概率分析和随机算法

本章介绍**概率分析**和**随机算法**。如果你不熟悉概率论的基本知识，应先阅读附录 C，复习这部分材料。我们将在本书中多次提到概率分析和随机算法。

5.1 雇用问题

假如你要雇用一名新的办公助理。你先前的雇用尝试都失败了，于是你决定找一个雇用代理。雇用代理每天给你推荐一个应聘者。你面试这个人，然后决定是否雇用他。你必须付给雇用代理一小笔费用，以便面试应聘者。然而要真的雇用一个应聘者需要花更多的钱，因为你必须辞掉目前的办公助理，还要付一大笔中介费给雇用代理。你承诺在任何时候，都要找最适合的人来担任这项职务。因此，你决定在面试完每个应聘者后，如果该应聘者比目前的办公助理更合适，就会辞掉当前的办公助理，然后聘用新的。你愿意为该策略付费，但希望能够估算该费用会是多少。

下面给出的 HIRE-ASSISTANT 过程以伪代码表示该雇用策略。假设应聘办公助理的候选人编号为 1 到 n。该过程中假设你能在面试完应聘者 i 后，决定应聘者 i 是否是你目前见过的最佳人选。初始化时，该过程创建一个虚拟的应聘者，编号为 0，他比其他所有应聘者都差。 114

```
HIRE-ASSISTANT(n)
1  best = 0          // candidate 0 is a least-qualified dummy candidate
2  for i = 1 to n
3      interview candidate i
4      if candidate i is better than candidate best
5          best = i
6          hire candidate i
```

这个问题的费用模型与第 2 章中描述的模型不同。我们关注的不是 HIRE-ASSISTANT 的执行时间，而是面试和雇用所产生的费用。表面上看起来，分析这个算法的费用与分析归并排序等的运行时间有很大不同。然而，我们在分析费用或者分析运行时间时，所采用的分析技术却是相同的。在任何情形中，我们都是在计算特定基本操作的执行次数。

面试的费用较低，比如为 c_i，然而雇用的费用较高，设为 c_h。假设 m 是雇用的人数，那么该算法的总费用就是 $O(c_i n + c_h m)$。不管雇用多少人，我们总会面试 n 个应聘者，于是面试产生的费用总是 $c_i n$。因此，我们只关注于分析 $c_h m$，即雇用的费用。这个量在该算法的每次执行中都不同。

这个场景用来作为一般计算范式的模型。我们通常通过检查序列中的每个成员，并且维护一个当前的“获胜者”，来找出序列中的最大值或最小值。这个雇用问题对当前获胜成员的更新频率建立模型。

最坏情形分析

在最坏情况下，我们实际上雇用了每个面试的应聘者。当应聘者质量按出现的次序严格递增时，这种情况就会出现，此时雇用了 n 次，总的费用是 $O(c_h n)$。

当然，应聘者并非总以质量递增的次序出现。事实上，我们既不知道他们出现的次序，也不能控制这个次序。因此，很自然地会问在一种典型或者平均的情形下，会有什么发生。

概率分析

概率分析是在问题分析中应用概率的理念。大多数情况下，我们采用概率分析来分析一个
115 算法的运行时间，有时也用它来分析其他的量，例如，过程 HIRE-ASSISTANT 中的雇用费用。为了进行概率分析，我们必须使用或者假设关于输入的分布。然后分析该算法，计算出一个平均情形下的运行时间，其中我们对所有可能的输入分布取平均值。因此，实际上，我们对所有可能输入产生的运行时间取平均。当报告此种类型的运行时间时，我们称其为**平均情况运行时间**。

我们在确定输入分布时必须非常小心。对于某些问题，我们可以对所有可能的输入集合做某种假定，然后采用概率分析来设计一个高效算法，并加深对问题的认识。对于其他一些问题，我们不能描述一个合理的输入分布，此时就不能采用概率分析。

在雇用问题中，我们可以假设应聘者以随机顺序出现。这一假设意味着什么？假定可以对任何两个应聘者进行比较，并决定哪一个更有资格；也就是说，所有应聘者存在一个全序关系(全序的定义可参见附录 B)。因此，可以使用从 1 到 n 的唯一号码对应聘者排列名次，用 $rank(i)$ 表示应聘者 i 的名次，并照常约定一个较高名次对应一个更好的应聘者。有序序列$\langle rank(1), rank(2), \cdots, rank(n)\rangle$是序列$\langle 1, 2, \cdots, n\rangle$的一个排列。称应聘者以随机顺序出现，等价于称这个排名列表是数字 1 到 n 的 $n!$ 种排列表中的任一个。或者，我们也称这些排名构成一个**均匀随机排列**；也就是说，在 $n!$ 种可能的排列中，每一种以等概率出现。

5.2 节包含这个雇用问题的一个概率分析。

随机算法

为了利用概率分析，我们需要了解关于输入分布的一些信息。在许多情况下，我们对输入分布了解很少。即使知道输入分布的某些信息，也可能无法从计算上对该分布知识建立模型。然而，我们通过使一个算法中某部分的行为随机化，常可以利用概率和随机性作为算法设计和分析的工具。

在雇用问题中，看起来应聘者好像以随机顺序出现，但我们无法知道是否的确如此。因此，为了设计雇用问题的一个随机算法，我们必须对面试应聘者的次序有更大的控制。所以，稍稍改变这个模型。假设雇用代理有 n 个应聘者，而且他们事先给我们一份应聘者名单。每天随机选择
116 某个应聘者来面试。尽管除了应聘者的名字外对其他信息一无所知，但我们已经做了一个显著的改变。不是像以前依赖于猜测应聘者以随机次序出现，取而代之，我们获得了对流程的控制并且加强了随机次序。

更一般地，如果一个算法的行为不仅由输入决定，而且也由**随机数生成器**(random-number generator)产生的数值决定，则称这个算法是**随机的**(randomized)。我们将假设有一个可以自由使用的随机数生成器 RANDOM。调用 RANDOM(a, b)将返回一个介于 a 和 b 之间的整数，并且每个整数以等概率出现。例如，RANDOM(0, 1)产生 0 的概率是 1/2，产生 1 的概率也是1/2。调用 RANDOM(3, 7)将返回 3、4、5、6 或 7，每个出现的概率都是 1/5。每次 RANDOM 返回的整数独立于前面调用的返回值。可以将 RANDOM 想象成掷一个($b-a+1$)面的骰子，获得出现的点数。(在实践中，大多数编程环境会提供一个**伪随机数生成器**，它是一个确定性算法，返回值在统计上看起来是随机的。)

当分析一个随机算法的运行时间时，我们以运行时间的期望值衡量，其中输入值由随机数生成器产生。我们将一个随机算法的运行时间称为**期望运行时间**，以此来区分这类算法和那些输入是随机的算法。一般而言，当概率分布是在算法的输入上时，我们讨论的是平均情况运行时间；当算法本身做出随机选择时，我们讨论其期望运行时间。

练习

5.1-1 证明：假设在过程 HIRE-ASSISTANT 的第 4 行中，我们总能决定哪一个应聘者最佳，

则意味着我们知道应聘者排名的全部次序。

***5.1-2**　请描述 RANDOM(a，b)过程的一种实现，它只调用 RANDOM(0，1)。作为 a 和 b 的函数，你的过程的期望运行时间是多少？

***5.1-3**　假设你希望以 1/2 的概率输出 0 与 1。你可以自由使用一个输出 0 或 1 的过程 BIASED-RANDOM。它以某概率 p 输出 1，概率 $1-p$ 输出 0，其中 $0<p<1$，但是 p 的值未知。请给出一个利用 BIASED-RANDOM 作为子程序的算法，返回一个无偏的结果，能以概率 1/2 返回 0，以概率 1/2 返回 1。作为 p 的函数，你的算法的期望运行时间是多少？ 117

5.2　指示器随机变量

为了分析雇用问题在内的许多算法，我们采用指示器随机变量(indicator random variable)。它为概率与期望之间的转换提供了一个便利的方法。给定一个样本空间 S 和一个事件 A，那么事件 A 对应的**指示器随机变量** $\mathrm{I}\{A\}$ 定义为：

$$\mathrm{I}\{A\} = \begin{cases} 1 & \text{如果 } A \text{ 发生} \\ 0 & \text{如果 } A \text{ 不发生} \end{cases} \tag{5.1}$$

举一个简单的例子，我们来确定抛掷一枚标准硬币时正面朝上的期望次数。样本空间为 $S=\{H, T\}$，其中 $\Pr\{H\}=\Pr\{T\}=1/2$。接下来定义一个指示器随机变量 X_H，对应于硬币正面朝上的事件 H。这个变量计数抛硬币时正面朝上的次数，如果正面朝上则值为 1，否则为 0。我们记成：

$$X_H = \mathrm{I}\{H\} = \begin{cases} 1 & \text{如果 } H \text{ 发生} \\ 0 & \text{如果 } T \text{ 发生} \end{cases}$$

在一次抛掷硬币时，正面朝上的期望次数就是指示器变量 X_H 的期望值：

$$\begin{aligned} \mathrm{E}[X_H] &= \mathrm{E}[\mathrm{I}\{H\}] = 1 \cdot \Pr\{H\} + 0 \cdot \Pr\{T\} \\ &= 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2 \end{aligned}$$

因此抛掷一枚标准硬币时，正面朝上的期望次数是 1/2。如下面引理所示，一个事件 A 对应的指示器随机变量的期望值等于事件 A 发生的概率。

引理 5.1　给定一个样本空间 S 和 S 中的一个事件 A，设 $X_A=\mathrm{I}\{A\}$，那么 $\mathrm{E}[X_A]=\Pr\{A\}$。 118

证明　由等式(5.1)指示器随机变量的定义，以及期望值的定义，我们有

$$\mathrm{E}[X_A] = \mathrm{E}[\mathrm{I}\{A\}] = 1 \cdot \Pr\{A\} + 0 \cdot \Pr\{\overline{A}\} = \Pr\{A\}$$

其中 $\overline{A}$ 表示 $S-A$，即 A 的补。 ■

虽然指示器随机变量看起来很麻烦，比如在计算单枚硬币一次投掷的正面次数期望时，但是它在分析重复随机试验时是有用的。例如，指示器随机变量为我们求等式(C.37)的结果提供了一个简单方法。在这个等式中，我们分别考虑出现 0 个、1 个、2 个…正面朝上的概率，以计算抛 n 次硬币时正面朝上的次数。等式(C.38)中给出了简单方法，隐含使用了指示器随机变量。为使讨论更清楚，我们设指示器随机变量 X_i 对应第 i 次抛硬币时正面朝上的事件：$X_i=\mathrm{I}\{$第 i 次抛掷时出现事件 $H\}$。设随机变量 X 表示 n 次抛硬币中出现正面的总次数，于是

$$X = \sum_{i=1}^{n} X_i$$

我们希望计算正面朝上次数的期望，所以对上面等式两边取期望，得到

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{n} X_i\right]$$

上面等式给出了 n 个指示器随机变量总和的期望值。由引理 5.1，我们容易计算出每个随机变量的期望值。根据反映期望线性性质的等式(C.21)，容易计算出总和的期望值：它等于 n 个随机变量期望值的总和。期望的线性性质利用指示器随机变量作为一种强大的分析技术；当随机变量

119 之间存在依赖关系时也成立。现在我们可以轻松地计算正面出现次数的期望：

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{E}[X_i] = \sum_{i=1}^{n} 1/2 = n/2$$

因此，和等式(C.37)中用到的方法相比，指示器随机变量极大地简化了计算过程。我们将在本书中一直采用指示器随机变量。

用指示器随机变量分析雇用问题

返回到雇用问题上来。我们希望计算雇用一个新的办公助理的期望次数。为了利用概率分析，假设应聘者以随机顺序出现，如前一节所述。(我们将看到在5.3节如何去除这个假设。)设X是一个随机变量，其值等于我们雇用一个新办公助理的次数。然后，应用等式(C.20)中期望值的定义，得到

$$\mathrm{E}[X] = \sum_{x=1}^{n} x\Pr\{X = x\}$$

但是这种计算会很麻烦。取而代之，我们将采用指示器随机变量来大大简化计算。

为了利用指示器随机变量，我们不是通过定义与雇用一个新办公助理所需次数对应的一个变量来计算$\mathrm{E}[X]$，而是定义n个变量，与每个应聘者是否被雇用对应。特别地，假设X_i对应于第i个应聘者被雇用该事件的指示器随机变量。因而，

$$X_i = \mathrm{I}\{\text{应聘者 } i \text{ 被雇用}\} = \begin{cases} 1 & \text{如果应聘者 } i \text{ 被雇用} \\ 0 & \text{如果应聘者 } i \text{ 不被雇用} \end{cases}$$

以及

120

$$X = X_1 + X_2 + \cdots + X_n \tag{5.2}$$

根据引理5.1，我们有

$$\mathrm{E}[X_i] = \Pr\{\text{应聘者 } i \text{ 被雇用}\}$$

因此必须计算HIRE-ASSISTANT中第5～6行被执行的概率。

在第6行中，应聘者i被雇用，正好应聘者i比从1到$i-1$的每一个应聘者优秀。因为我们已经假设应聘者以随机顺序出现，所以前i个应聘者也以随机次序出现。这些前i个应聘者中的任意一个都等可能地是目前最有资格的。应聘者i比应聘者1到$i-1$更有资格的概率是$1/i$，因而也以$1/i$的概率被雇用。由引理5.1，可得

$$\mathrm{E}[X_i] = 1/i \tag{5.3}$$

现在可以计算$\mathrm{E}[X]$：

$$\begin{aligned} \mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] && \text{(根据等式(5.2))} \qquad (5.4) \\ &= \sum_{i=1}^{n} \mathrm{E}[X_i] && \text{(根据期望的线性性质)} \\ &= \sum_{i=1}^{n} 1/i && \text{(根据等式(5.3))} \\ &= \ln n + O(1) && \text{(根据等式(A.7))} \qquad (5.5) \end{aligned}$$

尽管我们面试了n个人，但平均起来，实际上大约只雇用他们之中的$\ln n$个人。我们用下面的引理来总结这个结果。

引理5.2 *假设应聘者以随机次序出现，算法HIRE-ASSISTANT总的雇用费用平均情形下为$O(c_h \ln n)$。*

证明 根据雇用费用的定义和等式(5.5)，可以立即推出这个界，说明雇用的人数期望值大约是$\lg n$。 ■

121 平均情形下的雇用费用比最坏情况下的雇用费用$O(c_h n)$有了很大的改进。

练习

5.2-1　在 HIRE-ASSISTANT 中，假设应聘者以随机顺序出现，你正好雇用一次的概率是多少？正好雇用 n 次的概率是多少？

5.2-2　在 HIRE-ASSISTANT 中，假设应聘者以随机顺序出现，你正好雇用两次的概率是多少？

5.2-3　利用指示器随机变量来计算掷 n 个骰子之和的期望值。

5.2-4　利用指示器随机变量来解如下的**帽子核对问题**(hat-heck problem)：n 位顾客，他们每个人给餐厅核对帽子的服务生一顶帽子。服务生以随机顺序将帽子归还给顾客。请问拿到自己帽子的客户的期望数是多少？

5.2-5　设 $A[1..n]$是由 n 个不同数构成的数列。如果 $i<j$ 且 $A[i]>A[j]$，则称(i, j)对为 A 的一个**逆序对**(inversion)。(参看思考题 2-4 中更多关于逆序对的例子。)假设 A 的元素构成$\langle 1, 2, \cdots, n\rangle$上的一个均匀随机排列。请用指示器随机变量来计算其中逆序对的数目期望。

5.3　随机算法

在前面一节中，我们已说明了输入的分布是如何有助于分析一个算法的平均情况行为。许多时候，我们无法得知输入分布的信息，因而阻碍了平均情况分析。如 5.1 节中所提及，我们也许可以采用一个随机算法。

对于诸如雇用问题之类的问题，其中假设输入的所有排列等可能出现往往有益，通过概率分析可以指导设计一个随机算法。我们不是假设输入的一个分布，而是设定一个分布。特别地，在算法运行前，先随机地排列应聘者，以加强所有排列都是等可能出现的性质。尽管已经修改了这个算法，我们仍希望雇用一个新的办公助理大约需要 $\ln n$ 次期望值。但是现在我们期望对于所 122
有的输入它都是这种情况，而不是对于一个具有特别分布的输入。

我们来进一步探索概率分析和随机算法之间的区别。在 5.2 节中，我们断言：如果应聘者以随机顺序出现，则聘用一个新办公助理的期望次数大约是 $\ln n$。注意，这个算法是确定性的；对于任何特定输入，雇用一个新办公助理的次数始终相同。此外，我们雇用一个新办公助理的次数将因输入的不同而不同，而且依赖于各个应聘者的排名。既然次数仅依赖于应聘者的排名，我们可以使用应聘者的有序排名列表来代表一个特定的输入，例如$\langle rank(1), rank(2), \cdots, rank(n)\rangle$。给定排名列表 $A_1=\langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\rangle$，一个新的办公助理会雇用 10 次，因为每一个后来应聘者都优于前一个，在算法的每次迭代中，第 5～6 行都要被执行。给定排名列表 $A_2=\langle 10, 9, 8, 7, 6, 5, 4, 3, 2, 1\rangle$，一个新的办公助理只雇用一次，在第一次迭代中。给定排名序列 $A_3=\langle 5, 2, 1, 8, 4, 7, 10, 9, 3, 6\rangle$，一个新的办公助理会雇用三次，即面试排名为 5、8 和 10 的 3 位应聘者。回顾一下，算法的费用依赖于雇用一个新办公助理的次数。我们可以看到，有昂贵的输入(如 A_1)、不贵的输入(如 A_2)，以及适中贵的输入(如 A_3)。

另外，考虑先对应聘者进行排列，然后确定最佳应聘者的随机算法。此时，我们让随机发生在算法上，而不是在输入分布上。给定一个输入，如上面的 A_3，我们无法说出最大值会被更新多少次，因为此变量在每次运行该算法时都不同。第一次在 A_3 上运行这个算法时，可能会产生排列 A_1 并执行 10 次更新；但第二次运行算法时，可能会产生排列 A_2 并只执行 1 次更新。第三次执行时，可能会产生其他次数的更新。每次运行这个算法时，执行依赖于随机选择，而且很可能和前一次算法的执行不同。对于该算法及许多其他的随机算法，*没有特别的输入会引出它的最坏情况行为*。即使你最坏的敌人也无法产生最坏的输入数组，因为随机排列使得输入次序不再相关。只有在随机数生成器产生一个“不走运”的排列时，随机算法才会运行得很差。

123 对于雇用问题，代码中唯一需要改变的是随机地变换应聘者序列。

```
RANDOMIZED-HIRE-ASSISTANT(n)
1  randomly permute the list of candidates
2  best = 0            // candidate 0 is a least-qualified dummy candidate
3  for i = 1 to n
4      interview candidate i
5      if candidate i is better than candidate best
6          best = i
7          hire candidate i
```

通过这个简单改变，我们已经建立了一个随机算法，其性能和假设应聘者以随机次序出现所得结果是匹配的。

引理 5.3 过程 RANDOMIZED-HIRE-ASSISTANT 的雇用费用期望是 $O(c_h \ln n)$。

证明 对输入数组进行变换后，我们已经达到了和 HIRE-ASSISTANT 概率分析时相同的情况。 ■

比较引理 5.2 和引理 5.3 突出了概率分析和随机算法的差别。在引理 5.2 中，我们在输入上做了一个假设。在引理 5.3 中，我们没有做这种假设，尽管随机化输入会花费一些额外时间。为了保持术语的一致性，我们用平均情形下的雇用费用来表达引理 5.2，而用期望雇用费用来表达引理 5.3。在本节余下部分里，我们讨论关于随机排列输入的一些议题。

随机排列数组

很多随机算法通过对给定的输入变换排列以使输入随机化。(还有其他使用随机化的方法。)这里，我们将讨论两种随机化方法。不失一般性，假设给定一个数组 A，包含元素 1 到 n。我们的目标是构造这个数组的一个随机排列。

一个通常的方法是为数组的每个元素 $A[i]$赋一个随机的优先级 $P[i]$，然后依据优先级对数组 A 中的元素进行排序。例如，如果初始数组 $A=\langle 1, 2, 3, 4\rangle$，随机选择的优先级 $P=\langle 36, 3, 62, 19\rangle$，则将产生一个数组 $B=\langle 2, 4, 1, 3\rangle$，因为第 2 个优先级最小，接下来是第 4 个，
124 然后第 1 个，最后第 3 个。我们称这个过程为 PERMUTE-BY-SORTING：

```
PERMUTE-BY-SORTING(A)
1  n = A.length
2  let P[1..n] be a new array
3  for i = 1 to n
4      P[i] = RANDOM(1, n^3)
5  sort A, using P as sort keys
```

第 4 行选取一个在 $1\sim n^3$ 之间的随机数。我们使用范围 $1\sim n^3$ 是为了让 P 中所有优先级尽可能唯一。(练习 5.3-5 要求读者证明所有元素都唯一的概率至少是 $1-1/n$，练习 5.3-6 问如何在两个或更多优先级相同的情况下，实现这个算法。)我们假设所有的优先级都唯一。

这个过程中耗时的步骤是第 5 行的排序。正如我们将在第 8 章看到的那样，如果使用比较排序，排序将花费 $\Omega(n \lg n)$时间。我们可以达到这个下界，因为我们已经看到归并排序时间代价为 $\Theta(n \lg n)$。(我们将在第二部分看到，其他的比较排序花费时间代价为 $\Theta(n \lg n)$。练习 8.3-4 要求读者解决一个非常类似的问题，在 $O(n)$时间内对 $0\sim n^3-1$ 范围之内的整数排序。)排序以后，如果 $P[i]$是第 j 个最小的优先级，那么 $A[i]$将出现在输出位置 j 上。用这种方式，我们得到了一个排列。还需要证明这个过程能产生一个**均匀随机排列**，即该过程等可能地产生数字 $1\sim n$ 的每一种排列。

引理 5.4 假设所有优先级都不同，则过程 PERMUTE-BY-SORTING 产生输入的均匀随机

排列。

证明　我们从考虑每个元素 $A[i]$分配到第 i 个最小优先级的特殊排列开始，并说明这个排列正好发生的概率是 $1/n!$。对 $i=1, 2, \cdots, n$，设 E_i 代表元素 $A[i]$分配到第 i 个最小优先级的事件。然后我们想计算对所有的 i，事件 E_i 发生的概率，即

$$\Pr\{E_1 \cap E_2 \cap E_3 \cap \cdots \cap E_{n-1} \cap E_n\}$$

运用练习 C.2-5，这个概率等于

$$\Pr\{E_1\} \cdot \Pr\{E_2 \mid E_1\} \cdot \Pr\{E_3 \mid E_2 \cap E_1\} \cdot \Pr\{E_4 \mid E_3 \cap E_2 \cap E_1\}$$
$$\cdots \Pr\{E_i \mid E_{i-1} \cap E_{i-2} \cap \cdots \cap E_1\} \cdots \Pr\{E_n \mid E_{n-1} \cap \cdots \cap E_1\}$$

因为 $\Pr\{E_1\}$是从一个 n 元素的集合中随机选取的优先级最小的概率，所以有 $\Pr\{E_1\}=1/n$。接下来，我们观察到 $\Pr\{E_2 \mid E_1\}=1/(n-1)$，因为假定元素 $A[1]$有最小的优先级，余下来的 $n-1$ 个 125
元素都有相等的可能成为第二小的优先级别。一般地，对 $i=2, 3, \cdots, n$，我们有 $\Pr\{E_i \mid E_{i-1} \cap E_{i-2} \cap \cdots \cap E_1\}=1/(n-i+1)$。因为给定元素 $A[1]$到 $A[i-1]$(按顺序)有前 $i-1$ 小的优先级，剩下的 $n-(i-1)$个元素中，每一个都等可能具有第 i 小优先级。所以有

$$\Pr\{E_1 \cap E_2 \cap E_3 \cap \cdots \cap E_{n-1} \cap E_n\} = \left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\cdots\left(\frac{1}{2}\right)\left(\frac{1}{1}\right) = \frac{1}{n!}$$

并且我们已说明，获得等同排列的概率是 $1/n!$。

我们可以扩展这个证明，使其对任何优先级的排列都有效。考虑集合$\{1, 2, \cdots, n\}$的任意一个确定排列 $\sigma=\langle\sigma(1), \sigma(2), \cdots, \sigma(n)\rangle$。我们用 r_i 表示赋予元素 $A[i]$优先级的排名，其中优先级第 j 小的元素名次为 j。如果定义 E_i 为元素 $A[i]$分配到优先级第 $\sigma(i)$小的事件，或者 $r_i=\sigma(i)$，则同样的证明仍适用。因此，如果要计算得到任何特定排列的概率，该计算与前面的计算完全相同，于是得到此排列的概率也是 $1/n!$。 ■

你可能会这样想，要证明一个排列是均匀随机排列，只要证明对于每个元素 $A[i]$，它排在位置 j 的概率是 $1/n$。练习 5.3-4 证明这个弱条件实际上并不充分。

产生随机排列的一个更好方法是原址排列给定数组。过程 RANDOMIZE-IN-PLACE 在 $O(n)$ 时间内完成。在进行第 i 次迭代时，元素 $A[i]$是从元素 $A[i]$到 $A[n]$中随机选取的。第 i 次迭代以后，$A[i]$不再改变。

```
RANDOMIZE-IN-PLACE(A)
1  n = A.length
2  for i = 1 to n
3      swap A[i] with A[RANDOM(i, n)]
```

我们将使用循环不变式来证明过程 RANDOMIZE-IN-PLACE 能产生一个均匀随机排列。一个具有 n 个元素的 **k 排列**(k-permutation)是包含这 n 个元素中的 k 个元素的序列，并且不重复(参见附录 C)。一共有 $n!/(n-k)!$ 种可能的 k 排列。 126

引理 5.5　过程 RANDOMIZE-IN-PLACE 可计算出一个均匀随机排列。

证明　我们使用下面的循环不变式：

在第 2～3 行 **for** 循环的第 i 次迭代以前，对每个可能的$(i-1)$排列，子数组 $A[1..i-1]$包含这个$(i-1)$排列的概率是$(n-i+1)!/n!$。

我们需要说明这个不变式在第 1 次循环迭代以前为真，循环的每次迭代能够维持此不变式，并且当循环终止时，这个不变式提供一个有用的性质来说明正确性。

初始化：考虑正好在第 1 次循环迭代以前的情况，此时 $i=1$。由循环不变式可知，对每个可能的 0 排列，子数组 $A[1..0]$包含这个 0 排列的概率是$(n-i+1)!/n! = n!/n! = 1$。子数组 $A[1..0]$是一个空的子数组，并且 0 排列也没有元素。因而，$A[1..0]$包含任何 0 排列的概率是

1，在第 1 次循环迭代以前循环不变式成立。

保持：我们假设在第 i 次迭代之前，每种可能的$(i-1)$排列出现在子数组 $A[1..i-1]$中的概率是$(n-i+1)!/n!$，我们要说明在第 i 次迭代以后，每种可能的 i 排列出现在子数组 $A[1..i]$中的概率是$(n-i)!/n!$。下一次迭代 i 累加后，还将保持这个循环不变式。

我们来检查第 i 次迭代。考虑一个特殊的 i 排列，并以$\langle x_1, x_2, \cdots, x_i\rangle$来表示其中的元素。这个排列中包含一个$(i-1)$排列$\langle x_1, x_2, \cdots, x_{i-1}\rangle$，后面接着算法在 $A[i]$里放置的值 x_i。设 E_1 表示前 $i-1$ 次迭代已经在 $A[1..i-1]$中构造了特殊$(i-1)$排列的事件。根据循环不变式，$\Pr\{E_1\}=(n-i+1)!/n!$。设 E_2 表示第 i 次迭代在位置 $A[i]$放置 x_i 的事件。当 E_1 和 E_2 恰好都发生时，i 排列$\langle x_1, \cdots, x_i\rangle$出现在 $A[1..i]$中，因此，我们希望计算 $\Pr\{E_2 \cap E_1\}$。利用等式(C.14)，我们有

$$\Pr\{E_2 \cap E_1\} = \Pr\{E_2 \mid E_1\}\Pr\{E_1\}$$

概率 $\Pr\{E_2 \mid E_1\}$等于 $1/(n-i+1)$，因为在算法第 3 行，从 $A[i..n]$的 $n-i+1$ 个值中随机选取
127 x_i。因此，我们有

$$\Pr\{E_2 \cap E_1\} = \Pr\{E_2 \mid E_1\}\Pr\{E_1\} = \frac{1}{n-i+1} \cdot \frac{(n-i+1)!}{n!} = \frac{(n-i)!}{n!}$$

终止：终止时，$i=n+1$，子数组 $A[1..n]$是一个给定 n 排列的概率为$(n-(n+1)+1)/n! = 0!/n! = 1/n!$。

因此，RANDOMIZE-IN-PLACE 产生一个均匀随机排列。 ■

一个随机算法通常是解决一个问题最简单、最有效的方法。我们将在本书中偶尔用到随机算法。

练习

5.3-1 Marceau 教授不同意引理 5.5 证明中使用的循环不变式。他对第 1 次迭代之前循环不变式是否为真提出质疑。他的理由是，我们可以很容易宣称一个空数组不包含 0 排列。因此，一个空的子数组包含一个 0 排列的概率应是 0，从而第 1 次迭代之前循环不变式无效。请重写过程 RANDOMIZE-IN-PLACE，使得相关循环不变式适用于第 1 次迭代之前的非空子数组，并为你的过程修改引理 5.5 的证明。

5.3-2 Kelp 教授决定写一个过程来随机产生除恒等排列(identity permutation)外的任意排列。他提出了如下过程：

```
PERMUTE-WITHOUT-IDENTITY(A)
1  n = A.length
2  for i = 1 to n - 1
3      swap A[i] with A[RANDOM(i + 1, n)]
```

这段代码实现了 Kelp 教授的意图吗？

5.3-3 假设我们不是将元素 $A[i]$与子数组 $A[i..n]$中的一个随机元素交换，而是将它与数组任
128 何位置上的随机元素交换：

```
PERMUTE-WITH-ALL(A)
1  n = A.length
2  for i = 1 to n
3      swap A[i] with A[RANDOM(1, n)]
```

这段代码会产生一个均匀随机排列吗？为什么会或为什么不会？

5.3-4 Armstrong 教授建议用下面的过程来产生一个均匀随机排列：

```
PERMUTE-BY-CYCLIC(A)
1  n = A.length
2  let B[1..n] be a new array
3  offset = RANDOM(1, n)
4  for i = 1 to n
5      dest = i + offset
6      if dest > n
7          dest = dest − n
8      B[dest] = A[i]
9  return B
```

请说明每个元素 $A[i]$出现在 B 中任何特定位置的概率是 $1/n$。然后通过说明排列结果不是均匀随机排列，表明 Armstrong 教授错了。

*5.3-5　证明：在过程 PERMUTE-BY-SORTING 的数组 P 中，所有元素都唯一的概率至少是 $1-1/n$。

5.3-6　请解释如何实现算法 PERMUTE-BY-SORTING，以处理两个或更多优先级相同的情形。也就是说，即使有两个或更多优先级相同，你的算法也应该产生一个均匀随机排列。

5.3-7　假设我们希望创建集合$\{1, 2, 3, \cdots, n\}$的一个**随机样本**，即一个具有 m 个元素的集合 S，其中 $0 \leqslant m \leqslant n$，使得每个 m 集合能够等可能地创建。一种方法是对 $i=1, 2, \cdots, n$ 设$A[i]=i$，调用 RANDOMIZE-IN-PLACE(A)，然后取最前面的 m 个数组元素。这种方法会对 RANDOM 过程调用 n 次。如果 n 比 m 大很多，我们能够创建一个随机样本，只对 RANDOM 调用更少的次数。请说明下面的递归过程返回$\{1, 2, 3, \cdots, n\}$的一个随机 m 子集 S，其中每个 m 子集是等可能的，然而只对 RANDOM 调用 m 次。 129

```
RANDOM-SAMPLE(m, n)
1  if m == 0
2      return ∅
3  else S = RANDOM-SAMPLE(m − 1, n − 1)
4      i = RANDOM(1, n)
5      if i ∈ S
6          S = S ∪ {n}
7      else S = S ∪ {i}
8      return S
```

*5.4　概率分析和指示器随机变量的进一步使用

本节通过 4 个例子进一步阐释概率分析。第 1 个例子确定在一个有 k 个人的屋子中，某两个人生日相同的概率。第 2 个例子讨论把球随机投入箱子的问题。第 3 个例子探究抛硬币时连续出现正面的情况。最后一个例子分析雇用问题的一个变形，其中你必须在没有面试所有的应聘者时做出决定。

5.4.1　生日悖论

我们的第一个例子是**生日悖论**。一个屋子里人数必须要达到多少人，才能使其中两人生日相同的机会达到 50%？这个问题的答案是一个很小的数值，让人吃惊。下面我们将看到，所出现的悖论在于，这个数目实际上远小于一年中的天数，甚至不足一年天数的一半。

为了回答这个问题，我们用整数 1，2，…，k 对屋子里的人编号，其中 k 是屋子里的总人数。另外，我们不考虑闰年的情况，并且假设所有年份都有 $n=365$ 天。对于 $i=1, 2, \cdots, k$，设 b_i 表示编号为 i 的人的生日，其中 $1 \leqslant b_i \leqslant n$。还假设生日均匀分布在一年的 n 天中，因此对

$i=1, 2, \cdots, k$ 和 $r=1, 2, \cdots, n$，$\Pr\{b_i=r\}=1/n$。

两个人 i 和 j 的生日正好相同的概率依赖于生日的随机选择是否独立。从现在开始，假设生
130 日是独立的，于是 i 和 j 的生日都落在同一日 r 上的概率是

$$\Pr\{b_i = r \text{ 且 } b_j = r\} = \Pr\{b_i = r\}\Pr\{b_j = r\} = 1/n^2$$

这样，他们的生日落在同一天的概率是

$$\Pr\{b_i = b_j\} = \sum_{r=1}^{n} \Pr\{b_i = r \text{ 且 } b_j = r\} = \sum_{r=1}^{n} (1/n^2) = 1/n \tag{5.6}$$

更直观地说，一旦选定 b_i，b_j 被选在同一天的概率是 $1/n$。因此，i 和 j 有相同生日的概率与他们其中一个的生日落在给定一天的概率相同。然而需要注意，这个巧合依赖于各人的生日是独立的这个假设。

我们可以通过考察一个事件补的方法，来分析 k 个人中至少有两人生日相同的概率。至少有两个人生日相同的概率等于 1 减去所有人生日都不相同的概率。k 个人生日互不相同的事件为

$$B_k = \bigcap_{i=1}^{k} A_i$$

其中 A_i 是指对所有 $j<i$，i 与 j 生日不同的事件。既然可以写成 $B_k = A_k \cap B_{k-1}$，由公式(C.16)可得递归式

$$\Pr\{B_k\} = \Pr\{B_{k-1}\}\Pr\{A_k \mid B_{k-1}\} \tag{5.7}$$

其中取 $\Pr\{B_1\}=\Pr\{A_1\}=1$ 作为初始条件。换句话说，对 $i=1, 2, \cdots, k-1$，假设 $b_1, b_2, \cdots, b_{k-1}$ 两两不同，那么 $b_1, b_2, \cdots, b_k$ 两两不同的概率等于 $b_1, b_2, \cdots, b_{k-1}$ 两两不同的概率乘以 $i=1, 2, \cdots, k-1$ 时 $b_k \neq b_i$ 的概率。

如果 $b_1, b_2, \cdots, b_{k-1}$ 两两不同，对于 $i=1, 2, \cdots, k-1$，$b_k \neq b_i$ 的条件概率是 $\Pr\{A_k \mid B_{k-1}\} =$
131 $(n-k+1)/n$，这是因为 n 天中有 $n-(k-1)$ 天没被占用。我们反复应用递归式(5.7)得到

$$\begin{aligned}
\Pr(B_k) &= \Pr\{B_{k-1}\}\Pr\{A_k \mid B_{k-1}\} \\
&= \Pr\{B_{k-2}\}\Pr\{A_{k-1} \mid B_{k-2}\}\Pr\{A_k \mid B_{k-1}\} \\
&\vdots \\
&= \Pr\{B_1\}\Pr\{A_2 \mid B_1\}\Pr\{A_3 \mid B_2\}\cdots\Pr\{A_k \mid B_{k-1}\} \\
&= 1 \cdot \left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right)\cdots\left(\frac{n-k+1}{n}\right) \\
&= 1 \cdot \left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{k-1}{n}\right)
\end{aligned}$$

由不等式(3.12)，$1+x \leqslant e^x$，我们得出

$$\Pr\{B_k\} \leqslant e^{-1/n}e^{-2/n}\cdots e^{-(k-1)/n} = e^{-\sum_{i=1}^{k-1} i/n} = e^{-k(k-1)/2n} \leqslant 1/2$$

当 $-k(k-1)/2n \leqslant \ln(1/2)$ 时成立。当 $k(k-1) \geqslant 2n \ln 2$，或者，解二次方程，当 $k \geqslant (1+\sqrt{1+(8\ln 2)n})/2$ 时，所有 k 个生日两两不同的概率至多是 1/2。当 $n=365$ 时，必有 $k \geqslant 23$。因而，如果至少有 23 个人在一间屋子里，那么至少有两个人生日相同的概率至少是 1/2。在火星上，一年有 669 个火星日，所以达到相同效果须有 31 个火星人。

采用指示器随机变量的一个分析

我们可以利用指示器随机变量给出生日悖论的一个简单而近似的分析。对屋子里 k 个人中的每一对 (i, j)，对 $1 \leqslant i < j \leqslant k$，定义指示器随机变量 X_{ij} 如下：

$$X_{ij} = \mathrm{I}\{i \text{ 和 } j \text{ 生日相同}\} = \begin{cases} 1 & \text{如果 } i \text{ 和 } j \text{ 生日相同} \\ 0 & \text{其他} \end{cases}$$

根据等式(5.6)，两个人生日相同的概率是 $1/n$，因此据引理 5.1，我们有

$$\mathrm{E}[X_{ij}] = \Pr\{i \text{ 和 } j \text{ 生日相同}\} = 1/n$$

设 X 表示计数生日相同两人对数目的随机变量，我们有 132

$$X = \sum_{i=1}^{k} \sum_{j=i+1}^{k} X_{ij}$$

两边取期望，并应用期望的线性性质，我们得到

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{k} \sum_{j=i+1}^{k} X_{ij}\right] = \sum_{i=1}^{k} \sum_{j=i+1}^{k} \mathrm{E}[X_{ij}] = \binom{k}{2}\frac{1}{n} = \frac{k(k-1)}{2n}$$

因此，当 $k(k-1) \geqslant 2n$ 时，生日相同的两人对的期望数至少是 1。因此，若屋子里至少有 $\sqrt{2n}+1$ 个人，我们可以期望至少有两人生日相同。对于 $n=365$，若 $k=28$，生日相同人对数目的期望值为 $(28 \cdot 27)/(2 \cdot 365) \approx 1.035\,6$。因此，如果至少有 28 人，我们可以期望找到至少一对人生日相同。在火星上，一年有 669 个火星日，我们至少需要 38 个火星人。

第一种分析仅用了概率，确定了为使存在至少一对人生日相同概率大于 1/2 所需的人数；第二种分析使用了指示器随机变量，给出了相同生日期望数为 1 时的人数。虽然两种情形下人的准确数目不同，但它们在渐近阶数上是相等的，都为 $\Theta(\sqrt{n})$。

5.4.2　球与箱子

现在我们来考虑这样一个过程，即把相同的球随机投到 b 个箱子里，箱子编号为 1，2，…，b。每次投球都是独立的，每一次投球，球等可能落在每一个箱子中。球落在任一个箱子中的概率为 $1/b$。因此，投球的过程是一组伯努利试验(参见附录 C.4)，每次成功的概率是 $1/b$，其中成功是指球落入指定的箱子中。这个模型对分析散列(参见第 11 章)特别有用，而且我们可以回答关于该投球过程的各种有趣问题。(思考题 C-1 提出了另外一些关于球和箱子的问题。) 133

有多少球落在给定的箱子里？落在给定箱子里的球数服从二项分布 $b(k; n, 1/b)$。如果投 n 个球，公式(C.37)告诉我们，落在给定箱子里的球数期望值是 n/b。

*在平均意义下，我们必须要投多少个球，才能在给定的箱子里投中一个球？*直至给定箱子收到一个球的投球次数服从几何分布，概率为 $1/b$，根据等式(C.32)，成功的投球次数期望是 $1/(1/b)=b$。

我们需要投多少次球，才能使每个箱子里至少有一个球？一次投球落在空箱子里称为一次“命中”。我们想知道为了获得 b 次命中，所需的投球次数期望 n。

采用命中次数，可以把 n 次投球分为几个阶段。第 i 个阶段包括从第 $i-1$ 次命中到第 i 次命中之间的投球。第 1 阶段包含第 1 次投球，因为我们可以保证一次命中，此时所有的箱子都是空的。对第 i 阶段的每一次投球，有 $i-1$ 个箱子有球，$b-i+1$ 个箱子是空的。因而，对第 i 阶段的每次投球，得到一次命中的概率是 $(b-i+1)/b$。

设 n_i 表示第 i 阶段的投球次数。因而，为得到 b 次命中所需的投球次数为 $n = \sum_{i=1}^{b} n_i$。每个随机变量 n_i 服从几何分布，成功的概率是 $(b-i+1)/b$，根据公式(C.32)，于是有

$$\mathrm{E}[n_i] = \frac{b}{b-i+1}$$

根据期望的线性性质，我们有

$$\begin{aligned}\mathrm{E}[n] &= \mathrm{E}\left[\sum_{i=1}^{b} n_i\right] = \sum_{i=1}^{b} \mathrm{E}[n_i] = \sum_{i=1}^{b} \frac{b}{b-i+1} \\ &= b\sum_{i=1}^{b} \frac{1}{i} = b(\ln b + O(1)) \quad \text{(根据等式(A.7))}\end{aligned}$$

所以，在我们期望每个箱子里都有一个球之前，大约要投 $b \ln b$ 次。这个问题也称为**礼券收集者问题**，意思是一个人如果想要收集齐 b 种不同礼券中的每一种，大约需要 $b \ln b$ 张随机得到的礼

134 券才能成功。

5.4.3 特征序列

假设抛投一枚标准的硬币 n 次，最长连续正面的序列的期望长度有多长？答案是 $\Theta(\lg n)$，如以下分析所示。

首先证明最长的连续正面的特征序列的长度期望是 $O(\lg n)$。每次抛硬币时是一次正面的概率为 1/2。设 A_{ik} 为这样的事件：长度至少为 k 的正面特征序列开始于第 i 次抛掷，或更准确地说，k 次连续硬币抛掷 i，$i+1$，…，$i+k-1$ 得到的都是正面，其中 $1\leqslant k\leqslant n$，$1\leqslant i\leqslant n-k+1$。因为每次抛硬币是互相独立的，对任何给定事件 A_{ik}，所有 k 次抛掷都是正面的概率是

$$\Pr\{A_{ik}\}=1/2^k \tag{5.8}$$

对于 $k=2\lceil\lg n\rceil$，

$$\Pr\{A_{i,2\lceil\lg n\rceil}\}=1/2^{2\lceil\lg n\rceil}\leqslant 1/2^{2\lg n}=1/n^2$$

因而，长度至少为 $2\lceil\lg n\rceil$、起始于位置 i 的一个正面特征序列的概率是很小的。这种序列起始位置至多有 $n-2\lceil\lg n\rceil+1$ 个。所以长度至少为 $2\lceil\lg n\rceil$的正面特征序列开始于任一位置的概率是

$$\Pr\left\{\bigcup_{i=1}^{n-2\lceil\lg n\rceil+1}A_{i,2\lceil\lg n\rceil}\right\}\leqslant\sum_{i=1}^{n-2\lceil\lg n\rceil+1}1/n^2<\sum_{i=1}^{n}1/n^2=1/n \tag{5.9}$$

因为根据布尔不等式(C.19)，一组事件并集的概率至多是各个事件的概率之和。(注意，即使这些事件不独立，布尔不等式依然成立。)

我们现在利用不等式(5.9)来给出最长特征序列的长度界。对于 $j=0$，1，2，…，n，令 L_j 表示最长连续正面的特征序列长度正好是 j 的事件，并设最长特征序列的长度是 L。由期望值的定义，我们有

135
$$\mathrm{E}[L]=\sum_{j=0}^{n}j\Pr\{L_j\} \tag{5.10}$$

我们可以尝试用每个 $\Pr\{L_j\}$的上界来估计这个和，就像不等式(5.9)所计算的那样。遗憾的是，这种方法将导致弱的界。不过，我们可以用从上面分析得到的一些直观知识来得到一个好的界。然而非正式地说，我们观察到在等式(5.10)的总和中，没有任何一项同时让 j 和 $\Pr\{L_j\}$因子都是大的。为什么呢？当 $j\geqslant 2\lceil\lg n\rceil$时，$\Pr\{L_j\}$很小；当 $j<2\lceil\lg n\rceil$时，j 相当小。更正式地说，我们注意到对于 $j=0$，1，…，n，事件 L_j 是不相交的，因此长度至少为 $2\lceil\lg n\rceil$的连续正面特征序列起始于任一位置的概率为 $\sum_{j=2\lceil\lg n\rceil}^{n}\Pr\{L_j\}$。根据不等式(5.9)，我们有 $\sum_{j=2\lceil\lg n\rceil}^{n}\Pr\{L_j\}<1/n$。另外，注意到 $\sum_{j=0}^{n}\Pr\{L_j\}=1$，我们有 $\sum_{j=0}^{2\lceil\lg n\rceil-1}\Pr\{L_j\}\leqslant 1$。因此，我们得到

$$\begin{aligned}\mathrm{E}[L]&=\sum_{j=0}^{n}j\Pr\{L_j\}=\sum_{j=0}^{2\lceil\lg n\rceil-1}j\Pr\{L_j\}+\sum_{j=2\lceil\lg n\rceil}^{n}j\Pr\{L_j\}\\&<\sum_{j=0}^{2\lceil\lg n\rceil-1}(2\lceil\lg n\rceil)\Pr\{L_j\}+\sum_{j=2\lceil\lg n\rceil}^{n}n\Pr\{L_j\}\\&=2\lceil\lg n\rceil\sum_{j=0}^{2\lceil\lg n\rceil-1}\Pr\{L_j\}+n\sum_{j=2\lceil\lg n\rceil}^{n}\Pr\{L_j\}\\&<2\lceil\lg n\rceil\cdot 1+n\cdot(1/n)=O(\lg n)\end{aligned}$$

正面特征序列长度超过 $r\lceil\lg n\rceil$次抛掷的概率随着 r 变小而很快减少。对 $r\geqslant 1$，正面特征序列长度至少为 $r\lceil\lg n\rceil$，起始于位置 i 的概率是

$$\Pr\{A_{i,r\lceil\lg n\rceil}\}=1/2^{r\lceil\lg n\rceil}\leqslant 1/n^r$$

因此，最长特征序列长度至少为 $r\lceil\lg n\rceil$的概率至多是 $n/n^r=1/n^{r-1}$，或等价地，最长特征序列长

度小于 $r\lceil \lg n\rceil$的概率至少是 $1-1/n^{r-1}$。

看一个例子，抛掷 $n=1\,000$ 次硬币，最少出现 $2\lceil \lg n\rceil=20$ 次连续正面的几率至多是$1/n=1/1\,000$。长度超过 $3\lceil \lg n\rceil=30$ 次连续正面特征序列的几率至多是 $1/n^2=1/1\,000\,000$。

现在我们证明一个补充的下界：在 n 次硬币抛掷中，最长的正面特征序列的长度期望为 $\Omega(\lg n)$。为证明这个界，我们通过把 n 次抛掷划分成大约 n/s 个组，每组 s 次抛掷，来看长度为 136
s 的特征序列。如果选择 $s=\lfloor(\lg n)/2\rfloor$，可以说明这些组中至少有一组可能全是正面，因而可能最长特征序列的长度至少是 $s=\Omega(\lg n)$。然后将表明最长特征序列的长度期望是 $\Omega(\lg n)$。

我们把 n 次硬币抛掷划分成至少$\lfloor n/\lfloor(\lg n)/2\rfloor\rfloor$个组，每组$\lfloor(\lg n)/2\rfloor$次连续抛掷，然后对没有组全是正面的概率求界。根据等式(5.8)，从位置 i 开始都是正面的组的概率是

$$\Pr\{A_{i,\lfloor(\lg n)/2\rfloor}\}=1/2^{\lfloor(\lg n)/2\rfloor}\leqslant 1/\sqrt{n}$$

所以长度至少为$\lfloor(\lg n)/2\rfloor$的正面特征序列不从位置 i 开始的概率至多是 $1-1/\sqrt{n}$。既然$\lfloor n/\lfloor(\lg n)/2\rfloor\rfloor$个组是由彼此互斥、独立的抛掷硬币构成，其中每个组都不是长度为$\lfloor(\lg n)/2\rfloor$的特征序列的概率至多是

$$\begin{aligned}(1-1/\sqrt{n})^{\lfloor n/\lfloor(\lg n)/2\rfloor\rfloor}&\leqslant(1-1/\sqrt{n})^{n/\lfloor(\lg n)/2\rfloor-1}\leqslant(1-1/\sqrt{n})^{2n/\lg n-1}\\&\leqslant e^{-(2n/\lg n-1)/\sqrt{n}}=O(e^{-\lg n})=O(1/n)\end{aligned}$$

关于此论证，我们用到了不等式(3.12)，即 $1+x\leqslant e^x$，还用到了你可能想验证的一个事实：对足够大的 n，有$(2n/\lg n-1)/\sqrt{n}\geqslant\lg n$。

因此，最长特征序列超过$\lfloor(\lg n)/2\rfloor$的概率为

$$\sum_{j=\lfloor(\lg n)/2\rfloor}^{n}\Pr\{L_j\}\geqslant 1-O(1/n)\tag{5.11}$$

现在我们可以计算最长特征序列的长度期望的一个下界，从等式(5.10)开始，采用类似于我们上界分析的方式： 137

$$\begin{aligned}\mathrm{E}[L]&=\sum_{j=0}^{n}j\Pr\{L_j\}=\sum_{j=0}^{\lfloor(\lg n)/2\rfloor-1}j\Pr\{L_j\}+\sum_{j=\lfloor(\lg n)/2\rfloor}^{n}j\Pr\{L_j\}\\&\geqslant\sum_{j=0}^{\lfloor(\lg n)/2\rfloor-1}0\cdot\Pr\{L_j\}+\sum_{j=\lfloor(\lg n)/2\rfloor}^{n}\lfloor(\lg n)/2\rfloor\Pr\{L_j\}\\&=0\cdot\sum_{j=0}^{\lfloor(\lg n)/2\rfloor-1}\Pr\{L_j\}+\lfloor(\lg n)/2\rfloor\sum_{j=\lfloor(\lg n)/2\rfloor}^{n}\Pr\{L_j\}\\&\geqslant 0+\lfloor(\lg n)/2\rfloor(1-O(1/n))\qquad\text{(根据不等式(5.11))}\\&=\Omega(\lg n)\end{aligned}$$

和生日悖论一样，可以采用指示器随机变量来得到一个简单而近似的分析。设 $X_{ik}=\mathrm{I}\{A_{ik}\}$ 表示对应于特征序列长度至少为 k、开始于第 i 次抛掷硬币的指示器随机变量。为了计数这些特征序列的总数，定义

$$X=\sum_{i=1}^{n-k+1}X_{ik}$$

两边取期望并利用期望的线性性质，我们有

$$\mathrm{E}[X]=\mathrm{E}\left[\sum_{i=1}^{n-k+1}X_{ik}\right]=\sum_{i=1}^{n-k+1}\mathrm{E}[X_{ik}]=\sum_{i=1}^{n-k+1}\Pr\{A_{ik}\}=\sum_{i=1}^{n-k+1}1/2^k=\frac{n-k+1}{2^k}$$

通过代入不同的 k 值，可以计算出长度为 k 的特征序列的数目期望。如果这个数大(远大于1)，那么我们期望很多长度为 k 的特征序列会出现，而且出现一个的概率很高。如果这个数小 138
(远小于 1)，那么我们期望很少的长度为 k 的特征序列会出现，而且出现一个的概率很低。如果对某个正常数 c，有 $k=c\lg n$，那么可以得到

$$\mathrm{E}[X]=\frac{n-c\lg n+1}{2^{c\lg n}}=\frac{n-c\lg n+1}{n^c}=\frac{1}{n^{c-1}}-\frac{(c\lg n-1)/n}{n^{c-1}}=\Theta(1/n^{c-1})$$

如果 c 较大，长度为 $c\lg n$ 的特征序列的数目期望将很小，并且我们的结论是它们不大可能发生。另外，如果 $c=1/2$，那么 $\mathrm{E}[X]=\Theta(1/n^{1/2-1})=\Theta(n^{1/2})$，并且我们期望会有大量长度为 $(1/2)\lg n$ 的特征序列。所以，这种长度的特征序列很可能发生。仅通过这些粗略估计，我们可得出结论：最长特征序列的长度期望是 $\Theta(\lg n)$。

5.4.4 在线雇用问题

作为最后一个例子，我们考虑雇用问题的一个变形。假设现在我们不希望面试所有的应聘者以找到最好的一个。我们也不希望因为有更好的申请者出现，不停地雇用新人解雇旧人。取而代之，我们愿意雇用接近最好的应聘者，只雇用一次。我们必须遵守公司的一个要求：每次面试后，或者我们必须马上提供职位给应聘者，或者马上拒绝该应聘者。如何在最小化面试次数和最大化所雇用应聘者的质量两方面取得平衡？

我们可以通过如下方式对该问题建模。在面试一个应聘者之后，我们能够给每人一个分数；令 $score(i)$ 表示给第 i 个应聘者的分数，并且假设没有两个应聘者得到同样分数。在已看过 j 个应聘者后，我们知道这 j 人中哪一个分数最高，但是不知道在剩余的 $n-j$ 个应聘者中会不会有更高分数的应聘者。我们决定采用这样一个策略：选择一个正整数 $k<n$，面试然后拒绝前 k 个应聘者，再雇用其后比前面的应聘者有更高分数的第一个应聘者。如果最好的应聘者在前 k 个面试之中，那么将雇用第 n 个应聘者。我们形式化地表达该策略在过程 ON-LINE-MAXIMUM(k, n)
139 中，它返回的是我们希望雇用的应聘者下标。

```
ON-LINE-MAXIMUM(k, n)
1  bestscore = −∞
2  for i = 1 to k
3      if score(i) > bestscore
4          bestscore = score(i)
5  for i = k + 1 to n
6      if score(i) > bestscore
7          return i
8  return n
```

对每个可能的 k，我们希望确定能雇用最好应聘者的概率。然后选择最佳的 k 值，并用该值来实现这个策略。暂时先假设 k 是固定的。设 $M(j)=\max_{1\leqslant i\leqslant j}\{score(i)\}$ 表示应聘者 $1\sim j$ 中的最高分数。设 S 表示成功选择最好应聘者的事件，S_i 表示最好的应聘者是第 i 个面试者时成功的事件。既然不同的 S_i 不相交，我们有 $\Pr\{S\}=\sum_{i=1}^{n}\Pr\{S_i\}$。注意到，当最好应聘者是前 k 个应聘者中的一个时，我们不会成功，于是对 $i=1, 2, \cdots, k$，有 $\Pr\{S_i\}=0$。因而得到

$$\Pr\{S\}=\sum_{i=k+1}^{n}\Pr\{S_i\} \tag{5.12}$$

现在来计算 $\Pr\{S_i\}$。为了当第 i 个应聘者是最好时成功，两件事情必须发生。第一，最好的应聘者必须在位置 i 上，用事件 B_i 表示。第二，算法不能选择从位置 $k+1\sim i-1$ 中任何一个应聘者，而这个选择当且仅当满足 $k+1\leqslant j\leqslant i-1$ 时发生，在程序第 6 行有 $score(j)<bestcore$。（因为分数是唯一的，所以可以忽略 $score(j)=bestcore$ 的可能性。）换句话说，所有 $score(k+1)$ 到 $score(i-1)$ 的值都必须小于 $M(k)$；如果其中有大于 $M(k)$ 的数，则将返回第一个大于 $M(k)$ 的数的下标。我们用 O_i 表示从位置 $k+1$ 到 $i-1$ 中没有任何应聘者入选的事件。幸运的是，两个事件 B_i 和 O_i 是独立的。事件 O_i 仅依赖于位置 1 到 $i-1$ 中值的相对次序，而 B_i 仅依赖于位置 i 的值是否大于所有其他位置的值。从位置 1 到 $i-1$ 的排序并不应影响位置 i 的值是否大于上述所有

值，并且位置 i 的值也不会影响从位置 1 到 $i-1$ 值的次序。因而应用等式(C. 15)得到 140

$$\Pr\{S_i\} = \Pr\{B_i \cap O_i\} = \Pr\{B_i\}\Pr\{O_i\}$$

$\Pr\{B_i\}$的概率显然是 $1/n$，因为最大值等可能地是 n 个位置中的任一个。若事件 O_i 要发生，从位置 1 到 $i-1$ 的最大值必须在前 k 个位置的一个，并且最大值等可能地在这 $i-1$ 个位置中的任一个。于是，$\Pr\{O_i\}=k/(i-1)$，$\Pr\{S_i\}=k/(n(i-1))$。利用公式(5. 12)，我们有

$$\Pr\{S\} = \sum_{i=k+1}^{n}\Pr\{S_i\} = \sum_{i=k+1}^{n}\frac{k}{n(i-1)} = \frac{k}{n}\sum_{i=k+1}^{n}\frac{1}{i-1} = \frac{k}{n}\sum_{i=k}^{n-1}\frac{1}{i}$$

我们利用积分来近似约束这个和数的上界和下界。根据不等式(A. 12)，我们有

$$\int_k^n \frac{1}{x}\mathrm{d}x \leqslant \sum_{i=k}^{n-1}\frac{1}{i} \leqslant \int_{k-1}^{n-1}\frac{1}{x}\mathrm{d}x$$

求解这些定积分可以得到下面的界：

$$\frac{k}{n}(\ln n - \ln k) \leqslant \Pr\{S\} \leqslant \frac{k}{n}(\ln(n-1) - \ln(k-1))$$

这提供了 $\Pr\{S\}$的一个相当紧确的界。因为我们希望最大化成功的概率，所以关注如何选取 k 值使 $\Pr\{S\}$的下界最大化。(此外，下界表达式比上界表达式更容易最大化。)以 k 为变量对表达式 $(k/n)(\ln n-\ln k)$求导，得到

$$\frac{1}{n}(\ln n - \ln k - 1)$$

令此导数为 0，我们看到当 $\ln k=\ln n-1=\ln(n/\mathrm{e})$或等价地，$k=n/\mathrm{e}$ 时，概率下界最大化。因而，如果用 $k=n/\mathrm{e}$ 来实现我们的策略，那么将以至少 $1/\mathrm{e}$ 的概率成功雇用到最好的应聘者。 141

练习

5. 4-1　一个屋子里必须要有多少人，才能让某人和你生日相同的概率至少为 1/2？必须要有多少人，才能让至少两个人生日为 7 月 4 日的概率大于 1/2？

5. 4-2　假设我们将球投入到 b 个箱子里，直到某个箱子中有两个球。每一次投掷都是独立的，并且每个球落入任何箱子的机会均等。请问投球次数期望是多少？

***5. 4-3**　在生日悖论的分析中，要求各人生日彼此独立是否很重要？或者，是否只要两两成对独立就足够了？证明你的答案。

***5. 4-4**　一次聚会需要邀请多少人，才能让其中 3 人的生日很可能相同？

***5. 4-5**　在大小为 n 的集合中，一个 k 字符串构成一个 k 排列的概率是多少？这个问题和生日悖论有什么关系？

***5. 4-6**　假设将 n 个球投入 n 个箱子里，其中每次投球独立，并且每个球等可能落入任何箱子。空箱子的数目期望是多少？正好有一个球的箱子的数目期望是多少？

***5. 4-7**　为使特征序列长度的下界变得更精确，请说明在 n 次硬币的公平抛掷中，不出现比 $\lg n-2\lg\lg n$ 更长的连续正面特征序列的概率小于 $1/n$。 142

思考题

5-1　(*概率计数*)　利用一个 b 位的计数器，我们一般只能计数到 2^b-1。而用 R. Morris 的**概率计数法**，我们可以计数到一个大得多的值，代价是精度有所损失。

对 $i=0, 1, \cdots, 2^b-1$，令计数器值 i 表示 n_i 的计数，其中 n_i 构成了一个非负的递增序列。假设计数器初值为 0，表示计数 $n_0=0$。INCREMENT 运算单元工作在一个计数器上，它以概率的方式包含值 i。如果 $i=2^b-1$，则该运算单元报告溢出错误；否则，INCREMENT 运算单元以概率 $1/(n_{i+1}-n_i)$把计数器增加 1，以概率 $1-1/(n_{i+1}-n_i)$保持计

数器不变。

对所有的 $i \geqslant 0$，若选择 $n_i = i$，此计数器就是一个普通的计数器。若选择 $n_i = 2^{i-1}$（$i > 0$），或者 $n_i = F_i$（第 i 个斐波那契数，参见 3.2 节），则会出现更多有趣的情形。

对于这个问题，假设 n_{2^b-1} 已足够大，发生一个溢出错误的概率可以忽略。

a. 请说明在执行 n 次 INCREMENT 操作后，计数器所表示的数期望值正好是 n。

b. 分析计数器表示的计数的方差依赖于 n_i 序列。我们来看一个简单情形：对所有 $i \geqslant 0$，$n_i = 100i$。在执行了 n 次 INCREMENT 操作后，请估计计数器所表示数的方差。

5-2 （查找一个无序数组） 本题将分析三个算法，它们在一个包含 n 个元素的无序数组 A 中查找一个值 x。

考虑如下的随机策略：随机挑选 A 中的一个下标 i。如果 $A[i] = x$，则终止；否则，继续挑选 A 中一个新的随机下标。重复随机挑选下标，直到找到一个下标 j，使 $A[j] = x$，或者直到我们已检查过 A 中的每一个元素。注意，我们每次都是从全部下标的集合中挑选，于是可能会不止一次地检查某个元素。

a. 请写出过程 RANDOM-SEARCH 的伪代码来实现上述策略。确保当 A 中所有下标都被
143 挑选过时，你的算法应停止。

b. 假定恰好有一个下标 i 使得 $A[i] = x$。在我们找到 x 和 RANDOM-SEARCH 结束之前，必须挑选 A 下标的数目期望是多少？

c. 假设有 $k \geqslant 1$ 个下标 i 使得 $A[i] = x$，推广你对(b)部分的解答。在找到 x 或 RANDOM-SEARCH 结束之前，必须挑选 A 的下标的数目期望是多少？你的答案应该是 n 和 k 的函数。

d. 假设没有下标 i 使得 $A[i] = x$。在检查完 A 的所有元素或 RANDOM-SEARCH 结束之前，我们必须挑选 A 的下标的数目期望是多少？

现在考虑一个确定性的线性查找算法，我们称之为 DETERMINISTIC-SEARCH。具体地说，这个算法在 A 中顺序查找 x，考虑 $A[1]$，$A[2]$，$A[3]$，…，$A[n]$，直到找到 $A[i] = x$，或者到达数组的末尾。假设输入数组的所有排列都是等可能的。

e. 假设恰好有一个下标 i 使得 $A[i] = x$。DETERMINISTIC-SEARCH 平均情形的运行时间是多少？DETERMINISTIC-SEARCH 最坏情形的运行时间又是多少？

f. 假设有 $k \geqslant 1$ 个下标 i 使得 $A[i] = x$，推广你对(e)部分的解答。DETERMINISTIC-SEARCH 平均情形的运行时间是多少？DETERMINISTIC-SEARCH 最坏情形的运行时间又是多少？你的答案应是 n 与 k 的函数。

g. 假设没有下标 i 使得 $A[i] = x$。DETERMINISTIC-SEARCH 平均情形的运行时间是多少？DETERMINISTIC-SEARCH 最坏情形的运行时间又是多少？

最后，考虑一个随机算法 SCRAMBLE-SEARCH，它先将输入数组随机变换排列，然后在排列变换后的数组上，运行上面的确定性线性查找算法。

h. 设 k 是满足 $A[i] = x$ 的下标的数目，请给出在 $k = 0$ 和 $k = 1$ 情况下，算法 SCRAMBLE-SEARCH 最坏情形的运行时间和运行时间期望。推广你的解答以处理 $k \geqslant 1$ 的情况。

144 **i.** 你将会使用 3 种查找算法中的哪一个？解释你的答案。

本章注记

Bollobás[53]、Hofri[174]和 Spencer[321]介绍了大量高等概率技术。随机算法的优点在 Karp[200]和 Rabin[288]中有讨论和综述。Motwani 和 Raghavan[262]的教材中大量论述了随机算法。

雇用问题的很多变形已经得到广泛研究。这些问题通常被称为“秘书问题”。Ajtai、Meggido
145 和 Waarts[11]中给出了该领域中的一个例子。