

[Dashboard](#) / [My courses](#) / [Graph Theory-HK3-0405](#) / [Tuần 6 - 7 - Đường đi ngắn nhất trên đồ thị](#)

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

<b>Started on</b>	Tuesday, 24 June 2025, 10:41 PM
<b>State</b>	Finished
<b>Completed on</b>	Tuesday, 24 June 2025, 10:41 PM
<b>Time taken</b>	11 secs
<b>Marks</b>	1.00/1.00
<b>Grade</b>	<b>10.00</b> out of 10.00 ( <b>100%</b> )

Question 1

Correct

Mark 1.00 out of 1.00

Viết chương trình đọc vào một **đơn đồ thị có hướng, có trọng số**, kiểm tra xem nó có chứa chu trình âm hay không.

**Đầu vào (Input)**

Dữ liệu đầu vào được nhập từ dòng nhập chuẩn (bàn phím, stdin) 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 3 số nguyên  $u, v, w$  nói rằng cung  $(u, v)$  có trọng số  $w$ .

**Đầu ra (Output)**

- In ra màn hình **YES** nếu phát hiện có chu trình âm, ngược lại in ra **NO**.
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
4 4 1 2 1 2 3 -1 3 4 -1 4 1 -1	YES
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 -2 6 2 3 6 5 -3 7 6 2 7 8 2 8 5 -2 1	YES
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	NO

Answer: (penalty regime: 0 %)

```

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             for (v = 1; v <= pG->n; v++)
51                 if (pi[u][k] + pi[k][v] < pi[u][v]) {
52                     pi[u][v] = pi[u][k] + pi[k][v];
53                     next[u][v] = next[u][k];
54                 }
55
56     //Kiểm tra chu trình âm (nếu cần thiết)
57     int negative_cycle = 0;
58     for (u = 1; u <= pG->n; u++) {
59         if (pi[u][u] < 0) {
60             //Đồ thị có chứa chu trình âm
61             negative_cycle = 1;
62             break;
63         }
64     }
65     return negative_cycle;
66 }
67
68
69 int main() {
70     Graph G;
71     int n, m;
72     scanf("%d%d", &n, &m);
73     init_graph(&G, n);
74
75     for (int e = 0; e < m; e++) {
76         int u, v, w;
77         scanf("%d%d%d", &u, &v, &w);
78         add_edge(&G, u, v, w);
79     }
80
81     if (FloydWarshall(&G) == 1)
82         printf("YES\n");
83     else
84         printf("NO\n");
85
86     return 0;
87 }
88 }

```

Debug: source code from all test runs

Run 1



```

#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++)
            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];
                }

    //Kiểm tra chu trình âm (nếu cần thiết)
    int negative_cycle = 0;
    for (u = 1; u <= pG->n; u++) {
        if (pi[u][u] < 0) {
            //Đồ thị có chứa chu trình âm
            negative_cycle = 1;
            break;
        }
    }
    return 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);
    }
}

```

```
if (FloydWarshall(&G) == 1)
    printf("YES\n");
else
    printf("NO\n");

return 0;
}
```

Run 2



```

#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++)
            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];
                }

    //Kiểm tra chu trình âm (nếu cần thiết)
    int negative_cycle = 0;
    for (u = 1; u <= pG->n; u++) {
        if (pi[u][u] < 0) {
            //Đồ thị có chứa chu trình âm
            negative_cycle = 1;
            break;
        }
    }
    return 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);
    }
}

```



```
if (FloydWarshall(&G) == 1)
    printf("YES\n");
else
    printf("NO\n");

return 0;
}
```

Run 3

```

#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++)
            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];
                }

    //Kiểm tra chu trình âm (nếu cần thiết)
    int negative_cycle = 0;
    for (u = 1; u <= pG->n; u++) {
        if (pi[u][u] < 0) {
            //Đồ thị có chứa chu trình âm
            negative_cycle = 1;
            break;
        }
    }
    return 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);
    }
}

```





```
if (FloydWarshall(&G) == 1)
    printf("YES\n");
else
    printf("NO\n");

return 0;
}
```

Run 4

```

#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++)
            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];
                }

    //Kiểm tra chu trình âm (nếu cần thiết)
    int negative_cycle = 0;
    for (u = 1; u <= pG->n; u++) {
        if (pi[u][u] < 0) {
            //Đồ thị có chứa chu trình âm
            negative_cycle = 1;
            break;
        }
    }
    return 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);
    }
}

```



```
if (FloydWarshall(&G) == 1)
    printf("YES\n");
else
    printf("NO\n");

return 0;
}
```

Run 5

```

#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++)
            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];
                }

    //Kiểm tra chu trình âm (nếu cần thiết)
    int negative_cycle = 0;
    for (u = 1; u <= pG->n; u++) {
        if (pi[u][u] < 0) {
            //Đồ thị có chứa chu trình âm
            negative_cycle = 1;
            break;
        }
    }
    return 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);
    }
}

```



```

if (FloydWarshall(&G) == 1)
    printf("YES\n");
else
    printf("NO\n");

return 0;
}

```

	Input	Expected	Got	
✓	4 4 1 2 1 2 3 -1 3 4 -1 4 1 -1	YES	YES	✓
✓	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 -2 6 2 3 6 5 -3 7 6 2 7 8 2 8 5 -2 1	YES	YES	✓
✓	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	NO	NO	✓
✓	8 14 1 2 4 1 3 4 5 3 4 6 3 -2 4 1 3 4 3 2 5 4 1 5 7 -2 6 2 3 6 5 -3 7 6 2 7 8 2 5 8 -2 4 8 -4 4	YES	YES	✓

	Input	Expected	Got	
✓	8 14 1 2 4 1 3 4 5 3 4 6 3 -2 4 1 3 4 3 2 5 4 1 5 7 -2 6 2 3 6 5 -3 7 6 2 7 8 2 5 8 -2 4 8 -4 6	YES	YES	✓

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[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;

```

Correct

Marks for this submission: 1.00/1.00.

◀ Bài tập 9 - Thuật toán Floyd - Warshall (đường đi ngắn nhất giữa các cặp đỉnh)

Jump to...

Thuật toán Moore - Dijkstra ▶