2018 年全国青少年信息学奥林匹克
浙江省队选拔赛第二试题解

竞赛时间：4 月 26 日 8:00 – 13:00

题目名称	树	胖	保镖
目录	tree	joke	guard
可执行文件名	tree	joke	guard
输入文件名	tree.in	joke.in	guard.in
输出文件名	tree.out	joke.out	guard.out
每个测试点时限	3s	3s	4s
内存限制	512MB	512MB	512MB
测试点数目	10	10	10
每个测试点分值	10	10	10
是否有部分分	否	否	否
题目类型	传统型	传统型	传统型
是否有附加文件	是	是	是
是否有 Special Judge	否	否	是

提交源程序必须加后缀

对于 C++ 语言	tree.cpp	joke.cpp	guard.cpp
对于 C 语言	tree.c	joke.c	guard.c
对于 Pascal 语言	tree.pas	joke.pas	guard.pas

编译开关

对于 C++ 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 C 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 Pascal 语言	-O2	-O2	-O2

1 树

1.1 暴力算法 (20~30pts)

转化后就是要求对所有树的等价类（同构的算一类），将等价类的大小的 k 次方求和，最后再除以 $((n-1)!)^k$ 转化为概率。

1.2 初等算法 (60~70pts)

考虑这样记录状态，设 $f[i][j]$ 表示，jk 棵大小为 i 的树，两两同构的概率，最后的答案即为 $f[n][1]$，为了方便，实际的状态表示的是方案数，$((n-1)!)^k$ 最后再除。

假设已经计算好了 $f[1\ldots i-1][j]$，考虑 i 的子树构成情况，可以用类似背包的方式解决，但是需要注意一个问题，假如选择了 u 个大小为 v 的子树，在传统的计算方法中，这 u 棵子树中出现的同构关系将会不被考虑。

现在的子问题是，在计算 $f[i][j]$ 时选择了 u 棵大小为 v 的子树，考虑这 u 棵子树构成了若干等价类 $\{a_1, a_2, \ldots, a_m\}$，不妨使 $a_i \le a_{i+1}$。当 $m=1$ 时，答案即为 $f[v][a_1 j]$，否则进行如下容斥

- 先假设 $\{a_1\}$ 和 $\{a_2, \ldots, a_m\}$ 独立，递归计算这两个子问题，将答案直接相乘。
- 考虑不满足前一条的情况，枚举 a_1 与 a_i 同构，即递归计算 $\{a_2, \ldots, (a_i + a_1), \ldots a_m\}$。

加上记忆化后，上述的容斥复杂度与划分数有关，由于对 $f[n][1]$ 有用的子状态 $f[i][j]$ 满足 $i \cdot j \le n$，故可以通过 $n \le 50$ 的测试点，一个需要注意的细节是，当有多个大小相同的等价类时，对答案的贡献将会产生额外的系数。

注意到 i 最多有 i/v 个大小为 v 的子树，故可以通过预处理 v 比较小的情况（搜索所有树形态）来减小涉及到的划分数大小，可以通过 $n \le 200$ 的测试点。

1.3 标准算法 (100pts)

上一节中的容斥算法实际上是对等价类做的，既然提到了等价类那么考虑 burnside 引理。

设现在要计算有 m 个大小相同的子树的情况，那么就有 $m!$ 个置换作用其上，在传统 burnside 引理的计算方法中，对于每种在这个置换群下等价的方案，最终对答案的贡献将是 1，但在这里我们希望它对答案的贡献为 $\prod \frac{1}{(a_i!)^k}$，其中 a_i 为其实际各等价类的大小。换句话说，就是在这里统计的轨道是带权的。

为了统计带权的轨道，不妨让稳定化子也带权，仍然保留 burnside 引理的基本架构 $\frac{1}{m!}\sum |stab\ x|$，其中 x 为一个方案，$|stab\ x|$ 表示 x 的稳定子群大小，即在这些置换的作用下 x 不动，设 G_x 为与 x 同构的方案集合，在传统 burnside 中 $|G_x||stab\ x| = 1$，而我们需要 $\prod \frac{1}{(a_i!)^k}$。考虑 x 的每个等价类，对于那些使 x 不动的置换，各个等价类之间是独立的，所以每次只需要考虑一个等价类，可以如下递推构造稳定化子的系数

- 一个稳定化子（置换）的权是它的每一个置换环的权的乘积。
- 大小为 1 的环的权可以直接算出来。

- 假设已经算好了大小为 $1 \ldots i-1$ 的环的权，那么考虑一个大小为 i 的等价类，算出它所有只包含 $\leq i-1$ 长度置换环的稳定化子的权值之和，剩下的部分就由长度为 i 的稳定化子的权值组成。

计算出置换环的权值后，就能用类似背包的 dp 来计算答案，复杂度为 $O(n^2 \log n)$，也可以使用前一节提到的预处理方法来减小常数。

2 胖

因为 day1 大家的分太低了，所以临时加上了这一道送分题。

考虑每一个节点 i，在最短路算法运行的过程中，他会依次被经过瞭望塔 $x_{i,1}, \ldots, x_{i,k_i}$ 的路径更新，不难发现对于同一个 i, $x_{i,j}$ 是两两不同的。因此答案等于每一个瞭望塔更新的节点数之和。

接着考虑每一个瞭望塔 a_i，在算法运行的过程中，他会更新节点集合 S_i 的最短路，我们有如下结论：

Theorem 1 *S_i 是一个包含 a_i 的连续区间。*

首先显然有 $a_i \in S_i$。接着只考虑右边，若 $k \in S_i$ 且 $k > a_i$，那么经过 a_i 到 k 的最短路必定经过 $k-1$，因此在用经过 a_i 到 k 的路径更新 k 时，经过 a_i 到 $k-1$ 的路径必定为 $k-1$ 的当前最短路。

于是对于每一个 a_i，只需要求出 S_i 的左端点 l_i 和 r_i。考虑二分求 l_i，设当前点为 m，则需要判断经过 a_i 到 m 的路径是否能一度成为 m 的最短路。令 $d = |m - a_i|$，则这个条件等价于对于下标在 $[m-a_i, m+a_i]$ 中的所有关键点，到 m 的距离都不小于 a_i 到 m 的距离。

可以用二分查找找到下标 $[m-a_i, m+a_i]$ 对应的瞭望塔区间，然后再求一次区间极值。这可以用 ST 表解决。总时间复杂度 $O(n \log^2 n)$。

要注意考虑多个瞭望塔到同一个点的路径长度相同的情况，如果不加处理的话会算重。在这种情况下可以把这个点归给距离最近（有多个则选左侧）的瞭望塔。这样就不会算重复了。

3 保镖

3.1 Algorithm

首先假设没有三点共线，四点共圆，点数大于 3。

Theorem 2 *不经过反演中心的圆反演之后还是一个圆。经过反演中心的圆反演之后是一条线。*

证明略。

Definition 1 *一个点集的最近点 Voronoi 图是讲整个平面划分按照最近的点划分成若干个区域。*

最近点 Voronoi 图的对偶就是 Delaunay 三角划分，也就是每个三角形的外接圆不包含其他任何点。

Theorem 3 *一个大小为 n 的点集的 Delaunay 三角划分的三角形个数与凸包点数的和为 $2n-2$。*

可以通过欧拉定理 $V-E+F=2$ 来证明。这里的点数为 n。令 T 为 Delaunay 三角剖分后的三角形个数。这里的 F 为所有剖分出来的三角形加上一个无穷大的域。令凸包上的点数为 E_1，内部的边数为 E_2。考虑所有三角形的三条边，内部的边数被算入两次，凸包的边数被算一次。所以可以得到

$$\begin{aligned} E_1+2E_2 &= 3T \\ E_1+E_2 &= E \end{aligned}$$

带入欧拉定理得

$$n+1+T-\frac{3}{2}T-\frac{1}{2}E_1=2$$

化简得

$$T+E_1=2n-2$$

Definition 2 *定义内圆为边界上有三个点且内部没有点的圆，外圆为边界上有三个点且外部没有点的圆。*

Theorem 4 *内圆与最近点 Voronoi 图的顶点一一对应。*

首先 Voronoi 图的顶点为三个区域边界的交，那么这个点到这三个区域对应的点距离相同，并且因为是最近，所以其内部没有其他点。对于一个内圆，内部没有其他点，并且圆心到圆周上三个点距离相同，并且最近。根据 Voronoi 图的定义，一定是这三个区域的交。

Corollary 5 *一个点集凸包上的点数 $2n-2$ 减去内圆的个数。*

对于一个反演后的内圆，因为所有点都在这个圆的同一侧 (内部或者外部)，所以反演前还是所有的点也一定在这个圆的同一侧。也就是说反演后的内圆一定对应着反演前的内圆或者外圆。对于一个反演中心，如果在某个内圆之内，那么这个内圆会变成外圆，否则这个内圆依旧是内圆。如果在某个外圆之内，那么外圆变成内圆，否则这个外圆依旧是外圆。

于是对于每个外圆和内圆，可以根据期望的线性性统计对答案的贡献。对于一个内圆，如果反演中心在这个圆外部部，那么对答案贡献一个 -1。对于一个外圆，如果反演中心在这个圆内部，那么对答案贡献一个 -1。

3.2 Implementation

先来讲一下这个内圆和外圆的求法，内圆可以通过最近 Voronoi 图解决，外圆其实和最远点 Voronoi 图对应。但是直接求 Voronoi 图比较难处理一些边界情况，即三点共线四点共圆的情况。

我们在来考虑一下这些边界情况以及介绍一种比较好的处理方法。

首先如果所有点都共圆，那么无法区分外圆和内圆，那么这种情况下答案显然是 n，因为反演之后这些点还是共圆的，共线的概率为 0。

其次还要一些情况内圆或者外圆边界上有多个点，或者有些点在 Voronoi 图中对应的区域为无限大，所以我们采取一个更加成熟的方法去刻画这些圆。

我们把点 (x,y) 映射到一个抛物面 $z=x^2+y^2$ 上，即点 (x,y) 变为 (x,y,x^2+y^2)。那么对于平面上一个圆 $x^2+y^2+ax+by+c=0$，对应着一个三维空间的平面 $z+ax+by+c=0$。所有点都在圆的同一侧，那么对应着所有点在这个三维平面的同一侧。也就是说这些内圆或者外圆，对应着三维空间中的一个平面，平面上有至少三个点，并且所有的点在这个平面的同一侧，这样的面也就是三维凸包上的面。如果某个面上有 n_c 点，那么相当于内圆或者外圆的边界上有 n_c 个点。我们假想对这些点做一个微小的扰动，这 n_c 个点就形成了 n_c-2 个小三角形，也就是形成了 n_c-2 个面，所以在原问题上可以认为这个面出现了 n_c-2 次。

对于一个凸包上的面，可以知道它朝内凸包外部的法向量 (a,b,c)，那么面对应的圆为 $ax+by+c(x^2+y^2)\le d$。如果 $c>0$，那么这个圆为外圆，如果 $c<0$ 那么这个圆为内圆。并且反演中心如果在这个不等式定义的区域内部，会对答案有 -1 的贡献，所以这个时候我们求出这个圆和矩形的面积交，然后统计入答案即可。如果 $c=0$，也就是退化成一条线的情况，同样还是反演中心如果在这个不等式定义的区域内部就会对答案有贡献，只需要计算半平面和矩形的交。

所以这个题的算法就是首先把所有的点映射到抛物面上，然后求出三维凸包，然后对于每个面对应的平面上的区域分别计算即可，时间复杂度为 $O(n^2)$，当然三维凸包这个可以用 $O(n\log n)$ 的做法求，然后也可以使用 $O(n\log n)$ 最近点和最远点 Voronoi 图算法解决，但是不需要。

3.3 Precision Issues

在标程的实现中，三维凸包用卷包裹算法实现。在卷包裹算法的实现中无法处理共面的情况。通常的做法是给每个点一个随机的扰动，但随机扰动可能会让凸包上的点变少，比如一个三棱锥表面的点可能随机扰动后可能进入三棱锥内部。所以我们需要对于每个面重新计算一下有多少个点在这个上面。在这个题里面由于特殊性不存在一个点抖进凸包里面的情况。因为如果所有点共面，那么就对应着二维平面上共线或者共圆的情况，那么映射到那个平面上所有的点形成一个凸多边形，也就不存在一个点在别的点形成的三角形内部从而被抖进内部的情况。

在这个题中给每个点一开始加 10^{-7} 级别的随机扰动，可以通过所有的数据。但是要完全避免精度问题，可以这么做，还是给每个点一个随机的误差，但是我们讲每个点显式地表示成 $p+q\cdot\varepsilon$，其中 ε 可以认为是一个无穷小量，在运算的时候首先计算 p 部分的混合积，如果相同再考虑一阶无穷小量。这么做在前面求凸包的部分可以用全整数解决，但是在混合积中，数值大小可能达到 $|R|^4$ 的级别，所以得使用 int128 或者高精度解决。

最后在求圆交和矩形交的时候，圆心的坐标和半径会达到 $|R|^2$ 的级别，所以我们这里要求矩形的面积足够大来减少这部分计算带来的误差。