仙人掌的同构计数解题报告

长沙市雅礼中学 刘研绎

1 试题来源

经典问题。

2 试题大意

给出一个仙人掌，求他的自同构数量。

仙人掌：每条边只属于一个简单环。

自同构：从V到V的一个双射m，如果v_1, v_2由一条边连接，那么$m(v_1), m(v_2)$也由一条边连接。

$1 \le n \le 100000, 0 \le m \le 100000$

3 算法介绍

3.1 类比化归

不妨先与树的同构计数相类比。在无向树的同构中，我们先找一个根节点，将无向树变成有向的。类似树形动态规划，我们考虑以x为根的子树同构方案数如何计算。不妨枚举x所有子树的形态，令$subtree$为以x某个子节点为根的子树的一种形态，$tree$为以x为根的子树形态，$f[tree]$表示形态为$tree$的树的同构数，$num_x[tree]$表示以x为根的子树有多少形态为$tree$。那么：

$$f[tree] = \sum_{subtree} f[subtree]^{num_x[subtree]} \times num_x[subtree]!$$

对于形态为$subtree$的每一个子树都有$f[subtree]$种方案，而他们之间是同构的因此他们之间的对应关系一共有$num_x[subtree]!$种，两者相乘即可得到答案。判断两棵子树形态是否相同可以用树的hash。

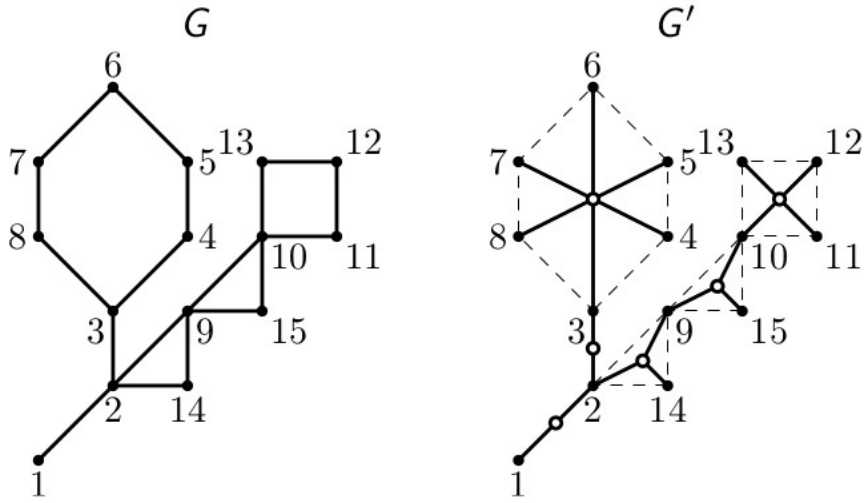
3.2 表示树的形态

我们可以用一个括号序列来表示一棵树的形态，左括号表示进入某个子树，右括号表示从某个子树回到他的父亲。具体的做法是，先递归求出所有当前节点子树的括号表示，再将他们按字典序排序后顺次拼接起来，最后在前后分别加上左右括号，就求出了当前子树的括号表示。

虽然这种做法是不会出错的，但直接保存括号表示的成本比较大，不妨对这个序列进行hash，就能快速表示出这个树的形态了。当然如果使用hash算法的话，只需要设计出将子树的hash值hash起来的函数就行了，这就有很多种方法了。

3.3 将仙人掌转化为树

接下来的问题就是如何将仙人掌同构转化到树上，不难发现我们可以为每一个环（将不在任何环中的边也看做一个环）新建一个节点连向环中的所有点，将环中本来有的边从图中删除，这样就能变成一棵树了。



如图所示，图G在变换后变成了一棵树G'。

对于原有的节点我们可以和计算树的同构一样，对于环节点，我们需要特殊处理。因为环节点的子树是不能简单的互相交换的，但要注意他们仍然有同构变换：翻转（对称）。

3.4 细节的处理

但需要注意的是，在计算树的同构数时，对于整棵树来说我们枚举的根节点不一定对应着自己，因此需要枚举根节点对应哪一个点。实际上没有必要进行这一步枚举，我们可以找一个树上“唯一”的一个点，那就是树的重心。

可以发现这棵树由仙人掌变过来的树的直径长度一定是偶数。不过重心可能在原节点上也有可能在环节点上，如果在环节点上需要特殊注意：

- 和普通的环节点一样，可能存在对称同构。
- 也可能存在循环同构，这样我们就要快速找出所有可能循环同构的，可以hash或者是KMP之类的。

3.5 总览

算法流程：

1. 利用DFS找出仙人掌所有环，并将其转化成树。
2. 找到其重心，以重心为根递归计算每棵子树的同构数。
3. 判断重心节点的情况并分类讨论。
4. 得出答案。

至此问题圆满解决。
时间复杂度：$O(n \log n)$