Solution

北京市第八十中学　罗剑桥

2013 年 3 月 16 日

Contents

1 Theresa

1.1 题意简述

在空间直角坐标系中，你要维护一个点的集合。最初集合中有N个点，然后依次进行Q次操作。操作有三种类型：

- **ADD** $x\ y\ z$
  在当前点集中添加一个新的坐标为(x, y, z)的点。
- **QUERY** $x\ y\ z\ r$
  询问位于正方体$(x, y, z) - (x + r, y + r, z + r)$内部(含边界)的点的数目。
- **CANCEL**
  撤销最近一次的ADD操作。

允许使用离线算法。

1.2 数据范围

10%的数据: $1 \leq N, Q \leq 10^3$.
此外10%的数据: $0 \leq x, y, z, r \leq 100$.
此外30%的数据: 没有ADD 操作和$CANCEL$ 操作。
此外20%的数据: 没有$CANCEL$ 操作。
以上70%的数据: $1 \leq N, Q \leq 50,000$.
100%的数据: $1 \leq N + Q \leq 10^5$, $0 \leq x, y, z, r \leq 10^7$. x, y, z是非负整数，r是正整数。

1.3 算法分析

【算法一】

用一个栈记录当前的点集，每次询问时枚举所有的点，依次判断是否在正方体中。
时间复杂度： $O(N * Q)$
期望得分： 10分

【算法二】

我们发现，$CANCEL$ 操作等价于添加一个权值为−1的点。本质上，只有ADD 和$QUERY$ 两种操作。对于10%的数据，每个点的坐标都比较小，因此可以使用三维的树状数组维护点集。询问时，我们使用容斥原理将正方体拆成8个以原点为顶点的正方体的答案的代数和与差的形式，而每个这样的正方体可以直接在三维树状数组中查询内部的点的权值和。
时间复杂度： $O((N + Q) * \log^3 101)$
期望得分： 20分(结合算法一)

【算法三】

注意到30%的数据没有ADD 操作和$CANCEL$ 操作，因此我们可以使用扫描线的方法对所有的$QUERY$操作一起处理。

具体来说，我们将每个$QUERY$操作中的正方体拆成z坐标上的两个正方形面，或者说是两个z坐标从0开始的正方体的答案的差。将所有点按z坐标排序，然后从小到大依次处理每个z坐标的事件。我们可以使用树状数组套平衡树存储每个时刻的(z坐标不大于当前值的)所有点。树状数组以x坐标为关键字，平衡树以y坐标维护树状数组中每个结点对应的x坐标在一定范围内的所有点。

这样的话，遇到加入点的事件，我们就直接更新这个数据结构，时间复杂度为$O(\log^2 n)$。遇到查询一个正方形面内的点权和的事件，由于没有了z坐标的限制，我们可以使用树套树直接查询满足两维限制的点的数目。当然，这里依然是采用做差的方法，即$\{x|l \le x \le r\} = \{x|0 \le x \le r\} - \{x|0 \le x \le l-1\}$。

时间复杂度： $O((N+Q) * \log^2 N)$

期望得分： 30分(结合算法一和算法二，期望得分50分)

【算法四】

实际上，我们可以直接采用三维的线段树代替算法二中的三维树状数组，这样就可以维护坐标范围很大的数据了。为了加快速度，我们可以先将所有点的坐标离散化。可惜由于常数太大，这个算法的期望得分并不乐观。

时间复杂度： $O((N+Q) * \log^3(N+Q))$

期望得分： $20 \sim 80$分

【算法五】

有一篇论文[1]介绍了一类专门针对数据结构问题的分治算法[2]，可以神奇地将在线问题转化为离线问题。

我们将所有操作一分为二，那么后半部分的修改不会影响前一半的查询的结果，而前一半的修改对后一半的查询的影响是与后一半的修改操作无关的。

因此，我们只需要分别递归地解决两个部分的规模缩小一半的子问题，然后高效地处理前一半的修改操作对后一半的询问操作的影响。这里就可以采用算法三中介绍的扫描线与树套树的算法。

时间复杂度： $O((N+Q) * \log^3(N+Q))$

期望得分： 100分

[1]陈丹琦，《从"$Cash$"谈一类分治算法的应用》，IOI2008国家集训队作业

[2]这种分治算法也是顾昱洲同学在2013年全国冬令营的讲课内容

2 Alice

2.1 题意简述

给出一个二分图，所有顶点分为X集合和Y集合。其中X集合有N个点，Y集合有M个点($N \leq M$)。已知X集合和Y集合中的哪些点之间有边。而Y集合中的点又分为两类，分别是A类点和B类点。每个Y集合中的点都有一定的权值。

你要从Y集合中选出恰好N个点，使得它们与X集合的点之间存在完备匹配。定义一个方案的代价为被选出的A类点的权值之和加上没有被选出的B类点的权值之和。求最小代价的方案。若方案不唯一，输出将选出的点按下标排序后字典序最小的方案。

每个测试点包含多组测试数据。

2.2 数据范围

20%的数据: $1 \leq N, M \leq 8$.

40%的数据: $1 \leq N, M \leq 16$.

此外20%的数据: Y集合的顶点的权值均为0。

70%的数据: $1 \leq N \leq 50$, $1 \leq M \leq 100$.

100%的数据: $1 \leq N \leq 400$, $1 \leq M \leq 500$, 权值是不超过10^4的非负整数，每个测试点不超过10组测试数据。

2.3 算法分析

【算法一】

我们依次枚举每个X集合中的点与Y集合中的哪个点匹配，每次得到一个合法方案就拿去更新当前的字典序最小的最优解。

时间复杂度： $O(N^2 * N!)$

期望得分： 20分

【算法二】

我们采用状态压缩动态规划的方法，设状态$F(i, S)$表示X集合中的前i个点与S集合中的点能否匹配。那么，最多会有2^M个不同的状态，而转移的复杂度是$O(M)$的。

时间复杂度： $O(M * 2^M)$

期望得分： 40分

【算法三】

我们尝试建立费用流模型解决这道题目。

添加源点S和汇点T。从源点S向X集合中的每个点x_i添加一条容量等于1、费用等于0的边。从Y集合中的每个点y_i向汇点T添加一条容量等于s、费用等于0的边。若原图中X集合中

的点x_i与Y集合中的点y_i之间有边，则添加一条从x_i向y_i的容量等于1、费用等于选择点y_i的代价的边。

为了计算不选某个点的费用，我们添加一个新点z。从源点S向z添加一条容量等于M、费用等于0的边。从z向Y集合中的每个点y_i添加一条容量等于1、费用等于不选择点y_i的代价的边。

这样建图以后，我们进行最小费用最大流算法即可得到最优解。为了得到最小字典序的解，我们按下标从小到大的顺序依次尝试能否选择点y_i。可以每次重新计算费用流，不过比较慢。也可以采用退流的方法，时间复杂度相当于只做一次费用流。

时间复杂度：　$O(mincostmaxflow())$或$O(M*mincostmaxflow())$

期望得分：　40 ~ 70分

【算法四】

算法三并没有利用题目中计算方案代价的方式的特殊性。我们发现，可以提前将B类点的权值和累加到答案中，然后将它们的权值取反，就成了A类点。这样就没有两类点的区别了。

此时，如果我们忽略最小字典序的要求的话，问题就转化为一个经典问题。即，Y集合中的每个点有权值，求选出的点的权值之和最小的完备匹配。一个正确的贪心算法是，将Y集合的所有点按权值从小到大的顺序做匈牙利算法。算法的正确性请读者自行思考。

至于字典序的问题，我们可以按下标从小到大的顺序依次尝试每个点，判断选择它以后能否得到最优解。如果重新计算就会太慢，必须采用退流的方法解决。

时间复杂度：　$O(N*M^2)$[3]

期望得分：　70 ~ 100分

【算法五】

如果我们对算法四中求最小字典序的算法进行分析，会发现它和另一个更加贪心的算法是等价的。即，将所有点按权值为第一关键字、下标为第二关键字排序，然后按此顺序进行匈牙利算法即可。

这个算法十分有意思，读者可以去尝试证明它的正确性[4]。

时间复杂度：　$O(N*M^2)$

期望得分：　100分

[3]虽然时间复杂度看起来比较高，但是匈牙利算法以及解决网络流模型的一类最短增广路算法的时间效率都是很高的，对于本题的数据规模可以在时限内轻松出解。

[4] 实际上，由此算法拓展我们可以发现此问题的一个和最小生成树问题相似的性质，即任意的最优方案中每个权值对应的点的数目是一定的。

3 Catherine

3.1 题意简述

给出一个N个点，M条边的无向图。你要选出一些顶点，使得这些顶点与它们之间的边形成的子图中每个点的度数不超过1。求最多选出多少个顶点。

每个测试点包含多组测试数据。

3.2 数据范围

10%的数据: $1 \leq N \leq 10$.

30%的数据: $1 \leq N \leq 20$.

50%的数据: $1 \leq N \leq 30$.

100%的数据: $1 \leq N \leq 40$, $0 \leq M \leq 780$, 每个测试点不超过50组测试数据。

3.3 算法分析

首先，我们发现这道题目和图论中经典的最大独立集问题很相似，并且数据规模比较小，因此猜测这是一个$NP-Complete(NPC)$问题。事实上，这的确是个NPC问题。下面给出证明：

- **这个问题是属于NP的。**
  显然，若给出一组解(即一个顶点集合)，可以在$O(N+M)$的时间复杂度内验证是否符合题目要求。

- **最大独立集问题可以在多项式时间内规约到这个问题。**
  设最大独立集问题对应的原图为图G_0。

  我们构造一个新图G_1。原图G_0的每个点v对应新图G_1中的两个点v_1和v_2。原图G_0的每条边(u,v)对应新图G_1中的边(u_1,v_1)。除此以外，我们附加$|V|$条边(v_1,v_2)，即在原图G_0的每个点对应的两个点之间添加一条边。接着，我们来证明原图G_0存在规模不小于K的独立集当且仅当新图G_1存在规模不小于$|V|+K$的$Catherine$问题的解。

  首先，我们说明任何无向图中必定存在一个$Catherine$问题的最优解使得所有度数等于1的顶点都被选出。这是因为，若某个度数等于1的顶点x没有出现在最优解中，那么与x有边相连的唯一的点y必定出现在最优解中，否则添加x以后答案更优。于是，我们可以去掉最优解中的点y，而加入点x，新方案依然满足条件而且不会更差。

  下面依次是充分性和必要性的证明：

  - **充分性**
    假设新图G_1存在规模不小于$|V|+K$的$Catherine$问题的解。那么一定存在某个最优解满足顶点集合$\{v_2\}$中的所有点都被选出。于是剩余的K个点之间必定没有连边。那么我们删去这$|V|$个点，剩下的点集就是一个规模不小于K的最大独立集。

– **必要性**

假设原图G_0存在规模不小于K的最大独立集。那么在新图G_1中，我们保留原图最大独立集中的所有点以及顶点集合$\{v_2\}$中的所有点。则每个点的度数不会大于1，于是得到了一个规模不小于$|V|+K$的$Catherine$问题的解。

这样，我们就证明了这个$Catherine$问题是NPC问题。鉴于目前此类问题尚未发现多项式解法，我们考虑采用搜索算法解决这道题目。

【算法一】

我们依次枚举每个点是否选取，然后判断得到的方案是否合法。

时间复杂度: $O((N+M)*2^N)$

期望得分: 10分

【算法二】

我们在枚举每个点是否选取时，检查一下它与之前已经选取的所有顶点之间的连边情况，以判断是否还能选择它。

期望得分: 30分

【算法三】

在枚举到某个点的时候，发现如果它与以后的点之间都没有连边则不能不选。

记录当前的最优解以便进行最优性剪枝。

将所有点按度数从大到小的顺序搜索。

期望得分: $50\sim80$分

【算法四】

在算法三的基础上，结合$Catherine$问题的特殊性，在搜索时提前选出所有仍然可以选择且度数不大于1的点, 并采用位运算等手段优化判断的时间效率。

期望得分: $70\sim100$分