差分约束

Difference Constraints

Idea：给定 n 个变量 $x_1,x_2,\cdots,x_n$ 和 m 个约束条件，每个约束条件都行如 $x_i-x_j\leqslant c_k$，称之为**差分约束系统**。一个满足所有约束条件的解 $x_1,x_2,\cdots,x_n$ 就是该差分约束系统的一个合法的解。

为求解差分约束系统，注意到 $x_i-x_j\leqslant c_k\iff x_i\leqslant x_j+c_k$，形似最短路的松弛条件：$\text{dis}_y\leqslant\text{dis}_x+w(x,y)$，所以我们可以从 j 到 i 连接一条长为 c_k 的有向边来表示这种关系。具体链接方式如下表所示：

题意	转化	连边
$x_i-x_j\leqslant c$	$x_i-x_j\leqslant c$	$j\xrightarrow{c}i$
$x_i-x_j\geqslant c$	$x_j-x_i\leqslant -c$	$i\xrightarrow{-c}j$
$x_i=x_j$	$x_i-x_j\leqslant 0,\ x_j-x_i\leqslant 0$	$i\xrightarrow{0}j$ $j\xrightarrow{0}i$

> 注：如果问题在 $\mathbb{Z}$ 上，那么 $x_i-x_j<c\iff x_i-x_j\leqslant c-1$.

转换问题后，若得到的图存在**负环**，则差分约束系统无解；否则，取每个点的最短路距离就是一组合法的解。

Code：

```
bool inq[N];
int dis[N], cnt[N];
bool SPFA(int s){
    queue<int> q;
    dis[s] = 0, q.push(s), inq[s] = 1, cnt[s]++;
    while(!q.empty()){
        int cur = q.front(); q.pop();
        inq[cur] = 0;
        for(auto &to : edge[cur]){
            if(dis[to.first] > dis[cur] + to.second){
                dis[to.first] = dis[cur] + to.second;
                if(!inq[to.first]){
                    q.push(to.first);
                    inq[to.first] = 1;
                    if(++cnt[to.first] > n) return false;
                }
            }
        }
    }
    return true;
}

int main(){
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= m; i++){
        int a, b, c; scanf("%d%d%d", &a, &b, &c);
        edge[b].pb(mp(a, c));
    }
    for(int i = 1; i <= n; i++) dis[i] = INF;
    bool ok = true;
    for(int i = 1; i <= n && ok; i++)
        if(!cnt[i]) ok &= SPFA(i);
    if(!ok) puts("NO");
    else{
```

```
        for(int i = 1; i <= n; i++) printf("%d ", dis[i]);
        puts("");
    }
    return 0;
}
```