Started on	Saturday, 21 June 2025, 10:17 PM
State	Finished
Completed on	Saturday, 21 June 2025, 10:54 PM
Time taken	36 mins 32 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 %)

```
1
2  #include <stdio.h>
```

```
#define MAX_N 1000
#define NO_EDGE -1
#define oo 99999

int pi[MAX_N];  //dinh u(pi/p)
int p[MAX_N];

//Biểu diễn đồ thị bằng phương pháp danh sách cung
typedef struct{
    int u,v;
    int w;
}Edge;

//GRAPH//
typedef struct{
    int n,m;
    Edge edges[MAX_N];
}Graph;
```

Debug: source code from all test runs

Run 1

```c
#include <stdio.h>
#define MAX_N 1000
#define NO_EDGE -1
#define oo 99999

int pi[MAX_N];  //dinh u(pi/p)
int p[MAX_N];

//Biểu diễn đồ thị bằng phương pháp danh sách cung
typedef struct{
    int u,v;
    int w;
}Edge;

//GRAPH//
typedef struct{
        int n,m;
    Edge edges[MAX_N];
}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++;
}

//Bellman_Ford
void BellmanFord (Graph *pG, int s){
    int u, v, w;
        //khoi tao cac dinh co pi = oo va p = 0
        for (u = 1; u <=pG->n; u++){
                pi[u] = oo;
        }

        //khoi tao dinh bat dau
        pi[s] = 0;
        p[s] = -1; //truoc dinh s khong co cha

    //lặp n-1 lần
        for (int it = 1; it < pG->n; it++){
                for (int k = 0; k < pG->m; k++){
            //duyệt qua các cung và cập nhật nếu thỏa
            u = pG->edges[k].u;
            v = pG->edges[k].v;
            w = pG->edges[k].w;
            if (pi[u] == oo){ //chưa có đường đi đến u, bỏ qua cung u v
                continue;
            }

            if (pi[u] + w < pi[v]){
                pi[v] = pi[u] + w;
                p[v] = u;
            }
                }
        }

}

int main (){
```

```
        Graph G;
        int n,m,u,v,w,s;
        scanf ("%d%d", &n, &m);
    init_graph(&G, n);

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

        scanf ("%d", &s);
    BellmanFord(&G, s);

    //
    int negative_cycle = 0;
    for (int k = 0; k < G.m; k++){
        u = G.edges[k].u;
        v = G.edges[k].v;
        w = G.edges[k].w;
        if (pi[u] == oo){
            continue;
        }
        if (pi[u] + w < pi[v]){
            negative_cycle = 1;
            break;
        }
    }

    if (negative_cycle){
        printf ("YES");
    }
    else{
        printf ("NO");
    }

        return 0;
}
```

Run 2

```c
#include <stdio.h>
#define MAX_N 1000
#define NO_EDGE -1
#define oo 99999

int pi[MAX_N];  //dinh u(pi/p)
int p[MAX_N];

//Biểu diễn đồ thị bằng phương pháp danh sách cung
typedef struct{
    int u,v;
    int w;
}Edge;

//GRAPH//
typedef struct{
        int n,m;
    Edge edges[MAX_N];
}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++;
}

//Bellman_Ford
void BellmanFord (Graph *pG, int s){
    int u, v, w;
        //khoi tao cac dinh co pi = oo va p = 0
        for (u = 1; u <=pG->n; u++){
                pi[u] = oo;
        }

        //khoi tao dinh bat dau
        pi[s] = 0;
        p[s] = -1; //truoc dinh s khong co cha

    //lặp n-1 lần
        for (int it = 1; it < pG->n; it++){
                for (int k = 0; k < pG->m; k++){
            //duyệt qua các cung và cập nhật nếu thỏa
            u = pG->edges[k].u;
            v = pG->edges[k].v;
            w = pG->edges[k].w;
            if (pi[u] == oo){ //chưa có đường đi đến u, bỏ qua cung u v
                continue;
            }

            if (pi[u] + w < pi[v]){
                pi[v] = pi[u] + w;
                p[v] = u;
            }
                }
        }

}

int main (){
```

```
    Graph G;
    int n,m,u,v,w,s;
    scanf ("%d%d", &n, &m);
   init_graph(&G, n);

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

    scanf ("%d", &s);
   BellmanFord(&G, s);

   //
   int negative_cycle = 0;
   for (int k = 0; k < G.m; k++){
    u = G.edges[k].u;
    v = G.edges[k].v;
    w = G.edges[k].w;
    if (pi[u] == oo){
        continue;
    }
    if (pi[u] + w < pi[v]){
        negative_cycle = 1;
        break;
    }
   }

   if (negative_cycle){
    printf ("YES");
   }
   else{
    printf ("NO");
   }

    return 0;
}
```

Run 3

```c
#include <stdio.h>
#define MAX_N 1000
#define NO_EDGE -1
#define oo 99999

int pi[MAX_N];  //dinh u(pi/p)
int p[MAX_N];

//Biểu diễn đồ thị bằng phương pháp danh sách cung
typedef struct{
    int u,v;
    int w;
}Edge;

//GRAPH//
typedef struct{
        int n,m;
    Edge edges[MAX_N];
}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++;
}

//Bellman_Ford
void BellmanFord (Graph *pG, int s){
    int u, v, w;
        //khoi tao cac dinh co pi = oo va p = 0
        for (u = 1; u <=pG->n; u++){
                pi[u] = oo;
        }

        //khoi tao dinh bat dau
        pi[s] = 0;
        p[s] = -1; //truoc dinh s khong co cha

    //lặp n-1 lần
        for (int it = 1; it < pG->n; it++){
                for (int k = 0; k < pG->m; k++){
            //duyệt qua các cung và cập nhật nếu thỏa
            u = pG->edges[k].u;
            v = pG->edges[k].v;
            w = pG->edges[k].w;
            if (pi[u] == oo){ //chưa có đường đi đến u, bỏ qua cung u v
                continue;
            }

            if (pi[u] + w < pi[v]){
                pi[v] = pi[u] + w;
                p[v] = u;
            }
                }
        }

}

int main (){
```

```
    Graph G;
    int n,m,u,v,w,s;
    scanf ("%d%d", &n, &m);
   init_graph(&G, n);

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

    scanf ("%d", &s);
   BellmanFord(&G, s);

   //
   int negative_cycle = 0;
   for (int k = 0; k < G.m; k++){
    u = G.edges[k].u;
    v = G.edges[k].v;
    w = G.edges[k].w;
    if (pi[u] == oo){
        continue;
    }
    if (pi[u] + w < pi[v]){
        negative_cycle = 1;
        break;
    }
   }

   if (negative_cycle){
    printf ("YES");
   }
   else{
    printf ("NO");
   }

    return 0;
}
```

Run 4

```c
#include <stdio.h>
#define MAX_N 1000
#define NO_EDGE -1
#define oo 99999

int pi[MAX_N];  //dinh u(pi/p)
int p[MAX_N];

//Biểu diễn đồ thị bằng phương pháp danh sách cung
typedef struct{
    int u,v;
    int w;
}Edge;

//GRAPH//
typedef struct{
        int n,m;
    Edge edges[MAX_N];
}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++;
}

//Bellman_Ford
void BellmanFord (Graph *pG, int s){
    int u, v, w;
        //khoi tao cac dinh co pi = oo va p = 0
        for (u = 1; u <=pG->n; u++){
                pi[u] = oo;
        }

        //khoi tao dinh bat dau
        pi[s] = 0;
        p[s] = -1; //truoc dinh s khong co cha

    //lặp n-1 lần
        for (int it = 1; it < pG->n; it++){
                for (int k = 0; k < pG->m; k++){
            //duyệt qua các cung và cập nhật nếu thỏa
            u = pG->edges[k].u;
            v = pG->edges[k].v;
            w = pG->edges[k].w;
            if (pi[u] == oo){ //chưa có đường đi đến u, bỏ qua cung u v
                continue;
            }

            if (pi[u] + w < pi[v]){
                pi[v] = pi[u] + w;
                p[v] = u;
            }
                }
        }

}

int main (){
```

```
    Graph G;
    int n,m,u,v,w,s;
    scanf ("%d%d", &n, &m);
   init_graph(&G, n);

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

    scanf ("%d", &s);
   BellmanFord(&G, s);

   //
   int negative_cycle = 0;
   for (int k = 0; k < G.m; k++){
    u = G.edges[k].u;
    v = G.edges[k].v;
    w = G.edges[k].w;
    if (pi[u] == oo){
        continue;
    }
    if (pi[u] + w < pi[v]){
        negative_cycle = 1;
        break;
    }
   }

   if (negative_cycle){
    printf ("YES");
   }
   else{
    printf ("NO");
   }

    return 0;
}
```

Run 5

```c
#include <stdio.h>
#define MAX_N 1000
#define NO_EDGE -1
#define oo 99999

int pi[MAX_N];  //dinh u(pi/p)
int p[MAX_N];

//Biểu diễn đồ thị bằng phương pháp danh sách cung
typedef struct{
    int u,v;
    int w;
}Edge;

//GRAPH//
typedef struct{
        int n,m;
    Edge edges[MAX_N];
}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++;
}

//Bellman_Ford
void BellmanFord (Graph *pG, int s){
    int u, v, w;
        //khoi tao cac dinh co pi = oo va p = 0
        for (u = 1; u <=pG->n; u++){
                pi[u] = oo;
        }

        //khoi tao dinh bat dau
        pi[s] = 0;
        p[s] = -1; //truoc dinh s khong co cha

    //lặp n-1 lần
        for (int it = 1; it < pG->n; it++){
                for (int k = 0; k < pG->m; k++){
            //duyệt qua các cung và cập nhật nếu thỏa
            u = pG->edges[k].u;
            v = pG->edges[k].v;
            w = pG->edges[k].w;
            if (pi[u] == oo){ //chưa có đường đi đến u, bỏ qua cung u v
                continue;
            }

            if (pi[u] + w < pi[v]){
                pi[v] = pi[u] + w;
                p[v] = u;
            }
                }
        }

}

int main (){
```

```
    Graph G;
    int n,m,u,v,w,s;
    scanf ("%d%d", &n, &m);
   init_graph(&G, n);

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

    scanf ("%d", &s);
   BellmanFord(&G, s);

   //
   int negative_cycle = 0;
   for (int k = 0; k < G.m; k++){
    u = G.edges[k].u;
    v = G.edges[k].v;
    w = G.edges[k].w;
    if (pi[u] == oo){
        continue;
    }
    if (pi[u] + w < pi[v]){
        negative_cycle = 1;
        break;
    }
   }

   if (negative_cycle){
    printf ("YES");
   }
   else{
    printf ("NO");
   }

    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	✔

	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	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	✔
✔	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;
```

```
12
13  typedef struct {
14      int n, m;
15      Edge edges[MAXM];
16  } Graph;
17
18  void init_graph(Graph *pG, int n) {
19      pG->n = n;
20      pG->m = 0;
21  }
22
```

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 ►