# Permanent 解题报告

浙江省镇海中学 杜瑜皓

## 1 试题来源

Codeforces #268E Pernament 原创

## 2 试题大意

给一个$n*n$的矩阵$A$，除了$k$个位置外全是1。

求perm($A$) mod $10^9+7$。

规定$n \le 10^5, k \le 50$。

## 3 算法介绍

积和式可以这样计算：对所有边集$(e_1, e_2, \cdots, e_t)$满足$x_{e_1}, x_{e_2}, \cdots, x_{e_t}$两两不同，$y_{e_1}, y_{e_2}, \cdots, y_{e_t}$两两不同，把$(n-t)!\prod_{i=1}^{t}(w_{e_i}-1)$加到答案中。

可以用如下式子证明：

$\text{perm}(A+B) = \sum_{s,t} \text{perm}(a_{ij})_{i\in s, j\in t}\text{perm}(b_{ij})_{i\in \bar{s}, j\in \bar{t}}$,

这里$s$和$t$是$1, 2, ..., n$大小相同的子集，$\bar{s}, \bar{t}$是他们的补集。

其实也可以直观理解，把$w_{e_i}$当成$w_{e_i}-1$和1两条边，分别计算贡献。

构造一个$2n$个结点的无向图$G, v_1, v_2, \cdots, v_{2n}$，$v_i$和$v_{n+j}$之间的边权为$A_{i,j}-1$。

我们只要知道所有选了$t$条边的匹配权值总和，匹配权值就是匹配中所有边权的积。

### 3.1 算法1

事实上，只要知道每个连通分量中选$x$条边的总和，然后背包组合一下就好了。

假设这个连通分量中有$s$个点，$t$条边。

#### 3.1.1 算法1.1

因为它是个二分图，所以一边的点数不超过$s/2$，所以可以使用状压dp解决，时间复杂度为$O(2^{s/2} * s^2)$。

#### 3.1.2 算法1.2

首先可以找到一棵生成树，那么生成树中的树边为$s-1$，非树边为$t-s+1$，所以可以首先枚举所有非树边选或不选，然后使用树形dp计算，时间复杂度为$O(2^{t-s} * s^2)$。

结合这两个算法，时间复杂度为$O(\min(2^{s/2}, 2^{k-s}) * s^2)) = O(2^{k/3} * k^2)$。

### 3.2 算法2

事实上我们考虑爆搜边的状态，一开始是全集。

如果当前的边集构成了超过一个连通块，可以考虑分成若干个连通块，然后分别计算，然后合并。

如果边的个数很少的时候可以使用记忆化。

每次选取一条边，将它选入匹配中，那么两个端点将被删去，或者不选入匹配中，那么这条边被删除。

这条边的选取是整个算法的关键，如果能迅速分裂到若干个小块那么效率就会变高。

一个参考做法是选择一条使得删去后最大连通块尽量小的边。

虽然无法给出复杂度的上界，但是在实际运行中能通过所有测试数据，并且我没有办法将它卡掉。

### 3.3 算法3

$dp[i][k][S]$表示从点1做到点$i$，已经选择了$k$条边，未被匹配的节点的集合是$S$的情况下所有匹配的权值的和。

对于每个节点$i$我们只要从$S$中选择一个节点匹配，或者让这个点不匹配就行了。我们只要考虑一些有边相连的节点。

这样做时间复杂度是$O(2^k * k^2)$。这样是很不高效的。

当我们考虑点$i$时，我们并不需要知道从1到$i$每一个节点是否被匹配了。如果对于一个节点$j(j < i)$，不存在节点$k(k > i)$使得$j$和$k$有边相连，那么我们并不需要知道$j$这个点的状态。

所以对于每个节点$i$，我们只要在$i+1$到$k$之间存下状态就行了，这里的$k$是最大的点使得$i$与$k$之间有边相连。

在最坏的情况下，这样做还是$O(2^k * k^2)$的。

但是我们并不需要从节点1做到节点$n$，我们可以改变点之间的顺序，使得dp的代价尽量小。

现在还剩两个问题：

1.最优的顺序下，这个算法的复杂度是多少？

2.如何找到一个合适的点的顺序？

考虑一棵树，我们把子树按大小从小到大排序，然后我们把dfs序当成dp的顺序。点$i$的状态要记录直到最后一个儿子。考虑点$u$，要记录哪些点的状态，首先只可能数父亲或祖先，如果是祖先，那么这个子树不是最后一个子树，也就是这个子树不是最大的子树，那么这个子树的大小之多为总大小/2，所以每个点要记录状态的点是$\log_2 n + O(1)$的，所以每个点要存储的状态是$O(n)$的。所以所有点的状态是$O(n^2)$的。

如果有非树边，那么状态数最多会乘上2。然而最优的顺序又不会差于直接状压$dp$。所以假设这个连通块的点数为$s$，我们再乘上记录选用边的个数和转移的代价，复杂度还是$O(\min(2^{s/2} * s^2, 2^{k-s+1} * s^4))$。

我们显然可以按上文所述的方式确定点的顺序，但是我们可以在这个基础上，使用随机调整等算法得到更好的顺序，然后在实践中跑得更快。

## 4 总结

这是我在Codeforces Round中出的E题。比赛中没有人通过这个题。

这个题还是一个比较精致的题。一开始我想到了算法1，也就是设定的算法。利用二分图的性质和生成树做到了一个很好的复杂度。

后来我想了想有没有普适的算法，也就是算法3，利用随机调整得到了一个好的dp顺序，然后直接做，这是一个很有趣的idea。并且能证明这个是靠谱的，给出一个上界。同时这个在实践中跑得十分快。

然而赛后有很多人使用算法2通过了这个题。

感谢hos.lyric和sillycross和我的交流。