F题题解报告:差分约束

这一题应该是图论专题里最简单的一道题目（或许没有之一）。

只要明白一个重要定理，轻松解决。

首先明确，改题目是一个差分约束的板子题。

什么是差分约束呢，就是所有不等式满足ai<=aj+c的形式，这时候我们可以把这类数学问题转化为图论问题（其实刚看到这个的时候我首先想到的是matlab直接调用lineprog函数求解线性规划问题）。

如何转化成图论问题呢？

首先，先把所有变量看作图上的顶点，然后把所有不等式转化成上述形式（如果出现ai<=aj+c(k!=1)的情况就寄了），然后依照着不等式在图中添加aj指向ai的有向边，权重即为c。

重点来了：若图中出现负环，则不等式组无解，输出NO；反之则有解，输出YES。

所以此时，F题就转化成了判断是否存在负环的情况了。

判断是否存在负环，有两种比较热门的方法，这里利用bellman_ford算法判断是否存在负环：若循环了n-1次以后，仍旧存在更新路径的情况，则说明图中存在负环，即不等式组不成立。

于是就有以下代码：

```
for(int i=0;i<n;++i){
 for(int j=0;j<m;++j){
  if(dist[e[j].v]>dist[e[j].u]+e[j].w)
  dist[e[j].v]=dist[e[j].u]+e[j].w;
 }
}
for(int i=0;i<m;++i){
 if(dist[e[i].v]>dist[e[i].u]+e[i].w){//若仍旧可以更新，则存在负环，返回false
  return false;
 }
}
```

所以本题解法一共有两大要点：一是利用有负边即无解的定理；而是利用bellman_ford算法判断是否存在负边。

下面来简短地、不严谨地解释下以上两个要点。

首先有观察下这两个式子：

> a1<=a2+1,a2<=a3+2;

如果我们把两个不等式左右相加，则会得到a1<=a3+3。 把第一个式子中的1看成a2->a1的权重，第二个式子的2看成a3->a2的权重。

那么a3->a1的权重就应该是3，与相加后的式子相符合。 据此，我们假设有一大堆式子，如果存在环的话，假设从a1开始最终指回a1，那么把这些不等式加起来，不等式左边一定是零

（参考上面加起来是a1<=a3+c，最终指回a1时，就是a1<=a1+c，消掉）。

零肯定不能小于等于负数，所以有了负环必定无解的结论。

至于bellman_ford算法循环n次判断负环，我们知道，bellman_ford循环n-1次后，正常情况下（即不存在负环）某个确定的节点对其他所有节点的最小路径就已经找到了。

但是如果存在负环的话，那么松弛将会一直存在（等到海枯石烂依旧在循环），故循环次数大于n-1时仍旧更新最小路径的话，那么肯定是因为存在负环了。

> **时间复杂度O（nm）：看核心代码的两层循环就知道了，外层循环n次，内层循环m次。**

附上AC代码如下：

```
#include<iostream>
#include<cstring>
using namespace std;
typedef struct edge{
 int u;
 int v;
 int w;
}edge;
const int inf=9999999;
const int maxn=1e4+5;
edge e[maxn];
int dist[maxn];
int cnt=0;
int n,m;
void add(int u,int v,int w){
 e[cnt].u=u;
 e[cnt].v=v;
 e[cnt++].w=w;
}
int main(){
 int u,v,w;
 scanf("%d%d",&n,&m);
 memset(dist,inf,n);
 for(int i=0;i<m;++i){
  scanf("%d%d%d",&u,&v,&w);
  add(v,u,w);
 }
 dist[e[0].u]=0;
 for(int i=0;i<n;++i){
  for(int j=0;j<m;++j){
   if(dist[e[j].v]>dist[e[j].u]+e[j].w)
   dist[e[j].v]=dist[e[j].u]+e[j].w;
  }
 }
 int key=1;
 for(int i=0;i<m;++i){
  if(dist[e[i].v]>dist[e[i].u]+e[i].w){
   key=0;
   break;
  }
 }
 if(!key)
  printf("NO\n");
 else
  printf("YES\n");
 return 0;
}
```