```c
// Dijkstra tim duong di ngan nhat tu (1 - n)
#include <stdio.h>

#define MAXN 1000
#define NO_EDGE 0
#define INFINITY 9999999

// Graph
typedef struct {
    int n;
    int L[MAXN][MAXN];
} Graph;

void init_graph(Graph* G, int n) {
    G->n = n;

    int i, j;
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j) {
            G->L[i][j] =NO_EDGE;
        }
    }
}

void add_edge(Graph* G, int x, int y, int w) {
    G->L[x][y] = w;
}

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

void Dijkstra(Graph* G, int s) {
    int i, j, it;
    for (i = 1; i <= G->n; ++i) {
        pi[i] = INFINITY;
        mark[i] = 0;
    }

    pi[s] = 0;
    p[s] = -1;

    for (it = 1; it < G->n; ++it) {
```

```
        int min_pi = INFINITY;
        for (j = 1; j <= G->n; ++j) {
            if (mark[j] == 0 && pi[j] < min_pi) {
                min_pi = pi[j];
                i = j;
            }
        }

        mark[i] = 1;
        for (j = 1; j <= G->n; ++j) {
            if (G->L[i][j] != NO_EDGE && mark[j] == 0) {
                if (pi[i] + G->L[i][j] < pi[j]) {
                    pi[j] = pi[i] + G->L[i][j];
                    p[j] = i;
                }
            }
        }
    }
}

int main() {
    Graph G;
    int n, m, u, v, w, e;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

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

    Dijkstra(&G, 1);

    if (pi[n] == INFINITY) {
        printf("-1");
    } else {
        printf("%d", pi[n]);
    }
    return 0;
}
```