



浅析 非完美算法 在信息学竞赛中的应用

长郡中学 胡伟栋

➤ 计算机科学中非完美的例子

■ 图片、音频、视频的压缩

- 很多压缩率比较高的压缩方法都是有损压缩
较小的磁盘空间

■ 密码验证

- 很多都是多对一，通过验证的不一定是正确的
安全、实用

■ 搜索引擎

- 不一定能搜索到所有匹配的内容
方便、快捷

➤ 非完美算法

- 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。
 - 时间使用较少
 - 空间使用较少
 - 实现较容易
 - 容易被接受

➤ 非完美算法的一些常见方法

- 随机贪心 —— 周咏基 《论随机化算法的原理与设计》

(*)

- 抽样测试

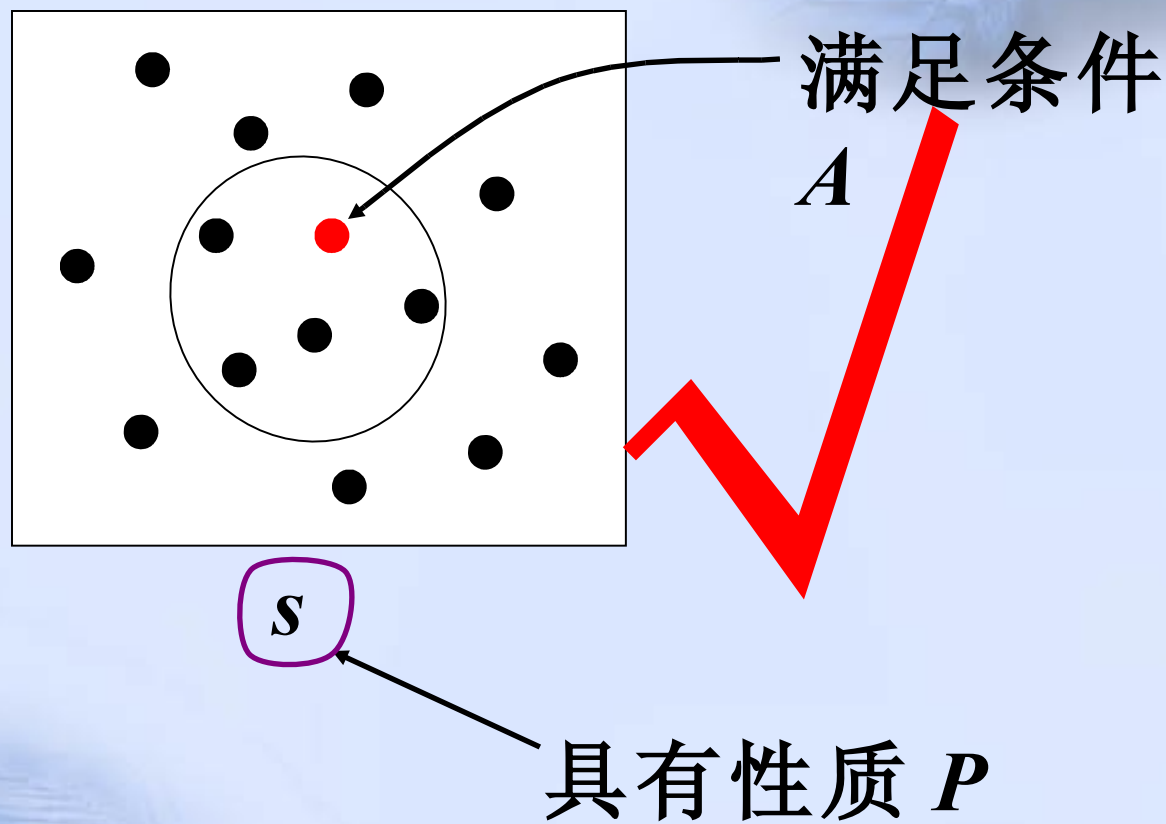
(*)

- 部分忽略

➤ 抽样测试法

- 抽样：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。

➤ 抽样测试法

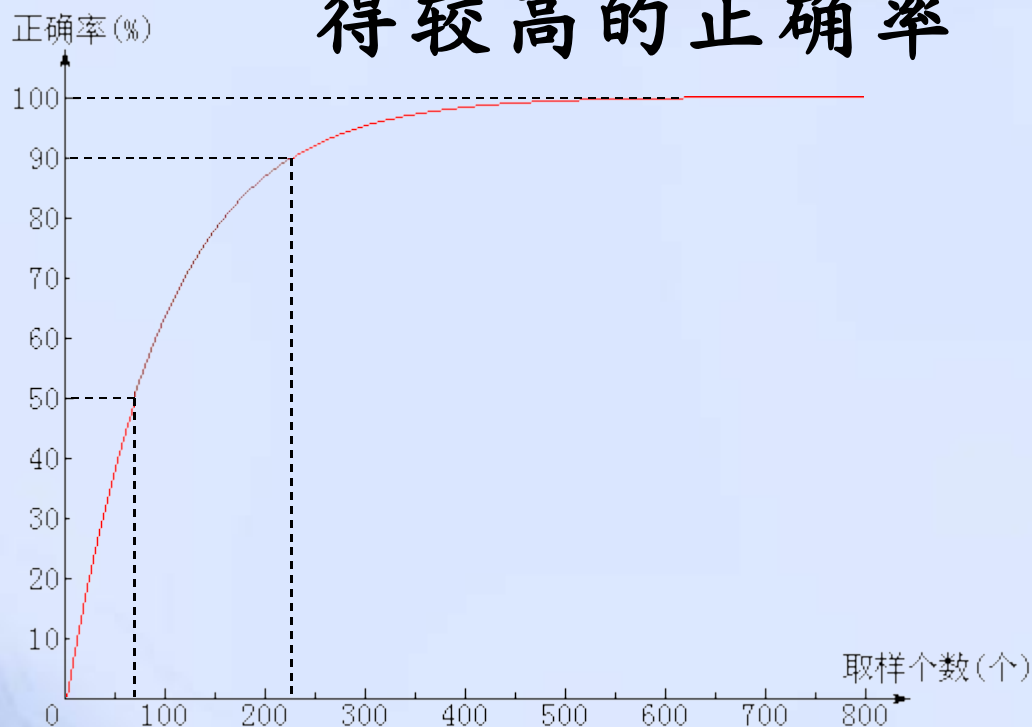


➤ 抽样测试法

10000 个元素

100 个满足条件

只要少量抽样就能
取得较高的正确率



➤ 抽样测试法——特殊抽样

- 在抽样测试时，有时总体中存在一些特殊的元素，这些元素满足条件的概率往往与其他元素满足条件的概率相差较大。如果特别的在这些元素中抽取一些进行测试，则可以加快出解的速度或增大解的正确率。

➤ 抽样测试法——特殊抽样

$$\alpha = \{'A', 'B', 'C', \dots, 'Z'\}$$

总体： α 的所有子

集
条件： 含 $\{'A', 'B', 'C', \dots, 'G'\}$ 的集
合

取特殊元素 α 即满足条件
!

➤ 质数判定

- 朴素的质数判定方法：
 - 用 $2 \sim \sqrt{n}$ 试除。 $O(n^{0.5})$

- 抽样测试法：
 - 在 $2 \sim \sqrt{n}$ 中抽取 k 个数试除。

➤ Strong Pseudoprime

- 对于奇数 n 和正整数 a ，设 $n-1=d \cdot 2^s$ (d 为奇数)，若：

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

或

- 存在 $0 \leq r < s$ ，使 $a^{d \cdot 2^r} \equiv -1 \pmod{n}$

则称 n 是以 a 为底的 强伪质数 (*Strong Pseudoprime*)。

判断： $O(\log_2 n)$

➤ 质数测试—— 抽样测试

- 当 a 不是 n 的整数倍时，质数 n 必然是以 a 为底的强伪质数。
- 在所有可能的 a 中，一个合数至多 $\frac{1}{4}$ 有 的机会为强伪质数。
- 抽样测试：随机抽取 k 个不同的 a 进行测试。
 - 正确率大于 $1 - 4^{-k}$
 - 时间复杂度为 $O(k \log_2 n)$ 。

➤ 质数测试——抽样测试

- 特殊抽样：让 a 取最小的若干个质数。
 - 只用 2 测试：最小的强伪质数为 2047，在小于 2.5×10^{10} 中有 4842 个强伪质数。
 - 只用 2,3 测试：最小强伪质数大于 1.3×10^6 。
 - 只用 2,3,5 测试：最小强伪质数大于 2.5×10^7 。
 - 只用 2,3,5,7 测试：最小强伪质数大于 3.2×10^9 。

➤ 质数测试——抽样测试

- 一般情况下，只要用 2,3,5,7 进行测试就能正确的判断一个数是否为质数。
- 时间复杂度： $O(\log_2 n)$

➤ 抽样测试法

抽样测试法


明显降低时间复杂度！

➤ 部分忽略法

- 在信息学中，可能会遇到这样情况：一个题的分类非常多，同时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，忽略这些复杂但对结果影响不大的情况仍然可以达到令人满意的结果。

➤ 部分忽略法

情况 出现率 处理用

Problem	A	30%	时少	✓
	B	40%	多	✓
	C	29%	少	✓
	D	1%	非常多	

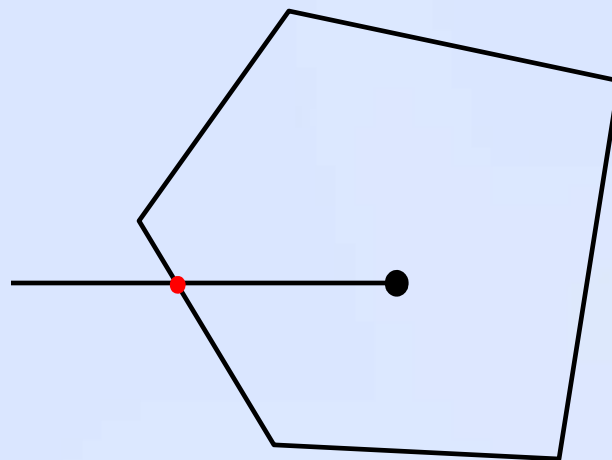
正确率：

99%

► 判断点是否在多边形内

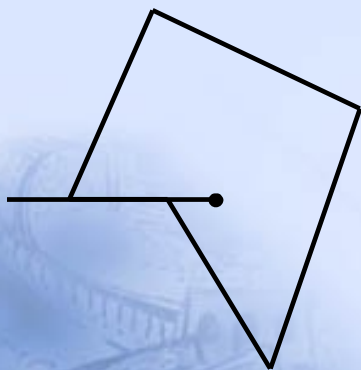
- 给出一个点和一个简单多边形（设点不在多边形的边上），判断点是否在多边形内。

- 方法：

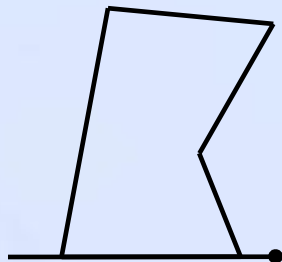


➤ 判断点是否在多边形内

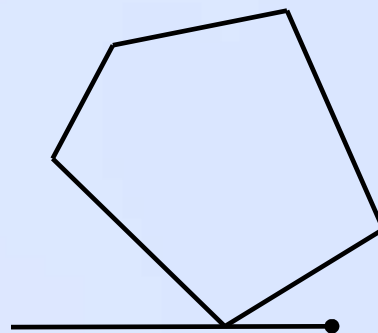
■ 特殊情况：



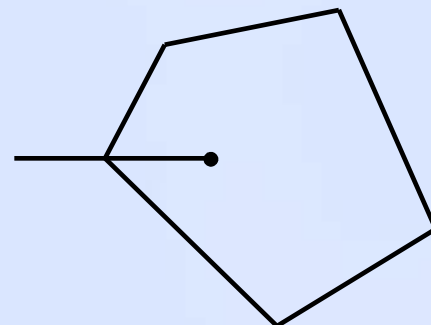
A ✓



B ✗



C ✓



D ✗

...

忽略特殊情况?

➤判断点是否在多边形内

- 射线经过多边形的顶点是一种非常特殊的情况。
- 而我们取的是一条非常特殊的射线——与 x 轴平行的射线。
- 经验表明，特殊的射线经过多边形的顶点的几率会大于一般的射线！
- 解决方法：

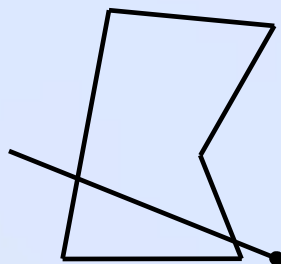
取一条一般的射线

！

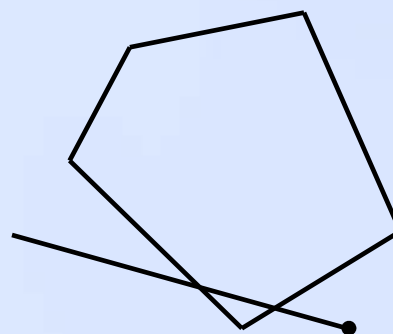
➤判断点是否在多边形内



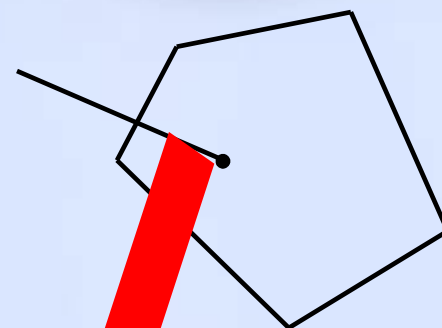
A



B



C



D

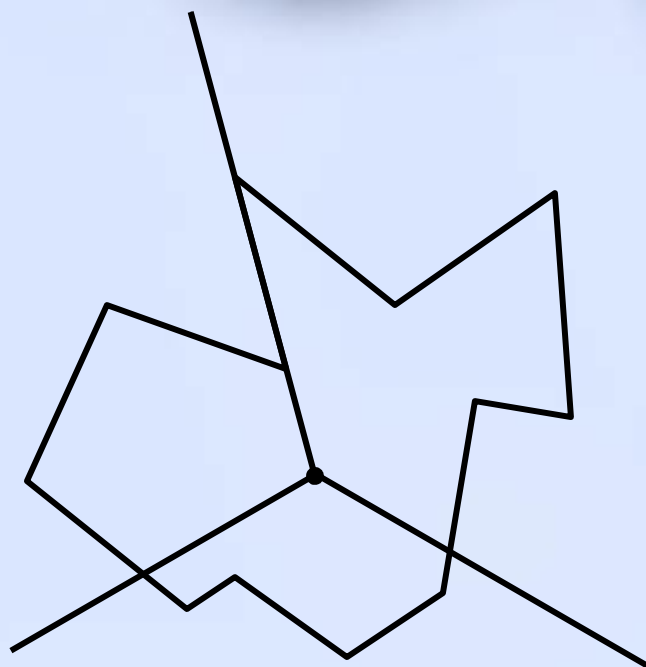


►判断点是否在多边形内

多取几条

将出现次数最多的结果作为正确结果

设取一条射线，其经过多边形顶点的概率为 x ，取 n 条，所得结果的正确率不小于 $1-x^{[n/2]}$



➤ 部分忽略

- 部分忽略法能减轻我们的思维负担和编程复杂性。
- 部分忽略法通常要加入一些小小的技巧。
使
被忽略的情况对结果影响尽量小
！

➤ 非完美算法

- 共同点：

- 不完全性

- 优点

- 时空消耗低
- 编程复杂度低
- 思维复杂度低
- 能减少编程错误

- 缺点

- 不完全正确

➤ 总结

- 在信息学竞赛以及实际应用中，并不是完全正确的算法就一定比非完美的算法表现得好，因为非完美算法的不完全性，反而使非完美算法在一些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取得非常令人满意的结果。

➤ 忠告

- 在能用完全正确的算法时要尽量使用完全正确的算法，只有当确实难想到很好的方法或时间比较紧时才使用非完美算法。

➤ 结束语

- 想了解更多，欢迎阅读我的论文。里面还有一些更好玩的例子，如：NOIP2003 的《传染病控制》、ACM 的《直觉主义逻辑》以及 IOI2004 的《Polygon》。

谢

谢

➤ RP 类问题 与 Monte-Carlo 算法

- 对于一个判定类问题，如果存在一个随机算法，当此算法判定结果为否时，原问题的结果必为否，同时，当此算法的判定结果为是时，原问题的结果为是的概率大于 50%，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。