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 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
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

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

```c
//Viết chương trình đọc vào một đơn đồ thị có hướng, có trọng số, Không có chu trình âm
//Áp dụng thuật toán Floyd - Warshall để tìm đường đi ngắn nhất giữa các cặp đỉnh.

#include <stdio.h>
#define MAX_N 105
#define INF 100000000
#define NO_EDGE -1


int pi[MAX_N][MAX_N]; //đường đi ngắn nhất u -> v
int next[MAX_N][MAX_N];   //đỉnh kế tiếp đỉnh u trên đường đi ngắn nhất từ u -> v
int path[MAX_N];

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

//khởi tạo ma trận k/c
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->W[u][v] = NO_EDGE;
        }
    }
}

void add_edge (Graph *pG, int u, int v, int w){
    pG->W[u][v] = w;
```

```
    pG->m++;
}

void FloydWarshall (Graph *pG){
    // trong đoạn đường từ i -> j thì có 1 đỉnh trung gian ở giữa là k
    int u, v, k;
    for (u = 1; u <= pG->n; u++){
        for (v = 1; v <= pG->n; v++){
            pi[u][v] = INF;
            next[u][v] = -1;
        }
    }

    for (u = 1; u <= pG->n; u++){
        pi[u][u] = 0;
        next[u][u] = u;
    }

    for (u = 1; u <= pG->n; u++){
        for (v = 1; v <= pG->n; v++){
            if (pG->W[u][v] != NO_EDGE){
                pi[u][v] = pG->W[u][v]; //đi trực tiếp từ u -> v
                next[u][v] = v;
            }
        }
    }

    for (k = 1; k <= pG->n; k++){
        for (u = 1; u <= pG->n; u++){
            for (v = 1; v <= pG->n; v++){
                if (pi[u][k] + pi[k][v] < pi[u][v]){
                    pi[u][v] = pi[u][k] + pi[k][v];
                    next[u][v] = next[u][k];
                }
            }
        }
    }
}

void print(Graph *pG) {
    for (int u = 1; u <= pG->n; u++) {
        for (int v = 1; v <= pG->n; v++) {
            printf("%d -> %d: ", u, v);
            if (next[u][v] == -1){
                printf("NO PATH\n");
            }
            else{
                printf("%d\n", pi[u][v]);
            }
        }
    }
}

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

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

    FloydWarshall(&G);
    print(&G);

    return 0;
}
```

	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 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	1 -> 1: 0 1 -> 2: 4 1 -> 3: 4 1 -> 4: 4 1 -> 5: 3 1 -> 6: 6 1 -> 7: 8 1 -> 8: 10 2 -> 1: NO PATH 2 -> 2: 0 2 -> 3: NO PATH 2 -> 4: NO PATH 2 -> 5: NO PATH 2 -> 6: NO PATH 2 -> 7: NO PATH 2 -> 8: NO PATH 3 -> 1: 3 3 -> 2: 5 3 -> 3: 0 3 -> 4: 0 3 -> 5: -1 3 -> 6: 2 3 -> 7: 4 3 -> 8: 6 4 -> 1: 3 4 -> 2: 7 4 -> 3: 2 4 -> 4: 0 4 -> 5: 1 4 -> 6: 4 4 -> 7: 6 4 -> 8: 8 5 -> 1: 4 5 -> 2: 8 5 -> 3: 3 5 -> 4: 1 5 -> 5: 0 5 -> 6: 5 5 -> 7: 5 5 -> 8: 7 6 -> 1: 1 6 -> 2: 3 6 -> 3: 0 6 -> 4: -2 6 -> 5: -3 6 -> 6: 0 6 -> 7: 2 6 -> 8: 4 7 -> 1: 3 7 -> 2: 5 7 -> 3: 2 7 -> 4: 0 7 -> 5: -1 7 -> 6: 2 7 -> 7: 0 7 -> 8: 2 8 -> 1: 2 8 -> 2: 6 8 -> 3: 1 8 -> 4: -1 8 -> 5: -2 8 -> 6: 3 8 -> 7: 3 8 -> 8: 0	1 -> 1: 0 1 -> 2: 4 1 -> 3: 4 1 -> 4: 4 1 -> 5: 3 1 -> 6: 6 1 -> 7: 8 1 -> 8: 10 2 -> 1: NO PATH 2 -> 2: 0 2 -> 3: NO PATH 2 -> 4: NO PATH 2 -> 5: NO PATH 2 -> 6: NO PATH 2 -> 7: NO PATH 2 -> 8: NO PATH 3 -> 1: 3 3 -> 2: 5 3 -> 3: 0 3 -> 4: 0 3 -> 5: -1 3 -> 6: 2 3 -> 7: 4 3 -> 8: 6 4 -> 1: 3 4 -> 2: 7 4 -> 3: 2 4 -> 4: 0 4 -> 5: 1 4 -> 6: 4 4 -> 7: 6 4 -> 8: 8 5 -> 1: 4 5 -> 2: 8 5 -> 3: 3 5 -> 4: 1 5 -> 5: 0 5 -> 6: 5 5 -> 7: 5 5 -> 8: 7 6 -> 1: 1 6 -> 2: 3 6 -> 3: 0 6 -> 4: -2 6 -> 5: -3 6 -> 6: 0 6 -> 7: 2 6 -> 8: 4 7 -> 1: 3 7 -> 2: 5 7 -> 3: 2 7 -> 4: 0 7 -> 5: -1 7 -> 6: 2 7 -> 7: 0 7 -> 8: 2 8 -> 1: 2 8 -> 2: 6 8 -> 3: 1 8 -> 4: -1 8 -> 5: -2 8 -> 6: 3 8 -> 7: 3 8 -> 8: 0	✔

Passed all tests! ✔

Question author's solution (C):

```c
#include <stdio.h>

#define MAXM 500
#define MAXN 100
#define oo 999999
#define NO_EDGE -999999

typedef struct {
    int n, m;
    int W[MAXN][MAXN];
```

```
    int W[MAXN][MAXN];
} 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->W[u][v] = NO_EDGE;
}

void add_edge(Graph *pG, int u, int v, int w) {
    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, ... %)

```c
#include <stdio.h>

#define MAXM 500
#define MAXN 100
#define oo 999999
#define NO_EDGE -999999

typedef struct {
    int n, m;
    int W[MAXN][MAXN];
} 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->W[u][v] = NO_EDGE;
}

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


#define oo 999999
int pi[MAXN][MAXN];
int next[MAXN][MAXN];
int FloydWarshall(Graph *pG) {
    int u, v, k;
```

```
    for (u = 1; u <= pG->n; u++)
        for (v = 1; v <= pG->n; v++) {
            pi[u][v] = oo;
            next[u][v] = -1;
        }

    for (u = 1; u <= pG->n; u++)
        pi[u][u] = 0;

    for (u = 1; u <= pG->n; u++)
        for (v = 1; v <= pG->n; v++)
            if (pG->W[u][v] != NO_EDGE) {
                pi[u][v] = pG->W[u][v]; //đi trực tiếp từ u -> v
                next[u][v] = v;
            }

    for (k = 1; k <= pG->n; k++)
        for (u = 1; u <= pG->n; u++) {
            if (pi[u][k] == oo)
                continue;
            for (v = 1; v <= pG->n; v++) {
                if (pi[k][v] == oo)
                    continue;

                if (pi[u][k] + pi[k][v] < pi[u][v]) {
                    pi[u][v] = pi[u][k] + pi[k][v];
                    next[u][v] = next[u][k];
                }
            }
        }

    //Kiểm tra chu trình âm (nếu cần thiết)
    return 0; //no negative cycle
}


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

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

    FloydWarshall(&G);

    for (int u = 1; u <= G.n; u++)
        for (int v = 1; v <= G.n; v++)
            if (pi[u][v] < oo) {
                printf("path(%d, %d): %d", u, v, u);
                int current = u;
                while (current != v) {
                    current = next[current][v];
                    printf(" -> %d", current);
                }
                printf("\n");
            } else
                printf("path(%d, %d): NO PATH\n", u, v);


    return 0;
}
```

	Input	Expected	Got	
✔	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	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	✔

	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):

```c
#include <stdio.h>

#define MAXM 500
#define MAXN 100
#define oo 999999
#define NO_EDGE -999999

typedef struct {
    int n, m;
    int W[MAXN][MAXN];
```

```
    int W[MAXN][MAXN];
} 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->W[u][v] = NO_EDGE;
}

void add_edge(Graph *pG, int u, int v, int w) {
    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) ►