

算法设计与分析

主讲人：权丽君

Email: *ljquan@suda.edu.cn*

苏州大学 计算机学院



第四讲 概率分析和随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法
- 概率分析和随机算法的应用：在线雇用问题，生日悖论，球与盒子



算法分析与输入分布

- 算法运行时间与输入规模和输入分布有关，如插入排序：

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1)$$

$$+ c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n-1)$$

INSERTION-SORT(A)

	cost	times
1 for ($j = 2; j <= \text{length}[A]; j++$)	c_1	n
2 key = $A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1 .. j-1]$	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 while ($i > 0 \ \&\& A[i] > \text{key}$)	c_5	
6 $A[i+1] = A[i]$	c_6	
7 $i = i-1$	c_7	
8 $A[i+1] = \text{key}$	c_8	$n-1$





分治法的基本思想

- 对于运算时间与输入数据分布有关的算法，时间复杂度分析一般有三种：最坏运行时间、最佳运行时间、平均运行时间。
- 平均运行时间是算法对所有可能情况的期望运行时间，与输入数据的概率分布有关。
- 分析算法的平均运行时间通常需要对输入分布做某种假定



第四讲 概率分析和随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法
- 概率分析和随机算法的应用：在线雇用问题，生日悖论，球与盒子



雇用问题

假如你要通过雇佣代理来寻找并雇佣一名新的办公助理。雇佣代理每天给你推荐一名应聘者。你面试这个人，然后决定是否雇佣他。

- 1) 雇佣代理每推荐你一名应聘者，你就要支付一小笔推荐费，记为 c_i ；
- 2) 如果你雇佣了其中一名应聘者，则需要花费更多的钱，记为 c_h ，包括辞掉目前办公助理的费用，并另付给雇佣代理一笔中介费。

你承诺在任何时候，只要当前应聘者比目前的办公助理更适合，就立刻辞掉目前的办公助理而雇佣新的办公助理，你愿意为此花费一笔钱。

2021/10/21 ⁶  现在你希望能够估算一下这笔费用会是多少。



雇用问题

雇用策略的伪代码如下：

- 设应聘者的编号为1到n
- 假设在面试完应聘者*i*后，可以决定应聘者*i*是否是你见过的最适当人选
- 为了初始化，建立一个虚拟的应聘者，编号为0，他比所有其他的应聘者都差。

HIRE-ASSISTANT(*n*)

best $\leftarrow 0$ // candidate 0 is a least-qualified dummy candidate

for *i* $\leftarrow 1$ to *n*

 do interview candidate *i*

cost times

c_i *n*

 if candidate *i* is better than candidate *best*

 then *best* $\leftarrow i$

 hire candidate *i*

cost times

- 费用：n个应聘者中雇用了m个，则该算法的总费用是 $O(nc_i + mc_h)$



雇用问题

分析：

设面试推荐费用为 c_i ，雇用费用为 c_h

- 假设总共面试n个人，其中雇用了m人，则该算法的总费用是 $O(c_i n + c_h m)$.
- 进一步观察，会发现面试费用 $c_i n$ 是恒定的，因为不管雇用多少人，总会面试n个应聘者。所以我们只关注于雇用费用 $c_h m$ 即可。
- 那么整个过程会产生多少雇用费用呢？



雇用问题

■ 最坏情形分析：

- **最坏情形**：实际雇用了每个面试的应聘者；
- 当应聘者的质量按出现的次序严格递增时，就会出现这种情况。
- 此时面试了 n 次，雇用了 n 次，则雇用总费用是 $O(c_h n)$ 。

■ 一般情况分析：

- **一般情形**：应聘者不会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

那么，在一般情形下，会发生什么呢？



概率分析

过程HIRE-ASSISTANT中，检查序列中的每个成员，维护一个当前“获胜者”（即best），最后找出序列中的最好者。

这里对当前获胜成员的**更新频率**建立模型，并引入“**概率分析**”对上述现象进行分析

概率分析：就是在问题分析中应用概率的方法。

- 概率分析可用于时间分析，也可用于其他量的分析。
- 这里用概率分析技术分析雇用费用。



概率分析

HIRE-ASSISTANT(n)

cost times

$best \leftarrow 0$ // candidate 0 is a least-qualified dummy candidate

for $i \leftarrow 1$ to n

do interview candidate i

c_i n

if candidate i is better than candidate $best$

then $best \leftarrow i$

hire candidate i

c_h m

假设应聘者的资质序列是任意的：

- 用1到 n 将应聘者排列名次，用 $rank(i)$ 表示应聘者 i 的名次，约定较高的名次对应较有资格的应聘者。
- 有序序列 $\langle rank(1), \dots, rank(n) \rangle$ 是 $\langle 1, \dots, n \rangle$ 的一个排列，例如 $\langle 5, 2, 16, 28, 9, \dots, 11 \rangle$
- 所有应聘者存在全序关系，即任意两个应聘者可以比较 $rank$ 值



概率分析

为了进行概率分析，对输入分布做如下假设：

假设雇佣问题中应聘者以随机顺序出现，并且这种随机性由输入自身决定。

- 应聘者以随机顺序出现等价于称排名列表 $\langle \text{rank}(1), \text{rank}(2), \dots, \text{rank}(n) \rangle$ 是数字1到n的 $n!$ 种排列中的任一个。
- 称这样的排列构成一个均匀随机排列，即在 $n!$ 种可能的排列中，每种情况以“等概率”情形出现。



概率分析

- 概率分析：在问题的分析中应用概率技术。
- 使用概率分析来分析一个算法的运行时间，或其它的量。

概率分析的本质：需已知或假定输入的概率分布

- 依据概率分布来求期望，期望值即是平均雇用的人数
- 5.2节介绍了雇佣问题的概率分析



概率分析

- 概率分析：使用关于**输入分布**的知识或者对其做的假设，然后**分析算法**，计算出一个期望的运行时间。
- 实际上是将所有可能输入的运行时间做平均。
- 确定输入的分布时必须非常小心：
 - 有些问题，对所有可能的输入集合可以做某种假定，也可以将概率分析作为一种手段来设计高效算法，并加深对问题的认识。
 - 有些问题可能无法描述一个合理的输入分布，则不能用概率分析方法。



随机算法

目的：为了利用概率分析，就要了解关于输入分布的一些信息。但在许多情况下，我们对输入分布了解很少。而且即使知道输入分布的某些信息，也无法从计算上对这种认知建立模型——输入不可控。

如何让输入变得可控？

对算法中的某部分的行为随机化，利用概率和随机性作为算法设计与分析的工具进行相关处理。



分析：在雇佣问题中，看起来，应聘者好像以随机顺序出现，但我们无法知道是否确实如此。我们必须对应聘者的出现次序进行更大的**控制**，使其达到一种“随机”出现的样子。

方法：

假设雇用代理有 n 个应聘者，他们可以事先给我们一份名单，我们每天随机选择某个应聘者来面试。

——尽管除了应聘者名字外，对其他信息一无所知，但不再像以前依赖于“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**并加强了随机次序。



随机算法

- **随机算法：**如果一个算法的行为不只是由输入决定，同时也由随机数生成器所产生的数值决定，则称这个算法是随机的。
- **随机数生成器RANDOM**
 - ◆ RANDOM(a, b)返回一个介于 a 与 b 的整数，而每个整数出现的机会均等。and each of the $b-a+1$ possible values of r is equally likely.
 - ◆ RANDOM实际上由一个确定的算法〔伪随机产生器〕产生，其结果表面上看上去像是随机数



随机算法

随机算法的输入次序最终由随机数发生器决定，我们将随机算法的运行时间称为**期望运行时间**。

一般而言，

- 当**概率分布是在算法的输入上时**，我们讨论算法的“**平均情况运行时间**”；
- 当**算法本身做出随机选择时**，我们讨论算法的“**期望运行时间**”。



第四讲 概率分析和随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法
- 概率分析和随机算法的应用：在线雇用问题，生日悖论，球与盒子



指示器随机变量

为了建立概率(probabilities)和期望(expectations)之间的联系，这里引进**指示器随机变量**(indicator random variables)，用于实现概率与期望之间的转换。

指示器随机变量：给定一个样本空间**S** (sample space) 和一个事件**A** (event) , 那么事件A对应的指示器随机变量 **I{A}** 定义为:

$$I\{A\} = \begin{cases} 1 & \text{if } A \text{ occurs ,} \\ 0 & \text{if } A \text{ does not occur .} \end{cases}$$



指示器随机变量

例：抛掷一枚硬币，求正面朝上的期望次数

这里，

- 样本空间 $S = \{H, T\}$ ，其中 $\Pr(H) = \Pr(T) = 1/2$.
- 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 。

X_H 记录一次抛硬币时正面朝上的次数：

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



指示器随机变量

一次抛掷硬币正面朝上的期望次数，即**指示器变量 X_H 的期望值**是：

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

注：一个事件对应的指示器随机变量的**期望值**等于该事件发生的概率。

$$E(X) = \sum_x x \cdot \Pr\{X = x\} \quad \text{P700}$$



指示器随机变量

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

证明：由指示器随机变量的定义以及期望值的定义，有：

$$\begin{aligned} E[X_A] &= E[I\{A\}] \\ &= 1 * \Pr\{A\} + 0 * \Pr\{\sim A\} \\ &= \Pr\{A\} \end{aligned}$$

其中， $\sim A$ 表示 $S - A$ ，即A的补。



复杂一点的例子

- 投 n 次硬币，正面向上的期望（平均数）

- ◆ 令随机变量 X 表示投 n 次硬币正面向上的数
 - ◆ 我们有

$$E[X] = \sum_{k=0}^n k \cdot \Pr\{X = k\}$$

计算比较麻烦

- ◆ 我们可以利用指示随机变量简化计算



复杂一点的例子

- 设随机变量 X 表示 n 次抛硬币中出现正面朝上的总次数。
- 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，即： $X_i = I\{\text{第 } i \text{ 次抛掷时出现事件 } H\}$ 。

则显然有：
$$X = \sum_{i=1}^n X_i .$$

则，两边取期望，计算正面朝上次数的期望，有：

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

即：总和的期望值等于 n 个指示器随机变量值和的期望，也等于 n 个指示器随机变量值期望的和。



复杂一点的例子

□ 引理

对事件 A , 设 $X_A = \mathbf{I}\{A\}$. 则 $E[X_A] = \Pr\{A\}$.

- 投 n 次硬币, 正面向上的期望〔平均数〕
 - ◆ 以上引理得到 $E[X_i] = \Pr\{H\} = 1/2$, $i=1, 2, \dots, n$

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^n X_i\right] \\ &= \sum_{i=1}^n E[X_i] \\ &= \sum_{i=1}^n 1/2 \\ &= n/2. \end{aligned}$$



雇用问题分析

- 假设应聘者以随机次序出现.
- 令随机变量 X 表示我们雇用新助手的人数，由期望定义：

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

计算繁琐。

- 定义新的指示随机变量：

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

- 以及 $X = X_1 + X_2 + \dots + X_n$
- 由定理5.1：

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

需计算该概率



HIRE-ASSISTANT(n)

$best \leftarrow 0$ // candidate 0 is a least-qualified dummy candidate

for $i \leftarrow 1$ to n

 do interview candidate i

 cost times

c_i n

 if candidate i is better than candidate $best$

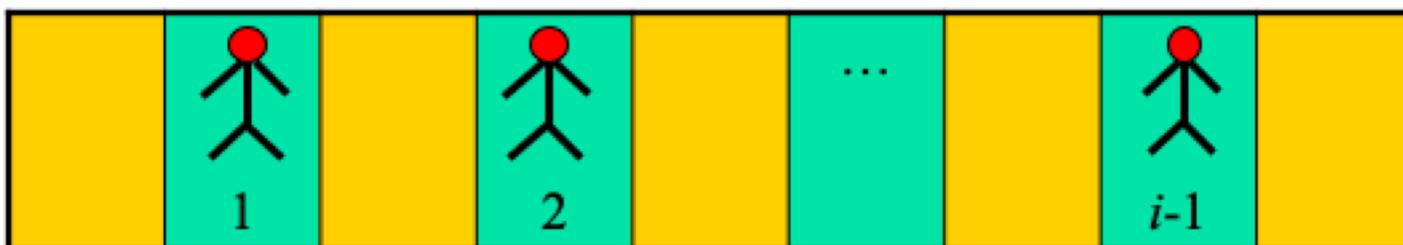
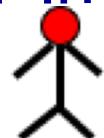
 then $best \leftarrow i$

 hire candidate i

c_h m

计算 $\Pr\{\text{应聘者 } i \text{ 被雇佣}\}$?

- ◆ i 被雇佣 $\Leftrightarrow i$ 比应聘者 $1, 2, \dots, i-1$ 都优秀.
- ◆ 若候选人随机到来，则每一个候选人为最佳人选的概率相等
- ◆ 因此, $E\{X_i\} = \Pr\{i \text{ 为前 } i \text{ 个应聘者中最优秀}\} = 1/i$?





计算 $E[X]$:

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

亦即，尽管面试了 n 个人，但平均起来，实际上大约只雇用了他们之中的 $\ln n$ 个人。



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)，可以直接推出这个界，说明雇用的人数期望值大约为 $\ln n$ 。

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



第四讲 概率分析和随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法
- 概率分析和随机算法的应用：在线雇用问题，生日悖论，球与盒子



随机算法

- 如前节所示，输入的分布有助于分析一个算法的平均情况行为。
- 但很多时候是无法得知输入分布的信息的。
- 采用随机算法，分析算法的期望值。
- 随机算法不是假设输入的分布，而是设定一个分布。



随机算法

雇用问题

上述关于雇用问题的算法，是确定性算法.

- 给定输入，则雇用的人数确定
- 雇用的人数〔资源消费〕依赖于输入
 - ◆ 某些排序的输入总是会产生很高的雇用费用，如： $<1,2,3,4,5,6>$ ，所有人都会被雇用。
 - ◆ 某些排序的输入总是产生很低的雇用费用，如： $<6,*,*,*,*,*>$ 。这时只有第一个应聘者会被雇用。
 - ◆ 大部分输入是中间情况。
- 假设公司人力资源部门有敌方公司的卧底，掌握了公司面试顺序，每次都给最差的排序，如何应对？



随机算法

先对应聘者进行排列，再确定最佳应聘者的随机算法。

- 随机化过程在算法中而不是在输入中体现
- Given a particular input, we can no longer say what its hiring cost will be. Each time we run the algorithm, we can get a different hiring cost. (算法的运行时间与输入无关)
- 算法的最坏运算时间不取决于特定的输入
- 只有当随机数产生器产生很不幸运的数时，算法的运算时间最差。



雇用问题的随机算法

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  $i$ 
```

- 根据引理5.2，如果应聘者以随机顺序出现，则聘用一个新办公助理的平均情况下雇佣次数大约是 $\ln n$ 。
- 现在，修改了算法，使得随机发生在算法上，那么聘用一个新办公助理的期望次数仍是 $\ln n$ 吗？



雇用问题的随机算法

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，我们已经达到了和引理5.2相同的情况。 ■

引理5.2和引理5.3的区别：

- 引理5.2中，假设输入是随机分布的，求的是平均情形下的雇用费用。
- 在引理5.3中，将随机化作用在算法上，求的是雇用费用的期望值。



随机排列数组

随机算法需要对给定的输入变换排列，以使输入随机化。

- 不失一般性，假设给定一个数组A，包含元素1到n。
- 目标：产生一个均匀随机排列，使得 $n!$ 种排列中的每一种被取到的概率一致($=1/n!$)

那么如何产生输入的随机排列呢？



- 这里介绍两种随机化方法。
- **方法一**: 为数组的每个元素**A[i]**赋一个随机的优先级**P[i]**, 然后根据优先级对数组中的元素进行排序。
 - ✓ 例: 设初始数组**A=<1,2,3,4>**, 随机选择的优先级是 **P=<36,3,62,19>**, 则将产生一个新数组:
B=<2,4,1,3>



上述策略的过程描述

PERMUTE-BY-SORTING(A)

```
1   $n = A.length$ 
2  let  $P[1..n]$  be a new array
3  for  $i = 1$  to  $n$ 
4       $P[i] = \text{RANDOM}(1, n^3)$ 
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。使用范围 $1 \sim n^3$ 是为了让 P 中所有优先级尽可能唯一。
- 第5步排序时间为 $O(n \log n)$
- 排序后，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。最后得到一个“随机”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：

首先考虑元素A[i]分配到第i个最小优先级的特殊排列，可以证明这个排列发生的概率是 $1/n!$ 。

证明如下：

设 E_i 代表元素A[i]分配到第i个最小优先级的事件 ($i=1,2,\dots,n$)，则对所有的 E_i ，整个事件发生的概率是：

$$\begin{aligned} & \Pr\{E_1 \cap E_2 \cap E_3 \cap \dots \cap E_{n-1} \cap E_n\} \\ = & \Pr\{E_1\} \cdot \Pr\{E_2 | E_1\} \cdot \Pr\{E_3 | E_2 \cap E_1\} \cdot \Pr\{E_4 | E_3 \cap E_2 \cap E_1\} \\ & \dots \Pr\{E_i | E_{i-1} \cap E_{i-2} \cap \dots \cap E_1\} \dots \Pr\{E_n | E_{n-1} \cap \dots \cap E_1\} \end{aligned}$$



这里

- $\Pr\{E_1\}$ 是从一个n元素的集合中随机选取的最小优先级的概率，有 $\Pr\{E_1\}=1/n$;
- $\Pr\{E_2|E_1\}=1/(n-1)$ ，因为假定了元素A[1]有最小的优先级，余下来的n-1个元素都有相等的可能成为第二小的优先级别
- 而一般对于*i*=2,3,..,n，有：

$$\Pr\{E_i|E_{i-1} \cap E_{i-2} \cap \dots \cap E_1\}=1/(n-i+1).$$

含义：给定元素A[1]到A[i-1]的前*i*-1个小的优先级，在剩下的*n-(i-1)*个元素中，每个都有可能是第*i*小优先级



最终有：

$$\begin{aligned} & \Pr\{E_1 \cap E_2 \cap E_3 \cap \cdots \cap E_{n-1} \cap E_n\} \\ = & \Pr\{E_1\} \cdot \Pr\{E_2 | E_1\} \cdot \Pr\{E_3 | E_2 \cap E_1\} \cdot \Pr\{E_4 | E_3 \cap E_2 \cap E_1\} \\ & \cdots \Pr\{E_i | E_{i-1} \cap E_{i-2} \cap \cdots \cap E_1\} \cdots \Pr\{E_n | E_{n-1} \cap \cdots \cap E_1\} \\ = & \left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right) \cdots \left(\frac{1}{2}\right)\left(\frac{1}{1}\right) = \frac{1}{n!} \end{aligned}$$

因此，获得该排列的概率是 $1/n!$ ，是一个均匀随机排列
。



扩展上述证明过程：

- 考虑集合 $\{1, 2, \dots, n\}$ 的任意一个确定排列

$$\sigma = <\sigma(1), \sigma(2), \dots, \sigma(n)>.$$

- 用 r_i 表示元素 $A[i]$ 优先级的排名，即 $r_i = \sigma(i)$ 。
- 定义 E_i 为元素 $A[i]$ 分配到优先级第 $\sigma(i)$ 小的事件(即 $r_i = \sigma(i)$)， 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 $1/n!$ 。



随机排列数组

The method is better

(2) 方法二：RANDOMIZE-IN-PLACE（原地排列给定的数列）

RANDOMIZE-IN-PLACE(A, n)

```
for( $i=1; i<=n; i++$ )
    swap( $A[i], A[\text{RANDOM}(i, n)]$ )
```

思路：

- 第 i 次迭代后，从 $A[i..n]$ 中随机选取 $A[i]$
- 第 i 次迭代后不再改变 $A[i]$

优势：

- 线性时间($O(n)$)，不用排序。
- 更少的随机比特数(n 个 1 到 n 范围的随机数，对比 n 个 1 到 n^3 范围的随机数) (仅需更小范围的随机数产生器)
- 不需要额外的辅助空间

1	2	3	...	n
$A(1)$	$A(2)$	$A(3)$...	$A(n)$

1	2	3	...	i_1	...	n
$A(i_1)$	$A(2)$	$A(3)$...	$A(1)$...	$A(n)$

1	2	3	...	i_2	...	n
$A(i_1)$	$A(i_2)$	$A(3)$...	$A(2)$...	$A(n)$



正确性:

- 给定一个包含 n 个元素的集合, 一个 k -排列是指一个包含其中 k 个元素的序列。
- K -排列的个数为 $n!/(n-k)!$

$$P_n^k = C_n^k \cdot P_k^k = \frac{n!}{k!(n-k)!} \cdot k! = \frac{n!}{(n-k)!}$$

定理 RANDOMIZE-IN-PLACE 产生一个均匀随机排列.

证明 利用循环不变式: 在第 i 次迭代之前, 对每个可能的 $(i-1)$ -排列, 子数组 $A[1 .. i-1]$ 选取任意一个 $(i-1)$ -排列的概率为 $(n-i+1)!/n! ?$



循环不变式: $A[1 \dots i-1]$ 包含一个 $(i-1)$ -排列 的概率为 $(n-i+1)!/n!$?

- **初始条件 :** 对 $i=1$, $A[1\dots 0]$ 包含某个 0-排列 的概率为 $n!/n!=1$, 其中 $A[1\dots 0]$ 为空子列 , 0-排列则是不包含任何元素的序列
- **归纳步骤 :** 假设第 i 次迭代前 , 设 $(i-1)$ -排列 $A[1 \dots i-1]$ 中 , 任一排列发生的概率为 $(n-i+1)!/n!$, 则需证明在第 i 次迭代后 , 任一 i -排列 的概率为 $(n-i)!/n!$
 - 考虑一个特殊的 i -排列 $R = < x_1, x_2, \dots, x_i >$, 其中包含一个 $(i-1)$ -排列 $< x_1, x_2, \dots, x_{i-1} >$, 算法在 $A[i]$ 放置 x_i
 - 令事件 $E1$ 表示算法实际输出 $(i-1)$ -排列 R' 为 $A[1 \dots i-1]$, 根据循环不变量 , $\Pr\{E1\} = (n-i+1)!/n!$
 - 令事件 $E2$ 表示第 i 次迭代后输出 $A[i]$ 为 x_i , 则当且仅当 $E1$ 和 $E2$ 同时发生时我们得到 i -排列 R 为 $A[1 \dots i]$, 其概率为 $\Pr\{E2 \cap E1\} = ?$



循环不变式: $A[1 \dots i-1]$ 包含一个 $(i-1)$ -排列的概率为 $(n-i+1)!/n!$?

- **初始条件 :** 对 $i=1$, $A[1\dots 0]$ 包含某个 0-排列 的概率为 $n!/n!=1$, 其中 $A[1\dots 0]$ 为空子列 , 0-排列则是不包含任何元素的序列
- **归纳步骤 :** 假设第 i 次迭代前 , 设 $(i-1)$ -排列 $A[1 \dots i-1]$ 中 , 任一排列发生的概率为 $(n-i+1)!/n!$, 则需证明在第 i 次迭代后 , 任一 i -排列 的概率为 $(n-i)!/n!$
 - 根据条件概率 : $\Pr\{E_2 \cap E_1\} = \Pr\{E_2 | E_1\} \Pr\{E_1\}$.
 - 给定 E_1 的条件下 , $A[i]$ 有 $n-i+1$ 种取法 , 取到 x 的概率为 $1/(n-i+1)$, 即 $\Pr\{E_2 | E_1\} = 1/(n-i+1)$
 - 因此

$$\begin{aligned}\Pr\{E_2 \cap E_1\} &= \Pr\{E_2 | E_1\} \Pr\{E_1\} \\ &= \frac{1}{n-i+1} \cdot \frac{(n-i+1)!}{n!} = \frac{(n-i)!}{n!}\end{aligned}$$



循环不变式: $A[1 .. i-1]$ 包含一个 $(i-1)$ -排列的概率为 $(n-i+1)!/n!$?

- **初始条件 :** 对 $i=1$, $A[1...0]$ 包含某个0-排列的概率为 $n!/n!=1$, 其中 $A[1...0]$ 为空子列 , 0-排列则是不包含任何元素的序列
- **归纳步骤 :** 假设第 i 次迭代前 , 设 $(i-1)$ -排列 $A[1 .. i-1]$ 中 , 任一排列发生的概率为 $(n-i+1)!/n!$, 则需证明在第*i* 次迭代后 , 任一 i -排列 的概率为 $(n-i)!/n!$
- **终止 :** 终止时 , $i=n+1$, $A[1...n]$ 是一个给定 n -排列的概率为 $(n-(n+1)+1)!/n! = 1/n!$
- 该算法又称为Fisher–Yates shuffle或者Knuth Shuffle



第四讲 概率分析和随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法
- 概率分析和随机算法的应用：在线雇用问题，生日悖论，球与盒子



在线雇用问题

➤ 情景描述：

假设现在我们不希望面试所有的应聘者来找到最好的一个，也不希望因为不断有更好的申请者出现而不停地雇用信任解雇旧人。我们愿意雇用接近最好的应聘者，只雇用一次。但是，我们必须遵守猎头公司的一个规定：在每次面试之后，必须给出面试结果，要么雇用候选人，要么拒绝。

➤ **Goal:** 最小化面试次数和最大化雇用应聘者的质量取得平衡。



解决思路：

- 1) 面试一个应聘者之后，给他一个分数，令 $score(i)$ 表示给第*i*个应聘者的分数，并且假设所有应聘者得分都不相同；
- 2) 面试前面k个 ($k < n$) 应聘者然后拒绝他们，再雇用其后比前面的应聘者更高分数的第一个应聘者。

ON-LINE-MAXIMUM (k, n)

```
bestscore ← -∞
for  $i \leftarrow 1$  to  $k$ 
    do if  $score(i) > bestscore$ 
        then  $bestscore \leftarrow score(i)$ 
for  $i \leftarrow k+1$  to  $n$ 
    do if  $score(i) > bestscore$ 
        then return  $i$ 
return  $n$ 
```



- 对每个可能k值，希望能确定雇用到最好应聘者的概率
 - 概率最高的k值可以用来最好的实现这个策略
 - K太小太大雇用到最好应聘者的概率都不高
-
- 设S是我们成功选择的是最好应聘者的事件， S_i 表示最好的应聘者是第i个面试者时成功的事件，则

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

互斥事件！

- S_i 发生表示两件事必须发生：1.最好的应聘者在第i个位置；2.i前面没有应聘者被雇用



- 设 B_i 表示最好的应聘者在第*i*个位置上的事件,
 $\Pr\{B_i\}=1/n$
- 设 O_i 表示*k+1*到*i-1*中没有应聘者被选取的事件，即当
 $k+1 \leq j \leq i-1$ 时， $\text{score}(j) < \text{bestscore}$ ，此事件仅取决于1
到*i-1*的排列情况，与 B_i 是独立事件。

- 前*i-1*个应聘者分数最高的可以是前*K*个中的任意一个
位置，所以 $\Pr\{O_i\}=k/(i-1)$
- 要想雇用到第*i*个应聘者(S_i)且为最佳应聘者，必须同
时有发生 B_i 和 O_i :

$$\Pr\{S_i\} = \Pr\{B_i \cap O_i\} = \Pr\{B_i\} \Pr\{O_i\} = \frac{k}{n(i-1)}$$



➤ 于是

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

➤ 利用积分性质进行缩放

$$\text{而: } \int_k^n \frac{1}{x} dx \leq \sum_{k=1}^{n-1} \frac{1}{i} \leq \int_{k-1}^{n-1} \frac{1}{x} dx$$

$$\text{所以: } \frac{k}{n}(\ln n - \ln k) \leq \Pr\{S\} \leq \frac{k}{n}(\ln(n-1) - \ln(k-1))$$

➤ 我们希望最大化成功概率，因而主要关注如何选取 k 的值，使其能最大化 $\Pr\{S\}$ 的下界：上式左侧当 $k = n/e$ 时取最大值 $1/e$ ，即如果用 $k=n/e$ 来实现我们的策略，则可以以至少 $1/e$ 的概率成功雇用到最有资格的应聘者



第四讲 概率分析和随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法
- 概率分析和随机算法的应用：在线雇用问题, 生日悖论, 球与盒子



生日悖论

一个屋子里的人数必须达到多少人，才能使其中两人生日相同的机会达到50%？

- 问题描述: 用整数 $1, 2, \dots, k$ 对屋子里的人编号, k 是总人数, $n=365$, 对于 $i=1, 2, \dots, k$, 设 b_i 表示编号为 i 的人的生日, 其中 $1 \leq b_i \leq n$, 假设生日均匀独立分布在一年 n 天中, 对于 $i=1, 2, \dots, k$ 和 $r=1, 2, \dots, n$, $\Pr\{b_i=r\}=1/n$
- 假设 i 和 j 的生日都落在同一日 r 上的概率是:

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



生日悖论



这样他们的生日落在同一天的概率是：

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

- 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
- 至少两个人生日相同的概率等于1减去所有人的生日都互不相同的概率。
- K个人的生日互不相同的概率: $B_k = \cap_{i=1}^k A_i$ ， 其中Ai是指对所有j<i, i与j生日不同的概率。
 - $B_k = A_k \cap B_{k-1}$
 - $\Pr\{B_k\} = \Pr\{B_{k-1}\} \Pr\{A_k | B_{k-1}\}$



生日悖论

- 如果 b_1, \dots, b_{k-1} 两两不同，则 $\Pr\{A_k | B_{k-1}\} = \frac{n-k+1}{n}$
- 我们有
$$\begin{aligned}\Pr(B_k) &= \Pr\{B_{k-1}\} \Pr\{A_k | B_{k-1}\} \\&= \Pr\{B_{k-2}\} \Pr\{A_{k-1} | B_{k-2}\} \Pr\{A_k | B_{k-1}\} \\&\quad \vdots \\&= \Pr\{B_1\} \Pr\{A_2 | B_1\} \Pr\{A_3 | B_2\} \cdots \Pr\{A_k | B_{k-1}\} \\&= 1 \cdot \left(\frac{n-1}{n}\right) \left(\frac{n-2}{n}\right) \cdots \left(\frac{n-k+1}{n}\right) \\&= 1 \cdot \left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right)\end{aligned}$$
- 由 $1 + x \leq e^x$ 得到： $k \geq (1 + \sqrt{1 + (8 \ln 2)n}) / 2$
- 当 $n=365$, $k \geq 23$ 可以保证以 $1/2$ 的概率，有两个人生日相同



生日悖论

解法二：利用指示器随机变量来分析

- 定义指示器随机变量

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

- 两个人生日相同的概率为 $n * (1/n) * (1/n) = 1/n$
- 所以 $E(X_{ij}) = 1/n$



生日悖论

- 令 X 表示具有相同生日的两人对数目，则

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

- 对 $n = 365$, $k = 28$ 时, X 期望值为1
- 注: 与前一种准确数目不等, 但渐近意义上是相等的, 都是 $\Theta(\sqrt{n})$), 渐进分析对 n 很大时才有意义。



第四讲 概率分析和随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法
- 概率分析和随机算法的应用：在线雇用问题，生日悖论，球与盒子

论，球与盒子



球与盒子



- 问题：把相同的球随机投到 b 个盒子里，在每个盒子都有球之前，要投多少个球

- 解答：

1. 定义“击中”为球落在空盒子里面，
2. 定义第 i 阶段为从第 $(i-1)$ 次击中到 i 次击中之间的投球。
3. 第 i 阶段，投球击中的概率为 $(b-i+1)/b$ ，因此投球数 n_i 的期望为 $E[n_i] = b/(b-i+1)$ ，
4. 所以，总的投球数期望为

$$E[n] = E\left[\sum_{i=1}^b n_i\right] = \sum_{i=1}^b E[n_i] = \sum_{i=1}^b \frac{b}{b-i+1} = b \sum_{i=1}^b \frac{1}{i} = b(\ln b + O(1))$$