IOI2013中国国家候选队作业
解题报告

湖南师大附中彭天翼

April 5, 2013

Contents

1 WinterCamp 2013 平面图

1.1 Description

题意：平面中有n个点和m条直线段，将平面分成了若干个有限区域和一个无限区域。

保证所有的点是联通的。每条直线段有一个权值。现在给出q个询问，每次询问两个点A(x1,y1),B(x2,y2)。请用一条曲线将它们连接起来，使得这条曲线通过的直线段的权值中最大值尽可能小。如果询问点位于无限区域上，请输出-1。

数据范围：$n, m \leq 10^5, q \leq 10^5, 0 \leq x, y \leq 10^7$
难度：★★★★

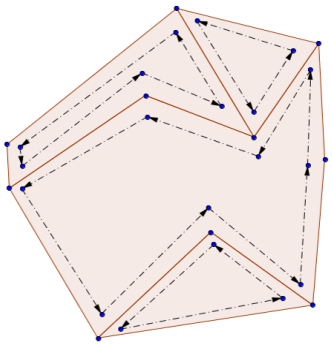
1.2 Analysis

算法：平面图的对偶图+点定位+ 最小生成树
分析：**步骤1：平面图的对偶图**
平面图的域：平面图的有限域为顶点和边所围成的简单多边形。如果两个域之间有公共边，则称它们相邻。

对偶图定义：对偶图中的每个点，与平面图中的域一一对应。对偶图中的两个点之间有边，当且仅当它们所对应的域相邻。

如何尽快得求出平面图的对偶图？下面介绍一种O(nlogn)的算法。

我们将平面图的边拆成两条方向相反的边，定义为反向边。从任意一条没有走过的边出发，每当我们到达一个顶点时，拐向顺时钟方向夹角最小的那条边，继续前进，直到回到初始边。这样我们就能得到一个域，标记下每条边所属的域。如下图所示：



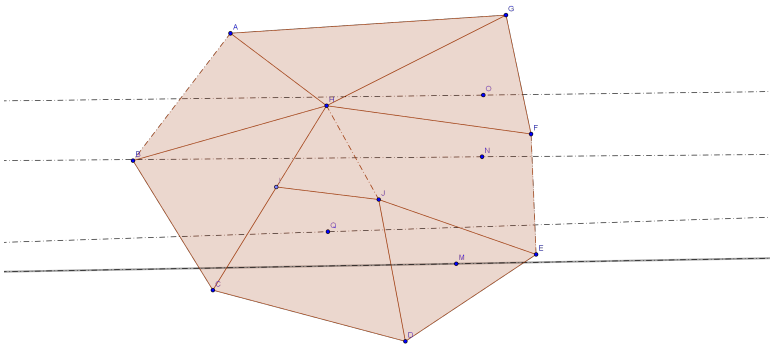
最后我们将每条边的反向边和正向边所对应的域建边，就得到了对偶图。怎样才能得到一条边沿顺时钟方向夹角最小的另一条边？我们只需要对每个顶点所发出去的边进行极角排序即可。所以复杂度为O(nlogn)。

步骤2：点定位

接下来我们需要知道每一个询问点它们属于哪个域。钱桥同学在冬令营时提到过的《梯形剖分》[1]可以优秀得解决这个问题。在这里介绍一种离线的O(nlogn)做法。

首先假设所有点的y坐标值互不相同（可以通过剪切变换使其满足该条件）。我们将所有的询问点和顶点一起按y坐标排序。通过扫描线从下往上扫描，用set维护当前的边。每当扫到一个顶点时，它向下的连边将从set中删除，向上的连边将加入set。set按从左至右的顺序维护每条边的关系。

可以发现，set中相邻的两条边中间夹着的就是一个唯一的域。所以当我们扫描到询问点时，直接查询此时它在set中的位置，便可以得知它所属于的域。每条边插入set 一次，从set中删除一次，所以复杂度为O(nlogn)。



步骤3：最小生成树

至此，我们已经将平面图问题转换成了我们所熟知的图论模型：在一张带权无向图中，每次询问两个点，请找出一条连接它们的路径，使得该路径上最大权值最小。对于这个问题，我们只需要求出该图的最小生成树，每次查询两个点的路径上最大权值的边即可。倍增算法和tarjian离线算法皆可完成。复杂度O(nlogn)或O(n)。

总结： 为了解决本题，我们分三个步骤化整为零，各个击破。这也提示给了我们解决复杂问题的思路：将复杂问题分解成很多我们熟知的简单问题，再逐一解决。

[1]NOI 官网下载地址

2 WinterCamp 2013 糖果公园

2.1 Description

题意：糖果公园是一棵有n个节点的树。每个节点上都放着一种糖果，用c[u]表示。共有m种糖果。游客第i次尝到第j种糖果，得到的喜悦值将是wi*vj。现在有q次操作，操作分为以下两种：

1. 询问：游客从u走到v，每经过一个点都品尝该点上的糖果，问喜悦值之和是多少？

2. 修改：将u点的糖果改为第x种

数据范围： $n, m, q \leq 10^5$
难度： ★★★

2.2 Analysis

算法：分块
分析：1、没有修改且树为一条链
性质：设当前从l走到r的答案为ans，第i 种糖果出现的次数为cnt[i]，则从l走到r+1的答案为：$ans + w[cnt[c[r+1]]+1] * v[c[r+1]]$。

这个性质提示我们可以在O(1)的时间内维护移动一步的答案。于是我们可以得到O(n*q)的方法：每次询问从l出发，走向r，同时维护好答案。

这个方法在求出答案的同时，还求出了一些不必要的值，所以我们可以进一步思考优化的空间。

考虑分块的做法：将链分成$\sqrt{n}$块，每个块的长度为$\sqrt{n}$。定义s[i][r] 为第i种颜色在前r个块中出现的次数，f[l][r]为从第l 个块的开端到第r个块的结尾的答案。它们可以在$O(n\sqrt{n})$时间内预处理出来。

查询时，设我们查询L到R的答案，L在第l 个块中，R在第r个块中。则我们在f[l+1][r-1]的基础上往左右移动，便可以在$O(\sqrt{n})$ 的时间内得到最终的答案。于是整个算法的复杂度为$O(n\sqrt{n})$。

2、带修改且树为一条链
如果加入修改之后呢？似乎无法在原有的方法上做出简单的扩展。我们考虑对操作也进行分块。将q个操作分为$q^{\frac{1}{3}}$个块，每个块的长度为$q^{\frac{2}{3}}$。将n分成$n^{\frac{1}{3}}$个块，每个块的长度为$n^{\frac{2}{3}}$。

此时对原序列的预处理时间复杂度变为$O(n^{\frac{4}{3}})$。我们每隔$q^{\frac{2}{3}}$个操作进行重做，总的预处理复杂度为$O(n^{\frac{5}{3}})$。每次询问时，我们在原块中进行移动，同时要考虑操作块内被修改的点对答案的影响，所以所有询问的总复杂度也为$O(n^{\frac{5}{3}})$，整个算法的复杂度同样为$O(n^{\frac{5}{3}})$。

3、带修改的树形结构

从序列变成树，我们需要再下一些功夫。首先将树的括号序列dfs出来。在树的括号序列中，每个节点出现了两次，分别是在dfs时入栈的出栈时刻记录的，记入栈点为L[u]，出栈点为R[u]。当我们查询树上的u-v的链时，改为查询括号序列中的R[u]-L[v]（设$R[u] < L[v]$），并且定义如果一个点在该序列中出现两次，则等价于没有出现。这样树上的每一条链就对应着括号序列的一个子串，注意该链的lca需要单独考虑。

现在我们要解决一个被加强了的序列问题：如果一个点出现两次，等价于没有出现。在这样一个新增的条件下，我们仍然能够在O(1) 的时间内维护好新增的答案，所以整个算法仍然可以解决该问题。总复杂度为$O(n^{\frac{5}{3}})$。

提示与补充：本问题有另外一种解法：莫队算法。两种算法都利用了题目的一个关键性质：在O(1)的时间维护好移动一格的答案。与本问题相似的题目还有：2010年国家集训队命题答辩《小Z的袜子》[2]。

[2]题目网址

3 WinterCamp 2013 小Q运动季

3.1 Description

题意：给定m个关于n个未知数的线性同余方程。形如下式：

$$\begin{aligned}
a_{11} * x1 + a_{12} * x_2 + a_{13} * x_3 + \cdots + a_{1n} * x_n &\equiv r_1 \pmod{p_1} \\
a_{21} * x1 + a_{22} * x_2 + a_{23} * x_3 + \cdots + a_{2n} * x_n &\equiv r_2 \pmod{p_2} \\
a_{i1} * x1 + a_{i2} * x_2 + a_{i3} * x_3 + \cdots + a_{in} * x_n &\equiv r_i \pmod{p_i} \\
a_{m1} * x1 + a_{m2} * x_2 + a_{m3} * x_3 + \cdots + a_{mn} * x_n &\equiv r_m \pmod{p_m}
\end{aligned}$$

请求一组n个未知数的解，使得满足的线性同余方程尽可能地多。

数据范围：提交答案
难度：★★★

3.2 Analysis

算法：解线性同余方程组

Case 1: 有5000个方程，只有一个未知数，取模的数相同都为$p = 1048576$。

分析：直接在[0,p-1]中枚举未知数的值，再计算满足的方程个数即可，可以在一分钟内出解。

Case 2: 50个方程，一个未知数，取模的数两两互质

分析：设方程为$ax \equiv b \pmod{c}$，方程有解等价于gcd(a, b, c) = gcd(a, c)。所以我们先将不可能有解的方程排除掉，剩下的再用中国剩余定理进行合并求解。需要高精度。

Case 3: 300个方程，300个未知数，取模的数都相同且为1e9+7。

分析：直接用高斯消元求解同余方程组，发现最后能满足300组方程。

Case 4: 20个未知数，845个方程。有20对方程，每对方程矛盾，每个方程重复20次。剩下45 个方程为普通情形。

分析：从每对方程中选出一个方程，能够恰好组成上三角矩阵，从而很快得到方程的解。还剩下45个方程。枚举从每对方程中取哪一个方程，再去检验45个方程中满足了多少个。能够在一分钟内出解。

Case 5: 100个未知数，100个方程。取模的数相同。

分析：尝试用高斯消元解同余方程，发现可以全部满足。

Case 6: 50个方程，50个未知数。取模的数都为1次质数的积。

分析：将取模的数分解为质数的积。然后分别考虑每个质数下的线性同余方程组。用高斯消元分别解之。最后再用中国剩余定理进行合并，可以发现所有方程都能够被满足。

Case 7: 50个未知数，200个方程。其中50个方程完全相同。另外50个未知数3个方程中选1 个。取模的数为176400。

分析：将每个未知数看成物品，每个物品有三种体积，我们需要在每个物品中选择一个体积，再线性组合起来，使得其满足那50个完全相同的方程。背包即可。要记录下中途转移的过程，

Case 8: 50个未知数，50个方程。和Case 6相同。

分析：解法同Case 6。解方程后发现会出现1组不满足，删除该组即可。

Case 9: 1个未知数，100个方程，系数为1。为三个方程重复了若干次。

分析：统计哪个方程出现的次数最多就行。

Case 10: 50个未知数，90个方程。和Case 6相同。

分析：用模拟退火等方法进行搜索，能够期望得到较高的分数。

提示与补充：提交答案类型的题目一般来说想得到很高的分数是相当困难的。我大多数情况下是先观察几个比较容易的点，然后剩下的点使用非完美算法（比如盲人爬山）去跑个10min左右，这样一般能得到不错的分数。

4 Codechef MileStones

4.1 Description

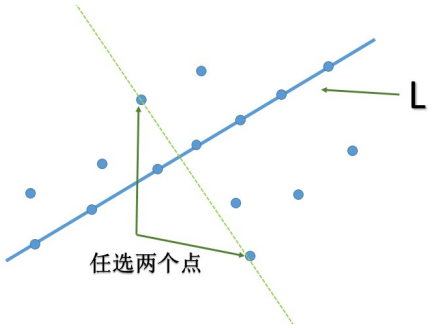
题意：平面上有N个互不重合的点，现在已知有不超过7条直线能够全部覆盖它们，请找到一条直线使得经过的点的数目最多，输出这个最大数目即可。

数据范围：$T \leq 30$（T为数据组数）,$N \leq 10000$
难度：★★

4.2 Analysis

算法：利用题目的特殊性
分析：**1、概率算法**
性质1：设覆盖点数最多的直线（即我们所求的答案）为L，覆盖的点数为ans。则由于所有的点能够被不超过7条直线覆盖，所以ans至少覆盖了N/7个点。

性质2：我们从N个点中任取两个点，它们所形成的的直线与L不重合的最坏概率是：$1-(\frac{1}{7})^2=\frac{48}{49}$。



有了上述两个性质，我们尝试从N个点中每次任取两个点，然后计算这两个点所形成的直线所过的点的个数，用来更新答案。

设我们取了k次，则仍然没有得到L的概率是$p=(\frac{48}{49})^k$。取k为1000，则$p \approx 1e-9$。也就是说取了1000次还没有取到最优解的概率比中百万大奖的概率还要低得多。如果你没有天天梦想着中百万大奖，那么你应该相信这个算法能够通过全部测试数据。

这个算法的复杂度为O(T*n*k)，在实际提交过程中，k可以设的更小一些（你还可以多次提交的哦）。

2、确定性算法

如果你是一个想到算法有彗星撞地球般的概率会出错便寝食难安的人，下面这个算法将会很合你的胃口。

性质1：不是覆盖所有点的7条直线中的直线最多只能覆盖7个点。因为8个点的话，该直线就会和前面某条直线重合了。

我们枚举每一个暂时没有打上标记的点，以它为中心排极角序。然后找到一条直线，使得它过该中心并且经过的点数最多。如果该直线经过了超过7个点，则我们找到了最后7条直线中的一条，将这条直线上的点打上标记（不需要再枚举它们了）。

这样我们最多枚举不超过50个点就能把L找出来，因为如果最后的某条直线没有覆盖超过7个点，则我们最坏需要在这条直线上的点枚举7次。而只有7条直线，所有最坏只需要枚举50次。

复杂度是O(T*50*n*logn)。

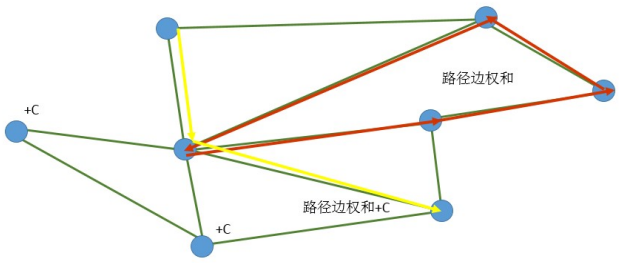
提示与补充：本题的关键性质在于只需要7条直线就能经过所有的点，依据这个性质，我们得出了两种能够解决本问题的算法。关于非确定性算法的经典题目还有:《Matrix Multiplication》。

5 Codechef Annual Parade

5.1 Description

题意：在一个n个点m条带**正**权边的有向图中，你需要安排若干条路径(si,ti)，路径本身不能是自环，而你的花费将由三部分构成：

1. 路径si− >ti上所经过的边的边权和，一条边被经过多次，就将被计算多次。
2. 如果si≠ti，那么该条路径额外增加C的费用。
3. 如果某个点没有被任何一条路径经过，也将额外增加C的费用。



现在将有Q个询问，每个询问是一个给定的C,你需要回答此时的最小花费。

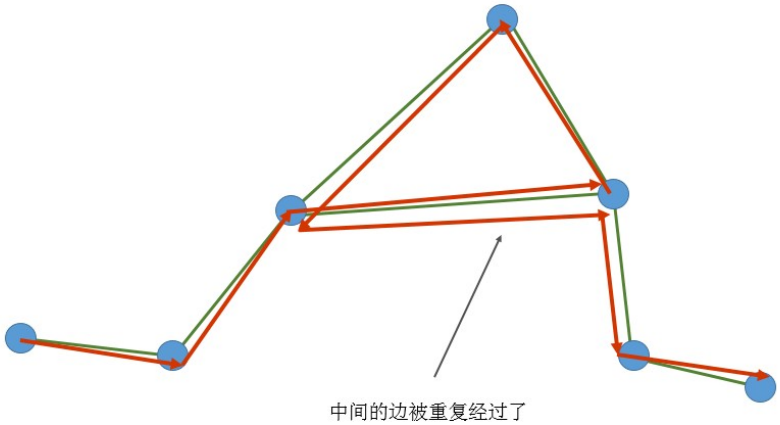
数据范围：$N \leq 250, M \leq 30000, Q \leq 10000, C \leq 10000, vij \leq 10000$
难度：★★★

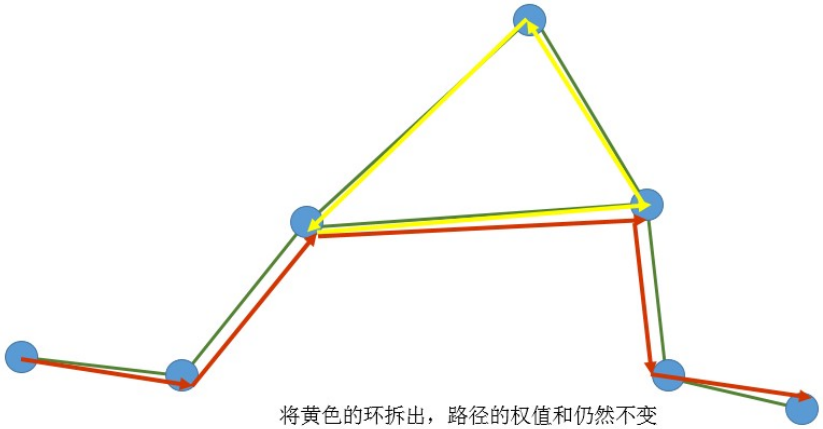
5.2 Analysis

算法：费用流
分析：性质1：最优方案的路径由简单路径和简单环构成。如果方案中出现了复杂的路径（某条边被经过了多次），则我们可以将该路径上的环拆出来，使得总花费不变。

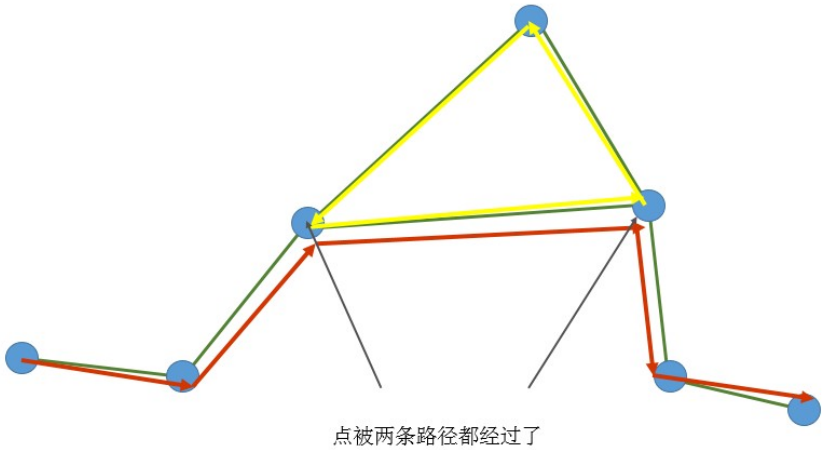
性质2：我们用floyd求出该图两两之间的最短路$d[i][j]$，用$d[i][j]$作为i到j的新边权。这样修改的目的是使得最优方案中的路径互不共点。这是因为如果存在两条路径共一个顶点u，则我们可以让其中某一条路径将u点去掉，改为u的前驱到u的后继，边权为它们之间的最短路径。

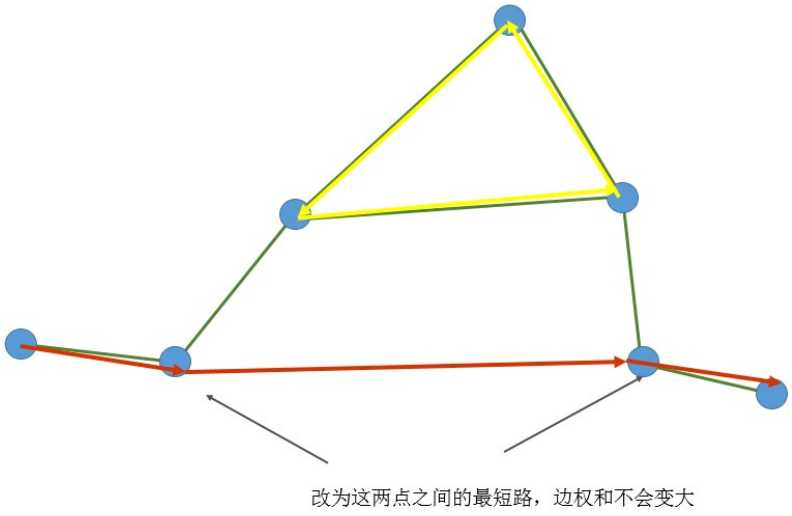
性质1：





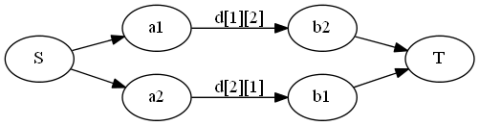
性质2：





至此，该问题的(si, ti)变为简单环或者简单路径，并且所有路径互不相交。回忆最小路径覆盖模型，我们建立如下的最小边权路径覆盖模型（最小费用最大流）：

最左边为S，连向左边n个点，右边n个点，连向T。如果i和j之间有边，则我们将左边的第i个点连向右边的第j个点(i≠j)，花费为边权。所有的边的容量均为1。



定义初始时所有的点都是独立的，花费为N*C。让我们思考：在这个模型中，每增广1的流量，意味着什么？

每增广1的流量，意味着多匹配了一条边，我们将这条边在原图中进行标记。接下来我们会发现这条边的出现导致了一些惊世骇俗的事情发生：

- 连接了两条原本不相交的路径，这时候费用将会减少C.
- 连接了一条路径的首尾，构成了一个环，费用同样减少C!

也就是说，增加1的流量一定会使得费用减少C，设本次增广的费用为V，则实际花费为V-C。由于费用流增广算法的性质，所以V是递增的，于是我们增广到$V > C$时便可以停止增广。

注意我们还有多组询问，所以我们可以将V 数组先预处理出来，每次询问时在V数组中二分出第一个$> C$的位置，便能在O(1)时间内求出答案。

提示与补充：奇妙的网络流。