QTA2025 暑期求职笔试训练营-week1 题目

作答区

2025 年 7 月 21 日

题目

问题 1. 单败淘汰赛

64 个人进行单败淘汰制竞赛 (每一轮随机分组两两 PK, 决出一位胜者进入下一轮)

1. 假如 64 个人实力有严格顺序, 实力强的人一定赢实力弱的人, 问最强的人甲、第二强的人乙在决赛相遇的概率是多少?
2. 假如 64 个人实力相当, 两两相遇胜率均为五五开, 问任意两人甲、乙在决赛相遇的概率是多少?

答：对于一个 64 人的单败淘汰赛：

- 总共需要进行 $64-1=63$ 场比赛。总共有 $\log_2(64)=6$ 轮比赛。
- 一个选手要赢得冠军，需要连赢 6 场。要进入决赛，需要连赢 5 场。

为了让甲和乙在决赛相遇，一个**必要的前提条件**是：**他们必须被分在整个赛区的不同半区**。

整个赛区有 64 个位置，可以被分成两个各有 32 个位置的上半区和下半区。

如果甲和乙在同一个半区，他们必然会在决赛前相遇，其中一人会被淘汰，因此他们不可能在决赛相遇。只有当他们分处不同半区时，才**有可能**在决赛相遇，故而我们首先需要考虑甲和乙分在不同半区的概率我们可以先计算这个前提条件的概率：

$$P(\text{甲乙在不同半区})=\frac{\text{另一半区的位置数}}{\text{总剩余位置数}}=\frac{32}{63}$$

现在，我们基于这个共同的前提来分析两种不同情况。

1 情况一：实力有严格顺序

步骤 1：计算甲乙被分在不同半区的概率。如上所述，这个概率是 $\frac{32}{63}$。

步骤 2：分析在该前提下，两人能否进入决赛。

- 甲是所有 64 人中实力最强的。一旦他被分入一个半区，无论对手是谁，他都必然会赢。因此，只要甲乙不在同一个半区，甲就**一定**会赢得他所在半区的所有比赛，进入决赛。这个概率是 1。
- 乙是第二强的选手。只要最强的甲不在他所在的半区, 则乙进入决赛。这个概率也是 1。

步骤 3：合并概率。 甲乙在决赛相遇的概率 = (甲乙在不同半区的概率) × (在该前提下甲进入决赛的概率) × (在该前提下乙进入决赛的概率)。

$$\begin{aligned} P(\text{甲乙决赛相遇}) &= P(\text{甲乙在不同半区}) \times P(\text{甲进决赛}|\text{不同半区}) \times P(\text{乙进决赛}|\text{不同半区}) \\ &= \frac{32}{63} \times 1 \times 1 = \frac{32}{63} \end{aligned}$$

结论

假如实力强的人一定赢，那么甲和乙在决赛相遇的概率即是不分在同一个半区的概率是 $\frac{32}{63}$。

2 情况二：实力相当

问题

任意两人甲、乙在决赛相遇的概率是多少?

类似的分析可知：甲乙在决赛相遇的概率 = (甲乙在不同半区的概率) × (甲在他所在半区获胜的概率) × (乙在他所在半区获胜的概率)。

$$\begin{aligned} P(\text{甲乙决赛相遇}) &= P(\text{甲乙在不同半区}) \times P(\text{甲连赢 5 场}) \times P(\text{乙连赢 5 场}) \\ &= \frac{32}{63} \times \frac{1}{32} \times \frac{1}{32} = \frac{1}{63 \times 32} = \frac{1}{2016} \end{aligned}$$

结论

假如所有人实力相当，任意两人甲、乙在决赛相遇的概率是 $\frac{1}{2016}$。

问题 2. 离散停问题

一直生成 $(0,1)$ 上独立同分布的随机数, 直到新的数比上一个小就停止生成, 问生成随机数个数的期望?

依据问题，我们考虑一个离散时间随机过程 $\{X_n\}_{n\geq 1}$，其中 X_n 是一系列独立同分布(i.i.d.)的随机变量，服从于状态空间 $S=(0,1)$ 上的均匀分布，即 $X_n \sim U(0,1)$。定义 $\mathcal{F}_n = \sigma(X_1, X_2, \ldots, X_n)$ 为该过程的自然信息流。$\mathcal{F}_n$ 代表了截至时间 n 的所有历史信息。

我们感兴趣的随机变量 N（生成的数的个数）是一个关于信息流 $\{\mathcal{F}_n\}$ 的停止时。其定义为:

$$N := \inf\{n \geq 2 : X_n < X_{n-1}\}$$

我们需要求解期望停止时生成数的个数的期望 $\mathbb{E}[N]$，当生成数为正整数时，其期望可以表示为:

$$\mathbb{E}[N] = \sum_{k=1}^{\infty} P(N \geq k)$$

这里的 $P(N \geq k)$ 是过程在时间 $k-1$ 时的存活概率。

过程存活至时间 k（即 $N \geq k$）的条件是，在所有 $j = 2, \ldots, k-1$ 时刻均未停止。这等价于前 $k-1$ 个随机变量构成了一个严格递增的序列。

$$\{N \geq k\} \iff \{X_1 < X_2 < \cdots < X_{k-1}\}$$

该事件是关于过程样本路径 $(X_1, \ldots, X_{k-1})$ 的一个性质。由于 $\{X_n\}$ 是独立同分布的，其联合分布具有排列对称性，任意一种特定排序的概率均为 $\frac{1}{(k-1)!}$。

$$P(N \geq k) = P(X_1 < X_2 < \cdots < X_{k-1}) = \frac{1}{(k-1)!}$$

这个公式对所有 $k \geq 1$ 均成立，严格的数学计算见附录

因此，停止时的期望为:

$$\begin{aligned}\mathbb{E}[N] &= \sum_{k=1}^{\infty} P(N \geq k) \\ &= \sum_{k=1}^{\infty} \frac{1}{(k-1)!} \\ &= \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots\end{aligned}$$

令 $j = k - 1$，上式变为:

$$\mathbb{E}[N] = \sum_{j=0}^{\infty} \frac{1}{j!} = e$$

问题 3. 次品称重问题

你有 12 个铁球, 其中包含一个次品球是比其它轻或者比其它重的, 给你一个天平, 最少称重几次可以保证找出这个球?

答：最少称重三次可以保证找出这个球

我们将 12 个球编号为 1, 2, 3, ..., 12。

第一次称重我们将球分成 ABC 三组，每组 4 个，并首先称重 A 和 B，

A 组：1, 2, 3, 4 B 组：5, 6, 7, 8 C 组：9, 10, 11, 12 将其进行称重，有三种情况，

- 情况一：如果 A 和 B 的重量相等，则次品球在 C 组中。将 C 组中的球分成两组，每组 2 个，随机取一组如 11,12，并取 A 组两个球作为标准球进行称重，有三种情况
  - 如果两组重量相等，则次品球在 9,10 中，那么将球 9 与标准球比较，如果球 9 比标准球重或者轻，则球 9 为次品球，如果球 9 与标准球重量相等，则球 10 为次品球。
  - 如果两组重量不相等，则次品球在 11,12 中，类似操作可确定次品球。
- 情况二：如果 A 和 B 的重量不相等，且 A 较重我们选择 1,2,5((2 个重嫌疑，1 个轻嫌疑)) 和 3,6,9 (1 个重嫌疑，1 个轻嫌疑，1 个标准球) 进行称重
  - 如果 1,2,5 较重，则次品球在 1,2,6 中，而且是 1，2 可能偏重，6 可能偏轻，再称 1,2, 如果相等，6 为次品球，如果不等，较重的为次品球。
  - 如果 1,2,5 较轻，则次品球在 5,3 中，那么将 9 与 5 比较即可。
  - 如果 1,2,5 和 3,6,9 重量相等，则次品球在 4,7,8 中，称重 7,8，如果相等，4 为次品球，如果不等，较轻的为次品球。
- 情况三：如果 A 和 B 的重量不相等，且 A 较轻此时可与情况二做类似操作，仍然是三次可以找出。

问题 4. 领导的红包

时近春节, 部门领导要给下属发年终奖。你的领导拿着三个红包 A、B、C, 其中只有一个红包是 200 元的, 其余两个都是 50 元, 让你从中挑选一个红包, 但是不能打开, 假设你选择了 A 红包, 然后领导当着你的面拆开了 B 红包, 发现里面有 50 元, 之后领导问你是否要用手中的 A 红包换剩下的 C 红包? 假设领导知道每个红包内的钱数且一定会打开 50 元的红包。请给出你的思路和理由。

问题 5. 幽灵宝藏

10 个房间排成一条直线, 只有一个房间有宝藏。每天宝藏都会移动到某个相邻房间。每天只能检查一个房间, 请问至少需要多少天可以确保找到宝藏? 给出你的策略。

至少需要 16 天可以确保找到宝藏。

现将房间命名为 1，2，3，4，5，6，7，8，9，10，因为宝藏每一次移动一定改变房间号的奇偶性，所以宝藏的房间号一定在 1-10 之间，与此同时为了不引起歧义，我们考虑宝藏在搜查第一天选定一个房间，并不进行移动，其与宝藏第一天开始移动的问题是等价的。如果我们假定宝藏的房间号在为偶数，那么我们依此搜寻 2,3,4,5,6,7,8,9 则一定能找到宝藏（因为我们每次搜寻的房间号和从偶数房间号出发的宝藏保持相同奇偶性，而我们这样搜查了所有这样的房间）如果搜寻 8 次后，仍然没发现宝藏，说明其从奇数房间出发，那么我们接下来遍历从奇数房间出发的所有可能即可。

具体来说，我们已经搜寻了八次，那么宝藏下一次检查时位于奇数房间（故而排除此时宝藏在房间 10），故而我们可以选择 9,8,7,6,5,4,3,2 作为我们接下来搜寻的房间（最后一个搜查的房间为 2，故而排除最后一天检查时位于房间 1 的情形），遍历所有从奇数房间房间出发的可能，而又因为每一次检查时与宝藏时同奇偶的，故而一定能找到宝藏。

接下来我们从一般意义上来说 16 天时确保找到宝藏的次数最小值。构造的策略如下：

1. **前进阶段 (共 $n-2$ 天)**: 从第 1 天到第 $n-2$ 天，依次搜查房间 $2, 3, \ldots, n-1$。
2. **返回阶段 (共 $n-2$ 天)**: 接着，从第 $n-1$ 天到第 $2n-4$ 天，依次搜查房间 $n-1, n-2, \ldots, 2$。

前进和返回阶段一共 2n-4 天，类似上面的分析可以知道，这个可以再第一阶段排除偶数房间移动的宝藏，第二阶段最终找到相应的宝藏。下面是证明 2n-4 天确实是这类问题的一个下届，即要确保找到宝藏，至少需要 2n-4 天。首先我们提出必胜策略的条件：对于任意一个内部房间 i，一个必胜的搜查方案必须满足以下两个条件：

- 至少在一个奇数天搜查过房间 i。
- 至少在一个偶数天搜查过房间 i。

我们使用反证法证明，假设存在一个必胜的搜查方案 $C = (C_1, C_2, \ldots, C_N)$，但它**从未在任何偶数天搜查过**某个特定的内部房间 i。即，对于所有奇数 d, $C_d \neq i$。我们设计一个宝藏的移动路径，使其永远不被抓住，从而与"C 是必胜方案"的前提产生矛盾。

- **第 1 天 (初始)**: 宝藏选择初始房间 $T_1 = i + -1$，使得宝藏不会在第一天被找到
- **第 2 天 (偶数天)**: 宝藏移动到 $T_2 = i$。由于第 2 天是偶数天，根据我们的假设，宝藏不会被找到。
- **第 3 天 (奇数天)**: 宝藏必须移动到相邻房间，即 $T_2 \in \{i-1, i+1\}$。由于我们的搜查方案在第 2 天只能检查一个房间 C_2，宝藏总可以选择未被搜查的那个邻近房间作为自己的位置。因此，宝藏总可以避免被找到。
- **第 4 天 (偶数天)**: 小偷从 T_2 移回至 $T_3 = i$。由于第 4 天是偶数天，根据假设，宝藏仍然不会被找到。

而第一个和最后一个为单侧的位置，故而可以不考虑，从而可以证明 2n-4 是这个问题答案的一个下界。

至少在一个偶数天搜查过房间 i。

问题 6. 归并排序的复杂度

侯哥在一台电脑上编写了一个程序对 256 个字符串进行排序 (采用 MergeSort 归并排序) 用时 2 秒钟, 那么用这个程序在给 512 个字符串进行排序的期望时间是多少秒?

这个题需要考虑归并排序算法的复杂度计算，在归并排序中我们先将列表二分到仅有一个元素，故而树的层数是 $log_2(n)$ 而在合并的每一层我们需要做 n 次操作，故而归并排序的复杂度为 $O(nlog_2(n))$。程序的运行时间 T(n) 与其算法的操作总量成正比，故而我们可以得到:

$$T(n) = k \times nlog_2(n)$$

其中 k 是常数，n 是字符串的数量。对于 256 个字符串，程序运行时间为 2 秒，故而:

$$T(512) = k \times 512 \times log_2(512) = k \times 512 \times 9$$

$$T(256) = k \times 256 \times log_2(256) = k \times 512 \times 8$$

$$\frac{T(512)}{T(256)} = \frac{512 \times 9}{256 \times 8} = 2.25$$

$$T(512) = 2.25 \times T(256) = 2.25 \times 2 = 4.5$$

故而程序用时共 4.5 秒。

问题 7. Python 小坑

python 语法中的一些细节规则是量化 OA 选择题中经常出现的考察点

1. python 的 copy 函数

   下述代码的输出结果是什么?

```
import copy
l=[1,2,3]
x={'a':l}
y= copy.copy (x)
y['a'][0]=11
print(x['a'][0])
```

输出结果为：[11]

原因：这是因为 copy.copy 方法执行的是浅拷贝，其创建了一个新的字典，但字典内部的元素是直接复制原容器（x）的引用，故而，x，y 是两个不同的对象，但是其中的值是相同的，故而对 y 的值进行新的赋值，y 的值也会发生改变。如果要使得对 y 的赋值不改变 x 的值，应该使用深拷贝，y = copy.deepcopy(x)。

2. python 函数中的默认参数值

下述代码会输出什么结果?

```
def fast(items=[]):
    items.append(1)
    return items

print(fast())
print(fast())
```

A. [1,1] [1,1] B. [1] [1] C. [1,1] [1] D. [1] [1,1]

输出结果为：D

原因：在 Python 中，函数也是对象，当第一次调用时，函数 fast 执行时，Python 会在内存中创建一个空的列表对象 []，并将这个唯一的列表对象的引用，在 Python 中，函数也是对象，当第一次调用时，函数 fast 执行时，Python 会在内存中创建一个空的列表对象 []，并将这个唯一的列表对象的引用，而在函数定义时创建后，就会一直存 D。直到程序结束。第一次调用时，item 首先创建一个空列表，函数 fast 执行时，items 添加了元素，再次调用时，items 列表对象的默认值变成 [1]，执行函数后，变成 [1,1]，故而选 D

3 附录

3.1 均匀分布推导

我们旨在计算概率 $P(N \geq k) = P(X_1 < X_2 < \cdots < X_{k-1})$，其中 $X_i \sim U(0,1)$ 且相互独立。该概率等价于计算下面这个 $(k-1)$ 维多重积分：

$$P(N \geq k) = \int_0^1 \int_0^{x_{k-1}} \int_0^{x_{k-2}} \cdots \int_0^{x_2} 1 \, dx_1 \, dx_2 \cdots dx_{k-2} \, dx_{k-1}$$

其具体的推导过程如下：

$$P(N \geq k) = \int_0^1 \int_0^{x_{k-1}} \cdots \int_0^{x_3} \int_0^{x_2} 1 \, dx_1 \, dx_2 \cdots dx_{k-1}$$

$$= \int_0^1 \int_0^{x_{k-1}} \cdots \int_0^{x_3} \left([x_1]_0^{x_2} \right) \, dx_2 \cdots dx_{k-1} \qquad \text{第一步：对 } x_1 \text{ 积分}$$

$$= \int_0^1 \int_0^{x_{k-1}} \cdots \int_0^{x_3} x_2 \, dx_2 \cdots dx_{k-1}$$

$$= \int_0^1 \int_0^{x_{k-1}} \cdots \int_0^{x_4} \left(\left[\frac{x_2^2}{2} \right]_0^{x_3} \right) dx_3 \cdots dx_{k-1} \qquad \text{第二步：对 } x_2 \text{ 积分}$$

$$= \int_0^1 \int_0^{x_{k-1}} \cdots \int_0^{x_4} \frac{x_3^2}{2!} \, dx_3 \cdots dx_{k-1}$$

$$\vdots \qquad \text{重复此过程}$$

$$= \int_0^1 \frac{x_{k-1}^{k-2}}{(k-2)!} \, dx_{k-1} \qquad \text{在对 } x_{k-2} \text{ 积分后，得到的结果}$$

$$= \frac{1}{(k-2)!} \left[\frac{x_{k-1}^{k-1}}{k-1} \right]_0^1 \qquad \text{最后一步：对 } x_{k-1} \text{ 积分}$$

$$= \frac{1}{(k-2)!} \left(\frac{1^{k-1}}{k-1} - \frac{0^{k-1}}{k-1} \right)$$

$$= \frac{1}{(k-2)!} \cdot \frac{1}{k-1}$$

$$= \frac{1}{(k-1)!}$$