算法设计与分析

主讲人：吴庭芳

Email: *tfwu@suda.edu.cn*

苏州大学 计算机学院

第四讲 概率分析与随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法

雇用问题







- 对于运行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**所有可能输入产生的运行时间求平均**，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**

雇用问题

情景：一个月内雇用最佳人选任办公室助理

- 猎头公司帮你物色办公助理候选人
- 每天推荐一名候选人（连续推荐 n 个）
- 面试一个候选人之后决定是否雇用，**如果这个应聘者比当前的办公室助理更优秀**，就会辞掉当前的办公室助理，聘用这个新的应聘者
- 假定面试一个候选人支付猎头公司的费用是 c_i
- **雇用一名候选人的费用是 c_h（解雇当前办公助理的费用 + 支付给猎头公司的中介费用）**

***Goal*: 该方案的费用是多少?**

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 是否是所见过的最优秀的人选
  - 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*)	cost	times
1. ***best*** = 0 // candidate 0 is a least-qualified dummy candidate		
2. for ***i*** = 1 to ***n***		
3. interview candidate ***i***	c_i	n
4. if candidate ***i*** is better than candidate ***best***		
5. ***best*** = ***i***		
6. hire candidate ***i***	c_h	m

  - 在上述过程中，检查序列中的每个应聘者，维护一个当前的获胜者 *best*，最后找出序列中最优秀的应聘者

雇用问题

- **雇用策略费用分析:**

设面试推荐费用为 c_i，雇用费用为 c_h

- 假设总共面试了 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，始终要面试 n 个应聘者。只需要专注于分析雇用费用 $c_h m$ 即可，$c_h m$ 与面试应聘者的顺序有关
- **那么整个过程会产生多少雇用费用呢?**

雇用问题

- **最坏情况分析：**
  - **最坏情形**：实际雇用了每个面试的应聘者
  - 即应聘者的优秀程度按出现的次序严格递增
  - 此时面试了 n 次，雇用了 n 次，则雇用总费用是 $O(c_h n)$

- **平均情况分析：**
  - **一般情形**：应聘者不会总以质量递增的次序出现（$1/n!$的概率）
  - 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，**在平均情况下怎么分析算法运行时间?**

雇用问题

- **雇用问题概率分析过程:**
  - 所有应聘者存在全序关系，即任意两个应聘者可以比较排列名次的 *rank* 值，并决定哪一个更有资格
  - 应聘者编号：1~n
  - 使用从 1 到 n 的唯一序号将应聘者进行名次排列，应聘者 i 名次为 $rank(i)$，$rank(i) \in [1, n]$，每个应聘者具有唯一一个名次。不失一般性，一个较高的名次对应一个更优秀的应聘者
  - 因此，n 个应聘者的排列名次序列 $\langle rank(1), rank(2), \ldots, rank(n)\rangle$，该序列是序列 $\langle 1, 2, \ldots, n\rangle$ 的一个排列

雇用问题

- **雇用问题概率分析过程**
  - 为了进行概率分析，对输入分布做如下**假设**：假设雇用问题中**应聘者以随机顺序出现**，并且这种随机性由输入自身决定
  - 每一个应聘者对应唯一一个排列名次，因此，**应聘者以随机顺序出现等价于排列名次序列 <*rank*(1), *rank*(2), …, *rank*(*n*)> 是数字 1 到 *n* 的 *n*! 种排列中的任意一个**
  - 或者，称这些排列名次构成一个**均匀随机排列**，即在 $n!$ 种可能的排列中，每种排列以**"等概率"**情形出现，这样就对雇用问题的输入做了一种分布假设

雇用问题

- **概率分析：在问题分析中应用概率技术**
  - 概率分析用来分析一个算法的运行时间，也可用于其他量的分析，例如利用概率分析技术来分析雇用费用
  - 概率分析的本质：需要使用或假定关于输入的分布
- **概率分析：首先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，计算出一个平均情况下的运行时间。当报告此种类型的运行时间时，称其为平均情况运行时间**

雇用问题

- **假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，从而可以将概率分析作为一种手段来设计高效算法
  - ✓ 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

雇用问题

- **随机算法**
  - 为了利用概率分析技术，就需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，也无法从计算上对该分布知识建立模型
  - **那么如何让输入变得可控?**

    通过对算法中的某部分的行为进行随机化，从而为输入强加一种分布，则可利用概率和随机性作为工具进行处理

雇用问题

雇用问题的随机算法分析

- 在雇用问题中，看起来应聘者好像以随机顺序出现，但是我们无法知道是否确实如此
- 因此，我们人为地对应聘者的出现次序进行更强的控制，使其达到一种“随机”的出现效果

雇用问题的随机算法

- 猎头公司预先提供 n 个应聘者名单
- 每天随机选择（通过随机生成器实现）某个应聘者进行面试
- 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

雇用问题

- **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的（Randomized）**
- **随机数生成器RANDOM：**
  - 调用 RANDOM(*a*, *b*) 将返回一个介于 *a* 和 *b* 之间（包含 *a* 和 *b*）的整数，并且每个整数以等概率出现

例： RANDOM(0, 1)：返回 0 和 1，每个出现的概率都为 1/2

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

  - 每次 RANDOM 返回的整数**都独立于**前面调用的返回值
  - 大多数编程环境中的 RANDOM 实际上由一个确定的算法模拟产生的（伪随机产生器），其结果表面上看上去像是随机数

雇用问题

- **期望运行时间：**随机算法的输入次序最终由随机数生成器决定，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 概率分析与随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法

指示器随机变量

- 引入指示器随机变量的目的：为了建立**概率** (probabilities) 和**期望** (expectations) 之间的联系，用于实现概率与期望之间的转换
- 指示器随机变量的定义：给定一个样本空间 S (samplespace) 和一个事件 A (event)，那么事件 A 对应的**指示器随机变量** $\mathrm{I}\{A\}$ 定义为：

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

指示器随机变量







□ **例：**抛掷一枚硬币，求**正面朝上**的期望次数。

解：首先，样本空间 $S=\{H, T\}$，其中 $\Pr(H)=\Pr(T)=1/2$

接下来定义一个指示器随机变量 $X_H=\mathrm{I}\{H\}$，对应于硬币正面朝上的事件 H:

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

指示器随机变量

□ **例：** 抛掷一枚硬币，求**正面朝上**的期望次数。

解： 则一次抛掷硬币正面朝上的期望次数，即指示器随机变量 $\mathrm{I}\{H\}$ 的期望值：

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

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

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，设事件 A 对应的指示器随机变量为 $X_A=\mathrm{I}\{A\}$，那么 $\mathrm{E}[X_A]=\Pr\{A\}$**

证明：由指示器随机变量的定义以及期望值的定义，有：

$$
\begin{aligned}
\mathrm{E}[X_A] &= \mathrm{E}[\mathrm{I}\{A\}] \\
&= 1\cdot\Pr\{A\} + 0\cdot\Pr\left\{\overline{A}\right\} \\
&= \Pr\{A\}
\end{aligned}
$$

其中，$\overline{A}$ 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?**

解：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 H，即：X_i =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朝上的总次数

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:

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

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?**

解： 由引理 5.1，每个指示器随机变量 X_i 的期望值为 1/2，则总和 X 的期望值为：

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

即：根据期望的线性性质，总和的期望值等于 n 个指示器随机变量期望值的总和

指示器随机变量

- **用指示器随机变量分析雇用问题的平均雇用费用**
  - ✓ 为了利用概率分析，假设应聘者以随机顺序出现
  - ✓ 设 X 是一个随机变量，表示雇用一个新办公助理的次数
  - ✓ 定义 n 个指示器随机变量 X_i，每个 X_i 与应聘者 i 的一次面试相对应，根据是否被雇用有：

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

以及 $X = X_1 + X_2 + \cdots + X_n = \sum_{i=1}^{n} X_i$

根据定理 5.1，有 $\mathrm{E}[X_i] = \Pr\{\text{应聘者}\, i\, \text{被雇用}\}$

应聘者 i 被雇用的概率是多少呢？是1/2吗？

指示器随机变量

- **应聘者 *i* 被雇用的概率：**

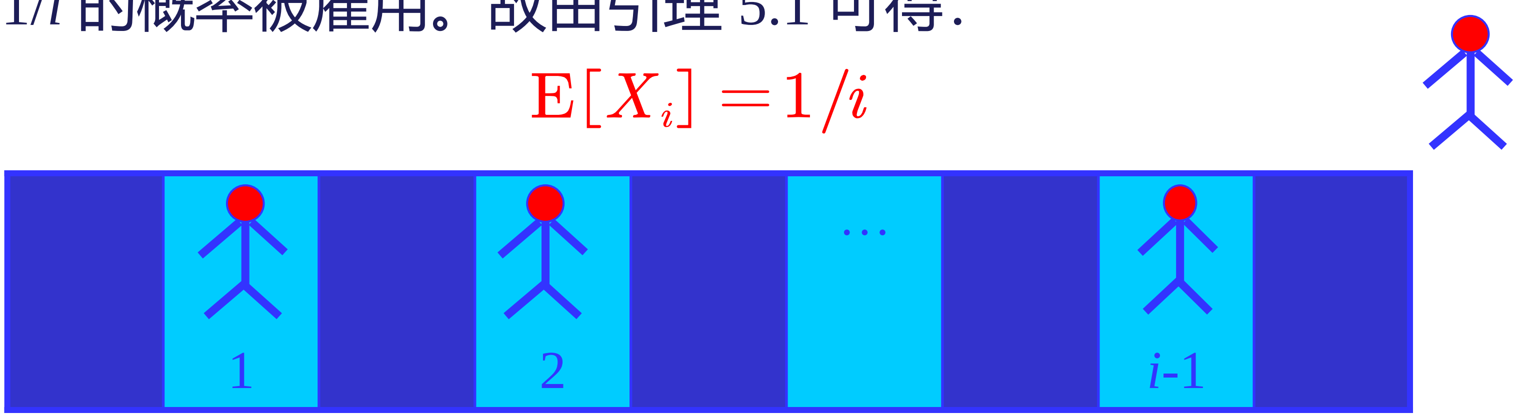
HIRE-ASSISTANT(n)	cost	times
1. *best* = 0 // candidate 0 is a least-qualified dummy candidate		
2. for i = 1 to n		
3. interview candidate i	c_i	n
4. if candidate i is better than candidate *best*		
5. *best* = i		
6. hire candidate i	c_h	m

➢ 在第 6 行中，若应聘者 i 被雇用，则需要应聘者 i 比前面 i-1 个应聘者都优秀。因此，我们需要求解出**应聘者 i 比前 i-1个应聘者更优秀的概率**

指示器随机变量

- **应聘者 *i* 比前 *i*-1个应聘者更优秀的概率?**
  - 因为应聘者以随机顺序出现，所以这 i 个应聘者也将以随机次序出现。因此，在这 i 个应聘者中，任何一个都等可能是目前最有优秀的
  - 所以，应聘者 i 比前 i-1 个应聘者更优秀的概率是 $1/i$，即它将有 $1/i$ 的概率被雇用。故由引理 5.1 可得：

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



指示器随机变量

- **应聘者 i 比前 i-1个应聘者更有资格的概率?**

可以出计算 $E[X]$:

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

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

指示器随机变量







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

证明：根据雇用费用的定义和等式(5.5)，可以直接得到这个界，说明雇用的人数期望值大约为 $\ln n$

可见，平均情况下的雇用费用 $c_h \ln n$ 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 概率分析与随机算法

内容提要：

- 雇用问题
- 指示器随机变量
- 随机算法

随机算法

- 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，从而阻碍了平均情况分析
- **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

- **随机算法和概率分析的不同点：**

概率分析算法是“确定”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，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

如，排名列表 A_1=<1,2,3,4,5,6,7,8,9,10>，新办公助理会雇用 10 次

排名列表 A_2=<10,9,8,7,6,5,4,3,2,1>，新办公助理只雇用 1 次

排名列表 A_3=<5,2,1,8,4,7,10,9,3,6>，新办公助理会雇用 3 次

随机算法

- **RANDOMIZED-HIRE-ASSISTANT是随机雇用算法，在算法运行前先随机地对应聘者进行排列：**
  - ✓ 先对应聘者进行重排列，然后确定最佳应聘者的随机算法。此时，让“随机”发生在**算法上**
  - ✓ 随机算法中，任何一个给定的输入，比如 A_3，都无法说出具体的雇用次数，因为在每次运行随机算法时雇用次数都不相同
  - ✓ 因此，对于同一个输入，每次运行随机算法时，每次输出的值（得到的代价）可能不一样，依赖于随机选择
  - ✓ 对于随机算法，没有特别的输入会引起它的最坏情况行为，因为随机化处理使得输入次序不再相关。只有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

随机算法

雇用问题的随机算法

- 对于雇用问题，伪代码中唯一需要改变的是**随机地变换应聘者的序列**：

```
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，假定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，那么雇用一个新办公助理的期望次数仍是 $\ln n$ 吗？

随机算法

- **引理 5.3 过程RANDOMIZED-HIRE-ASSISTANT的雇用费用期望仍然是 O($c_h$ln$n$)**

证明：对输入数组进行变换后，已经达到了和引理 5.2 相同的输入分布情况（随机顺序）

- **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**，求的是平均情况下的雇用费用
  - ✓ 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

随机算法

- **随机算法需要对给定的输入重新变换排列，使得输入随机化。**

  那么如何产生输入的一个均匀随机排列呢?

  - ✓ 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**

- **这里介绍两种随机化方法：**

方法一：随机排列给定数组。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：假设初始数组 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>

随机算法

随机排列策略的过程描述：

```
PERMUTE-BY-SORTING(A)
1  n =A. length
2  Let P[1..n] be a new array
3  for(i=1; i<=n; i++)
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，使用范围 $1\sim n^3$ 是为了让优先级数组 P 中所有优先级尽可能唯一
- 第 5 行排序时间为 $O(n\lg n)$
- 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“均匀随机”排列

随机算法

- **引理 5.4： 假设所有优先级都不同，则随机排列过程 PERMUTE-BY-SORTING 可以产生输入的一个均匀随机排列**

随机算法

- **这里介绍两种随机化方法：**

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] 中随机选取元素 A[RANDOM(i, n)] 进行交换

例如：

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)

随机算法

- **原址排列给定数组**
  - **RANDOMIZE-IN-PLACE过程如下：**

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

  - **RANDOMIZE-IN-PLACE 排序时间是 O(n)**
- **引理 5.5：原址排列过程 RANDOMIZE-IN-PLACE 也可以产生输入的一个均匀随机排列**

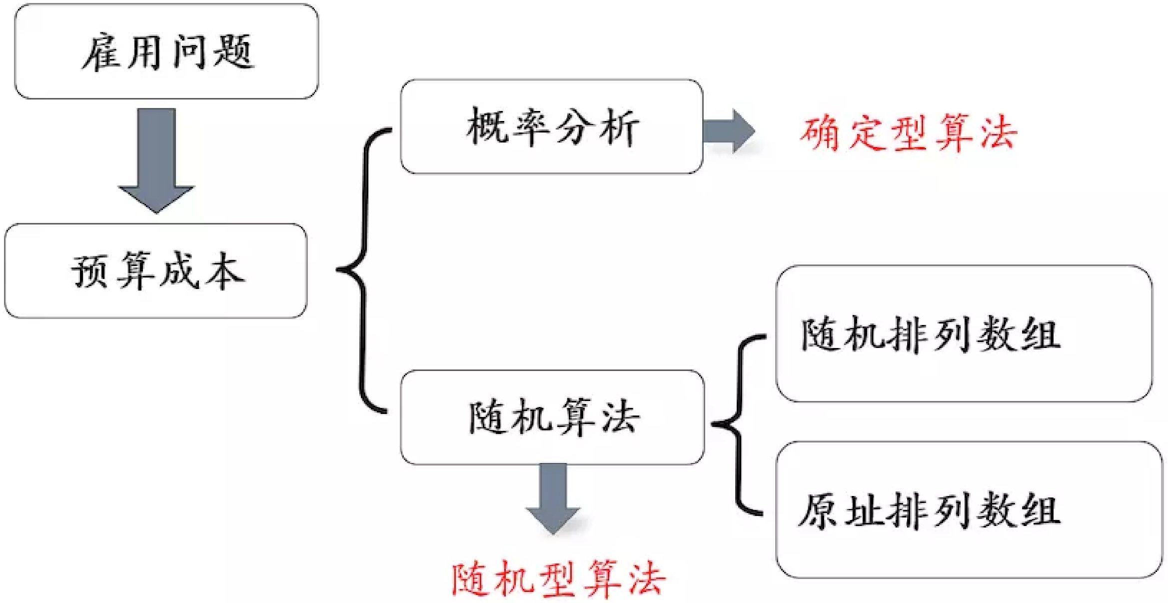
总结

- **概率分析 —— 确定型算法：**

  这个算法是随着输入的变化而变化，对于某个特定的输入，它总是会产生固定的结果

- **随机算法 —— 随机的算法**

  随机算法的随机发生在算法上，而不是发生在输入分布上。对输入进行随机再排列









谢谢！

Q & A

作业：5.2-1 5.2-4
5.3-3