首先输入数据 第一号：2 3 2（顶点2到顶点3距离为2，跟dijkstra的差不多）

第二号：1 2 -3

第三号：1 5 5

第四号：4 5 2

第五号：3 4 3

首先上代码，核心代码只有4行，很简洁。

for(k = 1;k <= n - 1;k++)

{

for(i = 1;i <= m;i++)

{

if(dis[v[i]] > dis[u[i]] + w[i])

{

dis[v[i]] = dis[u[i]] + w[i];

}

}

}

if语句很眼熟对不对，要更新的就是dis[v[i]]（源顶点1到v[i]点的距离），判断dis[u[i]] + w[i]（u[i]到v[i]的距离）是否比dis[v[i]]小，小的话就更新，这种操作就叫松弛。

那外面两个循环是什么回事呢？

第二个循环就是遍历每一个边，很好理解

第一个循环就是根据顶点数进行循环，第一轮循环结束后，得到的是从1号顶点出发，可以直接到达顶点的最短路径。第二轮循环结束后，得到的是从1号顶点出发，最多经过一个顶点的最短路径。以此类推，进行n-1轮。

到最后 ，这个版本的bellman-ford算法有很多地方要优化的地方。

最短路径最多有n-1条边，在每一轮，就是上面的第一个循环，我们都要执行松弛操作，有些边求得了最短路径了，在接下来的循环不会变，但我们还会处理它，这里就浪费了时间，可以优化。所以就有了bellman-ford的队列优化。

Bellman-ford算法队列优化

主要的思想就是，不用两个for循环来遍历了，用队列来存储点，松弛完就出队。

dis数组还是要的，作用一样。que数组存储各个顶点，一开始只有源顶点1.

Book数组判断该顶点有没有加入队列。用邻接表来存储图的数据

上代码：

n为顶点，m为边数

初始化dis数组

for（int i = 1;i <= n;i++）

Dis[i] = inf;

Dis[1] = 0;

初始化book数组全为0。

初始化first数组全为-1

用邻接表存储

for（int i = 1;i <= m;i++）

{

存入u[i],v[i],w[i];

Next[i] = first[u[i]];

First[u[i]] = i;

}

源顶点入队

Head = tail = 1;

Que[tail] = 1;

Tail++;

Book[1] = 1;

开始循环：

While（head<tail）//当队列为空就结束

{

K = first[que[head]];//每次读取队首的顶点，进行松弛操作

While(k != -1)//邻接表的遍历方式，扫描这个点的所有出边

{

If(dis[v[k]] > dis[u[k]] + w[k]])//核心bellman-ford算法语句,判断是否松弛成功

{

Dis[v[k]] = dis[u[k]] + w[k];

If(book[v[k]] == 0)//用book数组判断当前这个点是否在队列，用数组会很方便

{

Que[tail] = v[k];

Tail++;

Book[v[k]] = 1;

//入队操作

}

}

K = next[k];//邻接表遍历当前这个点的下一个出边的操作;

}

Book[que[head]] = 0;//出队操作

Head++;

}

循环输出dis数组。