R题题解：最小树形图

一句话题意：n个节点，m个边的有向图，求改图的最小树形图（即有向图的最小生成树）

相关解法：朱刘算法（Edmonds算法）和Tarjan的DMST算法。后者时间复杂度很低，但代码不易实现，~~本人学不会~~，为方便讲解，本题解采用朱刘算法求解最小树形图，时间复杂度为O(VE)。

核心思想：贪心+缩点

贪心体现在构建最短弧集的时候，即遍历所有边时，找到每个节点的入点的最小值。

缩点就是把有向环缩成一个点；缩点后的图若仍有有向环则继续缩，直到不存在有向环时。

这么做的原理是什么?跟着下面的算法步骤一起理解下吧。

算法实现：

- 构建最短弧集，找到所有点（不包括根节点）入边中最小的那个。假如存在某个点没有入边，那么就无法构建最小树形图。

```
for(int i = 0; i < m; ++ i){
    if(edge[i].u!= edge[i].v){
     if(edge[i].w < min_weight[edge[i].v]){//更小则更新
         min_weight[edge[i].v] = edge[i].w;//min_weight[x]用于存x的最小入边的权重
         pre[edge[i].v] = edge[i].u;
     }
    }
}
```

- 判断最短弧集中有没有有向环（假如有的话肯定就不是树形图了，要注意最短弧只有n-1条边）

```
for(int i = 1; i <= n; ++ i){
    //ans += min_weight[i];
    int v = i;
    while(vis[v] != i && re[v] == -1 && v != r){
        vis[v] = i;
        v = pre[v];//pre[u]表示指向u的节点
    }//在这个过程中，v不断往父节点走，假如存在环，则v必定会回到自己
    //所以有循环条件vis[v]!=i
    ……
```

- 有环，则缩点，然后继续判断是否有环

```
if(v != r && re[v] == -1){//v不是回到根节点（这样才有环）时才会进入此循环
     re[v] = ++tot;//tot是缩点后的编号
     for(int u = pre[v]; u != v; u = pre[u])
re[u] = tot;//环里的点都归属同一个编号
 }
```

- 无环，则拆点。注意到，有向环中只要要有一条边是要被删除的，用环外的另一条边来替换，拆点的过程其实就是实现这个的过程（具体体现在权重的变化上）。

```
    int vv = edge[i].v;
    edge[i].u = re[edge[i].u], edge[i].v = re[edge[i].v];
    if(edge[i].u != edge[i].v){
        edge[i ++].w -= min_weight[vv];
    }//假如同一条边的首尾节点归属的编号不同，，那么两个编号之间的权重更新为原边的权重减去子节点的原最小入边权重
    //因为这条边被删去了。对于每个最小树形图，除根节点外所有节点只有一个入边，而原本的入边会导致成环，故只能删掉
```

> 时间复杂度解析(O(VE))：主要是缩点操作。每次至少缩一个点，每缩一次点要m次操作，一个n个点，故时间复杂度(O(VE))。