最短路算法（二）SPFA

一、 Bellman-Ford 算法

对于单源最短路径的问题，前面介绍了 Dijkstra 算法，但是 Dijkstra 算法对于带负权边的图就无能为力了，而 Bellman-ford 算法可以解决这一问题。

Bellman-ford 算法根据发明者 Richard Bellman 和 LesterFord 命名，可以处理路径权值为负数时的单源最短路径问题。设想可以从图中找到一个环路(即从 v 出发，经过若干个点之后又回到 v)且这个环路中所有路径的权值之和为负。

那么通过这个环路，环路中任意两点的最短路径就可以无穷小下去。如果不处理这个负环路，程序就会永远运行下去。而 Bellman-ford 算法具有分辨这种负环路的能力。

Bellman-ford 算法的核心思想是松弛。

如果 dist[u]和 dist[v]满足 dist[v]>=dist[u]+map[u][v]，dist[v]就应该被更新为 dist[u]+map[u][v]。

反复地利用上式对 dist 数组进行松弛，如果没有负权回路的话，应当会在 n-1 次松弛后结束。原因在于考虑对每条边进行 1 次松弛的时候，得到的实际上是最多经过 0 个点的最短路径，对于每条边进行两次松弛的时候得到的是至多经过 1 个点的最短路径，如果没有负权回路，那么任意两点间的最短路径至多经过 n-2 个点，因此经过 n-1 次松弛操作后应当可以得到最短路径。

如果有负权回路，那么第 n 次松弛操作仍然会成功，Bellman-ford 算法就是利用这个性质判定负环

算法过程：时间复杂度 O(nm)

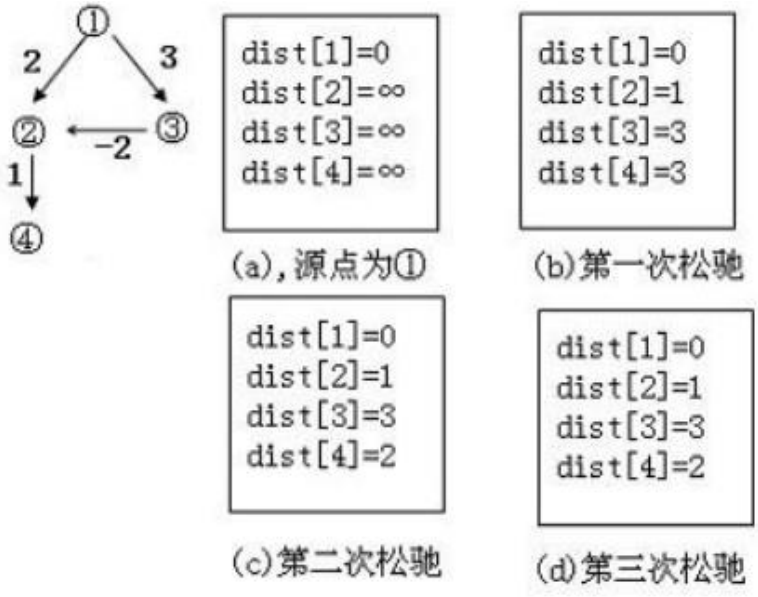
①初始化，所有点 i 初始化为 dist[i]=INF，出发点 s，dist[s]=0；

②对于每条边(u,v)，如果 dist[u]!=INF 且 dist[v]>dist[u]+map[u][v],则 dist[v]=dist[u]+map[u][v]。n

③循环步骤②n-1 次或直到某次操作中不再更新，进入步骤④；

④对于每条边(u,v)，如果 dist[u]!=INF 且 dist[v]>dist[u]+map[u][v]，则存在负权回路。

图例：



例 1：A2050 模板题

参考代码：

```
using namespace std;
#define N 10000
struct Node{
    int u,v,w;
}e[N<<1];//边目录
int dis[N];
int n,m,st,ed;
void init(){
    scanf("%d%d%d%d",&n,&m,&st,&ed);
    for(int i=1;i<=m;i++){
        scanf("%d%d%d",&e[i].u,&e[i].v,&e[i].w);
    }
}

void bfort(int s){
    memset(dis,0x3f,sizeof(dis));
    dis[s]=0;
    for(int i=1;i<n;i++)//松弛n-1次
        for(int j=1;j<=m;j++){//因为是无向图，所以相当于松弛2次
            if(dis[e[j].u ]+ e[j].w < dis[e[j].v])
                dis[e[j].v]=dis[e[j].u ]+ e[j].w;
            if(dis[e[j].v]+ e[j].w < dis[e[j].u])
                dis[e[j].u]=dis[e[j].v ]+ e[j].w;
        }
    //如果还能松弛就有负环
}
int main(){
    init();
    bfort(st);
    cout<<dis[ed];
    return 0;
}
```

二、SPFA

SPFA 是使用队列实现的 Bellman-Ford 算法。采取邻接表存储图，具体的算法过程如下：

初始队列和标记数组

1 源点入队。

2 对队首点出发的所有边进行松弛操作（即更新最小值）。

3 将不在队列中的尾结点入队。

4 队首点更新完其所有的边后出队。

三、SPFA 应用

例：模板 A2050

参考代码：

```
#include<bits/stdc++.h>
using namespace std;
#define N 2510
#define M 8010
```

```
struct Node{
    int u,v,w;
    Node(int u=0,int v=0,int w=0):u(u),v(v),w(w){}
}e[M*2];
int nxt[M*2],first[N]={}, cnt=0,d[N],vis[N];
int n,m,s,t;
void add(int u,int v,int w){
    e[++cnt].u=u;e[cnt].v=v;e[cnt].w=w;
    nxt[cnt]=first[u];first[u]=cnt;
}

void spfa(int s){
    queue<int>q;
    memset(d,0x3f,sizeof(d));
    memset(vis,0,sizeof(vis));
    q.push(s);vis[s]=1;d[s]=0;
    while(!q.empty()){
        int u=q.front();q.pop();
        vis[u]=0;
        for(int i=first[u];i;i=nxt[i]){
            int v=e[i].v;
            if(d[v]>d[u]+e[i].w){
                d[v]=d[u]+e[i].w;
                if(!vis[v]){
                    vis[v]=1;q.push(v);
                }
            }
        }

    }

}
int main(){
    scanf("%d%d%d%d",&n,&m,&s,&t);
    for(int i=1;i<=m;i++){
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        add(u,v,w);add(v,u,w);
    }
    spfa(s);
    cout<<d[t];
    return 0;
}
```

例 3：P10085　「一本通 3.3 练习 2」虫洞

【解析】本题主要是需要判断负环，SPFA 也可以判断图中是否出现负环，通过每个点进队的次数来判断，每个点最多被更新 N-1 次，超过 n-1 次则有负环。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
#define N 3010
struct node{
    int u,v,w,nxt;
}e[N*10];
int n,m,w,first[N],dis[N],c[N],cnt,vis[N],f;
inline void add(int u,int v,int w){
    e[++cnt].u=u;   e[cnt].v=v; e[cnt].w=w;
    e[cnt].nxt=first[u];    first[u]=cnt;
}
int spfa(int s){
    memset(dis,0x3f,sizeof(dis));
    memset(vis,0,sizeof(vis));
    memset(c,0,sizeof(c));
    queue<int>q;
    dis[s]=0;
    q.push(s);
    c[s]++;
    while(!q.empty()){
        int u=q.front();q.pop();
        vis[u]=0;
        for(int i=first[u];i;i=e[i].nxt){
            int v=e[i].v;
            if(dis[u]+e[i].w<dis[v]){
                dis[v]=dis[u]+e[i].w;
                if(++c[v]>=n) return 0;
                if(vis[v]) continue;//在队列中不进队
                vis[v]=1;q.push(v);
            }
        }
    }
    return 1;
}
int main(){
    cin>>f;
    while(f--){
        memset(first,0,sizeof(first));
```

```
        memset(e,0,sizeof(e));
        cnt=0;
        cin>>n>>m>>w;
        for(int i=1;i<=m;i++){
            int x,y,z;
            cin>>x>>y>>z;
            add(x,y,z);
            add(y,x,z);
        }
        for(int i=1;i<=w;i++){
            int x,y,z;
            cin>>x>>y>>z;
            add(x,y,-z);
        }
        for(int i=1;i<=n;i++)
            add(0,i,0);
        if(spfa(0)==0)
            cout<<"YES";
        else
            cout<<"NO";
        cout<<endl;
    }
    return 0;
}
```