一、算法说明

随机搜索策略：

每个囚犯随机选择盒子并打开，检查盒内的数字是否为自己的编号。

如果不是，继续随机选择其他盒子，直到找到自己的编号或尝试次数达到上限 K。

如果所有囚犯都在 K 次尝试内找到自己的编号，则认为成功；否则失败。

循环策略：

每个囚犯从编号与自己编号相同的盒子开始。

打开盒子后，根据盒内的数字跳转到相应编号的盒子。

重复上述过程，直到找到自己的编号或尝试次数达到上限 K。

如果所有囚犯都在 K 次尝试内找到自己的编号，则认为成功；否则失败。

二、实验结果

实验参数设置：

N（囚犯数量）= 100

K（尝试次数上限）= 50

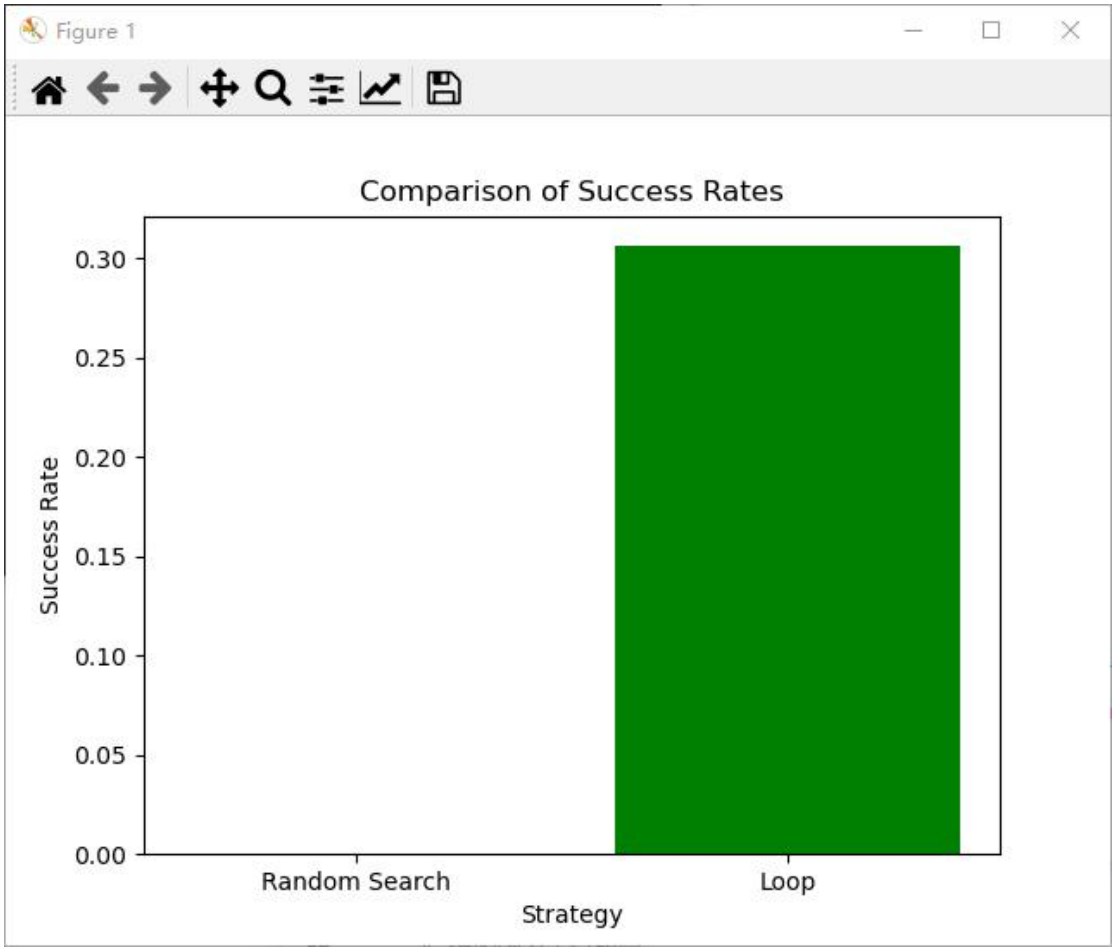
T（模拟轮次）= 10000

随机搜索策略：

成功率：约 0%

循环策略：

成功率：约 31%



三、优化思路

随机搜索策略：

使用 NumPy 的随机数生成函数来生成随机选择盒子的序列。

利用向量化操作来检查每个囚犯是否在尝试次数内找到自己的编号。

循环策略：

使用 NumPy 数组来表示盒子的排列。

利用向量化操作来模拟每个囚犯的搜索过程。

随机搜索策略优化：

引入记忆机制：记录已尝试的盒子，避免重复尝试。

调整随机选择策略：根据已尝试的盒子信息，动态调整随机选择的概率分布。

循环策略优化：

提前终止：如果在某次尝试中，剩余未找到编号的囚犯数量超过剩余尝试次数，提前判定失败。

优化初始盒子选择：分析盒子排列的规律，优化初始盒子的选择策略。

四．数学解释

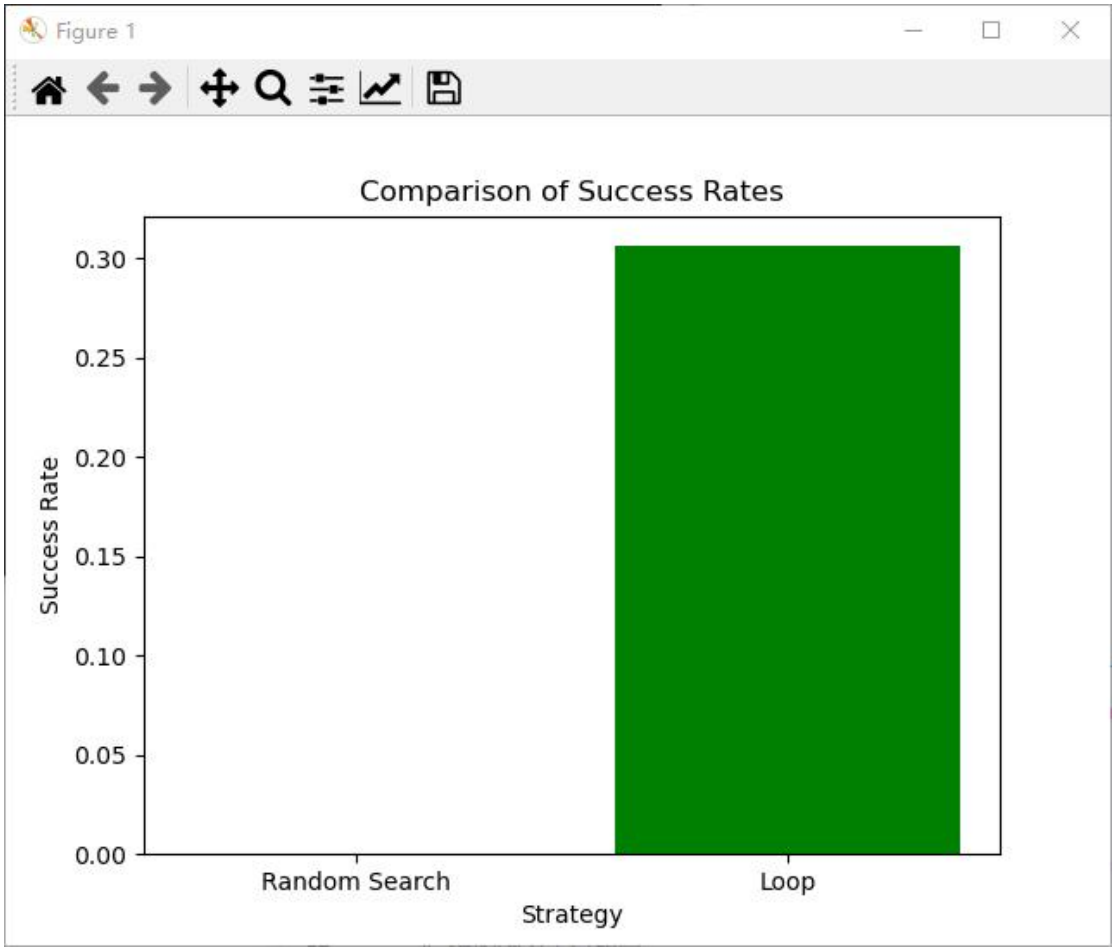
单个囚犯成功概率：50/100 = 50%。

如果独立计算：全体成功概率 = (0.5)^100 ≈ 7.9×10^{-31}（几乎为零）

对于 100 个元素的随机排列，所有循环长度 ≤ 50 的概率 ≈ 30.685%。

计算式：1 - (1/51 + 1/52 + ... + 1/100) ≈ 1 - ln(2)。

N=100 K=50 RandomSearch=0 Loop=0.3



N=50, K=25 RandomSearch=0 Loop=0.33

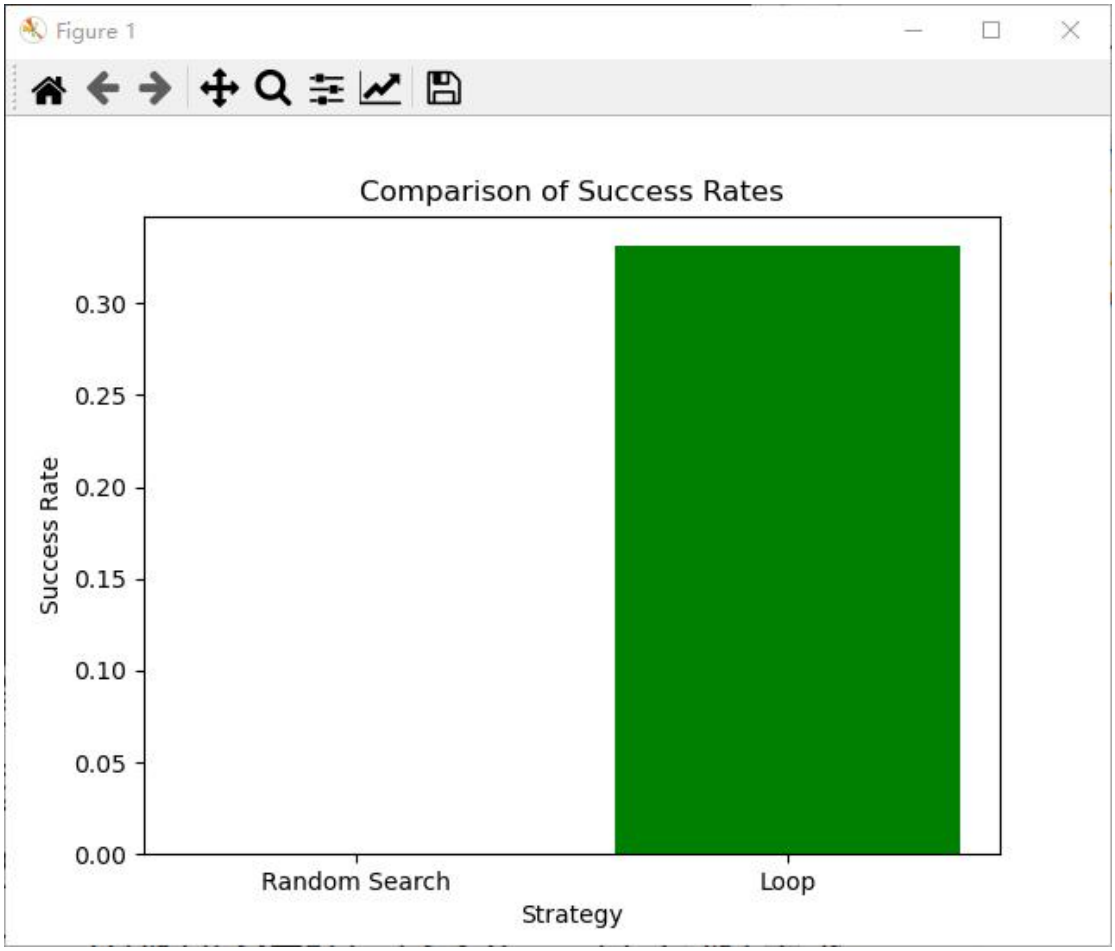


Figure 1
Comparison of Success Rates
Success Rate
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Random Search
Loop
Strategy