QTA 笔面试刷题 Week 5 完整解析

QTA

December 2025

目录

1 相关系数为 ρ 的取值范围

1.1 题目描述

设随机变量 X, Y, Z 两两相关系数均为 ρ，则其相关矩阵为：

$$R = \begin{pmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{pmatrix}$$

要求联合随机变量的相关矩阵为半正定（即 $R \succeq 0$），求 ρ 的取值范围。

1.2 解答与推导

矩阵 R 可以写成 $R = (1-\rho)I + \rho J$，其中 I 是 3×3 的单位矩阵，J 是全 1 矩阵。

1. **求解特征值：** 全 1 矩阵 $J_{3\times 3}$ 的特征值为 3（重数为1，对应特征向量 $[1,1,1]^T$）和 0（重数为2）。因此，矩阵 R 的特征值 λ 为：

$$\begin{aligned} \lambda_1 &= (1-\rho)\cdot 1 + \rho \cdot 3 = 1 + 2\rho \\ \lambda_2 &= (1-\rho)\cdot 1 + \rho \cdot 0 = 1 - \rho \\ \lambda_3 &= (1-\rho)\cdot 1 + \rho \cdot 0 = 1 - \rho \end{aligned}$$

2. **半正定条件：** 矩阵半正定要求所有特征值非负，即 $\lambda_i \geq 0$：$\begin{cases} 1 + 2\rho \geq 0 \implies \rho \geq -\frac{1}{2} \\ 1 - \rho \geq 0 \implies \rho \leq 1 \end{cases}$

结论： ρ 的取值范围是 $[-\frac{1}{2}, 1]$。

2 掷出连续 6 个 6 的期望投掷次数

2.1 题目描述

一个均匀的骰子，掷出连续 6 个 6 的期望次数是多少？

2.2 解答与推导

设 E_n 为掷出连续 n 个 6 所需的期望次数。我们可以建立递推关系：

- 假设我们已经掷出了连续 $n-1$ 个 6（耗费期望 E_{n-1} 次）。
- 再掷一次（第 $E_{n-1}+1$ 次）：
  - 如果是 6（概率 1/6），则达成连续 n 个 6，结束。
  - 如果不是 6（概率 5/6），则连续中断，一切从头开始（需要重新掷 E_n 次）。

根据全期望公式：

$$E_n = E_{n-1} + 1 + \frac{1}{6}(0) + \frac{5}{6}(E_n)$$

化简得：

$$\frac{1}{6}E_n = E_{n-1} + 1 \implies E_n = 6E_{n-1} + 6$$

已知 $E_0 = 0$，则：

$$\begin{aligned}
E_1 &= 6 \\
E_2 &= 6(6) + 6 = 6^2 + 6 \\
E_3 &= 6(6^2 + 6) + 6 = 6^3 + 6^2 + 6 \\
&\cdots \\
E_6 &= \sum_{i=1}^{6} 6^i
\end{aligned}$$

计算等比数列求和：

$$E_6 = \frac{6(1 - 6^6)}{1 - 6} = \frac{6(46656 - 1)}{5} = 55986$$

结论： 期望次数为 55986 次。

3 随机落座（疯子坐飞机）

3.1 题目描述

N 个座位，第 1 人（精神病）随机坐。后续正常人若座位空着就坐，被占则随机坐。求第 N 个人坐到自己座位的概率。

3.2 解答与推导

考虑第 1 个座位和第 N 个座位的对称性。对于任何一个人（从第 1 人到第 $N-1$ 人），如果他需要随机选择座位：

- 若选中 **1号座位**：由于1号座位对应的是那个“导致混乱的源头”（第1个人本该坐的位置），一旦1号座位被填满，后续所有人都可以坐回自己的位置，包括第 N 个人。
- 若选中 **N号座位**：第 N 个人的位置被占，第 N 个人必输。
- 若选中 **其他座位**（2 到 $N-1$）：混乱继续传递给下一个人。

直到游戏结束前，**1号座位**和**N号座位**在空座位集合中被选中的概率始终是相等的。因此，在这两个座位中，1号座位先被选中的概率等于N号座位先被选中的概率。

$$P(\text{第N人坐对}) = P(\text{1号座位先于N号座位被选中}) = \frac{1}{2}$$

结论： 概率为 0.5。

4 外星生物颜色

4.1 题目描述

红20，绿21，蓝22。规则：2个不同色 $\to$ 2个第三种色。问是否可能最后全为一种颜色？

4.2 解答与推导

设三种颜色的数量分别为 n_1, n_2, n_3。每次变换（例如红+绿→蓝），数量变化向量为 $(-1, -1, +2)$。考察模 3 的不变量。注意每次变化中：

$$\Delta n_i \equiv -1 \equiv 2 \pmod 3 \quad 或 \quad \Delta n_i = +2 \equiv 2 \pmod 3$$

这意味着在模 3 意义下，所有颜色的数量都在同时增加 2。更关键的不变量是**任意两种颜色的数量之差**（模 3）：

$$(n_1 - 1) - (n_2 - 1) = n_1 - n_2$$
$$(n_1 - 1) - (n_3 + 2) = n_1 - n_3 - 3 \equiv n_1 - n_3 \pmod 3$$

即：**任意两种颜色的数量差在模 3 下保持不变。**

初始状态：

$$n_{红} = 20 \equiv 2, \quad n_{绿} = 21 \equiv 0, \quad n_{蓝} = 22 \equiv 1$$

两两之差模 3 分别为 $2, 1, 1$（均不为 0）。

目标状态（假设全变成蓝色）：

$$n_{红} = 0, \quad n_{绿} = 0, \quad n_{蓝} = 63$$

此时 $n_{红} - n_{绿} = 0 \equiv 0 \pmod 3$。

由于初始状态 $n_{红} - n_{绿} = -1 \equiv 2 \neq 0$，故无法达到目标状态。

结论： 不可能。

5 比较排序的时间复杂度

5.1 题目描述

证明任何基于比较的排序算法，其最坏情况下的时间复杂度下界为 $O(n \log n)$。

5.2 证明

利用决策树模型（Decision Tree Model）：

1. 对于 n 个待排序元素，其全排列共有 $n!$ 种情况，这些是排序算法必须区分的输出状态（叶子节点）。

2. 比较排序的每一步操作（比较 $a < b$）将可能性空间一分为二。这就构成了一棵二叉树。

3. 设树的高度为 h（对应最坏情况下的比较次数）。二叉树最多有 2^h 个叶子节点。

4. 为了覆盖所有可能的排列，必须满足：

$$2^h \geq n!$$

5. 两边取对数：

$$h \geq \log_2(n!)$$

6. 根据斯特林公式（Stirling's Formula），$\ln(n!) \approx n \ln n - n$。

$$\log_2(n!) \approx n \log_2 n$$

结论： 下界为 $\Omega(n \log n)$。

6 交易逆序对总数

6.1 题目描述

输入股票记录，若前一天股价高于后一天股价，则构成“交易逆序对”。求逆序对总数。数据范围：$N \leq 50000$。

6.2 解答与算法

直接使用双重循环暴力求解的复杂度是 $O(N^2)$，对于 $N = 50000$ 会超时。必须使用 $O(N \log N)$ 的解法，通常使用**归并排序（Merge Sort）**的改版。

算法思路： 在归并排序的 ‘merge‘ 阶段：

- 假设我们将左右两个有序数组 ‘L‘ 和 ‘R‘ 合并。
- 双指针 ‘i‘ 指向 ‘L‘，‘j‘ 指向 ‘R‘。
- 如果 $L[i] > R[j]$：说明 $L[i]$ 比 $R[j]$ 大。
- 关键点：因为 ‘L‘ 是有序的，所以 $L[i]$ 之后的所有元素（共 ‘mid - i + 1‘ 个）都必定大于 $R[j]$。
- 这些都构成了逆序对，将数量加到总数中，并将 $R[j]$ 放入临时数组。

6.3 代码实现 (Python)

```python
class Solution:
    def reversePairs(self, record: list[int]) -> int:
        self.count = 0
        self.merge_sort(record, 0, len(record) - 1)
        return self.count

    def merge_sort(self, nums, left, right):
        if left >= right:
```

```
            return

        mid = (left + right) // 2
        self.merge_sort(nums, left, mid)
        self.merge_sort(nums, mid + 1, right)
        self.merge(nums, left, mid, right)

    def merge(self, nums, left, mid, right):
        temp = []
        i, j = left, mid + 1

        while i <= mid and j <= right:
            if nums[i] <= nums[j]:
                temp.append(nums[i])
                i += 1
            else:
                # 48□□□□□□nums                                  [i] > nums[j]
                # 48□□□□□□□□□                                  i 48     mid 的所有元素都比 nums[j]
```

大self.count += (mid - i + 1)temp.append(nums[j])j += 1 处理剩余元素while i <= mid:temp.append(nums[i])i += 1while j <= right:temp.append(nums[j])j += 1 拷回原数组for k in range(len(temp)):nums[left + k] = temp[k]