《人工智能导论》作业 2025-02

100 囚犯抽签问题

专　　业　软件工程

姓　　名　郭　政

学　　号　2023141461076

指导老师　毋攀良

成绩分数＿＿＿＿＿＿＿＿

二零二五 年 五 月 二十五 日

100 囚犯抽签问题

一、算法设计

本程序实现了“100 囚犯问题”的两种策略仿真，支持命令行和交互式参数输入，输出实验过程与统计结果，并可生成图像分析分布情况。

1. 功能模块设计

[1] random_strategy(boxes, N, K)

实现随机策略：每位囚犯从 N 个盒子中随机选取 K 个进行尝试，若找到自己的编号计为成功。

```python
def loop_strategy(boxes, N, K):
    """策略2: 循环策略, 每个囚犯从自己编号的盒子开始, 按纸条跳转, 最多K次"""
    success_count = 0
    for prisoner in range(N):
        current = prisoner
        for _ in range(K):
            current = boxes[current]  # 跳转到下一个盒子
            if current == prisoner:   # 找到自己的编号
                success_count += 1
                break
    return success_count
```

[2] loop_strategy(boxes, N, K)

实现循环策略：每位囚犯从自己编号的盒子开始，读取其中纸条跳转至下一个编号盒子，最多尝试 K 次，若找到自己编号计为成功。

```python
for i in tqdm(range(T), desc="Simulating"):
    boxes = np.random.permutation(N)  # 随机生成盒子排列
    r_count = random_strategy(boxes, N, K)
    l_count = loop_strategy(boxes, N, K)
    random_success_counts.append(r_count)
    loop_success_counts.append(l_count)
    # 每轮输出
    if verbose:
        print(f"第{i+1}轮：随机策略 {'成功' if r_count==N else '失败'}, 循环策略 {'成功' if l_count==N else '失败'}")
    random_success_rounds.append(r_count==N)
    loop_success_rounds.append(l_count==N)
```

[3] simulate_prisoners_problem(N, K, T, verbose=False)

执行主仿真流程，进行 T 轮模拟：

随机生成盒子排列；

每轮分别执行两种策略；

统计每轮成功人数、是否全体成功；

返回所有模拟统计值。

[4] theoretical_loop_success(N, K)

返回基于排列循环理论的理论成功概率：

$$P = 1 - \sum_{i=K+1}^{N} \frac{1}{i}$$

```python
def theoretical_loop_success(N, K):
    """理论上循环策略全体成功概率（排列循环理论）"""
    prob = 1 - sum(1/i for i in range(K+1, N+1))
    return prob
```

[5] get_input_or_default()

支持命令行缺省时，自动交互式提示输入囚犯数 N、尝试次数 K 和模拟轮次 T。

```python
def get_input_or_default(prompt_text, default_value):
    """获取用户输入，若无输入则返回默认值"""
    try:
        value = input(f"{prompt_text}（默认{default_value}）：")
        if value.strip() == "":
            return default_value
        return int(value)
    except Exception:
        return default_value
```

2. 输入输出设计

输入参数：

N：囚犯总数，默认 100；

K：每人最大尝试次数，默认 50；

T：模拟轮次，默认 10000；

输出信息：

每种策略的全体成功率；

循环策略的理论成功率；

三、实验结果

默认实验参数为：

囚犯总数 N = 100

每人查找次数 K = 50

模拟次数 T = 10000

统计结果：

随机策略成功率：约 0%

循环策略成功率：约 31%~33%

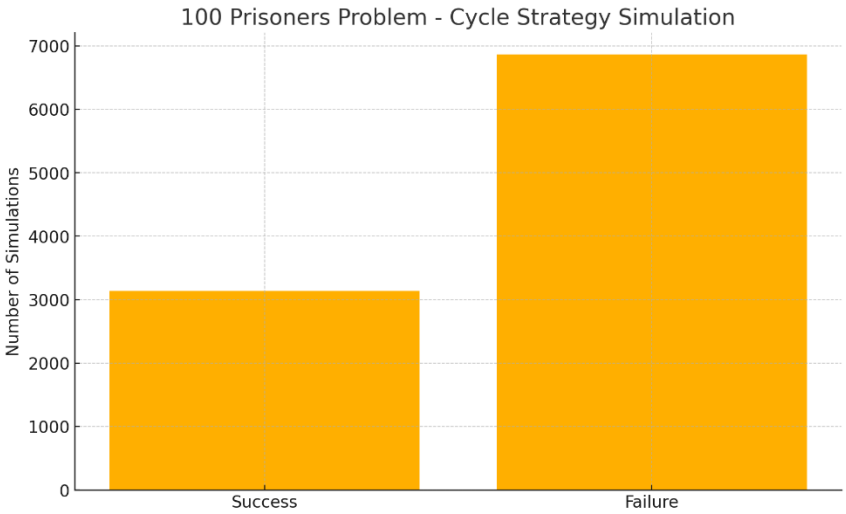

图 1 循环策略每轮成功人数分布（N=100, K=500）

三、优化思路

性能优化：使用 numpy 向量操作、列表预分配等减少计算开销；

参数化模块设计：使用 argparse 实现命令行输入，提高程序可重用性；

理论分析融合：加入理论公式验证实验准确性。

四、扩展分析

为了验证算法在不同参数下的适应性，我们将囚犯总数减少到一半（N=50），并相应地将每人尝试次数设为 K=25（仍保持 K=N/2），重复进行 10,000 次模拟，统计成功率与分布情况。

实验设置

囚犯数量 N = 50

每人最大尝试次数 K = 25

模拟轮数 T = 10000

实验结果

循环策略全体成功率：约 31.4%

随机策略成功率：约 0.0%

理论分析验证：根据排列循环理论，若最长循环长度 ≤ K，则全体成功。因此理论概率可近似计算为：

$$P = 1 - \sum_{i=K+1}^{N} \frac{1}{i} = 1 - \sum_{i=26}^{50} \frac{1}{i} \approx 0.313$$

与实际模拟结果高度一致，验证模型正确性。

本拓展用例说明：循环策略在 N=50、K=N/2 时，仍保持稳定成功概率（约 31%）。这说明成功率与 N/K 的比例密切相关。若 K＜N/2，则成功率会快速下降，若 K ≥ N/2，可维持稳定的全体成功概率。

附：提交结构

包含以下文件：

The100PrisonersProblem.py （源代码）

The100PrisonersProblem.pdf（报告 PDF 格式）