Chapter 5

Probabilistic Analysis and Randomized Algorithms

吕志鹏

zhipeng.lv@hust.edu.cn

群名称：算法设计与分析

群　号：271069522

群名称：算法设计与分析2020

群　号：271069522

5.1 雇佣问题

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。

现在你希望能够估算一下这笔费用会是多少。

- 不失一般性，假设应聘办公助理的候选人编号为1到n。
- 并假设该过程在面试完应聘者i后，就能决定应聘者i是否是你目前见过的最佳人选。

雇佣算法可描述如下：

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

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?**

- **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?**

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。

- 应聘者编号：1~n
- 应聘者名次：rank(i)，每个应聘者有唯一一个名次，rank(i)表示应聘者i的名次，rank(i) $\in$[1,n]，不失一般性，一个较高的名次对应一个更好的应聘者：

任意有序序列<*rank*(1), *rank*(2),…,*rank*(n)>是序列<1,2,…,n>的一个排列。

- 所有应聘者存在全序关系，即任意两个应聘者可以比较rank值并决定哪一个更有资格。

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<rank(1), rank(2), …,rank(n)>是数字1到n的n!种排列表中的任一个。**
- 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。

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：

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）。

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，且每个整数以等概率出现。
- 同时每次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)：返回3，4，5，6，7

每个出现的概率为1/5；

期望运行时间:

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”。

指示器随机变量

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：

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

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：

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

则，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：

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

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。

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，~A表示S-A，即A的补。

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{第i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：

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

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。

由引理5.1，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：

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

用指示器随机变量分析雇用问题

- 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，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 $X = X_1 + X_2 + \cdots + X_n$

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？

▪ **应聘者i被雇用的概率：**

```
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
```

在第6行中，若应聘者i被雇用，则就要有应聘者i比前面i-1个应聘者都优秀。

- **应聘者i比前i-1个应聘者更有资格的概率是?**

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

- 因为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。则在这i个应聘者中，任意一个都可能是目前最有资格的。
- 所以，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5.1可得：

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

计算E[X]：

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

（P673，调和级数）

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$\ln n$。

5.3 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。

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)
randomly permute the list of candidates
best = 0                // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
        interview candidate i
        if candidate i is better than candidate best
                best = i
                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，那么雇用一个新办公助理的期望次数仍是ln n 吗？

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。■

关于算法的进一步讨论：

- 如果不考虑随机处理，则算法就是“确定”的：

　　雇用新办公助理的次数依赖于n个应聘者的排列。对于任何特定输入，如果不改变应聘者的排列，则雇用一个新办公助理的次数始终相同。

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。

- **RANDOMIZED-HIRE-ASSISTANT是随机算法**，在算法运行前先随机地排列应聘者：
  - ✓ 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，都无法说出最大值会被更新多少次的。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。
- 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。

随机算法需要对给定的输入变换排列，以使输入随机化。那么**如何产生输入的随机排列呢?**

- 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：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] = RANDOM(1, n^3)
5  sort A, using P as sort keys
```

- 第4行选取一个**在1~n^3之间的随机数**。使用范围1~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(nlogn)
- 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,...,n)，则对所有的E_i，整个事件发生的概率是：

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

- $Pr\{E_1\}$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:

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

最终有：

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

$$
= \left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\cdots\left(\frac{1}{2}\right)\left(\frac{1}{1}\right) = \frac{1}{n!}
$$

因此，获得该排列的概率是1/n!，是一个均匀随机排列。

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。

过程如下：

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

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

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。（自学）

5.4 概率分析和指示器随机变量的进一步使用(自学)

5.4.1 生日悖论

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%?

5.4.2 球和箱子

5.4.3 特征序列

5.4.4 在线雇佣问题