```c
// BellMan Ford tim duong di ngan nhat tu 1 den cac dinh con lai
//pi[1] = 0, p[1] = -1
//pi[2] = 9, p[2] = 1
//pi[3] = 4, p[3] = 1
#include <stdio.h>

#define MAXN 1000
#define NO_EDGE 0
#define INFINITY 999

// Graph
typedef struct {
    int u, v;
    int w;
} Edge;
typedef struct {
    int n, m;
    Edge edges[1000];
} Graph;

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

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

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

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

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

```
    for (it = 1; it < G->n; ++it) {
        for (j = 0; j < G->m; ++j) {
            int u = G->edges[j].u;
            int v = G->edges[j].v;
            int w = G->edges[j].w;
            if (pi[u] + w < pi[v]) {
                pi[v] = pi[u] + w;
                p[v] = u;
            }
        }
    }
}

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);
    }

    BellmanFord(&G, 1);

    for (int i = 1; i <= n; ++i) {
        printf("pi[%d] = %d, p[%d] = %d\n", i, pi[i], i, p[i]);
    }

    return 0;
}
```