -> 6 -> 5 path(3, 6): 3 -> 6 path(3, 7): 3 -> 6 -> 5 -> 7 path(3, 8): 3 -> 6 -> 5 -> 7 -> 8 path(4, 1): 4 -> 1 path(4, 2): 4 -> 1 -> 2 path(4, 3): 4 -> 3 path(4, 4): 4 path(4, 5): 4 -> 3 -> 6 -> 5 path(4, 6): 4 -> 3 -> 6 path(4, 7): 4 -> 3 -> 6 -> 5 -> 7 path(4, 8): 4 -> 3 -> 6 -> 5 -> 7 -> 8 path(5, 1): 5 -> 4 -> 1 path(5, 2): 5 -> 4 -> 1 -> 2 path(5, 3): 5 -> 4 -> 3 path(5, 4): 5 -> 4 path(5, 5): 5 path(5, 6): 5 -> 4 -> 3 -> 6 path(5, 7): 5 -> 7 path(5, 8): 5 -> 7 -> 8 path(6, 1): 6 -> 5 -> 4 -> 1 path(6, 2): 6 -> 2 path(6, 3): 6 -> 5 -> 4 -> 3 path(6, 4): 6 -> 5 -> 4 path(6, 5): 6 -> 5 path(6, 6): 6 path(6, 7): 6 -> 5 -> 7 path(6, 8): 6 -> 5 -> 7 -> 8 path(7, 1): 7 -> 6 -> 5 -> 4 -> 1 path(7, 2): 7 -> 6 -> 2 path(7, 3): 7 -> 6 -> 5 -> 4 -> 3 path(7, 4): 7 -> 6 -> 5 -> 4 path(7, 5): 7 -> 6 -> 5 path(7, 6): 7 -> 6 path(7, 7): 7 path(7, 8): 7 -> 8 path(8, 1): 8 -> 5 -> 4 -> 1 path(8, 2): 8 -> 5 -> 4 -> 1 -> 2 path(8, 3): 8 -> 5 -> 4 -> 3 path(8, 4): 8 -> 5 -> 4 path(8, 5): 8 -> 5 path(8, 6): 8 -> 5 -> 4 -> 3 -> 6 path(8, 7): 8 -> 5 -> 7 path(8, 8): 8	path(1, 1): 1 path(1, 2): 1 -> 2 path(1, 3): 1 -> 3 path(1, 4): 1 -> 3 -> 6 -> 5 -> 4 path(1, 5): 1 -> 3 -> 6 -> 5 path(1, 6): 1 -> 3 -> 6 path(1, 7): 1 -> 3 -> 6 -> 5 -> 7 path(1, 8): 1 -> 3 -> 6 -> 5 -> 7 -> 8 path(2, 1): NO PATH path(2, 2): 2 path(2, 3): NO PATH path(2, 4): NO PATH path(2, 5): NO PATH path(2, 6): NO PATH path(2, 7): NO PATH path(2, 8): NO PATH path(3, 1): 3 -> 6 -> 5 -> 4 -> 1 path(3, 2): 3 -> 6 -> 2 path(3, 3): 3 path(3, 4): 3 -> 6 -> 5 -> 4 path(3, 5): 3 -> 6 -> 5 path(3, 6): 3 -> 6 path(3, 7): 3 -> 6 -> 5 -> 7 path(3, 8): 3 -> 6 -> 5 -> 7 -> 8 path(4, 1): 4 -> 1 path(4, 2): 4 -> 1 -> 2 path(4, 3): 4 -> 3 path(4, 4): 4 path(4, 5): 4 -> 3 -> 6 -> 5 path(4, 6): 4 -> 3 -> 6 path(4, 7): 4 -> 3 -> 6 -> 5 -> 7 path(4, 8): 4 -> 3 -> 6 -> 5 -> 7 -> 8 path(5, 1): 5 -> 4 -> 1 path(5, 2): 5 -> 4 -> 1 -> 2 path(5, 3): 5 -> 4 -> 3 path(5, 4): 5 -> 4 path(5, 5): 5 path(5, 6): 5 -> 4 -> 3 -> 6 path(5, 7): 5 -> 7 path(5, 8): 5 -> 7 -> 8 path(6, 1): 6 -> 5 -> 4 -> 1 path(6, 2): 6 -> 2 path(6, 3): 6 -> 5 -> 4 -> 3 path(6, 4): 6 -> 5 -> 4 path(6, 5): 6 -> 5 path(6, 6): 6 path(6, 7): 6 -> 5 -> 7 path(6, 8): 6 -> 5 -> 7 -> 8 path(7, 1): 7 -> 6 -> 5 -> 4 -> 1 path(7, 2): 7 -> 6 -> 2 path(7, 3): 7 -> 6 -> 5 -> 4 -> 3 path(7, 4): 7 -> 6 -> 5 -> 4 path(7, 5): 7 -> 6 -> 5 path(7, 6): 7 -> 6 path(7, 7): 7 path(7, 8): 7 -> 8 path(8, 1): 8 -> 5 -> 4 -> 1 path(8, 2): 8 -> 5 -> 4 -> 1 -> 2 path(8, 3): 8 -> 5 -> 4 -> 3 path(8, 4): 8 -> 5 -> 4 path(8, 5): 8 -> 5 path(8, 6): 8 -> 5 -> 4 -> 3 -> 6 path(8, 7): 8 -> 5 -> 7 path(8, 8): 8	✓

Passed all tests! ✓

Question author's solution (C):

1	#include <stdio.h>
2	
3	#define MAXM 500
4	#define MAXN 100
5	#define oo 999999
6	#define NO_EDGE -999999
7	
8	typedef struct {
9	int n, m;
10	int w[MAYM][MAYM];



```
10     int w[u][v] = NO_EDGE;
11 } Graph;
12
13 void init_graph(Graph *pG, int n) {
14     pG->n = n;
15     pG->m = 0;
16     for (int u = 1; u <= n; u++)
17         for (int v = 1; v <= n; v++)
18             pG->W[u][v] = NO_EDGE;
19 }
20
21 void add_edge(Graph *pG, int u, int v, int w) {
22     pG->W[u][v] = w;
```

Correct

Marks for this submission: 1.00/1.00.

[← Bài tập 8 - Extended traffic](#)

Jump to...

[Bài tập 10 - Thuật toán Floyd - Warshall \(kiểm tra chu trình âm\) ►](#)