NOI练习赛 解题报告
1fd280991d4ca87950774e6459256d0d

1 电阻网络

1.1 20分算法

$n \leq 5$时，只可能是若干电阻串联，直接输出$n-1$。

1.2 50分算法

列基尔霍夫方程组，高斯消元解之。

1.3 70分算法

列基尔霍夫方程组，用比高斯消元更快的方法解之。

1.4 100分算法

直接递归搜索。

具体地，假设所有的边都是从左到右的，那么接点（除了整个电路的负极以外）可以分成3类：出度为2的（并联电路的开始）、出度为1且出边

边权为0的（并联电路的结束）、出度为1且出边边权为1的（是一个电阻）。那么就很好解决了。标程长度不超过700B。

2 收钱

2.1 问题转化

容易看出，本题中所有的COLLECT操作都是无用的。因为这不会对总钱数对人数的模数产生任何影响。第一次IN/OUT之前与最后一次IN/OUT之后的所有PAY操作也是无用的。而连续的几次PAY操作，相当于一次PAY操作，其k值为各次之和。所有的OUT k操作均可以看做是IN $-k$。

只有在IN操作时才会做除法，才有可能产生分数运算。所以对于每一次IN操作，我们可以得到一个信息：从上次IN之后到这次IN之前，所有的PAY操作的k值总和，除以上一次IN之后时当时的人数，余数为0。

设初始人数为x，则对于每次IN操作，我们能够得到1个方程。形式为：$a \bmod (x+b) = 0$，其中a与b均为常数。事实上，a为从上次IN操作之后到现在的PAY操作的k 值总和，b为在此之前所有IN操作的k值总和。总计有$O(N)$个方程。

2.2 30分算法

考虑x的范围。由于a最大可以为$O(Nk)$，而x是a的约数，所以x也是$O(Nk)$的。

枚举x依次代入检验，可以在$O(N^2k)$的时间复杂度解决问题。特殊地，如果方程的个数为0，则应直接输出SIZE $>= M$，其中M为最小的满足全程都至少有1个人的初始人数。

2.3 60分算法

实际上，我们所要求的是，对于$O(Nk)$范围内的每个x，分别满足了上述方程的多少个。显然我们对于以上的每个方程，可以求a的所有约数再减去b，直接找出所有符合该方程的x。

对于每个方程，我们需要用$O(\sqrt{k})$的时间处理，则对于n个方程，我们就可以在$O(N\sqrt{k})$的时间内统计出每个x符合的方程个数，再从上一节所说的M到$O(Nk)$(x可能取到的最大值)枚举x扫一遍即可。总的时间复杂度为$O(Nk)$。

2.4 100分算法

在上面的算法中，实际上我们对数组只进行了$O(N\sqrt{k})$次修改，所以，$O(Nk)$扫一遍是没有必要的。我们需要用数据结构维护这个数组，例如BST。（哈希表的遍历可能会有困难）

用C++ STL中的map可以很方便地实现，从而将其优化到$O(N\sqrt{k}\log N\sqrt{k})$。

3 旅行者

3.1 30分算法

对于每个询问使用状态压缩动态规划暴力求解，可以得到30分。由于与正解无关且过于暴力故略。

3.2 100分算法

考虑这样的一个暴力：

设现在要从a走到b，我们枚举路径上权值最小的点c，现在的问题是确定是否存在一条从a到b的简单路径经过了c，这等价于，从c出发分别存在

到a和b的两条不相交路径。

这个问题可以转化为网络流问题。用(u,v,w)表示从u到v，容量为c的有向弧。建立源S和汇T，连边$(S,c,2),(a,T,1),(b,T,1)$；对于原图的每一条边(u,v)，连边$(u,v,1),(v,u,1)$。如果最大流是2，那么我们就从c找到了两条分别前往a和b的路径。

这样找到的两条路径可能会相交，所以还需要对除了a以外的点进行1的流量限制。这只需把每个点u拆成入点u_0和出点u_1，原来与u相连的边，根据方向分别改接到u_0或u_1上，然后连边$(u_0,u_1,1)$就可以了。

这个算法每次询问的时间是$O(m)$，总时间$O(qm)$。

但是这样并不能解决更大规模的问题。如果问题规模更大呢?

先考虑整个图都是点双连通图的情况。

引理1. *在点双连通图中，对于任意的点$a,b,c(a\neq b)$，总是存在从a到b经过c的简单路径。*

证明. 如果图的顶点数不超过2，结论显然正确。

对于至少有3个点的点双连通图，它一定也是边双连通图，所以删掉任何一个点或任何一条边，图依然是连通的。

考虑之前的网络流建图方式，欲证最大流=2，根据最大流最小割定理，只需证最小割=2，这等价于证不存在权值为1的割。而权值为1的边有以下三种：

对于$(a,T,1),(b,T,1)$这两条边，割掉以后图显然仍然连通。

对于形如$(u_0,u_1,1)$的边，割掉这条边相当于将这个点的容量设为0，即相当于在原图上删掉这个点，根据之前的结论图依然连通。

对于形如$(u_1,v_0,1)$的边，即使把这条边与$(v_1,u_0,1)$一起割掉，也只等价于在原图上删掉这条边，图依然连通。

故不存在权值为1的割，所以最大流为2，证毕。

□

所以对于双连通图，我们只需要用数据结构维护一个（可重）集合，支持修改一个元素和查询最小元素，这是很容易做到的。

对于非双连通图，将它用Tarjan算法求出所有的割点和点双连通分量（以下简称“块”），把所有的割点和块抽象成顶点建一张新图，如果某个块包含了某个割点，就在它们之间连一条边，显然新图是一棵树。定义：

$$treenode(u) = \begin{cases} \text{树中}u\text{对应的顶点}, & u\text{是割点} \\ \text{树中}u\text{所在的块的顶点}, & u\text{不是割点} \end{cases}$$

引理2. 对于点$u, v (u \neq v)$，可能成为答案的点恰好是$treenode(u)$和$treenode(v)$的树上路径经过的所有点。

证明. 先证明所有经过的点都可能成为答案：

对于路径上除了两个端点以外的点，证明是显然的；对于端点$treenode(u)$是割点的情况，证明也是显然的。

现在$treenode(u)$不是割点，设x是$treenode(u)$对应块的任何一个点：

如果u和v在同一个块中，根据引理1，存在从u到v经过x的路径；

如果u和v不在同一个块中，设树上路径的第二个点是割点c，根据引理1，存在从u到c经过x的路径，而这个路径显然不会与从c到v的路径相交。

然后证明没经过的点都不可能成为答案：

对于一个没经过的点x，它走到树上路径上必然会经过某个割点，故从u走到x之后不可能回到树上路径，即不可能走到v。

□

所以我们只需要维护树上路径点权最小值，每次询问直接查询，每次修改先把所在块的最小权值计算出来，再在树上修改这个块。但是如果一个割点被很多块包含，这种方法就会使每次操作的时间退化为线性。

一个比较简便的方法是：对于每个块只维护孩子割点，不维护父亲割点。这样对于割点的修改就只需要修改它的父亲块。然而，在查询树上路径时，如果发现LCA是块，就要把这个块的父亲割点也考虑到答案当中。

用轻重链剖分或Link-Cut Tree维护树上点权，总的时间复杂度为$O(n + m + q \log^2 n)$或$O(n + m + q \log n)$。