Bellman-Ford 算法

【算法概述】

Bellman-Ford 算法计算源点到其他的最短路径问题，相比与Dijstra算法，Dijstra算法求最短路径问题的时候图的连接权重不能为负。在没有负权回路的情况下，采用Bellman-Ford算法可以正确求出最短路径。缺点是时间复杂度为O(VE)，复杂度比较高。

【算法思路】

1、用数组d来表示源点到其他点的距离。初始化数组d[0]为0，d[i]为无穷大。

2、遍历所有边，进行松弛计算。松弛计算为：

对于每一条边edge(start, end)，如果d[start] + weight(start,end) < d[end]，则令d[end] = di[start] + weight(start, end)。 若上述操作没有对dis更新，说明最短路径查找完毕，或者部分点不可达到，跳出循环。否则执行下次循环。

3、遍历所有的变(edge(u, v))，判断是否存在这样的情况： d(v) > d(u) + w(u,v)，则返回false，表示图中存在从源点可达的权为负的回路。

【伪代码】

```
BELLMAN-FORD(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 for i ← 1 to |V[G]| - 1
3       do for each edge (u, v) ∈ E[G]
4             do RELAX(u, v, w)
5 for each edge (u, v) ∈ E[G]
6       do if d[v] > d[u] + w(u, v)
7             then return FALSE
8 return TRUE
```

【代码】

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

const int maxnum=100;   //最大边数
const int maxint=9999;  //源点和某点不可达时的距离

//有向边的结构体
typedef struct Edge
{
    int Start; //有向边边的起始点
    int End;   //有向边的终点
    int Weight;//边的权重
}Edge;

Edge edge[maxnum];//有向边的数组
int dist[maxnum]; //距离数组

//节点的数目、边的数目、源节点的下标
int nodenum,edgenum,source;

//初始化函数
void Init()
{
    cin>>nodenum>>edgenum>>source;

    for(int i=1;i<=nodenum;i++)
        dist[i]=maxint;

    dist[source]=0;

    for(int i=1;i<=edgenum;i++)
    {
        cin >> edge[i].Start >> edge[i].End >> edge[i].Weight;
        if(source==edge[i].Start)
        {
            dist[edge[i].End]=edge[i].Weight;
        }
    }
}

//松弛函数
void Relax(int Start,int End,int Weight)
{
    if(dist[End]>dist[Start]+Weight)
        dist[End]=dist[Start]+Weight;
}

//贝尔曼福特函数
bool Bellman_Ford()
{
    for(int i=1;i<=nodenum-1;i++)
    {
```

```
        for(int j=1;j<=edgenum;j++)
        {
            Relax(edge[j].Start,edge[j].End,edge[j].Weight);
        }
    }
    bool flag=1;
    for(int i=1;i<=edgenum;i++)
    {
        //判断是否存在负回路
        if(dist[edge[i].End]>dist[edge[i].Start]+edge[i].Weight)
        {
            flag=0;
            break;
        }
    }
    return flag;
}

int main()
{
    freopen("out.txt","r",stdin);//打开txt
    Init();
    if(Bellman_Ford())
    {
        for(int i=1;i<nodenum;i++)
            cout<<dist[i]<<endl;//打印源节点到每个节点的距离
    }
    return 0;
}
```