Started on	Tuesday, 24 June 2025, 9:22 PM
State	Finished
Completed on	Tuesday, 24 June 2025, 9:46 PM
Time taken	23 mins 39 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 một **đơn đồ thị có hướng, có trọng số (có thể âm)** từ bàn phím và in ra chiều dài đường đi ngắn nhất từ đỉnh s đến đỉnh t (s và t cũng được đọc từ bàn phím).

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 $(1 \leq n < 100; 0 \leq m < 500)$
- m dòng tiếp theo mỗi dòng chứa 3 số nguyên u, v, w mô tả cung (u, v) có trọng số w $(0 \leq w \leq 100)$.
- Dòng cuối cùng chứa 2 số nguyên s và t.
- Dữ liệu đầu vào được đảm bảo là đồ thị không chứa chu trình âm.

Đầu ra (Output)

- In ra màn hình chiều dài của đường đi ngắn nhất từ s đến t. Nếu không có đường đi từ 1 đến n, in ra -1.
- Xem thêm ví dụ bên dưới.

Gợi ý

- Khi xét cung (u, v) nếu pi[u] = oo thì bỏ qua cung này, không xét.
- Sau khi kết thúc thuật toán nếu pi[u] = oo thì có nghĩa là không có đường đi từ s đến t.

For example:

Input	Result
3 3 1 2 3 2 3 -5 1 3 4 1 3	-2
3 1 1 2 5 1 3	-1
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 -10 5 6 9 1 6	10

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

```
#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 start, end;
    scanf("%d %d", &start,&end);

    BellmanFord(&G,start);
    if(pi[end] == oo)
        printf("-1");
    else
        printf("%d", pi[end]);


    return 0;
}
```

	Input	Expected	Got	
✔	3 3 1 2 3 2 3 -5 1 3 4 1 3	-2	-2	✔
✔	3 1 1 2 5 1 3	-1	-1	✔
✔	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 -10 5 6 9 1 6	10	10	✔
✔	8 15 1 2 9 1 6 6 1 7 15 2 3 10 6 3 18 6 5 30 6 7 -8 7 5 20 3 5 -16 5 4 11 4 3 6 3 8 19 4 8 6 5 8 16 7 8 44 1 8	19	19	✔
✔	6 9 1 2 7 1 3 9 1 5 14 2 3 10 2 4 1 3 4 -11 3 5 3 4 6 6 5 6 9 1 6	4	4	✔

Passed all tests! ✔

Question author's solution (C):

```c
#include <stdio.h>

#define MAXM 500
#define MAXN 100
#define oo 999999
#define NO_EDGE -1

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.

Question **2**

Correct

Mark 1.00 out of 1.00

Viết chương trình đọc một **đơn đồ thị có hướng, có trọng số (có thể âm)** từ bàn phím và in ra đường đi ngắn nhất từ đỉnh s đến đỉnh t (s và t cũng được đọc từ bàn phím).

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 $(1 \leq n < 100; 0 \leq m < 500)$
- m dòng tiếp theo mỗi dòng chứa 3 số nguyên u, v, w mô tả cung (u, v) có trọng số w $(0 \leq w \leq 100)$.
- Dòng cuối cùng chứa 2 số nguyên s và t.
- Dữ liệu đầu vào được đảm bảo là đồ thị không chứa chu trình âm.
- Luôn có đường đi từ s đến t.

Đầu ra (Output)

- In đường đi ngắn nhất từ s đến t theo mẫu:

```
s -> u1 -> u2 -> ... -> t
```

- Xem thêm ví dụ bên dưới.

Gợi ý

- Lần ngược theo p[u] để có được đường đi ngắn nhất.

For example:

Input	Result
3 3 2 1 3 1 3 -5 2 3 4 2 3	2 -> 1 -> 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 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
```

```
42
43      // lặp n-1 lần
44      for (it = 1; it < pG->n; it++) {
45          // Duyệt qua các cung và cập nhật (nếu thoả)
46          for (k = 0; k < pG->m; k++) {
47              u = pG->edges[k].u;
48              v = pG->edges[k].v;
49              w = pG->edges[k].w;
50
51              if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
52                  continue;
53
54              if (pi[u] + w < pi[v]) {
55                  pi[v] = pi[u] + w;
56                  p[v] = u;
57              }
58          }
59      }
60      //Làm thêm 1 lần nữa để kiểm tra chu trình âm (nếu cần thiết)
61      for (k = 0; k < pG->m; k++) {
62          u = pG->edges[k].u;
63          v = pG->edges[k].v;
64          w = pG->edges[k].w;
65
66          if (pi[u] == oo)    //chưa có đường đi từ s -> u, bỏ qua cung này
67              continue;
68
69          if (pi[u] + w < pi[v]) {
70              return 1;
71          }
72      }
73      return 0;
74  }
75
76
77
78  int main() {
79      Graph G;
80      int n, m;
81      scanf("%d%d", &n, &m);
82      init_graph(&G, n);
83
84      for (int e = 0; e < m; e++) {
85          int u, v, w;
86          scanf("%d%d%d", &u, &v, &w);
87          add_edge(&G, u, v, w);
88      }
89      int start, end;
90      scanf("%d %d", &start,&end);
91
92      BellmanFord(&G,start);
93
94          //Tìm đường đi ngắn nhất
95      int path[MAXN]; //lưu các đỉnh trên đường đi
96      int k = 0;      //số đỉnh của đường đi
97      int current = end;// bat dau duyet tu cuoi duong
98
99      //Lần ngược theo p để lấy đường đi
100     while (current != -1) {
101         path[k] = current; k++;
102         current = p[current];
103     }
104
105     //In ra màn hình theo chiều ngược lại
106     printf("%d", path[k-1]);
107
108     for (int u = k-2; u >= 0; u--)
109         printf(" -> %d", path[u]);
110
111     return 0;
112 }
```

	Input	Expected	Got	
✔	3 3 2 1 3 1 3 -5 2 3 4 2 3	2 -> 1 -> 3	2 -> 1 -> 3	✔

	Input	Expected	Got	
✔	3 4 1 2 5 2 3 -6 1 3 2 3 2 7 1 2	1 -> 2	1 -> 2	✔
✔	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 -10 5 6 9 1 6	1 -> 3 -> 4 -> 6	1 -> 3 -> 4 -> 6	✔
✔	8 15 1 2 9 1 6 6 1 7 15 2 3 10 6 3 18 6 5 30 6 7 -8 7 5 20 3 5 -16 5 4 11 4 3 6 3 8 19 4 8 6 5 8 16 7 8 44 1 8	1 -> 2 -> 3 -> 5 -> 8	1 -> 2 -> 3 -> 5 -> 8	✔
✔	6 9 1 2 7 1 3 9 1 5 14 2 3 10 2 4 1 3 4 -11 3 5 3 4 6 6 5 6 9 1 6	1 -> 3 -> 4 -> 6	1 -> 3 -> 4 -> 6	✔

Passed all tests! ✔

Question author's solution (C):

```c
#include <stdio.h>

#define MAXM 500
#define MAXN 100
#define oo 999999
#define NO_EDGE -1

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 7 - Thuật toán Bellman - Ford (kiểm tra chu trình âm)

Jump to...

Bài tập 8 - Extended traffic ►