随机化算法在信息学竞赛中的应用

湖南师大附中　胡泽聪

引言

什么是随机化算法？

引言

什么是随机化算法？

使用了随机函数，并且随机函数的返回值直接或间接地影响了算法的执行流程或执行结果，这类算法叫**随机化算法**。

引言

什么是随机化算法？

使用了随机函数，并且随机函数的返回值直接或间接地影响了算法的执行流程或执行结果，这类算法叫**随机化算法**。

近几年来，随机化算法出现的频率越来越高。
随机化算法并非只能用来骗分。
我们将探究“靠谱”的随机化算法。 □

分类

随机化算法分为以下三类：

分类

随机化算法分为以下三类：

- **数值概率算法，**通过随机选取元素从而求得在数值上的近似解。

分类

随机化算法分为以下三类：

- **数值概率算法**，通过随机选取元素从而求得在数值上的近似解。
- **Monte Carlo 算法**，总是能在确定的的运行时间内出解，但是得到的解有一定概率是错的。

分类

随机化算法分为以下三类：

- **数值概率算法**，通过随机选取元素从而求得在数值上的近似解。
- **Monte Carlo 算法**，总是能在确定的的运行时间内出解，但是得到的解有一定概率是错的。
- **Las Vegas 算法**，总是能返回正确的结果，但是其运行时间不确定。

分类

随机化算法分为以下三类：

- **数值概率算法**，通过随机选取元素从而求得在数值上的近似解。
- **Monte Carlo 算法**，总是能在确定的的运行时间内出解，但是得到的解有一定概率是错的。
- **Las Vegas 算法**，总是能返回正确的结果，但是其运行时间不确定。

由于随机化算法包含的范围较广，并不存在通用的解决问题的方法，只能具体问题具体分析。我们将通过一些例题来探究随机化算法的应用与效果。 □

例题一：MSTONE[1]

平面上有 n 个互不重合的点，已知存在不超过 7 条直线可以覆盖全部的点，问在平面上作一条直线，最多能覆盖多少个点。

$n \le 10000$。 □

[1] 题目来源：CodeChef。

例题一：MSTONE

先考虑最朴素的算法：枚举两个点，确定一条直线，然后判断有多少个点在这条直线上。

复杂度为 $O(n^3)$，无法在时间限制内出解。

例题一：MSTONE

先考虑最朴素的算法：枚举两个点，确定一条直线，然后判断有多少个点在这条直线上。

复杂度为 $O(n^3)$，无法在时间限制内出解。

我们尝试向朴素算法中加入随机，将其改造成 Monte Carlo 算法。问题在于，在哪一个部分随机？

题目有一个很关键的条件：存在不超过 7条直线可以覆盖全部的点。这个条件能给我们来带什么？ □

例题一：MSTONE

由题目的条件可以得出下面的引理

Lemma 1.1

覆盖了最多点的直线覆盖了至少 $\left\lceil \frac{n}{7} \right\rceil$ 个点。

用鸽巢原理易证。

例题一：MSTONE

由题目的条件可以得出下面的引理

Lemma 1.1

覆盖了最多点的直线覆盖了至少 $\left\lceil \frac{n}{7} \right\rceil$ 个点。

用鸽巢原理易证。从而引出下面的定理

Theorem 1.2

从给定的 n 个点中随机选择两个点，过这两个点的直线与覆盖了最多点的直线重合的概率为 $\frac{1}{49}$。

$\square$

例题一：MSTONE

我们将"枚举两个点"的步骤改为"随机选择两个点"。

运行这个算法一次的复杂度为 $O(n)$，正确率为 $\frac{1}{49}$。

例题一：MSTONE

我们将“枚举两个点”的步骤改为“随机选择两个点”。
运行这个算法一次的复杂度为 $O(n)$，正确率为 $\frac{1}{49}$。

设算法运行 k次，取 $k = 1000$，可以求出算法的正确率为

$$1 - \left(1 - \frac{1}{49}\right)^k \approx 1 - 10^{-9}$$

这个正确率相当令人满意，而且运行时间方面也可以接受。 □

例题一：MSTONE

事实上，例题一存在确定性算法。这道题的确定性算法需要进一步分析题目的性质，与这里介绍的随机化算法相比，思维难度更高。

由此可见，引入随机化可以在一定程度上降低思维难度、简化算法，同时对于一些题目来说，随机化算法还可以达到更优的复杂度。 □

例题六：Graph Reconstruction[2]

给定一个含有 n 个顶点和 m 条边的无向图，其中每个顶点的度数不超过 2，且图中无自环与重边。构造一个新图，满足以下条件：

- 新图含有 n 个顶点与 m 条边；
- 新图中每个顶点的度数不超过 2，且不含有自环与重边；
- 假设在原图中顶点 u 和 v 之间存在一条边，则在新图中顶点 u 和 v 之间不得有边。

输出任意一个满足条件的新图，或指出无解。

$1 \le m \le n \le 100000$。 □

[2] 题目来源：Codeforces Round #192 (Div. 1) - Problem C。

例题六：Graph Reconstruction

我们先分析题目中给出的图的性质。

例题六：Graph Reconstruction

我们先分析题目中给出的图的性质。

由于每个顶点的度数不超过 2，因此原图应该是一些环和链。而且在原图的补图中，每个顶点的度数不小于 $n-3$。

问题实际上就是在原图的补图中找出 m条边，满足给出的条件。 $\square$

例题六：Graph Reconstruction

先考虑问题在什么条件下一定有解。

例题六：Graph Reconstruction

先考虑问题在什么条件下一定有解。
考虑一个较强的约束：求出的新图应为一个哈密尔顿回路。

例题六：Graph Reconstruction

先考虑问题在什么条件下一定有解。

考虑一个较强的约束：求出的新图应为一个哈密尔顿回路。关于哈密尔顿回路的存在性，有如下定理：

Theorem 6.1 (Ore's Theorem)

对于一个含有 n 个顶点的无向图，其中存在哈密尔顿回路的充分条件是，对于任意两个不相邻的顶点 u 和 v，满足

$$\deg u + \deg v \geq n$$

其中 $\deg u$ 代表顶点 u 的度数。

此题中，$\deg u \geq n-3$ 对于每个顶点都成立，条件等价于

$$2(n-3) \geq n$$

即 $n \geq 6$。因此，对于任意 $n \geq 6$ 的情况，问题始终有解。 □

例题六：Graph Reconstruction

在下面的分析中，我们只考虑 $n \geq 6$的情况。如果我们求出了原图的补图的一个哈密尔顿回路，我们只需任选 m条边作为新图即可。问题在于如何求出哈密尔顿回路。

众所周知，求哈密尔顿回路是 NPC 问题，不存在高效的算法。这里我们介绍一个 Las Vegas 算法，并尝试求出其期望时间复杂度。 □

例题六：Graph Reconstruction

算法流程如下：

1. 随机一个 $1 \sim n$的排列 $p_1, \ldots, p_n$；
2. 对于 $1 \le i < n$，检查无向边 (p_i, p_{i+1})是否存在于原图中，同时检查无向边 (p_n, p_1)是否存在于原图中；
3. 如果检查的所有边都不存在于原图中，那么我们成功地找到了一个哈密尔顿回路；否则则返回第1步。 □

例题六：Graph Reconstruction

算法的步骤一和二的时间复杂度可以做到 $O(n\log n)$，我们关注的是期望意义下算法的运行次数。

例题六：Graph Reconstruction

算法的步骤一和二的时间复杂度可以做到 $O(n \log n)$，我们关注的是期望意义下算法的运行次数。

可以发现，这是一个由 Monte Carlo 算法改造成的 Las Vegas 算法，因此计算复杂度的关键在于求出其正确率。不过对于这个算法而言，求出精确的正确率相当麻烦，因此我们考虑求出近似的正确率。 □

例题六：Graph Reconstruction

假设原图是一个长度为 n的环。

考虑逐一确定排列的每一位，令 f_i^n表示，排列长度为 n且确定了前 i位时，排列合法的概率。

特别地，我们直接用 f^n表示 f_n^n。

由于只是近似，我们不考虑具体哪些点已经加入排列，只考虑一次选择中合法的概率。 □

例题六：Graph Reconstruction

显然我们有 $f_1^n = 1$，对于任意 $i (i > 1)$，令

$$
\begin{aligned}
p_1 &= \binom{i-2}{2} \\
p_2 &= (i-2)(n-i-1) \\
p_3 &= \binom{n-i+1}{2}
\end{aligned}
$$

例题六：Graph Reconstruction

显然我们有 $f_1^n = 1$，对于任意 $i(i > 1)$，令

$$
\begin{aligned}
p_1 &= \binom{i-2}{2} \\
p_2 &= (i-2)(n-i-1) \\
p_3 &= \binom{n-i+1}{2}
\end{aligned}
$$

我们可以做如下的估计

$$
\begin{aligned}
f_i^n &= f_{i-1}^n \frac{1}{\binom{n-1}{2}} \left(p_1 + p_2 \cdot \frac{n-i}{n-i+1} + p_3 \cdot \frac{n-i-1}{n-i+1} \right) \\
&= \frac{n-3}{n-1} f_{i-1}^n \\
&= \left(\frac{n-3}{n-1} \right)^{i-1}
\end{aligned}
$$

结果十分简洁。

例题六：Graph Reconstruction

那么对于 f^n 有

$$f^n = \left(\frac{n-3}{n-1}\right)^{n-1}$$

对其求极限可知

$$\lim_{n\to\infty} f^n = \frac{1}{e^2} \approx 0.135335$$

而当 $n \geq 9$时，已经有 $f^n \geq 0.1$。

例题六：Graph Reconstruction

那么对于 f^n 有

$$f^n = \left(\frac{n-3}{n-1}\right)^{n-1}$$

对其求极限可知

$$\lim_{n\to\infty} f^n = \frac{1}{e^2} \approx 0.135335$$

而当 $n \geq 9$时，已经有 $f^n \geq 0.1$。

通过进一步的计算可得，这个算法的期望运行次数 $k = (f^n)^{-1}$。

对于足够大的 n，我们可以认为 $k \approx 10$，在运行时间上可以说是绰绰有余。

至于 $n < 9$的情况，我们可以直接用暴力算法求解。 $\square$

对比

题目	思维难度	代码难度	时间复杂度
例题一	✓	✓	✓
例题二	不存在可以通过的确定性算法		
例题三	✓	✓	
例题四	✓		
例题五	不存在可以通过的确定性算法		
例题六	✓	✓	

打钩的项目代表该题的随机化算法在这个方面优于确定性算法。 □

总结

随机化算法可以运用在各种类型的题目中。相比起确定性算法，随机化算法有如下的优势：

- 不需要对问题的性质做过多分析；
- 编程复杂度较低；
- 一些情况下，可以达到更优的时间复杂度；
- 对于有些题目而言，随机化算法是唯一的正确算法。

总结

随机化算法可以运用在各种类型的题目中。相比起确定性算法，随机化算法有如下的优势：

- 不需要对问题的性质做过多分析；
- 编程复杂度较低；
- 一些情况下，可以达到更优的时间复杂度；
- 对于有些题目而言，随机化算法是唯一的正确算法。

当然，也有劣势：

- 需要比较严谨的复杂度与正确率的证明，证明可能会比较复杂；
- 并非所有问题都存在随机化的做法，有时随机化只能作为获得部分分的工具。 □

致谢

感谢中国计算机学会提供学习和交流的平台。
感谢胡伟栋教练和余林韵教练的帮助与指点。
感谢我的教练李淑平老师对我的指导与关心。
感谢父母对我学习信息学竞赛的支持与鼓励。
感谢一路陪我走来的许许多多的同学的帮助。

The End

Thanks for listening,
Questions are welcomed!