# 概述

参考：

<https://oi-wiki.org/misc/rand-technique/>

<https://www.dotcpp.com/course/1029>

## 随机化算法

随机化算法是一种利用随机数来增加算法效率或减少算法复杂度的算法。其原理是通过引入随机因素，使得算法的执行过程不再是确定性的，而是具有一定的随机性。

这样可以使得算法的执行效率更高，同时也可以避免一些特定输入数据导致算法复杂度过高的问题。

## 拒绝采样

拒绝采样算法（Rejection Sampling Algorithm）是一种用于从一个概率分布中抽样的方法，特别适用于无法直接抽样的分布。其基本思想是通过一个易于抽样的辅助分布来生成样本，并根据一定的条件接受或拒绝这些样本，从而获得符合目标分布的样本。这个辅助分布通常是一个比目标分布容易抽样的分布，称为提议分布（proposal distribution）。

算法步骤如下：

1、从提议分布中抽取一个样本。

2、从均匀分布中随机抽取一个值作为接受阈值。

3、如果提议样本的密度值乘以接受阈值小于目标样本的密度值，则接受该提议样本；否则拒绝该提议样本。

4、重复上述步骤，直到获得足够多的符合目标分布的样本。

拒绝采样算法的主要缺点是可能会产生大量的被拒绝样本，尤其是在提议分布与目标分布相差较大时。因此，在实际应用中，需要选择合适的提议分布以减少被拒绝样本的数量。

拒绝采样算法（Rejection Sampling Algorithm）和随机化算法（Randomized Algorithm）是两种不同的概念。

拒绝采样算法是一种用于从一个概率分布中抽样的方法，其基本思想是通过一个易于抽样的辅助分布来生成样本，并根据一定的条件接受或拒绝这些样本，从而获得符合目标分布的样本。这个辅助分布通常是一个比目标分布容易抽样的分布，称为提议分布（proposal distribution）。

随机化算法是一种算法设计范例，其中算法的行为受到随机性的影响。随机化算法通常利用随机性来解决问题的不确定性部分，从而减少算法的平均时间复杂度或空间复杂度，或提高算法的正确性概率。

虽然这两种概念都涉及随机性，但它们的应用领域和实现方式有所不同。拒绝采样算法是用于生成符合某一特定概率分布的样本，而随机化算法是一种通用的算法设计方法，用于解决各种类型的问题。

# 分类

随机算法通常分成4类：

## 数值随机算法（Numericalrandomized algorithm）/概率算法

随机算法有时也称概率算法（probabilistic algorithm），但也有人对两者这样区分。

如果取得结果的途径是随机的，则称为随机算法，如拉斯维加斯算法；

如果取得的解是否正确存在随机性，称为概率算法，如蒙特卡罗算法。

## 蒙特卡罗算法

蒙特卡罗算法（MonteCarlo）：不保证所求得的解是正确的，也就是说，蒙特卡罗算法求得的解有时是错误的。不过，由于可以设法控制这类算法得到错误解的概率，并因它的简单高效，是很有价值的一类随机算法。一般情况下，蒙特卡罗算法求得正确解的概率随计算时间的增加而增大。但无论如何不能确保解的正确性，而且通常无法有效地判断所求得的解究竟是否正确，这是蒙特卡罗算法的缺陷。

## 拉斯维加斯算法

拉斯维加斯算法（LasVegas）：求得的解总是正确的，但有时拉斯维加斯算法可能始终找不到解。使用拉斯维加斯算法求解同一问题的同一实例，能够得到相同的结果，但算法的执行时间会不一样。一般情况下，求得正确解的概率随计算时间的增加而增大。因此，为了减少求解失败的概率，可以使用一个拉斯维加斯算法对同一实例，重复多次执行该算法。

## 舍伍德算法

舍伍德算法（Sherwood）：总能求得问题的正确解。当一个确定性算法在最坏情况下的计算复杂度与其在平均情况下的计算复杂度两者相差较大时，可以在这个确定算法中引入随机性将它改造成一个舍伍德算法，用来消除或减少问题的不同实例之间这种在计算时间上的差别。舍伍德算法的精髓不是避免算法的最坏情况行为，而是设法消除这种最坏行为与特定实例之间的关联性。

# 原理

随机化算法的原理可以简单概括为以下几点：

1、随机数的引入。

随机数可以用来产生随机的初始状态，或者在算法的执行过程中用来产生随机决策。

2、概率分析。

由于随机化算法的执行过程具有一定的随机性，因此需要采用概率分析来评估算法的正确性和效率。

3、随机性和确定性的平衡。

随机化算法需要在随机性和确定性之间取得平衡，既要保证算法的正确性，又要提高算法的执行效率。

# 相关算法

## 洗牌算法

参考：

<https://www.cnblogs.com/z-sm/p/12393211.html>

<https://cloud.tencent.com/developer/article/2154920>

[算法思维：洗牌算法](https://github.com/labuladong/fucking-algorithm/blob/master/%E7%AE%97%E6%B3%95%E6%80%9D%E7%BB%B4%E7%B3%BB%E5%88%97/%E6%B4%97%E7%89%8C%E7%AE%97%E6%B3%95.md)

[三种洗牌算法shuffle](https://blog.csdn.net/qq_26399665/article/details/79831490)

[Fisher-Yates洗牌算法](https://styx11.github.io/blog/JavaScript/#%E7%AE%97%E6%B3%95%E6%AD%A5%E9%AA%A4)

## 模拟退火

参考：

爬山算法：<https://oi-wiki.org/misc/hill-climbing/>

模拟退火算法：<https://oi-wiki.org/misc/simulated-annealing/>

模拟退火是一种随机化算法。当一个问题的方案数量极大（甚至是无穷的）而且不是一个单峰函数时，我们常使用模拟退火求解。

根据爬山算法的过程，我们发现：对于一个当前最优解附近的非最优解，爬山算法直接舍去了这个解。而很多情况下，我们需要去接受这个非最优解从而跳出这个局部最优解，即为模拟退火算法。

什么是退火？

由于退火的规律引入了更多随机因素，那么我们得到最优解的概率会大大增加。于是我们可以去模拟这个过程，将目标函数作为能量函数。

# 应用

## 打乱数组

## 随机数索引

## 链表随机节点

## 按权重随机选择

## O(1) 时间插入、删除和获取随机元素

## 用Rand7()实现Rand10()