Started on	Tuesday, 24 June 2025, 9:22 PM
State	Finished
Completed on	Tuesday, 24 June 2025, 9:22 PM
Time taken	19 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 đơn đồ thị có hướng, có trọng số, áp dụng thuật toán Bellman – Ford kiểm tra xem nó có chứa chu trình âm hay không *khi ta tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại.*

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.
- Dòng cuối cùng chứa đỉnh s.

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.

Chú ý

- Nếu không có đường đi từ s đến u thì không tính các chu trình âm chứa u.

For example:

Input	Result
4 4 1 2 1 2 3 -1 3 4 -1 4 1 -1 2	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 %)

```
#include <stdio.h>

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

typedef struct {
    int u, v;
    int w;
} Edge;

```

```c
typedef struct {
    int n, m;
    Edge edges[MAXM];
} Graph;

void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
}

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

    pG->m++;
}


int pi[MAXN];
int p[MAXN];

int BellmanFord(Graph *pG, int s) {
    int u, v, w, it, k;
    for (u = 1; u <= pG->n; u++) {
        pi[u] = oo;
    }
    pi[s] = 0;
    p[s] = -1; //trước đỉnh s không có đỉnh nào cả

    // lặp n-1 lần
    for (it = 1; it < pG->n; it++) {
        // Duyệt qua các cung và cập nhật (nếu thoả)
        for (k = 0; k < pG->m; k++) {
            u = pG->edges[k].u;
            v = pG->edges[k].v;
            w = pG->edges[k].w;

            if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                continue;

            if (pi[u] + w < pi[v]) {
                pi[v] = pi[u] + w;
                p[v] = u;
            }
        }
    }
    //Làm thêm 1 lần nữa để kiểm tra chu trình âm (nếu cần thiết)
    for (k = 0; k < pG->m; k++) {
        u = pG->edges[k].u;
        v = pG->edges[k].v;
        w = pG->edges[k].w;

        if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
            continue;

        if (pi[u] + w < pi[v]) {
            return 1;
        }
    }
    return 0;
}



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);
    }
    int s;
    scanf("%d", &s);

    if (BellmanFord(&G, s) == 1)
        printf("YES\n");
    else
        printf("NO\n");


    return 0;
}
```

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 u, v;
    int w;
} Edge;

typedef struct {
    int n, m;
    Edge edges[MAXM];
} Graph;

void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
}

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

    pG->m++;
}


int pi[MAXN];
int p[MAXN];

int BellmanFord(Graph *pG, int s) {
        int u, v, w, it, k;
        for (u = 1; u <= pG->n; u++) {
                pi[u] = oo;
        }
        pi[s] = 0;
        p[s] = -1; //trước đỉnh s không có đỉnh nào cả

        // lặp n-1 lần
        for (it = 1; it < pG->n; it++) {
                // Duyệt qua các cung và cập nhật (nếu thoả)
                for (k = 0; k < pG->m; k++) {
                        u = pG->edges[k].u;
                        v = pG->edges[k].v;
                        w = pG->edges[k].w;

                        if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                            continue;

                        if (pi[u] + w < pi[v]) {
                                pi[v] = pi[u] + w;
                                p[v] = u;
                        }
                }
        }
        //Làm thêm 1 lần nữa để kiểm tra chu trình âm (nếu cần thiết)
        for (k = 0; k < pG->m; k++) {
                u = pG->edges[k].u;
                v = pG->edges[k].v;
                w = pG->edges[k].w;

                if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                    continue;

                if (pi[u] + w < pi[v]) {
                        return 1;
                }
        }
        return 0;
}



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);
    }
    int s;
    scanf("%d", &s);

    if (BellmanFord(&G, s) == 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 u, v;
    int w;
} Edge;

typedef struct {
    int n, m;
    Edge edges[MAXM];
} Graph;

void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
}

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

    pG->m++;
}


int pi[MAXN];
int p[MAXN];

int BellmanFord(Graph *pG, int s) {
        int u, v, w, it, k;
        for (u = 1; u <= pG->n; u++) {
                pi[u] = oo;
        }
        pi[s] = 0;
        p[s] = -1; //trước đỉnh s không có đỉnh nào cả

        // lặp n-1 lần
        for (it = 1; it < pG->n; it++) {
                // Duyệt qua các cung và cập nhật (nếu thoả)
                for (k = 0; k < pG->m; k++) {
                        u = pG->edges[k].u;
                        v = pG->edges[k].v;
                        w = pG->edges[k].w;

                        if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                            continue;

                        if (pi[u] + w < pi[v]) {
                                pi[v] = pi[u] + w;
                                p[v] = u;
                        }
                }
        }
        //Làm thêm 1 lần nữa để kiểm tra chu trình âm (nếu cần thiết)
        for (k = 0; k < pG->m; k++) {
                u = pG->edges[k].u;
                v = pG->edges[k].v;
                w = pG->edges[k].w;

                if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                    continue;

                if (pi[u] + w < pi[v]) {
                        return 1;
                }
        }
        return 0;
}



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);
    }
    int s;
    scanf("%d", &s);

    if (BellmanFord(&G, s) == 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 u, v;
    int w;
} Edge;

typedef struct {
    int n, m;
    Edge edges[MAXM];
} Graph;

void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
}

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

    pG->m++;
}


int pi[MAXN];
int p[MAXN];

int BellmanFord(Graph *pG, int s) {
        int u, v, w, it, k;
        for (u = 1; u <= pG->n; u++) {
                pi[u] = oo;
        }
        pi[s] = 0;
        p[s] = -1; //trước đỉnh s không có đỉnh nào cả

        // lặp n-1 lần
        for (it = 1; it < pG->n; it++) {
                // Duyệt qua các cung và cập nhật (nếu thoả)
                for (k = 0; k < pG->m; k++) {
                        u = pG->edges[k].u;
                        v = pG->edges[k].v;
                        w = pG->edges[k].w;

                        if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                            continue;

                        if (pi[u] + w < pi[v]) {
                                pi[v] = pi[u] + w;
                                p[v] = u;
                        }
                }
        }
        //Làm thêm 1 lần nữa để kiểm tra chu trình âm (nếu cần thiết)
        for (k = 0; k < pG->m; k++) {
                u = pG->edges[k].u;
                v = pG->edges[k].v;
                w = pG->edges[k].w;

                if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                    continue;

                if (pi[u] + w < pi[v]) {
                        return 1;
                }
        }
        return 0;
}



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);
    }
    int s;
    scanf("%d", &s);

    if (BellmanFord(&G, s) == 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 u, v;
    int w;
} Edge;

typedef struct {
    int n, m;
    Edge edges[MAXM];
} Graph;

void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
}

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

    pG->m++;
}


int pi[MAXN];
int p[MAXN];

int BellmanFord(Graph *pG, int s) {
        int u, v, w, it, k;
        for (u = 1; u <= pG->n; u++) {
                pi[u] = oo;
        }
        pi[s] = 0;
        p[s] = -1; //trước đỉnh s không có đỉnh nào cả

        // lặp n-1 lần
        for (it = 1; it < pG->n; it++) {
                // Duyệt qua các cung và cập nhật (nếu thoả)
                for (k = 0; k < pG->m; k++) {
                        u = pG->edges[k].u;
                        v = pG->edges[k].v;
                        w = pG->edges[k].w;

                        if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                            continue;

                        if (pi[u] + w < pi[v]) {
                                pi[v] = pi[u] + w;
                                p[v] = u;
                        }
                }
        }
        //Làm thêm 1 lần nữa để kiểm tra chu trình âm (nếu cần thiết)
        for (k = 0; k < pG->m; k++) {
                u = pG->edges[k].u;
                v = pG->edges[k].v;
                w = pG->edges[k].w;

                if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                    continue;

                if (pi[u] + w < pi[v]) {
                        return 1;
                }
        }
        return 0;
}



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);
    }
    int s;
    scanf("%d", &s);

    if (BellmanFord(&G, s) == 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 u, v;
    int w;
} Edge;

typedef struct {
    int n, m;
    Edge edges[MAXM];
} Graph;

void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
}

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

    pG->m++;
}


int pi[MAXN];
int p[MAXN];

int BellmanFord(Graph *pG, int s) {
        int u, v, w, it, k;
        for (u = 1; u <= pG->n; u++) {
                pi[u] = oo;
        }
        pi[s] = 0;
        p[s] = -1; //trước đỉnh s không có đỉnh nào cả

        // lặp n-1 lần
        for (it = 1; it < pG->n; it++) {
                // Duyệt qua các cung và cập nhật (nếu thoả)
                for (k = 0; k < pG->m; k++) {
                        u = pG->edges[k].u;
                        v = pG->edges[k].v;
                        w = pG->edges[k].w;

                        if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                            continue;

                        if (pi[u] + w < pi[v]) {
                                pi[v] = pi[u] + w;
                                p[v] = u;
                        }
                }
        }
        //Làm thêm 1 lần nữa để kiểm tra chu trình âm (nếu cần thiết)
        for (k = 0; k < pG->m; k++) {
                u = pG->edges[k].u;
                v = pG->edges[k].v;
                w = pG->edges[k].w;

                if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
                    continue;

                if (pi[u] + w < pi[v]) {
                        return 1;
                }
        }
        return 0;
}



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);
    }
    int s;
    scanf("%d", &s);

    if (BellmanFord(&G, s) == 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 2	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	NO	NO	✔

	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):

```c
#include <stdio.h>

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

typedef struct {
    int u, v;
    int w;
} Edge;

typedef struct {
    int n, m;
    Edge edges[MAXM];
} Graph;

void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
}

```

Correct

Marks for this submission: 1.00/1.00.

◄ Bài tập 5* - Ô kiều (Ngưu Lang - Chức Nữ)

Jump to...

Bài tập 6 - Thuật toán Bellman - Ford ►