Started on	Tuesday, 24 June 2025, 10:02 PM
State	Finished
Completed on	Tuesday, 24 June 2025, 10:08 PM
Time taken	6 mins 20 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Extended traffic (sửa lại từ: https://lightoj.com/problem/extended-traffic)

Thành phố Dhaka ngày càng đông dân và ồn ào hơn. Một số con đường luôn đông nghẹt và kẹt xe. Để thuyết phục người dân tránh đi các tuyến đường ngắn nhất để giảm tải cho các con đường, thị trưởng thành phố đã lập một kế hoạch như sau. Mỗi giao lộ u của thành phố được gán 1 con số nguyên dương (không quá 20) cho biết mức độ bận rộn (busy-ness) của giao lộ này, kí hiệu là $b[u]$. Mỗi khi một người dân đi từ giao lộ u đến một giao lộ v, thành phố sẽ có thêm một lượng bận rộn là $(b[v]-b[u])^3$.

Thị trưởng nhờ bạn kiểm tra xem nếu một người đi từ giao lộ s đến giao lộ t, anh ta sẽ làm tăng thêm một lượng bận rộn ít nhất là bao nhiêu.

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 giao lộ và số con đường nối hai giao lộ lại với nhau.
- Dòng thứ hai chứa n số nguyên mô tả độ bận rộn b[u] của các giao lộ.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v nói rằng có một con đường nối hai giao lộ u và v. Tất cả các con đường đều là đường 1 chiều.
- Dòng cuối cùng chứa 2 số nguyên s và t.

Đầu ra (Output)

- In ra màn hình lượng bận rộn ít nhất khi 1 người đi từ giao lộ s đến giao lộ t. Nếu không có đường đi thì in ra ?
- Xem thêm ví dụ bên dưới.

Gợi ý

- Đưa bài toán về đồ thị có hướng có trọng số
- Tìm đường đi ngắn nhất từ s đến t

For example:

Input	Result
5 6 6 7 8 9 10 1 2 2 3 3 4 1 5 5 4 4 5 1 2	1
5 6 6 7 8 9 10 1 2 2 3 3 4 1 5 5 4 4 5 1 4	3
5 6 6 7 8 9 10 1 2 2 3 3 4 1 5 5 4 4 5 1 5	4

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;

```

```
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 cube(int x) {
    return x*x*x;
}

int main() {
    Graph G;
    int n, m;

    int b[MAXN];

    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int u = 1; u <= n; u++)
        scanf("%d", &b[u]);

    for (int e = 0; e < m; e++) {
        int u, v;
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v, cube(b[v] - b[u]));
        //add_edge(&G, v, u, cube(b[u] - b[v]));
    }
    int s, t;
    scanf("%d%d", &s, &t);
```

```
100
101     BellmanFord(&G, s);
102     if (pi[t] == oo)
103         printf("?\n");
104     else
105
106         printf("%d\n", pi[t]);
107
108     return 0;
109 }
```

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 cube(int x) {
    return x*x*x;
}

int main() {
```

```
    Graph G;
    int n, m;

    int b[MAXN];

    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int u = 1; u <= n; u++)
        scanf("%d", &b[u]);

    for (int e = 0; e < m; e++) {
        int u, v;
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v, cube(b[v] - b[u]));
        //add_edge(&G, v, u, cube(b[u] - b[v]));
    }
    int s, t;
    scanf("%d%d", &s, &t);

    BellmanFord(&G, s);
    if (pi[t] == oo)
        printf("?\n");
    else

        printf("%d\n", pi[t]);

    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 cube(int x) {
    return x*x*x;
}

int main() {
```

```
    Graph G;
    int n, m;

    int b[MAXN];

    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int u = 1; u <= n; u++)
        scanf("%d", &b[u]);

    for (int e = 0; e < m; e++) {
        int u, v;
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v, cube(b[v] - b[u]));
        //add_edge(&G, v, u, cube(b[u] - b[v]));
    }
    int s, t;
    scanf("%d%d", &s, &t);

    BellmanFord(&G, s);
    if (pi[t] == oo)
        printf("?\n");
    else

        printf("%d\n", pi[t]);

    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 cube(int x) {
    return x*x*x;
}

int main() {
```

```
    Graph G;
    int n, m;

    int b[MAXN];

    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int u = 1; u <= n; u++)
        scanf("%d", &b[u]);

    for (int e = 0; e < m; e++) {
        int u, v;
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v, cube(b[v] - b[u]));
        //add_edge(&G, v, u, cube(b[u] - b[v]));
    }
    int s, t;
    scanf("%d%d", &s, &t);

    BellmanFord(&G, s);
    if (pi[t] == oo)
        printf("?\n");
    else

        printf("%d\n", pi[t]);

    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 cube(int x) {
    return x*x*x;
}

int main() {
```

```
    Graph G;
    int n, m;

    int b[MAXN];

    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int u = 1; u <= n; u++)
        scanf("%d", &b[u]);

    for (int e = 0; e < m; e++) {
        int u, v;
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v, cube(b[v] - b[u]));
        //add_edge(&G, v, u, cube(b[u] - b[v]));
    }
    int s, t;
    scanf("%d%d", &s, &t);

    BellmanFord(&G, s);
    if (pi[t] == oo)
        printf("?\n");
    else

        printf("%d\n", pi[t]);

    return 0;
}
```

	Input	Expected	Got	
✔	5 6 6 7 8 9 10 1 2 2 3 3 4 1 5 5 4 4 5 1 2	1	1	✔
✔	5 6 6 7 8 9 10 1 2 2 3 3 4 1 5 5 4 4 5 1 4	3	3	✔
✔	5 6 6 7 8 9 10 1 2 2 3 3 4 1 5 5 4 4 5 1 5	4	4	✔
✔	4 1 10 11 12 13 1 2 2 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 -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 6 - Thuật toán Bellman - Ford

Jump to...

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