//#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <iostream>

#include <cstring>

#include <string>

#include <algorithm>

#include <queue>

#include <stack>

#include <vector>

#include <map>

#include <set>

#include <list>

#include <cstdio>

#include <utility>

#include <bitset>

#include <ctime>

#include <random>

#include <iomanip>

#include <cmath>

using namespace std;

#define IOS ios::sync\_with\_stdio(false);cin.tie(nullptr);cout.tie(nullptr);

#define endl '\n'

#define mem(f,z) memset(f,z,sizeof f)

#define INF 0x7fffffff

#define INFF 0x3f3f3f3f

#define ll long long

#define ull unsigned long long

#define PP pair<int ,int>

int n, m;

const int N = 2e5 + 500;

//前项星存图、卡vector存图时可以使用

const int Maxm = N;

const int Maxn = N;

int head[Maxn];

int cnt = 0;

struct edge

{

int to, next, val;

}e[Maxm];

inline void add(int u, int v, int w)

{

e[cnt].to = v;

e[cnt].val = w;

e[cnt].next = head[u];

head[u] = cnt++;

}

//SPFA算法

ll dist[N];

bool inq[N];

//用来记录路径，只求最短路时不需要

//int lj[N];

//int p[N];

//int c = 0;

void spfa(int root)

{

memset(dist, 0x3f, sizeof(dist));

dist[root] = 0;

inq[root] = true;

queue<int> q;

q.push(root);

while (!q.empty())

{

auto p = q.front();

q.pop();

inq[p] = false;

for (int i = head[p]; i != -1; i = e[i].next)

{

int j = e[i].to;

if (dist[j] > dist[p] + e[i].val)

{

dist[j] = dist[p] + e[i].val;

//lj[j] = t;用来记录上一个结点是谁，只求最短路时不需要

if (!inq[j])

{

q.push(j);

inq[j] = true;

}

}

}

}

// 输出路径，可以用stack<int> 代替p[]数组。

/\*for (int i = n; i != 0; i = lj[i])

{

p[c++] = i;

}

if (p[c - 1] != root)

{

cout << -1 << endl;

return;

}

for (int i = c - 1; i >= 0; i--)

{

cout << p[i] << ' ';

}

cout << endl;\*/

}

int main()

{

IOS;

cin >> n >> m;

memset(head, -1, sizeof(head));

while (m--)

{

int a, b, c;

cin >> a >> b >> c;

add(a, b, c);

add(b, a, c);

}

spfa(1);

return 0;

}