浅析非完美算法在信息学竞赛中的应用

湖南省长沙市长郡中学 胡伟栋

【目录】

【摘要】

非完美算法就是用算法正确性的少量损失来换取时间、空间效率以及编程复杂度的算法。本文介绍了非完美算法的几种重要方法及非完美算法的优缺点，从中，可以看出：并不是完全正确的算法就一定好过非完美算法，有可能因为非完美算法的不完全性，反而使不正确的算法在很多方面比正确算法表现得更好。

【关键字】

非完美算法 随机化贪心法 抽样测试法 部分忽略法

【正文】

一、引言

在平时，我们研究的算法基本都是完全正确的算法，我们所追求的，也是尽量使我们的算法正确。但在实际应用中，有很多情况都不会对数据进行天衣无缝的正确处理。如：图片、音频、视频的压缩存储，很多压缩率比较高的方法都是有损压缩；很多密码验证的方法都是采用多对一的运算方法，通过验证的不代表密码真正正确；搜索引擎所提供的搜索，并不能将所有满足条件的都找到……显然，我们不会因为它们的不完美而不使用它们，它们的存在，有着它们的实际意义：图片、音频、视频的压缩存储，如果不损失一些，不可能将文件压缩得很小；密码验证虽是多对一，但找到两个所对的是同一个何尝容易；搜索引擎虽不能搜索到100%的结果，但其方便、快捷是无以伦比的……

那么，在信息学竞赛中是否也可以用非完美算法解题呢？

本文试图介绍一些比较有用的常见非完美算法，并希望读者能从此得到一些启发。

在开始此文前，希望读者能认同一点：在信息学乃至整个计算机科学领域，

不一定绝对正确的算法就是最好的算法，有可能一个在绝大多数情况下正确的算法(非完美算法)比一个完全正确的算法更有前途。或者，由于它没有完全正确的算法考虑得全面，而使得它的空间或时间使用得较少；或者，由于它没有正确的算法那么严谨，使得编程实现时较容易；或者，由于所用的知识没有正确算法那么深奥，使得它更容易被接受。

二、非完美算法的一些基本方法

1.1 随机化贪心法

随机化贪心是目前用得较广泛的一种非完美算法。特别是对求较优解的题目，这种方法可以说是首选。

随机贪心，就是用随机与贪心结合，在算法的每一步，都尽量使决策取得优，但不一定是最优决策。通过多次运行，使得算法能取得一个较优的解。有时，由于决策的优劣判断较复杂，直接用多次随机也能得到较优的结果。

例：传染病控制

□ 题目大意

给出一棵传染病传播树，其中根结点是被感染结点。每个时刻，被感染结点都会将传染病传播到它的子结点。但是，如果某时刻 t 切段 A 与其父结点 B 之间的边，则时刻 t 以后，传染病不会从 B 传染给 A(即 A 不会患病)。

每个时刻，只能切断一条传播途径。问每个时刻怎样切段传播途径，使最少的人被感染。

□ 说明

本题的标准算法是搜索。关于如何用搜索解决此题，请参见附件《传染病控制解题报告》，此解题报告由周戈林同学提供。

随机化贪心在周咏基前辈的《论随机化算法的原理与设计》中有写过，为了此文的完整性，这里将通过一个实例简要的说明。

题目来源：NOIP2003，原题请见附录。

□ 分析

显然，如果某个结点已被感染，则以后没有必要切断它与它的父结点之间的传播途径，因为这样和不切断没有区别(即切断已被感染的结点与其父结点之间的传播途径是无效的)；如果在一个时刻，切断 A 与 A 的父结点之间的传播途径是有效的，切断 B 与 B 的父结点之间的传播途径也是有效的，同时，B 是 A 的子孙结点，则切断 A 与 A 的父结点的传播途径优于切断 B 与 B 的父结点的传播途径。由于第 t 时刻被感染的显然只可能是前 t 层的。因此可得到，第 t 时刻切断的必然是第 t 层与第 t+1 层之间的传播途径。

本题可以用随机贪心来做。

根据经验，每次将一个结点数最多的子树从原树中切开，结果会比较好。但这样并不能适应所有情况，有时，选结点数第二多的或第三多的反而能使以后的最终结果要好一些。所以，可以多次贪心，每次都随机的选一个能切断的结点数较多的方案切(或者说每次使能切断的结点数多的方案数被切的几率大一些)，这样，虽然大多数情况下其结果都比贪心要差，但只要随机到一定的数量，这中间出现正确答案的概率就很大了。

上面加概率的随机贪心有一点难处理的就是怎样设置每种情况被选择的概率。其实，对于此题，只要将每种情况被选到的概率都设成同样大(即被选中的概率与其子结点数无关)就可以了。这样，这种方法就变成了纯粹的随机法，这种方法也能使此题得到满分。

小结

随机贪心是信息学竞赛中的一个重要工具，由于一般情况下它都是基于简单的贪心，所以往往编程复杂性会比较低，同时运行的时间复杂度也会比较低。随机多次是随机贪心的一个重要手段，通过多次运行，往往能使程序得到非常优的结果。

1.2 抽样测试法

抽样，即从统计总体中，任意抽出一部分单位作为样本，并以其结果推算总体的相应指标。

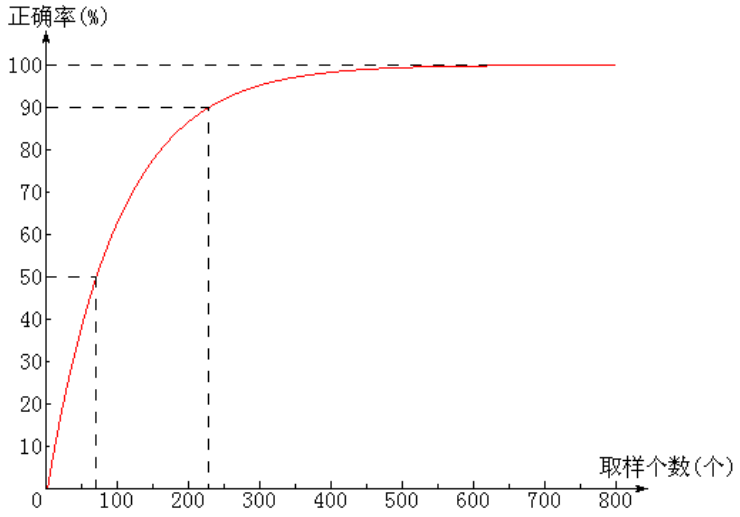
在某些题目中，题设会给出一系列需要测试的测试元(总体)S，若存在某一个测试元满足某个条件 A，则说 S 满足性质 P，若所有测试元都不满足条件 A，则说 S 不满足性质 P。

对于这个总体 S，判断它是否满足性质 P，通常要将 S 中所有的测试元全部列举出来才能知道。如果不列举出全部(哪怕只有一个没有列举到)且当前已测试的测试元都不满足条件 A，则不敢保证剩下的测试元不满足条件，也就不能断定 S 是否满足性质 P。

但是，若总体具有下面的性质：它或者不含满足条件的测试元，或者满足条件的测试元的个数比较多。这时，只要选取一些具有代表性的很少一部分测试元进行测试，就是保证结果基本正确。所谓有代表性，就是指取一些很特殊的测试元，同时取一些不特殊的测试元——就像问卷调查一样，从各种工作岗位、各种年龄阶段的人群中取得的问卷调查结果往往比在同一个工作岗位且年龄相仿的人群中取得的调查结果更让人信服。

下面，我们来看一个例子：一个总体 S 中含有 10000 个测试元，其中有 100 个测试元满足条件 A，现在，若从 S 中取出 100 个测试元进行测试，则检查出 S 满足性质 P 的概率约为：63.6%，若取 200 个测试元，则检查出的概率为 86.9%，若取 300 个，则检查出的概率可达 95.3%。

下图中给出了取的个数与正确率之间的关系(注意：这里只画了取 0 至 800 个时的图像，当取 800 个以上时，由于其正确率太接近 100%，其图像近似一条水平直线而没有再画出的意义)：



从图中还可以看出，只要抽样大约 70 个，就有 50%的正确率，抽样大约 230 个，就有 90%的正确率。

再看一个例子：如果总体 S 是{'**A**','**B**',...,'**Z**'}的所有子集，而条件 A 是同时含有' **A**'~'**G**'的子集。按照上面，根据上面一例，可以想到，如果取一定量的

子集测试，效果也会比较好，但好，对于总体 S，它存在一些特殊的子集——如空集ô 、全集{'**A**','**B**',...,'**Z**'}、只含一个元素的集合{'**A**'},{'**B**'}……这些都是我们认为比较特殊的集合，如果从这些特殊的集合中先取出几个测试，则可能在这些集合中就能找到满足条件 A 的，如此例中取全集显然就会满足条件。

下面看一个具体的实例：

例：Intuitionistic Logic(直觉主义逻辑)

□ 题目大意

给定一个有向无环图 $G=(V,E)$，在顶点 V 之间建立一些偏序关系：对于 x,y ô V，定义 $x\leq y$ 当且仅当 x 到 y 之间存在路径(可能长度为 0)。对于一个顶点集合 S，定义 **Max**(S)为 S 中所有最大元素的集合(即 **Max**(S)中的任意一个顶点都不可能通过一条或多条 E 中的边到达 S 中的其他顶点)，写成表达式的形式为：

Max(S)={x□S|不存在y□S,$x\neq y$,$x\leq y$}

令 β 为所有满足 Max(S)=S 的集合所组成的集合，即：

β={S|**Max**(S)=S}

令 **0**=**Max**(V)，**1**=□，显然 **0** 和 **1** 都属于 β

定义 β 中元素的一些操作：

P=>Q={xô Q|不存在y□P,$x\leq y$},

Pô Q=**Max**($P\cup Q$),

题目来源：ACM ICPC 2002-2003, Northeastern European Region, Northern Subregion，原题请见附件。

$P \oplus Q$=**Max**($\{x \in V$|存在$y \in P, z \in Q, x \le y, x \le z\}$),

$\neg P$=(P=>**0**),

$P \equiv Q$=((P=>Q) $\wedge$ (P=>Q))

问题：对于一个含有变量的表达式，问是否无论变量如何在β中取值，其运算结果都是**1**？

数据范围：其中$|V| \le 100$；$|E| \le 5000$；对于同一个图，要处理k个表示式，$k \le 20$；每个表达式长度不超过254个字符；$|\beta| \le 100$；$\sum_{1 \le j \le k} H^{v_j} \le 10^6$ (v_j是在第j个表达式中用到的变量个数)

分析

本题的最后一个数据范围$\sum_{1 \le j \le k} H^{v_j} \le 10^6$其实暗示了我们，此题可以用枚举来做，即枚举所有变量的所有可能取值。显然，如果要求完全正确，变量取值的总方案数是不可能减少的，即可能达到10^6，这时，就得设法减少计算的复杂度。显然，上面的基本运算都是$O(|V|)$或$O(|E|)$或更高的，而计算一个表达式可能要使用上百次基本运算，这样，尽管此题有5秒的时限，这样的枚举也是很难出解的。

由$|\beta| \le 100$，可以想到，将β中的每一个元素先用一个整数代替，并计算出对这些数进行上面的基本运算所得的值(当然也用整数表示)。对这些值列一个表，以后计算时就只要从表中查找结果了。这样，基本运算的复杂度就可以降到

$O(1)$，总的复杂度就可以降为 $O(\sum_{1\le j\le k} H^{v_j} * |B|)$。

此题还有一种解决方法，那就是抽样。将变量取 β 中一些比较特殊的值(如 **0**，**1**，……)看作特殊样品，其它的看作一般样品。抽取一些特殊样品和一些一般样品，对它们进行测试，如果对于一个式子，所有的样品都满足条件，则说明式子满足条件，否则说式子满足条件。实践证明，只要抽取不到 1000 个样品即可得到比较高的正确率。

当然，对于这类多测试数据的题目，要基本保证一个测试点对，每组测试数据的正确概率必须更高一些。如：对于有 20 个组测试数据的测试点，如果保证每组数据的正确概率为 95%，则整个测试点的正确概率只有 0.95^{20}，还不到 36%；如果保证每组数据的正确概率为 99%，整个测试点的正确概率才 82%；如果每组数据正确概率达到 99.9%，则整个测试点的正确概率就会有 98%。

这里特别要说明的是：一般情况下，抽样的数目应该尽量多(在保证不超时的前提下)，这样才能保证正确概率较大。

抽样法的实际运用：

在测试质数时，抽样法是一个非常有用的工具。下面给出一种质数判定方法：

对于待判定的整数 n。设 $n-1=d*2^s$(d 是奇数)。对于给定的基底 a，若

$a^d \equiv 1 \pmod{n}$

或存在 $0\le r<s$ 使

$$a^{d*2^r} \equiv -1 \pmod{n}$$

则称 n 为以 a 为底的强伪质数(Strong Pseudoprime)。利用二分法，可以在 $O(\log_2 n)$的时间内判定 n 是否为以 a 为底的强伪质数。

对于合数 c，以小于 c 的数为底，c 至多有 1/4 的机会为强伪质数(**Monier** 1980, **Rabin** 1980)。

根据这条，判断一个数 n 是否为质数，可以随机地抽取小于 n 的 k 个数为底。这样，正确的概率不小于 $1-4^{-k}$。显然，这已经非常优秀了，通常只要取几十组就可以保证基本正确。

如果不是随机抽样，而是抽样特殊情况——最小的几个质数，则：

如果只用 2 一个数进行测试，最小的强伪质数(反例)是 2047(所以一个数显然不够)；

如果用 2 和 3 两个数进行测试，最小的强伪质数大于 10^6；

如果用 2,3,5 进行测试，最小的强伪质数大于 2×10^7；

如果用 2,3,5,7 进行测试，最小的强伪质数大于 3×10^9，已经比 32 位带符号整数的最大值(Pascal 中的 Maxlongint)还大了。

可见，通常只要抽取 2,3,5,7 这几个固定的数进行测试就能保证测试的正确性了。

具体的最小强伪质数列表请见附录。

我们知道，最朴素的质数测试方法是用 2 至$\sqrt{n}$ 的数对 n 进行试除。这样，测试一个数的复杂度为 $O(n^{0.5})$，用上面的方法，每次测试只要 $O(\log_2 n)$的复杂度，由于只要测试很少的几次，所以，其总复杂度仍是 $O(\log_2 n)$的。可见，用抽样的方法对质数进行测试的算法明显优于朴素的质数测试法。

小结

从上面的例子可以看出，由于抽样法只对总体中的一部分更行测试，所以它要测试的次数非常少，从而使得算法需要的时间比完全枚举少很多，算法的效率也会高很多。

1.3 部分忽略法

在信息学中，可能会遇到这样情况：一个题的分类非常复杂，且某些情况的处理非常复杂，但这些情况往往不是主要情况(即出现的概率很小或不处理这些情况对答案不会造成很大的影响)。有时，忽略这些复杂却影响不大的情况会使程序达到令人满意的结果。

例：Polygon

□ 题目大意

在此题中，所有的多边形均指凸多边形。

给定两个多边形 A 和 B，A 和 B 的 **Minkowski 和**是指包含所有形如(x_1+x_2, y_1+y_2) 的点的多边形，其中(x_1,y_1) 是 A 中的点，(x_2,y_2)是 B 中的点。

题目来源：IOI2004，中文原题请见附录，英文原题在附件中《polygon-1.2》。

我们考虑 **Minkowski 和**的一种逆运算。给定一个多边形 P, 我们寻找两个多边形 A 和 B，满足：

- P 是 A 和 B 的 **Minkowski 和**；

- A 有 2 到 4 个不同的顶点(线段、三角形或四边形)；

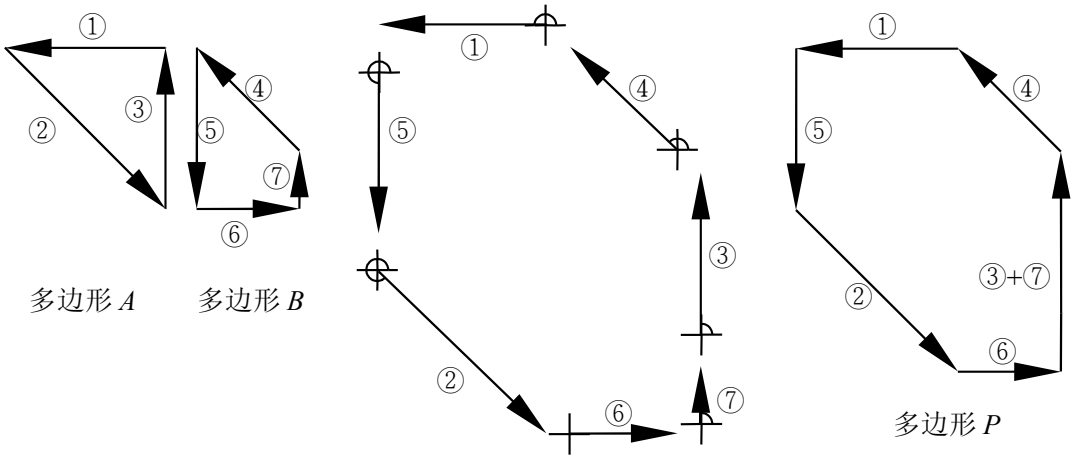
- A 必须包含尽可能多的顶点。

说明：此题附官方解答，其复杂度为 $O(N^2m^2)$，其中 N 为多边形 P 的顶点数，m 为 P 的边上的整点个数的最大值

□ 分析

由于本题中，位置并不是很难处理，所以此题将不考虑多边形的位置问题，只考虑其形状和大小。

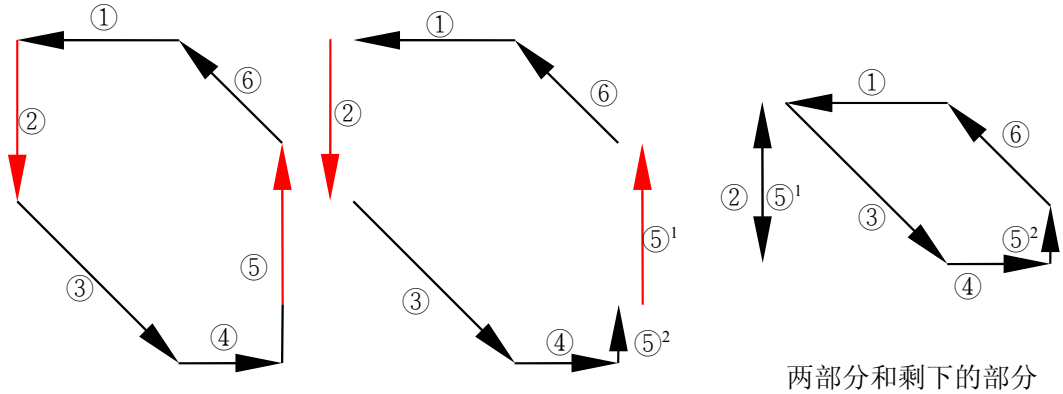
根据定义可以知道，对两个多边形 A、B 求 **Minkowski 和**，就是将它们的所有的边看作有顺序的向量，对所有向量按其级角排序，然后将所有的向量顺次连接，最后将共线同向的向量合并的过程。



由上面加法的过程不难想到，原来的多边形 A、B 的每条边在 P 中仍存在一条边与原边共线同向且模不小于原边，因此：

将一个多边形 P 拆成边数为 K 的多边形 A 与不定边数的多边形 B 相加的形式，就是要从 P 中找出 K 条边，从每边取出一段(或整条边)，使这 K 段正好能围成一个多边形，这个多边形就是 A，剩下的按照向量首尾相连可得到 B。

如：从上图的多边形 P 中取一个边数为 2 的多边形：



从 P 中取出一个三角形、四边形，也是同样的方法。

这个算法看起来很简单，但实现起来并没有想象中那么轻松。

找一个二边形，只要找到一对共线向量即可。

找一个三角形，可以枚举三条边，然后判断，其复杂度是 $O(N^3m^3)$。在提交答案类题看来，已经很不错了。

找一个四边形，如果枚举四条边，然后判断，其复杂度是 $O(N^4m^4)$，因其中的 m 是未知的，所以不敢轻易使用。这时可以枚举三条边，另一条边通过对边的二分查找得到。由于第四条边可能是多边形 P 中某条边的一段，而 P 中一共可能有 $N*m$ 段，因 m 未知，使得空间分配也不好处理；同时，由于存储时不仅要存储每段所对应的原来的边的序号，还要存储每段是占原来的几分之几……此时遇到的问题可能还远不只这些，这时，就很难保证程序不出错。

这种，不妨做一个大胆的假设：假设第四条边是 P 中的一条完整的边。这样，只要存储 P 条边即可，且这 P 条边都是原来多边形中的完整的边，所以编程中要考虑的问题就少多了，同时出错的可能性也小多了。

这样做，对于官方的数据，有 9 个测试点可以出解，所出的解全部是最优解。另一个测试点可以用手做。

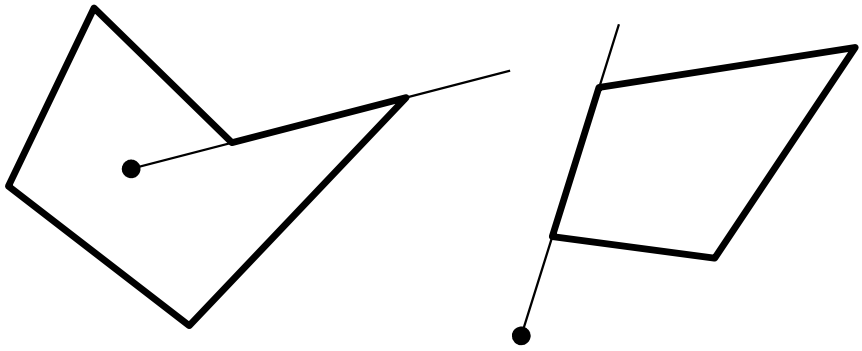
部分忽略法的实际应用

下面看一下判断一个点 P 是否在一个简单多边形内的算法：

假设已经判断了 P 在多边形边上这种特殊情况，剩下的只有点在多边形内和外的问题。

传统的判断算法是：以 P 为端点作一条射线，然后根据射线与多边形的交点个数是奇数个还是偶数个来判断其是否在多边形内。

但是，在计算交点个数的时候，有一种特殊情况：所作的射线经过多边形的某个顶点(有可能更特殊的与多边形的一条边重合)。如下图：



(*a*) (*b*)

在(*a*)中，处理的结果应为射线与多边形有奇数个交点，而(*b*)中，处理的结果应为射线与多边形有偶数个交点，如果不加特殊处理，则两种情况所得到的结果是相同的。

如果采用部分忽略法似乎会出现错误。但是，如果取的射线是一条很“一般”的射线：如在平面内随机取一个异于 P 的点 Q，且保证 Q 的坐标有若干位小数，从 P 作一条射线，使射线经过 Q，则，可以说，要出错的概率是非常小的。如果多取几个点，分别对这几个点进行判断，然后取这些结果中出现得最多的结果，则要出错就更不可能了。

小结

通过上面的例题可以看出，能过忽略部分复杂情况，能使算法的思考复杂性和编程复杂性降低。仅管它可能造成算法在一定程度上的错误，但这种错误通常是很小的。所以，有时采用部分忽略法仍是非常好的选择。

三、非完美算法的依据——RP 类问题与 Monte-Carlo 算法

前面所讲的一些方法，似乎都是靠的“运气”成分。但是，这些算法有它的依据：

如果一个问题存在一个随机算法，使得它有50%以上的概率得到期望的结果，那么这个问题属于RP类问题。该算法称为Monte-Carlo算法。

如果一个问题是RP类问题，可以通过多次运行它的一个Monte-Carlo算法而得到“几乎每次都是正确”的算法。

由于RP类问题与Monte-Carlo算法不是本文研究的重点，这里不再多做介绍，有兴趣的读者可以参阅相关的资料。

四、非完美算法的共性

非完美算法有很多种，但是，它们并不是完全不同的，它们之间存在一个共同的性质，那就是不完全性。上面的非完美算法都是由于其不完全性而使得它们具有一些完全正确的算法所不能具备的优势，同时，其不完全性也导致了算法不是完全正确的。

五、非完美算法的优点与缺点

4.1 优点

从上面的分析和举例，相信读者对非完美算法的优点已经有所了解了，现整理如下：

①时空消耗低：这是非完美算法的一个最突出的优点，也是大多数情况下使用它的原因。

②编程复杂度低：因为非完美算法会可能绕过纷繁的处理或会忽略掉非常难处理的一些情况，其编程复杂度可以得到一定的降低。在比赛中，低的编程复杂度往往比低的算法时间复杂更容易得到令人满意的结果。

③思维复杂度低：同样也是因为非完美算法可能忽略掉非常难处理的一些情况。

④能减少编程错误：通过前面几点，这点也就显然了。也许，一个非常优秀的算法，因为它的一点点小错误，就导致其前功尽弃，而一个非完美的算法，

因为它较容易实现，减少或避免了编程错误，反而能得到意想不到的好结果。

4.2 缺点

非正确性：这是不言而喻的，非完美算法，值得利用的就是它的非正确性。但非正确性使得算法不仅依赖算法的好坏、代码的好坏，还依赖于数据。如果数据较弱，非完美算法可能得到较高的分数，如果数据较强，其结果不一定很理想。

【总结】

本文主要介绍了非完美算法的几种重要方法。从上面的算法可以看到，并不是正确的算法就一定好过不完全正确的算法，因为非完美算法的不完全性，反而使非完美算法在很多方面比正确算法表现得更好。因此，合理的使用非完美算法能使我们得到令人满意结果。

【感谢】

衷心感谢向期中老师在我写这篇论文时对我的指导和帮助。
衷心感谢刘汝佳教练对我的指导和启发。
衷心感谢王俊、任恺同学对我的支持和帮助。
衷心感谢肖湘宁、周戈林同学在临近期末考试时还能为我看论文，并向我提很多宝贵的意见，特别感谢周戈林同学提供NOIP2003《传染病控制》的解题报告。

【参考文献】

《论随机化算法的原理与设计》——周咏基,1999

《The CRC Concise Encyclopedia of Mathematics》——Eric W. Weisstein,CRC Press

《计算几何——算法分析与设计》——周培德,清华大学出版社,广西科学技术出版社

《算法艺术与信息学竞赛》——刘汝佳、黄亮，清华大学出版社

《IOI'04: Solution of Polygon》——Ioannis Emiris and Elias Tsigaridas,2004

【附录】

最小强伪质数表

使用的最小质数个数	所使用的质数	最小强伪质数
1	2	2047
2	2,3	1373653
3	2,3,5	25326001
4	2,3,5,7	3215031751
5	2,3,5,7,11	2152302898747
6	2,3,5,7,11,13	3474749660383
7	2,3,5,7,11,13,17	341550071728321
8	2,3,5,7,11,13,17,19	341550071728321
9	2,3,5,7,11,13,17,19,23	≤41234316135705689041
10	2,3,5,7,11,13,17,19,23,29	≤1553360566073143205541002401
11	2,3,5,7,11,13,17,19,23,29,31	≤5689719352694202437032697232l

传染病控制原题(NOIP2003)

【问题背景】

近来，一种新的传染病肆虐全球。蓬莱国也发现了零星感染者，为防止该病在蓬莱国大范围流行，该国政府决定不惜一切代价控制传染病的蔓延。不幸的是，由于人们尚未完全认识这种传染病，难以准确判别病毒携带者，更没有研制出疫苗以保护易感人群。于是，蓬莱国的疾病控制中心决定采取切断传播途径的方法控制疾病传播。经过 WHO（世界卫生组织）以及全球各国科研部门的努力，这种新兴传染病的传播途径和控制方法已经研究消楚，剩下的任务就是由你协助蓬莱国疾控中心制定一个有效的控制办法。

【问题描述】

研究表明，这种传染病的传播具有两种很特殊的性质；

第一是它的传播途径是树型的，一个人 X 只可能被某个特定的人 Y 感染，只要 Y 不得病，或者是 XY 之间的传播途径被切断，则 X 就不会得病。

第二是，这种疾病的传播有周期性，在一个疾病传播周期之内，传染病将只会感染一代患者，而不会再传播给下一代。

这些性质大大减轻了蓬莱国疾病防控的压力，并且他们已经得到了国内部分易感人群的潜在传播途径图（一棵树）。但是，麻烦还没有结束。由于蓬莱国疾控中心人手不够，同时也缺乏强大的技术，以致他们在一个疾病传播周期内，只能设法切断一条传播途径，而没有被控制的传播途径就会引起更多的易感人群被感染（也就是与当前已经被感染的人有传播途径相连，且连接途径没有被切断的人群）。当不可能有健康人被感染时，疾病就中止传播。所以，蓬莱国疾控中心要制定出一个切断传播途径的顺序，以使尽量少的人被感染。你的程序要针对给定的树，找出合适的切断顺序。

【输入格式】

输入格式的第一行是两个整数 n（1≤n≤300）和 p。接下来 p 行，每一行有两个整数 i 和 j，表示节点 i 和 j 间有边相连（意即，第 i 人和第 j 人之间有传播途径相连）。其中节点

1 是已经被感染的患者。

【输出格式】

只有一行，输出总共被感染的人数。

【输入样例】

7 6
1 2
1 3
2 4
2 5
3 6
3 7

【输出样例】

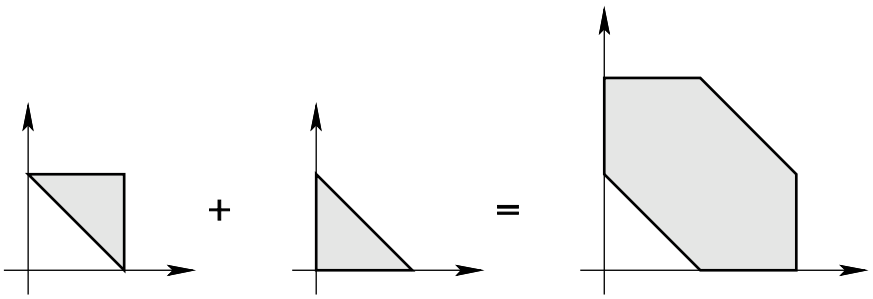
3

Polygon中文原题

【问题描述】

凸多边形的定义如下：多边形内任意两点 X 和 Y 的连线上的所有点都在多边形内部。在本题中的所有多边形都是拥有至少两个顶点的凸多边形并且所有顶点互不相同，且具有整数坐标。多边形的任意三个顶点不共线。在下文中的“多边形” 一词都是指这样的多边形。

给定两个多边形 A 和 B， A 和 B 的 *Minkowski 和* 包含所有形如 (x_1+x_2, y_1+y_2) 的点，其中 (x_1, y_1) 是 A 中的点，(x_2, y_2) 是 B 中的点。多边形的 *Minkowski* 和也是一个多边形。下图是一个例子：两个三角形和它们的 *Minkowski 和*。



我们考虑 *Minkowski 和* 的一种逆运算。给定一个多边形 P, 我们寻找两个多边形 A 和 B，满足:

- P 是 A 和 B 的 *Minkowski 和*,
- A 有 2 到 4 个不同的顶点，例如，它是一条线段（2 个顶点）, 一个三角形（3 个顶点），或一个四边形（4 个顶点），
- A 必须包含尽可能多的顶点，例如：
  - A 如果可能的话，必须是一个四边形，
  - 如果 A 不可能是一个四边形而有可能是一个三角形，则它必须是一个三角形，

ô 否则它是一条线段。

很显然， A 和 B 都不能等于 P，因为如果其中一个等于 P， 则另一个只能是一个点，而点不是一个合法的多边形。

给定多个输入文件，每个输入文件描述一个多边形 P。 对于每一个输入文件，你必须找出满足上述条件的 A 和 B， 并且创建一个文件来描述 A 和 B 。 对于所有给定的输入文件， A 和 B 一定存在。如果存在多组解，只需要输出其中的任意一组。不要提交源程序，只需要提交输出文件。

【输入格式】

共有十组输入文件从 polygon1.in 到 polygon10.in, 在 polygon 后的数字是输入文件序号。每个输入文件的构成如下：第一行包含一个整数 N： 多边形 P 的顶点个数。接下来的 N 行以逆时针方向描述了 N 个顶点的坐标，每个顶点一行。第 I+1 (I = 1, 2, ..., N) 行包含两个整数 X_I 和 Y_I， 以一个空格隔开，表示第 I 个顶点。所有坐标为非负整数。

【输出格式】

针对给定的输入文件，你需要给出相应的 10 个输出文件用以描述相应的 A 和 B。输出文件的第一行应包含：

```
#FILE polygon I
```

这里整数 I 表示相应的输入文件序号。

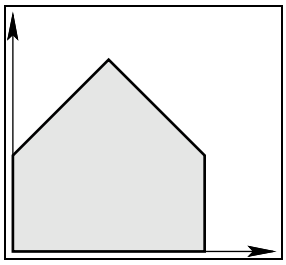
输出文件的格式与输入文件类似。第二行包含一个整数 N_A：A 中的顶点数 ($2 \le N_A \le 4$)。接下来的 N_A 行以逆时针方向描述 A 的 N_A 个顶点，每个顶点一行。第 I+2 (I = 1, 2, ..., N_A) 行包含两个整数 X 和 Y, 以一个空格隔开：表示 A 的第 I 个顶点。

第 N_A +3 行包含一个整数 N_B： 表示 B 中的顶点个数 ($2 \le N_B$)。接下来的 N_B 行以逆时针方向描述 B 的 N_B 个顶点，每个顶点一行。第 N_A +J+3 (J = 1, 2, ..., N_B) 行包含两个整数 X 和 Y， 以一个空格隔开：表示 B 的第 J 个顶点。

【输入输出样例】

polygon0.in

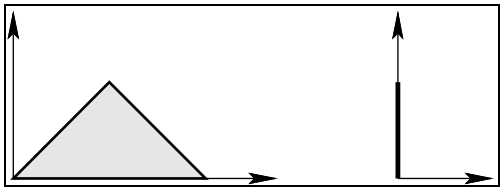
```
5
0 1
0 0
2 0
2 1
1 2
```



对于上面的输入，下面两种输出都是正确的，因为在每种输出中 A 都是一个三角形，而 A 不可能是一个四边形。

```
#FILE polygon
0
3
0 0
```

```
2 0
1 1
2
0 1
0 0
```



```
#FILE polygon
0
3
0 0
1 0
1 1
3
0 1
0 0
1 0
```