

第一部分：基础知识

INTRODUCTION TO

ALGORITHMS

THIRD EDITION

概率分析和随机算法

《算法导论》—— 第4讲

jiacaicui@163.com

内容提要

- 通过 “**雇佣问题**” 来学习算法分析与设计中的 “**概率**” 手段
 - 概率分析
 - 指示器随机变量
- 随机算法
 - 随机排列数组
- 问题选讲
 - 生日悖论
 - 球与箱子
 - 特征序列
 - 在线雇佣问题

雇佣问题

低价面试与高价雇佣



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  $i$ 
```

- 分析该雇佣策略所产生的费用。
 - 面试费用为 c_i ，雇佣费用为 c_h ，面试费用较低，雇佣费用较高。
 - 假设雇佣了 m 个人，则总费用为 $c_i n + c_h m$ 。
 - 最坏情况：应聘者质量按照出现的次序严格递增，总费用为 $c_i n + c_h n$ ；
 - 最好情况：第一个应聘者就是质量最好的，总费用为 $c_i n + c_h$ ；
 - 平均情况 / 一般情况下呢？

概率分析

- 概率分析：假定输入满足某种概率分布，藉此分析算法运行时间的数学期望，称为平均情况运行时间。
 - 通常假设输入的各种情况满足均匀分布，即所有可能的输入之间是等可能的。
- 算法第 4 行说明应聘者之间存在全序关系。
 - 用 $rank(i)$ 表示第 i 个应聘者的序（越大表示质量越高），不妨设有序序列 $\langle rank(1), rank(2), \dots, rank(n) \rangle$ 是 $\langle 1, 2, \dots, n \rangle$ 的一个排列。
 - 如果 $\langle rank(1), rank(2), \dots, rank(n) \rangle$ 的 $n!$ 种所有可能情况等概率出现，称其构成一个均匀随机排列。

随机算法

- 如果我们能有一份应聘者的名单，可以每天**随机**从名单上挑应聘者来面试，自己创造一个**均匀随机排列**的输入。
- 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**产生的数值决定，则称这是个**随机算法**。
 - 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**。
- 确定性算法（输入是随机的） .vs 随机算法
 - 确定性算法：**平均情况运行时间**
 - 随机算法：**期望运行时间**
 - 同一个输入同一个算法运行两次可能会花费不同的时间。

关于“费用”和“时间”

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  $i$ 
```

- 无论是分析“费用”，还是分析运行“时间”，本质上都是在分析关键步骤的执行次数。
 - 分析“费用”，我们选取第6行作为关键步骤，因为那是“变数”且是主要费用。

指示器随机变量

期望计算神器



指示器随机变量

- 事件 A 的指示器随机变量 (indicator random variable) :

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

- 引理5.1 : $E(I\{A\}) = \Pr\{A\}$ 。

- 证明 : $E[I\{A\}] = 1 \cdot \Pr\{A\} + 0 \cdot \Pr\{\bar{A}\} = \Pr\{A\}$ 。

- 例 : 求抛 n 次硬币正面朝上次数的期望。

- 记第 i 次抛出时正面朝上为事件 H_i , $X_i = I\{H_i\}$, X 为总次数, 则

$$E[X] = E\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n \Pr\{H_i\} = \frac{n}{2}$$

分析雇佣费用

目标：计算 m 的期望值 —— 即期望的雇佣次数。

- 假设应聘者以随机顺序出现，用随机变量 X 表示雇佣的次数，直接用定义有

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

计算会很麻烦。

- 使用指示器随机变量，记应聘者 i 被雇佣为事件 H_i ， $X_i = I\{H_i\}$ ，不难看出

$$X = X_1 + X_2 + \cdots + X_n。$$

- 应聘者 i 被雇佣的概率就是 $\text{rank}(i)$ 为 $\text{rank}(1) \sim \text{rank}(i)$ 中最大值的概率，

易有 $\Pr\{H_i\} = \frac{1}{i}$ ，于是

$$E[X] = E\left[\sum_{x=1}^n X_i\right] = \sum_{x=1}^n E[X_i] = \sum_{x=1}^n \Pr\{H_i\} = \sum_{x=1}^n \frac{1}{i} = \ln n + O(1)$$

分析雇佣费用

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

- 尽管我们面试了 n 个人，但平均起来，实际上大约只雇佣其中的 $\ln n$ 个人。
- 平均情况下的雇佣费用比最坏情况下的雇佣费用 $O(c_h n)$ 有了很大的改进。

随机算法

故布疑阵的妙用



从确定性算法的概率分析到随机算法

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  $i$ 
```

- 确定性算法的概率分析：让“随机”发生在输入上；
 - 假如应聘者密谋以质量严格递增的次序参与招聘怎么办？
- 随机算法：让“随机”发生在算法上。
 - 算法第一步就是随机打乱，自己创造均匀分布的条件。

从确定性算法的概率分析到随机算法

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

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
```

怎么做到的？

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

随机排列数组

输入：一个长度为 n 的数组 A 。

输出： A 的一个均匀随机排列。

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
```

约定： $\text{RANDOM}(a, b)$ 是一个随机数生成器，返回整数 $a \sim b$ 之间的一个随机整数。例如 $\text{RANDOM}(0, 1)$ 以 $\frac{1}{2}$ 的概率返回 0 或者 1。

随机排列数组

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
```

- 算法第4行选取一个 $1 \sim n^3$ 之间的随机数，使用这个范围是为了让 P 的中所有优先级尽量唯一，概率至少是 $1 - \frac{1}{n}$ （见练习4-2/问题3）。
- 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。

引理5.4 证明

证明：考虑 $A[1..n]$ 的任意一个确定排列 $B[1..n]$ ，定义 $A[1..n]$ 中的元素 $B[i]$ 分配到第 i 小的优先级为事件 E_i ，则我们只需要证明

$$\Pr\{E_1 E_2 \dots E_{n-1} E_n\} = \frac{1}{n!} \Leftrightarrow \prod_{i=1}^n \Pr\{E_i | E_1 E_2 \dots E_{i-1}\} = \frac{1}{n!}$$

这里应用了乘法公式，记 $\Pr\{E_0\} = 1$ 。 $\Pr\{E_i | E_1 E_2 \dots E_{i-1}\}$ 说的其实就是 $B[i]$ 是 $B[i..n]$ 中优先级最小的元素的概率，从而 $\Pr\{E_i | E_1 E_2 \dots E_{i-1}\} = \frac{1}{n-i+1}$ 。

则：

$$\Pr\{E_1 E_2 \dots E_{n-1} E_n\} = \prod_{i=1}^n \Pr\{E_i | E_1 E_2 \dots E_{i-1}\} = \prod_{i=1}^n \frac{1}{n+1-i} = \frac{1}{n!}$$

于是，我们说明了获任何给定排列的概率都是 $\frac{1}{n!}$ ，即产出了一个均匀随机排列。

更好的打乱办法：原址排列给定数组

RANDOMIZE-IN-PLACE(A)

1 $n = A.length$

2 **for** $i = 1$ **to** n

3 swap $A[i]$ with $A[\text{RANDOM}(i, n)]$

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：在第 2 – 3 行 for 循环的每次迭代开始前，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 $\frac{(n-i+1)!}{n!}$ 。

- 初始化： $i = 1$ ， $A[1..0]$ 包含 0 排列的概率是 1，显然成立。
- 保持：（见下一页PPT）。
- 终止：最后一次循环结束后， $i = n + 1$ ，根据循环不变式，对每个可能的 n 排列，数组 $A[1..n]$ 包含这个 n 排列的概率是 $\frac{1}{n!}$ ，是均匀随机排列。

引理5.5证明——“保持”

RANDOMIZE-IN-PLACE(A)

1 $n = A.length$

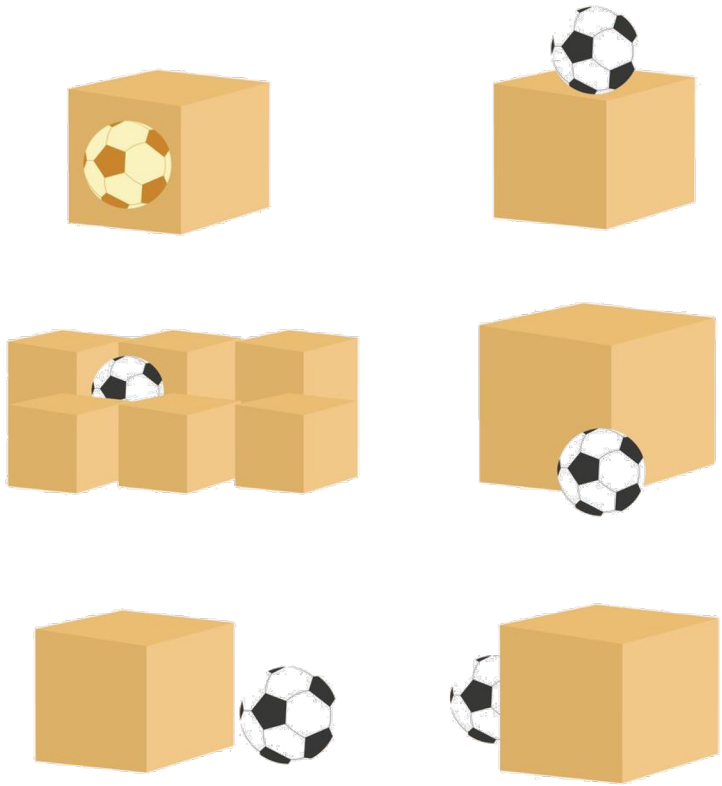
2 **for** $i = 1$ **to** n

3 swap $A[i]$ with $A[\text{RANDOM}(i, n)]$

- 保持：第 i 次迭代时，考虑任意一个特殊的 i 排列 $\langle x_1, x_2, \dots, x_i \rangle$ ，记 $A[1..i-1]$ 包含 $\langle x_1, x_2, \dots, x_{i-1} \rangle$ 这个 $(i-1)$ 排列为事件 E_1 ，记在 $A[i]$ 位置放置 x_i 为事件 E_2 ，下面证明 $\Pr\{E_1 E_2\} = \frac{(n-i)!}{n!}$ 。
- 假设第 i 次迭代前不变式成立，则 $\Pr\{E_1\} = \frac{(n-i+1)!}{n!}$ ；当 E_1 发生后，算法第 3 行从 $A[i..n]$ 中随机挑选了一个放在 $A[i]$ 处，这种情况下 $A[i] = x_i$ 的概率，也就是 $\Pr\{E_2|E_1\} = \frac{1}{n-i+1}$ 。
- 于是 $\Pr\{E_1 E_2\} = \Pr\{E_1\} \cdot \Pr\{E_2|E_1\} = \frac{(n-i+1)!}{n!} \cdot \frac{1}{n-i+1} = \frac{(n-i)!}{n!}$ 。

问题选讲

举一反三



生日悖论

- 一个屋子里人数必须要达到多少人，才能期望其中有两人生日相同？

- 不考虑闰年的情况，假设一年有 n 天，屋子里有 k 个人，记第 i 个人和第 j 个人生日相同为事件 H_{ij} ，定义指示器随机变量 $X_{ij} = I\{H_{ij}\}$ 。

- 记第 i 个人的生日为 b_i ，不难发现， $\Pr\{b_i = r\} = \frac{1}{n}$ 。

- 不妨假设任意两个人的生日之间是独立的，则 $\Pr\{b_i = r, b_j = r\} = \Pr\{b_i = r\} \cdot \Pr\{b_j = r\} = \frac{1}{n^2}$ 。

- 于是：

$$E[X_{ij}] = \Pr\{H_{ij}\} = \Pr\{b_i = b_j\} = \sum_{r=1}^n \Pr\{b_i = r, b_j = r\} = \sum_{r=1}^n \frac{1}{n^2} = \frac{1}{n}$$

生日悖论

设有 X 对人生日相同，则

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

于是

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

令 $E[X] \geq 1$ ，解得 $k \geq \sqrt{2n} + 1 = \Theta(\sqrt{n})$ 。

- 生日悖论：一个屋子里人数不必达到很多人，我们就能期望其中有两人生日相同。具体地，只需 $\Theta(\sqrt{n})$ 人， n 为一年的天数。
 - 在地球上，将 $n = 365$ 代入，有 $k \geq 28$ 。如果房间里面至少有 28 人，我们可以期望至少一对人生日相同。
 - 在火星上，一年有 $n = 669$ 个火星日，我们至少需要 38 个火星星人。

球与箱子

- 把相同的球不断地投到 b 个盒子里，球会随机落到其中一个盒子中，在我们期望每个箱子里都有一个球之前，至少需要投多少次球？
 - 假设我们需要投 n 次球，将其分为几个阶段，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；其中，“命中”指的是球投入一个空盒子中。
 - 第 i 个阶段，有 $b-i+1$ 个非空盒子，投一次的命中概率为 $\frac{b-i+1}{b}$ 。
 - 设 n_i 表示第 i 阶段的投球次数，则 $n_i \sim g\left(\frac{b-i+1}{b}\right)$ ，于是

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

(几何分布详见附录C.2)

球与箱子

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

- 在我们期望每个箱子都有一个球之前，大约要投 $b \ln b$ 次。
- 礼券收集者问题：收集齐 b 种不同礼券中的每一种，大约需要 $b \ln b$ 张随机得到的礼券才能成功。
 - 假设小浣熊干脆面中的 108 种水浒卡是均匀随机分布的，收集齐所有的武将大约需要购买 $108 \ln 108 \approx 506$ 包干脆面。

特征序列

- 抛一枚标准的硬币 n 次，最长连续正面的序列的期望长度有多长？

- 记从第 i 次抛硬币开始，连续至少 k 次正面朝上为事件 A_{ik} ，由于每次抛硬币都是独立的，不难得出 $\Pr\{A_{ik}\} = \frac{1}{2^k}$ 。

- 考虑 $X_{ik} = I\{A_{ik}\}$ ，用 X 表示长度为 k 的特征序列的数目，有

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

- 令 $E[X] = c \geq 1$ ，有 $n+1 = k + c2^k$ ，即 $\Theta(n) = \Theta(2^k)$ ，不难看出 $k = \Theta(\log n)$ ，使用代入法即可严谨证明。

- 最长特征序列的长度期望为 $\Theta(\log n)$ 。

在线雇佣问题

- 动机：**避免频繁的新人换旧人。**
 - 在**最小化雇佣次数**和**最大化所雇佣应聘者质量**之间取得平衡。
 - 限制条件：每次面试后必须立刻拒绝或者雇佣。
 - 策略：面试前 k 个，记下最高分，并拒绝；如果后 $n - k$ 个中有更好的，就选更好的那个，否则选最后一个。

ON-LINE-MAXIMUM(k, n)

```
1  bestscore =  $-\infty$ 
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  $n$ 
```

在线雇佣问题

ON-LINE-MAXIMUM(k, n)

```
1  bestscore =  $-\infty$ 
2  for  $i = 1$  to  $k$ 
3      if  $\text{score}(i) > \text{bestscore}$ 
4           $\text{bestscore} = \text{score}(i)$ 
5  for  $i = k + 1$  to  $n$ 
6      if  $\text{score}(i) > \text{bestscore}$ 
7          return  $i$ 
8  return  $n$ 
```

- 选择一个怎样的 k 能够让 ON-LINE-MAXIMUM(k, n) 以最大的概率取最好的应聘者？

- 记成功面试到最好的应聘者的事件为 S ，最好的应聘者是第 i 个面试者时成功为事件 S_i ，则显然有

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

在线雇佣问题

ON-LINE-MAXIMUM(k, n)

```

1  bestscore =  $-\infty$ 
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  $n$ 

```

- 当最好的应聘者是前 k 个时，一定不会成功，所以 $1 \leq i \leq k$ 时， $\Pr\{S_i\} = 0$ 。
- $i \geq k + 1$ 时， S_i 发生需要两个条件：
 - 最好的应聘者在位置 i 上，记为事件 B_i ，显然 $\Pr\{B_i\} = \frac{1}{n}$ 。
 - $k + 1 \sim i - 1$ 没有任何应聘者入选，记为事件 O_i ，这意味着 $1 \sim i - 1$ 中的最优秀的人只能在前 k 个，于是 $\Pr\{O_i\} = \frac{k}{i-1}$ 。
 - B_i 与 O_i 是独立的， $1 \sim i - 1$ 如何排列并不影响 i 是否比 $1 \sim i - 1$ 都优秀。

在线雇佣问题

ON-LINE-MAXIMUM(k, n)

```

1  bestscore =  $-\infty$ 
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  $n$ 

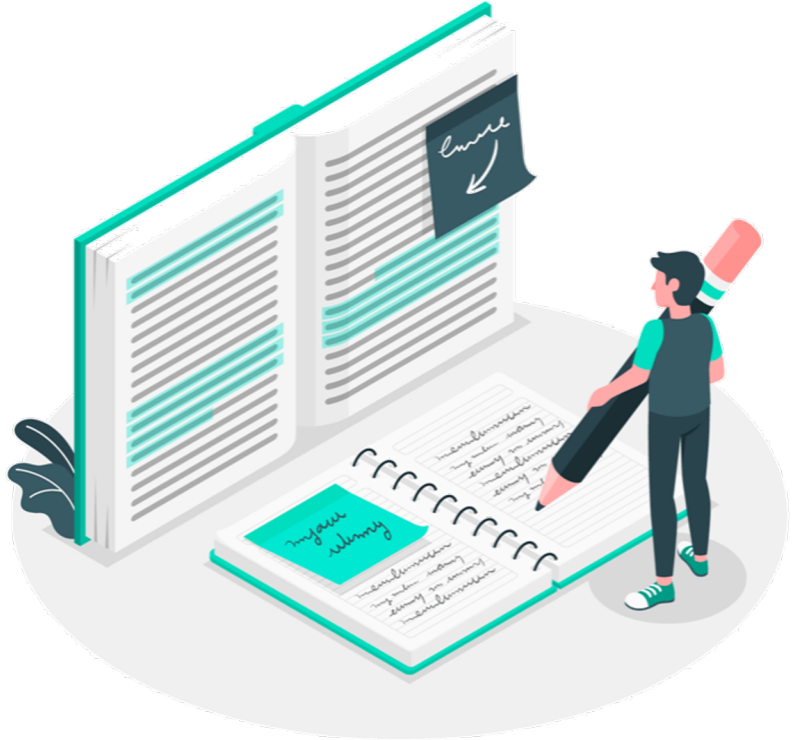
```

• 综上：

$$\begin{aligned}
 \Pr\{S\} &= \sum_{i=1}^n \Pr\{S_i\} = 0 + \sum_{i=k+1}^n \Pr\{S_i\} = \sum_{i=k+1}^n \Pr\{B_i O_i\} \\
 &= \sum_{i=k+1}^n \Pr\{B_i\} \Pr\{O_i\} = \sum_{i=k+1}^n \frac{1}{n} \cdot \frac{k}{i-1} = \frac{k}{n} \cdot \sum_{i=k}^{n-1} \frac{1}{i} \\
 &\geq \frac{k}{n} \int_k^n \frac{1}{x} dx = \frac{k}{n} (\ln n - \ln k) = \frac{k}{n} \ln \frac{n}{k}
 \end{aligned}$$

• 又 $f(x) = \frac{\ln x}{x}$ 在 $x = e$ 时取最大值，所以 $\frac{n}{k} = e$ 时， $\Pr\{S\}$ 的下界 $f\left(\frac{n}{k}\right)$ 最大化为 $\frac{1}{e}$ 。因此，取 $k = \frac{n}{e}$ 时，至少有 $\frac{1}{e}$ 的概率雇佣到最好的应聘者。

本讲小结



内容提要

- 通过 “**雇佣问题**” 来学习算法分析与设计中的 “**概率**” 手段
 - 概率分析
 - 指示器随机变量
- 随机算法
 - 随机排列数组
- 问题选讲
 - 生日悖论
 - 球与箱子
 - 特征序列
 - 在线雇佣问题

The End!

