# spfa求最短路

适用于负边情况

## 变量及其初始化

```
vector<PII> g[maxn];
int dis[maxn],cnt[maxn];
bool vis[maxn];

void init(int n)
{
    for (int i = 0; i <= n; ++i)
        g[i].clear();
    memset(dis, 0x3f, sizeof dis);
    memset(cnt, 0, sizeof cnt);
    memset(vis, 0, sizeof vis);
}
```

## spfa函数

```
bool spfa(int n, int start)
{
    //返回true表示无负环
    queue<int> q;
    q.push(start);
    dis[start] = 0;
    vis[start] = true;
    ++cnt[start];
    while (!q.empty())
    {
        int now = q.front();
        q.pop();
        vis[now] = 0;
        for (auto j : g[now])
        {
            if (dis[j.second] > dis[now] + j.first)
            {
                if (!vis[j.second])
                {
                    q.push(j.second);
                    ++cnt[j.second];
                    vis[j.second] = true;
                    if (cnt[j.second] >= n)
                        return false;
                }
                dis[j.second] = dis[now] + j.first;
            }
        }
```

```
    }
    return true;
}
```

## 完整代码

```
#include <bits/stdc++.h>
#define endl '\n'

using namespace std;

typedef long long ll;
typedef vector<ll> VI;
typedef pair<int, int> PII;

const int maxn = 2e5 + 5;
const ll mod = 1e9+7;

vector<PII> g[maxn];
int dis[maxn],cnt[maxn];
bool vis[maxn];

void init(int n)
{
    for (int i = 0; i <= n; ++i)
        g[i].clear();
    memset(dis, 0x3f, sizeof dis);
    memset(cnt, 0, sizeof cnt);
    memset(vis, 0, sizeof vis);
}
bool spfa(int n, int start)
{
    //返回true表示无负环
    queue<int> q;
    q.push(start);
    dis[start] = 0;
    vis[start] = true;
    ++cnt[start];
    while (!q.empty())
    {
        int now = q.front();
        q.pop();
        vis[now] = 0;
        for (auto j : g[now])
        {

            if (dis[j.second] > dis[now] + j.first)
            {
                if (!vis[j.second])
                {
                    q.push(j.second);
                    ++cnt[j.second];
                    vis[j.second] = true;
```

```
                    if (cnt[j.second] >= n)
                        return false;
                }
                dis[j.second] = dis[now] + j.first;
            }
        }
    }
    return true;
}

inline void solve()
{
    int n, m;
    cin >> n >> m;
    init(n);
    for (int i = 0; i < m; ++i)
    {
        int u, v, l;
        cin >> u >> v >> l;
        g[u].push_back({ l,v });
        //g[v].push_back({ l,u });
    }
    spfa(n, 1);
    for (int i = 2; i <= n; ++i)
        cout << dis[i] << endl;
}

int main()
{
    int T = 1;
    //cin >> T;
    while (T--)
        solve();
    return 0;
}
```