

100 囚犯抽签问题

一、算法说明

两种搜索策略的成功率对比：

1. 随机搜索策略：

每个囚犯随机选择 K 个盒子检查

整体成功率为独立事件概率的乘积： $(K/N)^N$

当 $N=100$, $K=50$ 时，成功率约为 $(1/2)^{100} \approx 7.89e-31$

2. 循环搜索策略：

囚犯从自己编号的盒子开始，按盒内纸条跳转

成功率取决于盒子排列中最长循环的长度是否 $\leq K$

理论成功率约为 $1 - \ln 2 \approx 30.68\%$

3. 理论计算：

循环策略的理论成功率公式为： $1 - \sum (1/i)$ ， i 从 $K+1$ 到 N

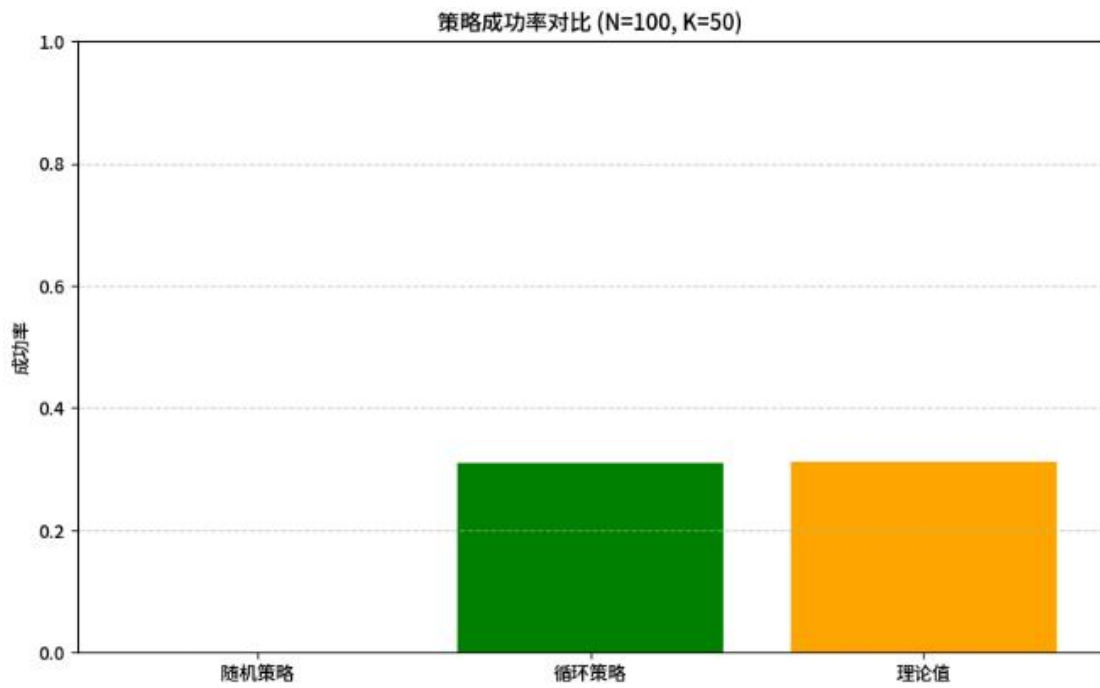
当 $K=N/2$ 时，近似为 $1 - \ln 2$

二、实验结果

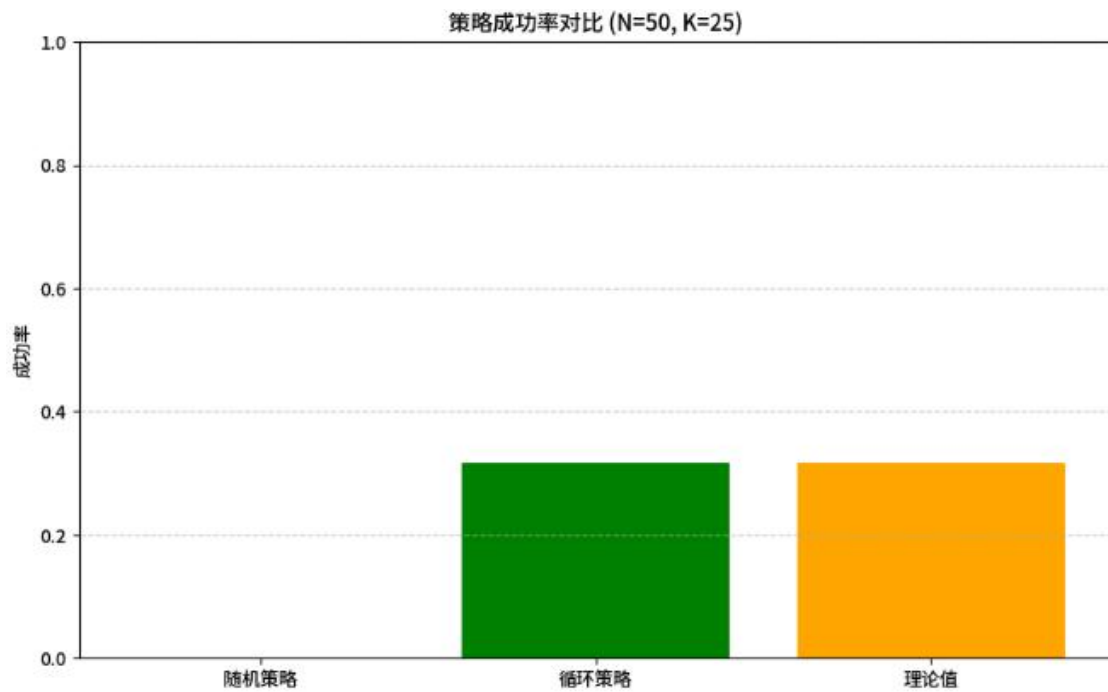
1. 结果

```
囚犯数量: 100, 尝试次数: 50, 模拟轮次: 10000
随机搜索策略成功率: 0.000000
循环搜索策略成功率: 0.309900
理论成功率: 0.311828
调整参数后的模拟 (N=50, K=25):
囚犯数量: 50, 尝试次数: 25, 模拟轮次: 10000
随机搜索策略成功率: 0.000000
循环搜索策略成功率: 0.316600
理论成功率: 0.316753
```

2. $N=100$, $K=50$ 时两种策略成功率的对比图



3. N=50, K=25 时两种策略成功率的对比图



三、优化思路

1. 原代码每次模拟都遍历所有囚犯，可以优化为直接检测最长循环长度，使用 `visited` 数组标记已访问的盒子，避免重复计算。
2. 添加循环长度分布直方图，直观展示排列特性。
3. 显示不同 N/K 值的成功率变化曲线，分析参数敏感性。
4. 计算不同 K 值对应的最优成功率，找到临界点。
5. 统计验证: 计算置信区间，评估模拟结果的可靠性；添加统计显著性检验，确认两种策略差异
6. 使用多进程或线程并行运行模拟，提高效率。
7. 优化内存：对于极大 N 值，使用位运算表示访问状态或考虑使用生成器模式减少内存占用。