



华中科技大学

计算机科学与技术学院

School of Computer Science & Technology, HUST

算法设计与分析

刘渝

Liu_yu@hust.edu.cn

2023秋季-华科-计算机

22级CS

Anytime·Everywhere
Computing

计算·无限





算法分析与设计

第五章

概率分析和随机算法

Time



目 录

01、概率分析

02、随机算法

概率分析

概率分析：就是在问题分析中应用概率的理念

首先对**输入分布**做出假设。

例如：**随机顺序**——这种随机性由输入自身决定（不是你决定的）

Example: 雇佣问题

5.1 雇佣问题

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。

Example: 雇佣问题

- 假设雇用代理推荐过来的应聘候选人有 n 人，编号为1到 n 。
- 并假设你在面试完一个应聘者后，就能决定该应聘者是否是你目前见过的最佳人选，如果是，则立即雇用。

雇佣算法可描述如下：

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  $i$ 
```

Example: 雇佣问题

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  $i$ 
```

该过程中检查序列中的每个成员，所以面试的总人数就是 n 。其中若干人被雇用，记被雇用总人数为 m 。

设介绍一个人面试的费用为 c_i ，发生一次雇用的费用为 c_h ，
则该算法的总费用是 $O(c_i n + c_h m)$ 。

Example: 雇佣问题

算法的总费用是: $O(c_i n + c_h m)$.

- 进一步会发现介绍面试的总费用 $c_i n$ 是恒定的, 因为不管雇用多少人, 总会面试 n 个应聘者。
- 而发生雇用的总费用 $c_h m$ 是变数, 因为 m 是未知的。
- 所以我们只需关注于雇用总费用 $c_h m$ 即可。因此, 求解雇佣问题就是对算法中best的**更新频率 m (次数)**建立模型。

Example: 雇佣问题

算法的总费用是: $O(c_i n + c_h m)$.

■ 最坏情形分析:

- 当应聘者质量按出现的次序严格递增时, 就会出现最坏情况: 最坏情形下实际雇用了所有参加面试的应聘者。
- 此时面试了 n 次, 雇用了 n 次, 所以雇用总费用是 $O(c_h n)$ 。

■ 一般情况分析:

- 事实上, 一般情形下应聘者不会总以质量递增的次序出现。
- 那么整个过程可能有多少人被雇用呢?

*Example:*雇佣问题

一般情况下需要注意的事实

- 应聘者会以任意可能的次序来应聘，有好有坏，事先既不知道他们出现的次序，也不能控制这个次序（假设雇用代理也不关心好坏的话）

我们用概率分析的方法分析上述问题的雇用费用。假设雇佣问题中应聘者以随机顺序出现。这也意味着所有应聘者是一种全序关系，即任意两个应聘者可以比较好坏并决定哪一个更有资格。

Example: 雇佣问题

用 $\text{rank}(i)$ 表示应聘者质量

- 不失一般性，设 $\text{rank}(i)$ 为整数，取值 $1 \sim n$ ，1代表最差， n 代表最好，且所有应聘者的 rank 各不相同；
- 则任意序列 $R = \langle \text{rank}(1), \text{rank}(2), \dots, \text{rank}(n) \rangle$ 将是 $\langle 1, 2, \dots, n \rangle$ 的一个排列。注：
： $\langle 1, 2, \dots, n \rangle$ 的排列共有 $n!$ 种。

R 是数字1到 n 的 $n!$ 种排列中的任一个，随机出现，且为**均匀随机排列**，即在 $n!$ 种可能的排列中，每种排列以 $1/n!$ 的可能性“等概率”出现。

随机算法

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，输入分布是未知的。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**。

输入如何可控？

使一个算法中的某部分的行为随机化，然后利用**概率**和**随机性**作为算法设计与分析的工具进行相关处理。

随机出现

在雇佣问题中，看起来应聘者好像以随机顺序出现，但无法知道是否真的如此？

为了分析雇佣问题一般情况下的解，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

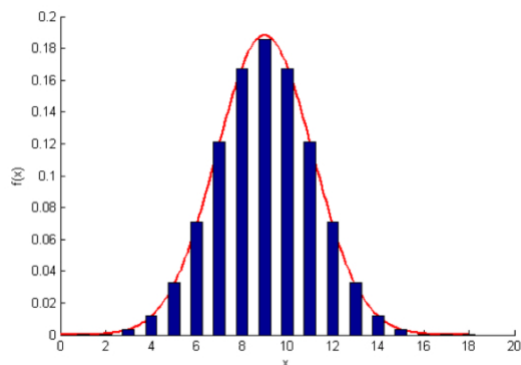
方法

假设雇用代理推荐了 n 个应聘者，可以事先给我们一份名单，而我们每天随机选择某个应聘者来面试

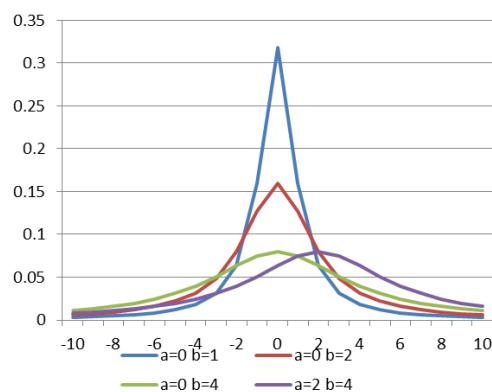
——这有什么不同呢？

数据分布

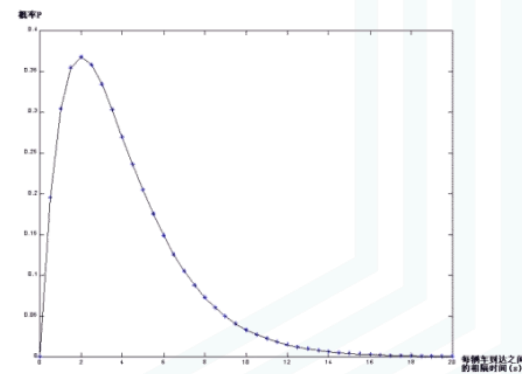
正态分布



柯西分布



泊松分布



随机算法

如果一个算法的行为不仅由输入决定，而且也由一个**随机数生成器**产生的数值决定，则称这个算法是随机化的（Randomized），这样的算法称为**随机算法**。

随机数生成器

- 调用RANDOM(a,b)将返回一个介于a和b之间的整数，且每个整数以等概率出现。
- 而且每次RANDOM返回的整数**都独立于**前面调用的返回值。

例：RANDOM(0,1)产生0和1的概率都为1/2；

RANDOM(3,7)可以返回3, 4, 5, 6, 7, 每个出现的概率为1/5；

期望运行时间

我们将一个随机算法的平均运行时间称为期望运行时间。

- 此时，算法的最终输入由随机数发生器产生
- 一般而言，当概率分布是发生在算法的原始输入上时，我们讨论算法的“平均情况运行时间”，而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”。

指示器随机变量

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

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：

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

Example: 指示器随机变量

例：硬币有正反两面，抛掷一枚硬币，求正面朝上的期望次数。

将硬币正面朝上的事件记为H，反面朝上的事件记为T。从而有样本空间 $S=\{H, T\}$ ，且 $\Pr(H)=\Pr(T)=1/2$ 。

对于正面朝上的事件H，定义一个指示器随机变量 X_H ，记录在一次抛硬币时正面朝上的次数：

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

Example: 指示器随机变量

例：硬币有正反两面，抛掷一枚硬币，求正面朝上的期望次数。

一次抛掷硬币正面朝上的期望次数：

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

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

指示器随机变量 引理

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。

指示器随机变量

n次抛掷硬币正面朝上的期望次数是多少？

- 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：

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

指示器随机变量

n次抛掷硬币正面朝上的期望次数是多少？

$X_i = I\{\text{第}i\text{次抛掷时出现事件}H\}$ 。 $X = \sum_{i=1}^n X_i$ 。

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之和的期望，也等于n个指示器随机变量值期望的和。

指示器随机变量 引理

若每个指示器随机变量的期望值为 $1/2$ ，则总和 X 的期望值为：

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

指示器随机变量 雇佣问题

计算雇用新办公助理的期望次数：

- 假设应聘者以随机顺序出现。
- 设 X 是一个随机变量，其值等于雇用新办公助理的总次数。
- 定义 n 个指示器随机变量 X_i ，与应聘者 i 的一次面试相对应，根据 i 是否被雇用有：

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

$$X = X_1 + X_2 + \cdots + X_n$$

根据定理，有 $E[X_i] = \Pr\{\text{应聘者 } i \text{ 被雇用}\}$

应聘者 i 被雇用的概率是多少呢？

指示器随机变量 雇佣问题

```
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
```

计算过程HIRE-ASSISTANT中第5~6行被执行的概率：

在第6行中，若应聘者*i*被雇用，则他就要比前面*i*-1个应聘者更优秀。

因为已经假设应聘者以随机顺序出现，所以这*i*个应聘者也以随机次序出现。而且在这*i*个应聘者中，任意一个都可能是最有资格的。那么，应聘者*i*比前*i*-1个应聘者更有资格的概率是 $1/i$ ，即它将有 $1/i$ 的概率被雇用。

□ 故由引理可得： $E[X_i] = 1/i$

指示器随机变量 雇佣问题

$$E[X_i] = 1/i \quad X = X_1 + X_2 + \cdots + X_n$$

计算 $E[X]$:

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

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



总结：



华中科技大学
计算机科学与技术学院
School of Computer Science & Technology, HUST

概率分析的含义：在算法分析中应用概率的思想

随机算法：将输入随机化，使算法不依赖于原始输入。

随机数发生器：Random

平均计算时间：概率分布定义在输入上。

期望运行时间：算法本身做出随机选择。





总结：



华中科技大学
计算机科学与技术学院
School of Computer Science & Technology, HUST

指示器随机变量：
$$I\{A\} = \begin{cases} 1 & \text{if } A \text{ occurs,} \\ 0 & \text{if } A \text{ does not occur.} \end{cases}$$

概率和期望的关系：一个事件A对应的指示器随机变量的期望值等于该事件发生的概率。

用指示器随机变量分析雇用问题：

- 应聘者*i*被雇佣的概率是 $1/i$ 。
- 面试*n*个人，平均起来，实际上大约只雇佣了他们之中的 $\ln n$ 个人



指示器随机变量 引理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 $O(c_h \ln n)$.

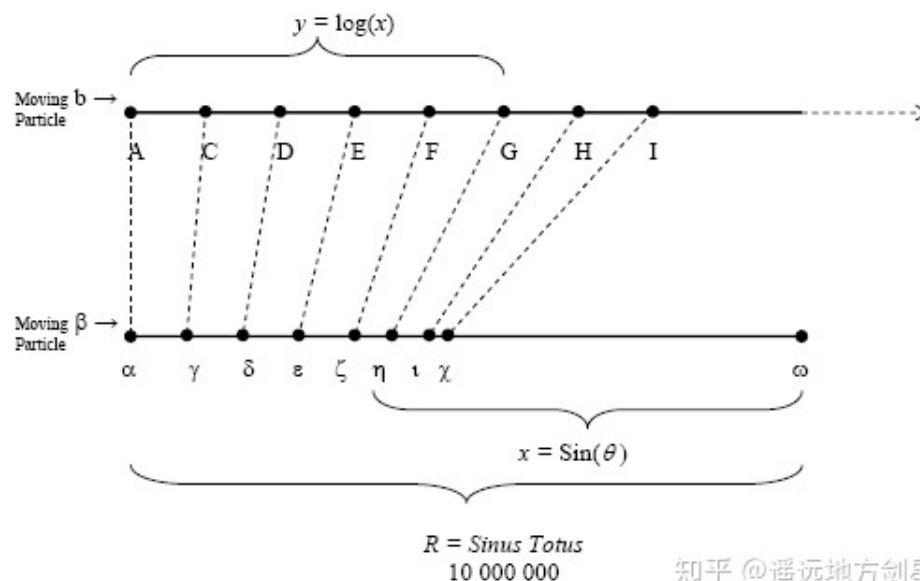
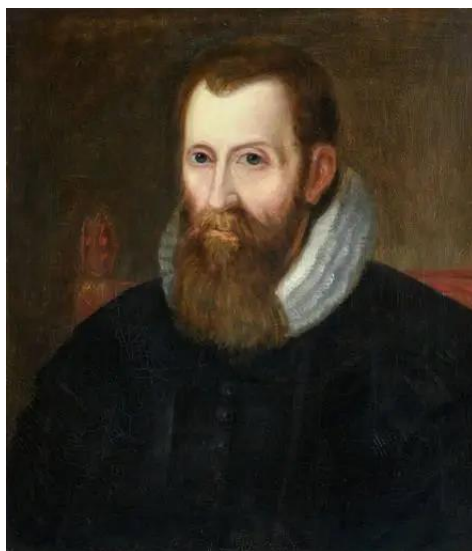
可见，平均情形下的雇用费用 $c_h \ln n$ 比最坏情况下的雇用费用 $O(c_h n)$ 有了很大的改进。

自然对数

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e$$

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \dots$$

约翰.纳皮尔



知乎 @遥远地方剑星



总结：

- 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。
- 采用随机算法，分析算法的期望值



华中科技大学
计算机科学与技术学院
School of Computer Science & Technology, HUST



雇用问题的随机算法

- 随机算法在算法运行前**先随机地排列应聘者**，体现所有排列都是等可能出现的性质。
- 核心思路：随机算法不是假设输入的分布，而是**设定一个分布**。
 - 根据前面的讨论，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，虽然修改了算法，使得随机发生在算法上，但雇用一个新办公助理的期望次数仍大约是 $\ln n$ 。

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  $i$ 
```

指示器随机变量 引理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，已经达到了和HIRE-ASSISTANT概率分析时相同的情况

引理2 与 引理3

引理2在输入上做了随机分布的假设，求的是平均情形下的雇用费用。

引理3的随机化作用在算法上，求的是雇用费用的期望值。



思考：



华中科技大学
计算机科学与技术学院
School of Computer Science & Technology, HUST

- 如果不考虑随机处理，算法是不是“确定”的？

此时，雇用新办公助理的次数依赖于初始时各个应聘者的排名，对于任何特定输入，雇用一个新办公助理的次数始终相同。





思考：



华中科技大学
计算机科学与技术学院
School of Computer Science & Technology, HUST

- 如果不考虑随机处理，算法是不是“确定”的？

如：排名列表 $A_1 = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle$ ，新办公助理会雇用 10 次。

排名列表 $A_2 = \langle 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 \rangle$ ，新办公助理会雇用 1 次。

排名列表 $A_3 = \langle 5, 2, 1, 8, 4, 7, 10, 9, 3, 6 \rangle$ ，新办公助理会雇用 3 次。





总结：

- 算法RANDOMIZED-HIRE-ASSISTANT在算法运行前先随机地排列应聘者，是**随机算法**。
 - ✓ 此时，“随机”发生在算法上，而不是在输入分布上。
 - ✓ **随机算法**中，任何一个给定的输入，如 A_1 或 A_3 ，都无法说出best会被更新多少次的。因为随机发生在算法中，而不是直接在输入分布上。**没有特别的输入会引出它的最坏行为**（当然也没有特别的输入不会引出它的最坏行为）。



随机数组

随机算法需要通过**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 。**随机化的目标是构造这个数组的一个均匀随机排列**

介绍两种随机化方法

随机生成数组 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：如果初始数组 $A = \langle 1, 2, 3, 4 \rangle$ ，随机选择的优先级是 $P = \langle 36, 3, 62, 19 \rangle$ ，
则将产生一个数组： $B = \langle 2, 4, 1, 3 \rangle$

随机生成数组 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。

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。

方法一 引理4

假设所有优先级都不同，则过程PERMUTE-BY-SORTING产生的输入是均匀随机排列的

证明：

从考虑每个元素 $A[i]$ 分配到第 i 小优先级的特殊排列开始讨论，说明这个排列正好发生的概率是 $1/n!$ 。

方法一 引理4

假设所有优先级都不同，则过程PERMUTE-BY-SORTING产生的输入是均匀随机排列的

证明：

设 E_i 代表元素 $A[i]$ 分配到第 i 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： $\Pr\{E_1 \cap E_2 \cap E_3 \cap \dots \cap E_{n-1} \cap E_n\}$

这里， $\Pr\{E_1\}$ 是为第一个元素选择的优先级恰好为第1小优先级的概率，故有

$\Pr\{E_1\}=1/n$ 。

方法一 引理4

假设所有优先级都不同，则过程PERMUTE-BY-SORTING产生的输入是均匀随机排列的

证明：

对于 $i=2,3,\dots,n$ ，一般有：

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

即：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，而恰好选中第 i 小优先级的概率是 $1/(n-i+1)$ 。

方法一 引理4

假设所有优先级都不同，则过程PERMUTE-BY-SORTING产生的输入是均匀随机排列的

证明：

最终有：

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

说明获得等同排列的概率是 $1/n!$ 。

因此PERMUTE-BY-SORTING过程能产生一个均匀随机排列

引理4 扩展

上述证明过程，对任何优先级的排列都有效

- 考虑集合 $\{1, 2, \dots, n\}$ 的任意一个确定排列

$$\sigma = \langle \sigma(1), \sigma(2), \dots, \sigma(n) \rangle。$$

- 由于优先级各不相同，所以可以重新定义映射： $r(\sigma(i)) \rightarrow i$ 。
- 定义 E_i 为“**元素 $A[i]$ 分配到优先级是第 $\sigma(i)$ 小的事件**”，则其等价于“**元素 $A[i]$ 分配到优先级是第 $r(\sigma(i))$ 小的事件**”，从而还原到“**元素 $A[i]$ 分配到第 i 小优先级**”的情形，所以同样的证明仍适用。
- 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 $1/n!$** 。

随机生成数组 方法二

原址排列给定数组。第 i 次迭代时，元素 $A[i]$ 从元素 $A[i]$ 到 $A[n]$ 中随机选取

RANDOMIZE-IN-PLACE(A)

```
1   $n = A.length$   
2  for  $i = 1$  to  $n$   
3      swap  $A[i]$  with  $A[\text{RANDOM}(i, n)]$ 
```

第 i 次迭代后， $A[i]$ 不再改变

方法二 引理5

过程RANDOMIZE-IN-PLACE可计算出一个均匀随机排列

RANDOMIZE-IN-PLACE(A)

```
1   $n = A.length$   
2  for  $i = 1$  to  $n$   
3      swap  $A[i]$  with  $A[\text{RANDOM}(i, n)]$ 
```

第 i 次迭代后, $A[i]$ 不再改变

证明: 使用循环不变式来证明该过程能产生一个均匀随机排列。



作业：



华中科技大学
计算机科学与技术学院
School of Computer Science & Technology, HUST

当人数达到多少，可使得这些人中有相同生日的可能性达到50%？

