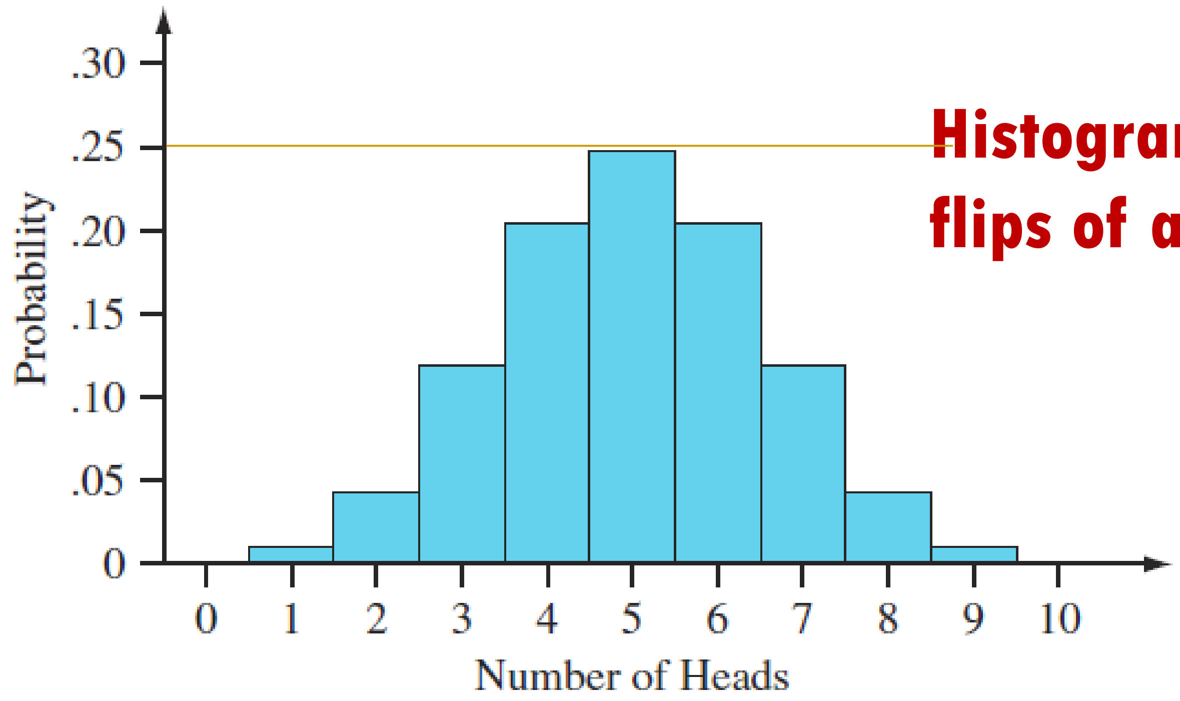
计算机问题求解 – 论题2-8 - 概率分析与随机算法

2016年04月14日

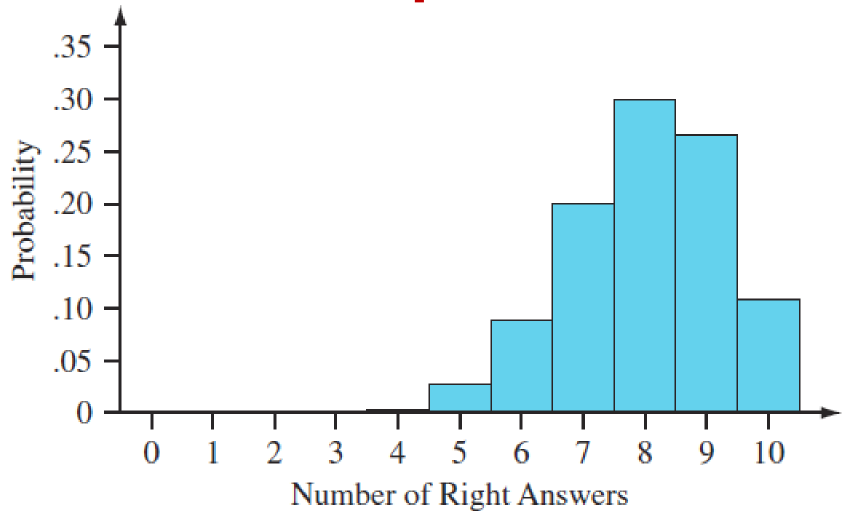


Histogram for 10 flips of a coin

问题1:

你能否结合这个图解释与随机变量相关的以下的概念: 期望值、分布以及它们之间的关系?

Histogram for answers for a 10-problem test



问题2：图中纵坐标值是怎么得出来的？

贝努利试验过程：

$$P(\text{exactly } k \text{ successes}) = \binom{n}{k} p^k (1-p)^{n-k}$$

当k=8, $\binom{10}{8} \cdot 0.8^8 \cdot 0.2^2 \approx 0.302$

```
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 $i$
```

那么，有没有问：
一般情况下，代价
会如何？

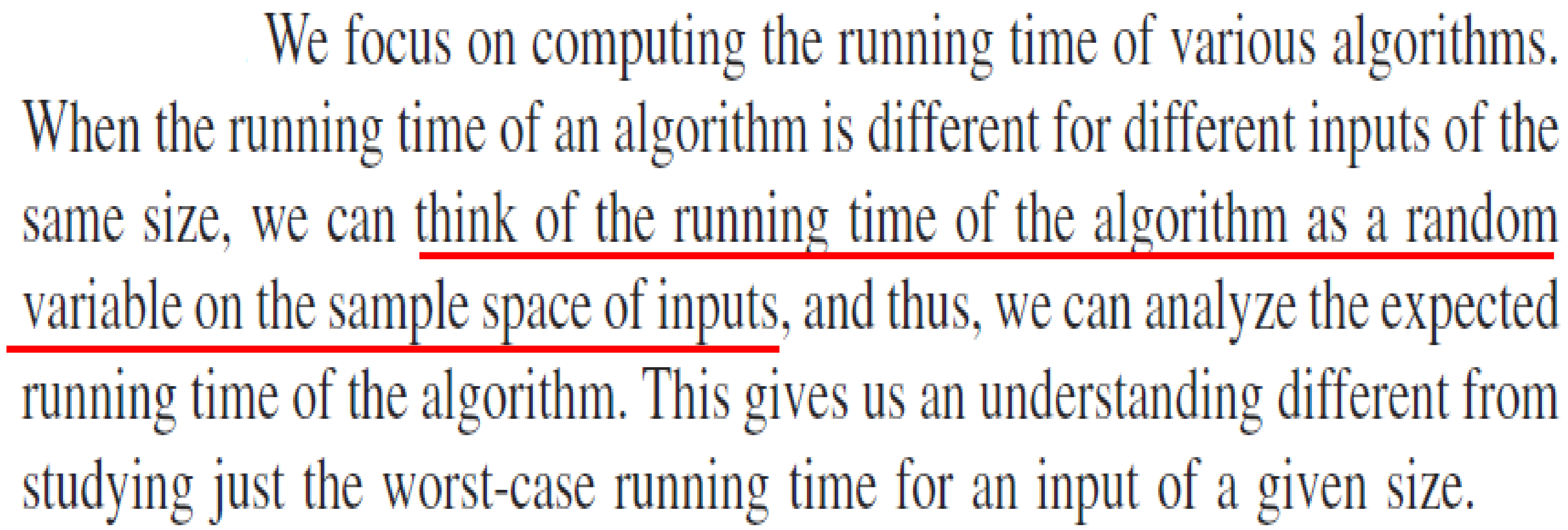
问题3：

这个算法是“确定”的吗？

什么是“随机”的呢？

它的best和worst case是什么？

Average-Case Analysis of Algorithms


We focus on computing the running time of various algorithms. When the running time of an algorithm is different for different inputs of the same size, we can think of the running time of the algorithm as a random variable on the sample space of inputs, and thus, we can analyze the expected running time of the algorithm. This gives us an understanding different from studying just the worst-case running time for an input of a given size.

Average case Analysis of an Algorithm

In order to perform a probabilistic analysis, we must use knowledge of, or make assumptions about, the distribution of the inputs. Then we analyze our algorithm, computing an average-case running time, where we take the average over the distribution of the possible inputs. Thus we are, in effect, averaging the running time over all possible inputs. When reporting such a running time, we will refer to it as the ***average-case running time***.

```
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 $i$
```

问题4：

在hiring problem算法中假设“应聘者随机到达”意味着什么？

```
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 $i$
```

问题5：

在hiring problem算法中分析算法所需要的随机变量是什么？

Hiring-Assistant算法的平均情况分析

涉及的随机变量：

- Hiring操作执行次数：*X*；

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

- 事件“第*i*个候选人被雇用”的indicator：X_i；

$$\boxed{X = X_1 + X_2 + \cdots + X_n}$$

Indicator Random Variable

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

Indicator random variables provide a convenient method for converting between probabilities and expectations.

问题6：

这句话是什么意思？

Hiring-Assistant算法的平均情况分析

涉及的随机变量：

- Hiring操作执行次数：X；
- 事件“第i个候选人被雇用”的indicator：X_i；

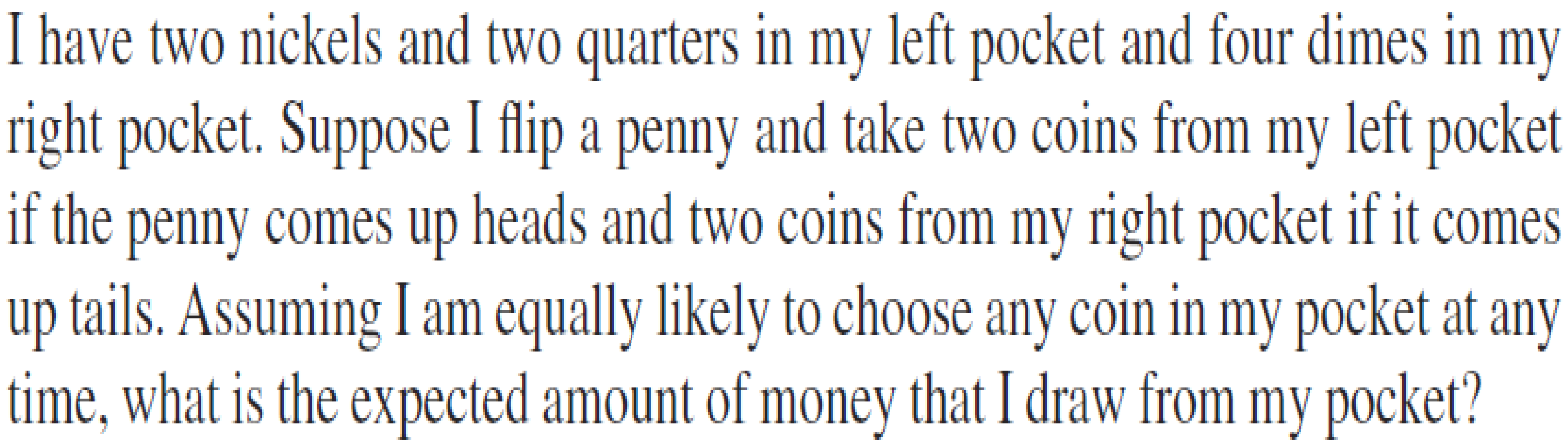
$$\boxed{X = X_1 + X_2 + \cdots + X_n}$$

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] && \text{(by equation (5.2))} \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] && \text{(by linearity of expectation)} \\
&= \sum_{i=1}^{n} 1/i && \text{(by equation (5.3))} \\
&= \ln n + O(1) && \text{(by equation (A.7))} \;.
\end{aligned}$$

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

为什么？

让随机“更随机”



I have two nickels and two quarters in my left pocket and four dimes in my right pocket. Suppose I flip a penny and take two coins from my left pocket if the penny comes up heads and two coins from my right pocket if it comes up tails. Assuming I am equally likely to choose any coin in my pocket at any time, what is the expected amount of money that I draw from my pocket?

条件期望值

- 将penny也纳入： HNQ, HQN, HQQ, HNN, and TDD

$$30\left(\frac{1}{6}\right)+30\left(\frac{1}{6}\right)+50\left(\frac{1}{12}\right)+10\left(\frac{1}{12}\right)+20\left(\frac{1}{2}\right)=25$$

- 引入条件期望值： $E(X|F)=\sum_{i=1}^{k} x_i P((X=x_i)|F)$

- *X*的期望值： $E(X)=\sum_{i=1}^{n} E(X|F_i)P(F_i)$

这里*n*=2； *P*(F_1)=*P*(F_2)=0.5

∴ *E*(*X*) = (30+20)/2 = 25

  - 若*F*= “Head” ： $E(X\mid F)=30\left(\frac{1}{3}\right)+30\left(\frac{1}{3}\right)+50\left(\frac{1}{6}\right)+10\left(\frac{1}{6}\right)=30$
  - 若*F*= “Tail” ： $E(X\mid F)=20(1)=20$

基于概率的算法分析与随机算法

- 输入数据的概率模型
  - outcomes，sample space，probability
  - probability variant
  - 通过分布、期望来进行平均时间开销分析
- 难以给出概率模型时：
  - “制造”一个随机分布，进而利用这个随机分布进行概率分析

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 $i$
```

相当于用抛硬币或者掷色子的方式决定下一步该干什么！

算法的执行不再依赖于输入数据，关键取决于一个“随机数”

这个随机现象，我们可以清楚认识它的概率模型

随机算法

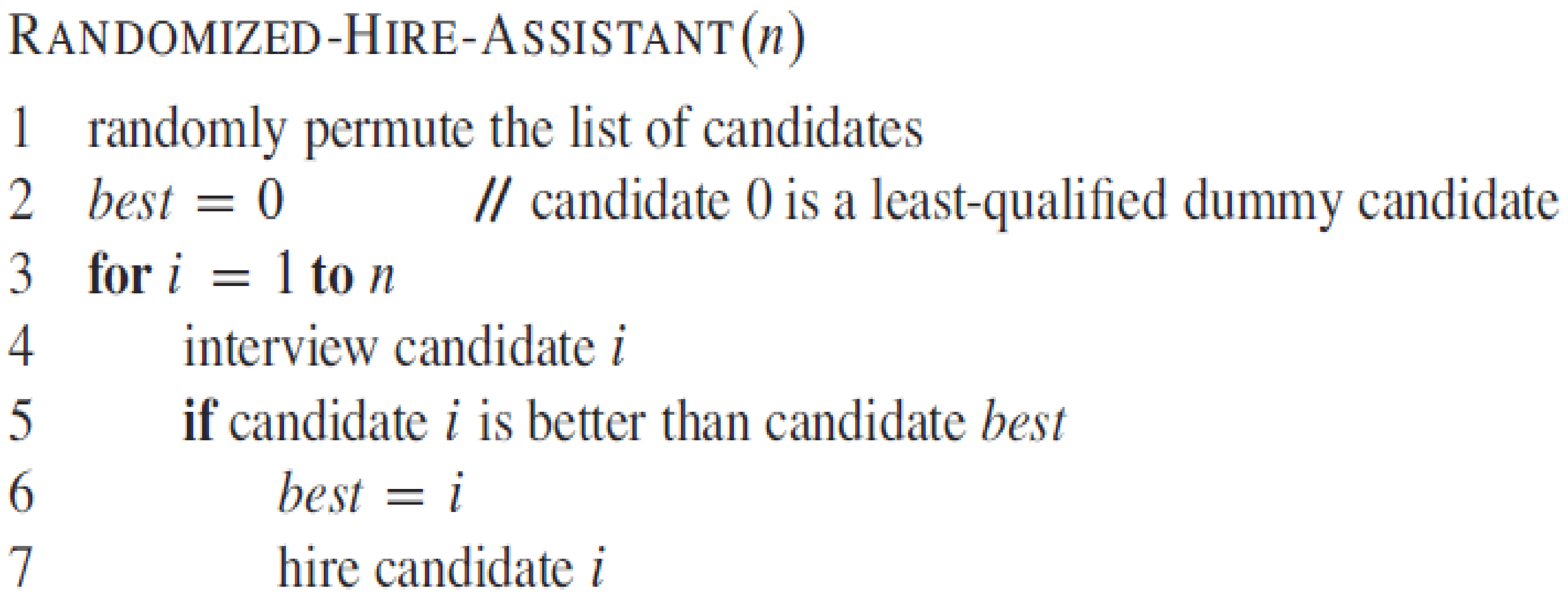
```
RANDOMIZED-HIRE-ASSISTANT(n)
1  randomly permute the list of candidates
2  best = 0          // candidate 0 is a least-qualified dummy candidate
3  for i = 1 to n
4      interview candidate i
5      if candidate i is better than candidate best
6          best = i
7          hire candidate i
```

问题7:

这个算法还有“最好/坏/平均情况”吗？

该如何分析这个算法的性能呢？

随机算法



```
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 $i$
```

问题8:
什么叫：RANDOMLY PERMUTE?

生成序列的“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
```

$O(n\log n)$

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

$O(n)$

对两个算法都必须回答同样的问题：

是否得到任意可能的排列的机会是一样的(uniformly distribution)?

前提是p[i]唯一

什么是：得到任意可能的排列的机会是一样的

对n个元素进行排列，其样本空间是：nN!

算法总是可以得到一个排列T：P(i)各不相同

如果存在一个T，算法得到T的概率大于1/n!，NO！

任意给定T，算法得到T的概率等于1/n!，YES!

试试看：

对一个特别的排列，证明其概率是1/*n*!：A中第*i*个元素“权”恰好是第*i*个最小。

用E_i表示对特定的*i*上述条件成立的事件，则要求的排列生成的概率是：

$$\Pr\{E_1 \cap E_2 \cap E_3 \cap \cdots \cap E_{n-1} \cap E_n\}$$

按照条件概率的定义式，并利用归纳法加以推广，上面的式子可以写为：

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

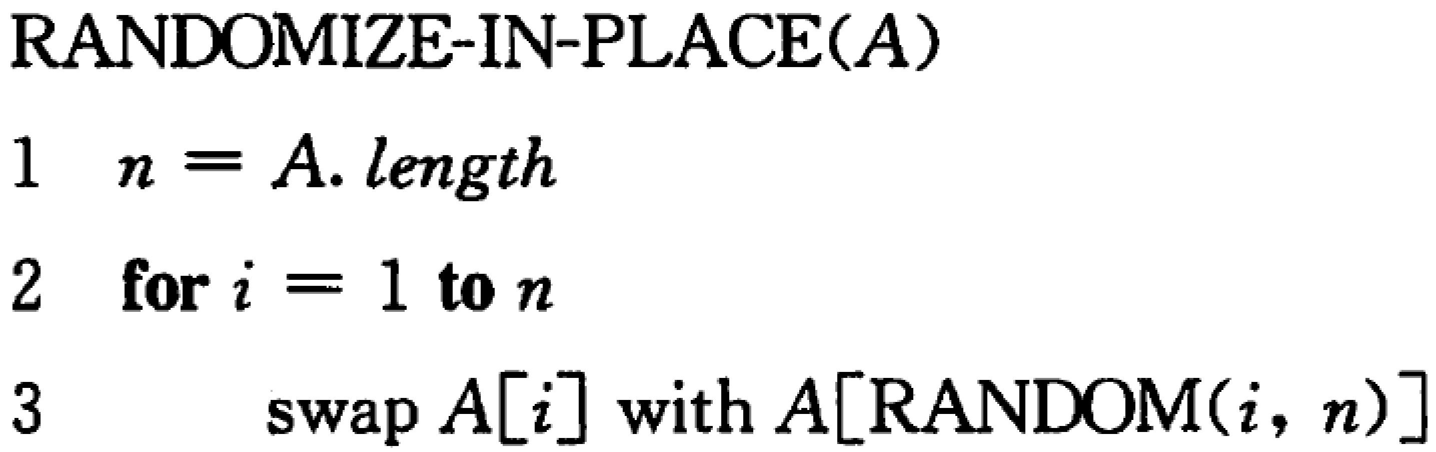
于是：

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

只要对上述的“最小”加以不同的解释，这个证明适用于任意排列！

在by-sorting算法中依次将*n*个随机生成的“权”赋给各元素。

如何证明InPlace算法也能产生URP?



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

循环不变量

利用循环不变式的归纳

针对 In-place 算法 可定义如下的不变式：

Just prior to the ith iteration of the **for** loop of lines 2–3, for each possible $(i-1)$-permutation of the n elements, the subarray $A[1..i-1]$ contains this $(i-1)$-permutation with probability $(n-i+1)!/n!$.

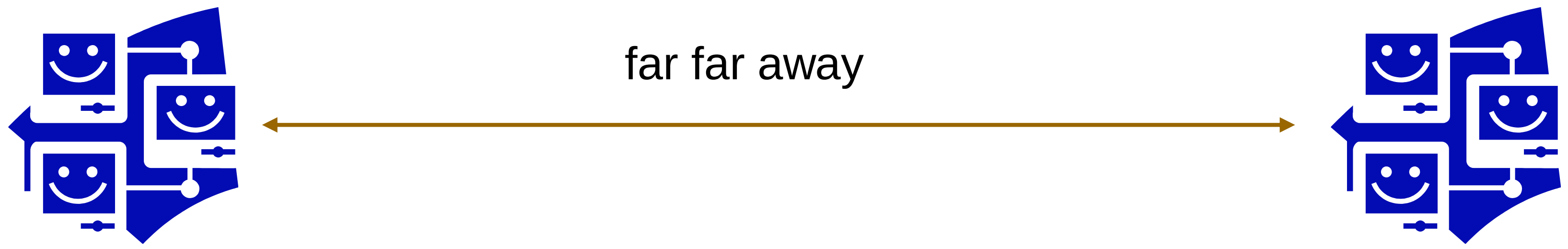
考虑一个特定排列：$\langle x_1, x_2, \ldots, x_i \rangle$
如果第 i 次循环开始前，前i -1个元素恰好是$\langle x_1,x_2,\ldots x_{i-1}\rangle$（事件$E_1$)，定义事件$E_2$：第$i$次循环恰好将$x_i$放到了第$i$个位置。则生成上述排列的概率是：

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

放在第i个位置的元素是在$[i,n]$中选的

从n个元素中任选i个排列的种数的倒数。

一个关于网络通信的例子



Computer A

Computer B

We want to verify the consistency between the two databases of size *n* (*n*=10^{16}, say) located on *A* and *B*.

For a deterministic answer, we may have to transfer a message of at least *n* bits, and (!) without an error on the way. It doesn't look a pleasant task.

用随机的方法解决上述问题

- 将逐位比较改为比较两个整数x 和 y的值。
- 并不直接比较它们的值，而是采用以下方法：
  - 计算 $s = x \bmod p$, (p 是一个质数)
  - 计算 $q = y \bmod p$
  - 通过比较 q 和 s 来判断 x 是否等于y 。
- p 是在 $[2, n^2]$ 区间随机选择的质数。

协议描述

Phase 1: R_I chooses a prime p from PRIM(n^2) at random. Every prime from PRIM(n^2) has the same probability $1/\text{Prim}(n^2)$ to be chosen.

Phase 2: R_I computes the integer

$$s = \text{Number}(x) \bmod p$$

(i.e., the remainder of the division Number(x) : p) and sends the binary representations of

$$s \text{ and } p$$

to R_{II}.

Phase 3: After reading s and p, R_{II} computes the number

$$q = \text{Number}(y) \bmod p.$$

If $q \neq s$, then R_{II} outputs "unequal".
If $q = s$, then R_{II} outputs "equal".

问题8：

你能看出这样做的好处吗？

关于通信量

原来最坏情况下要传输 *n* 位信息。

现在只需要传输两个不大于n^2的信息。每个信息的位数为： $\lceil \log_2 n^2 \rceil \leq 2 \cdot \lceil \log_2 n \rceil$

Let us see what that means for $n = 10^{16}$. As already mentioned, the best deterministic protocol cannot avoid the necessity of communicating at least

$$10^{16} \text{ bits}$$

for some inputs. Our protocol WITNESS always works within

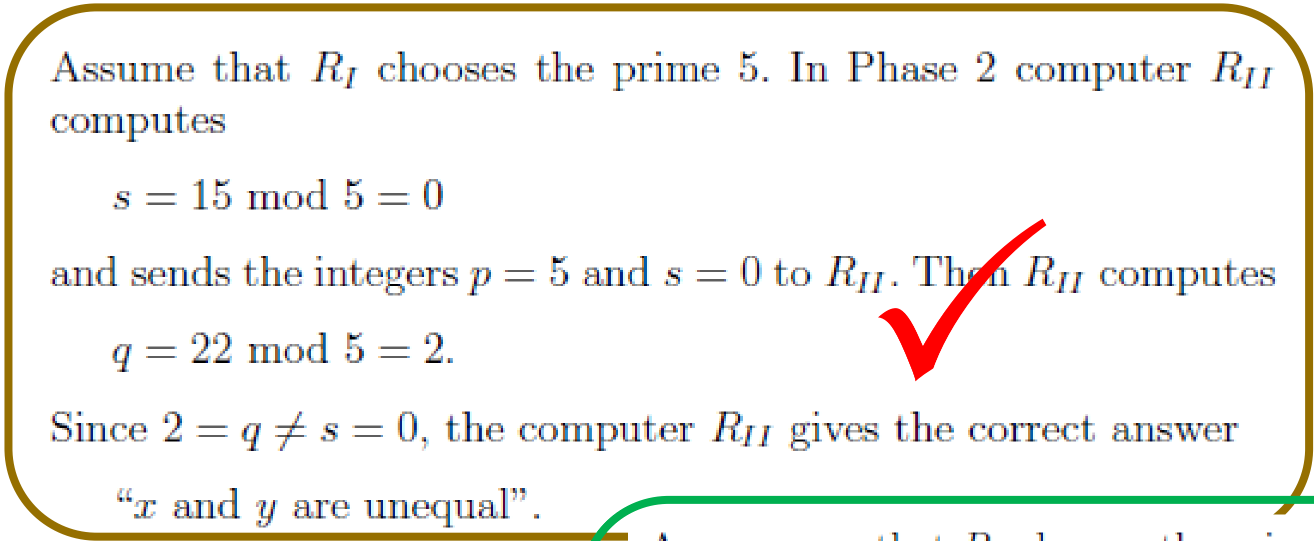
$$4 \cdot \lceil \log_2(10^{16}) \rceil \leq 4 \cdot 16 \cdot \lceil \log_2 10 \rceil = 256 \text{ communication bits.}$$

问题9：

你觉得这个结果可靠吗？

假设 $x = 01111_2 = 15$; $y = 10110_2 = 22$; $n=5$

PRIM(25) ={2, 3, 5, 7, 11, 13, 17, 19, 23};



Assume that R_I chooses the prime 5. In Phase 2 computer R_{II} computes

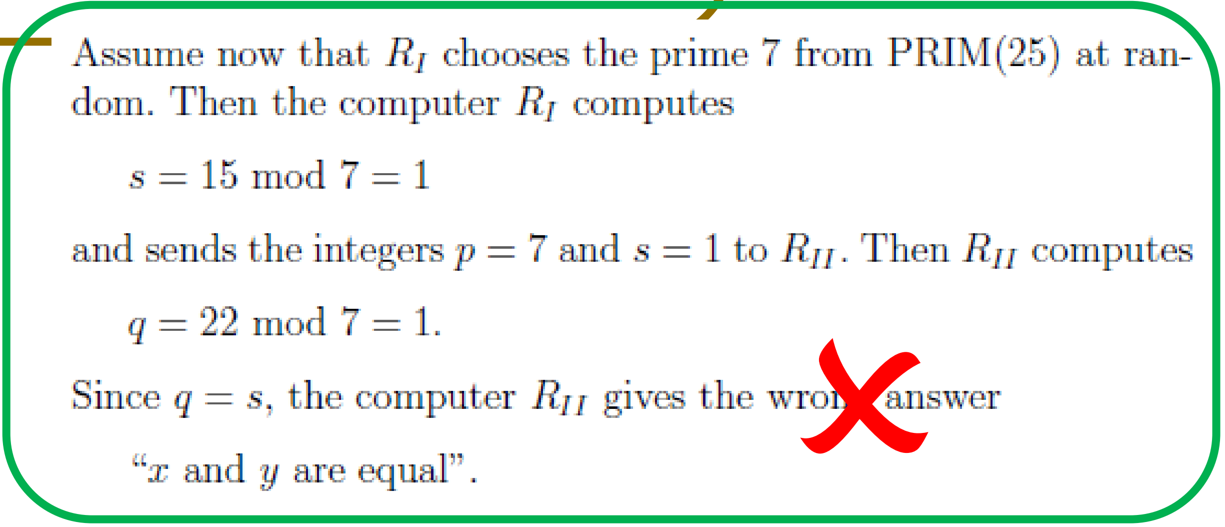
$$s = 15 \bmod 5 = 0$$

and sends the integers $p = 5$ and $s = 0$ to R_{II}. Then R_{II} computes ✓

$$q = 22 \bmod 5 = 2.$$

Since $2 = q \neq s = 0$, the computer R_{II} gives the correct answer

"x and y are unequal".



Assume now that R_I chooses the prime 7 from PRIM(25) at random. Then the computer R_I computes

$$s = 15 \bmod 7 = 1$$

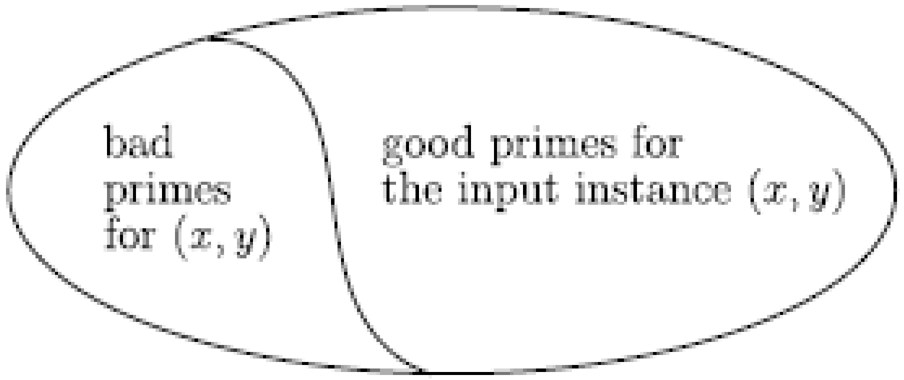
and sends the integers $p = 7$ and $s = 1$ to R_{II}. Then R_{II} computes

$$q = 22 \bmod 7 = 1.$$

Since $q = s$, the computer R_{II} gives the wrong answer ✗

"x and y are equal".

出错的概率有多大?



$$\mathrm{Error}_{WITNESS}(x, y) = \frac{\text{the number of bad primes for } (x, y)}{\mathrm{Prim}(n^2)}$$

关键是什么：

这个概率有多小？

两个关于质数的结论

对任意m>67，$\mathrm{Prim}(m) > \frac{m}{\ln m}$ ，因此对任意*n*≥9，$\mathrm{Prim}(n^2) > \frac{n^2}{2\ln n}$

for every problem instance (x,y), the number of bad primes for (x,y) is at most $n-1$,

注意：如果*x*=*y*, 没有bad prime。

出错率大致的概念

$$\begin{aligned}\text{Error}_{WITNESS}(x,y) &= \frac{\text{the number of bad primes for } (x,y)}{\text{Prim}(n^2)} \\ &\le \frac{n-1}{n^2/\ln n^2} \\ &\le \frac{2\ln n}{n}.\end{aligned}$$

Hence, the error probability of WITNESS for problem instances (x, y) with $x \neq y$ is at most $2 \ln / n$, which is for $n = 10^{16}$ at most

$$\frac{0.36841}{10^{14}}.$$

问题10:

如果x,y确实相等，则没有“bad prime”。这个结论对降低出错率有什么意义？

A randomized algorithm is often the simplest and most efficient way to solve a problem. We shall use randomized algorithms occasionally throughout this book.

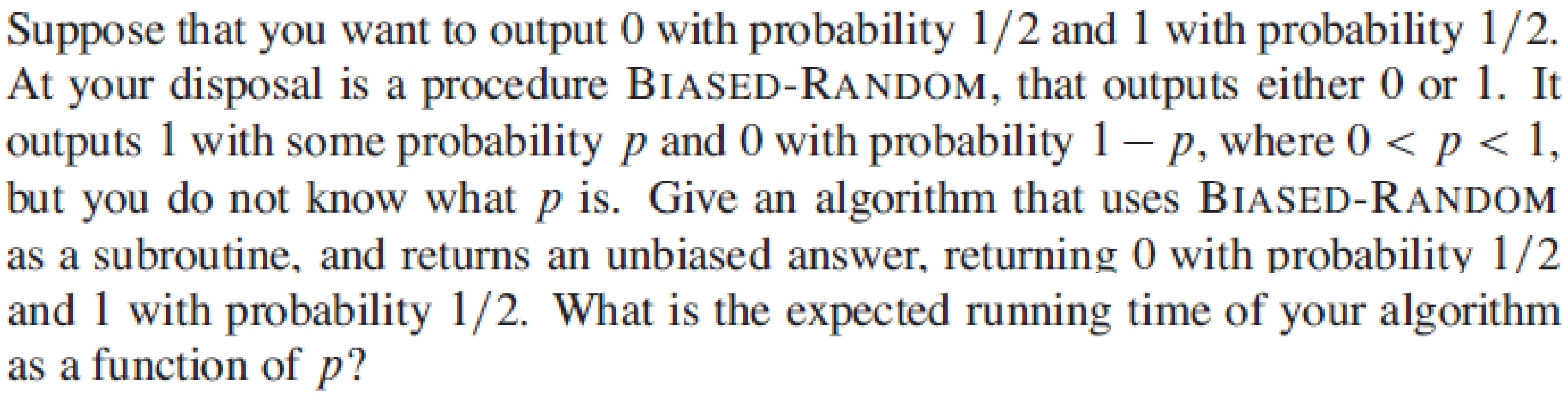
简单、高效是有代价的，所以随机算法经常用于那些“难”问题，牺牲“一点点”正确性。

问题11：

经证明正确的确定算法不会出错吗？

Open Topics：

- 1，如何扩展证明引理5.4？
- 2:



Suppose that you want to output 0 with probability 1/2 and 1 with probability 1/2. At your disposal is a procedure BIASED-RANDOM, that outputs either 0 or 1. It outputs 1 with some probability p and 0 with probability $1-p$, where $0 < p < 1$, but you do not know what p is. Give an algorithm that uses BIASED-RANDOM as a subroutine, and returns an unbiased answer, returning 0 with probability 1/2 and 1 with probability 1/2. What is the expected running time of your algorithm as a function of p?

课外作业

- CS pp.340-: 4, 8
- CS pp.355-: 1, 2, 4, 6, 12, 16, 18
- TC pp.122-: Ex.5.2-1, 5.2-2, 5.2-4
- TC pp.128-: Ex.5.3-1 – Ex.5.3-4
- TC pp.143-: prob.5-2

如何识别bad primes?

$$\text{Number}(x) = h_x \cdot p + s, \qquad \text{Number}(y) = h_y \cdot p + s,$$

$$\begin{array}{r} \text{Number}(x) \\ -\text{Number}(y) \\ \hline \text{Dif}(x,y) \end{array} \qquad \begin{array}{r} h_x \cdot p + s \\ -h_y \cdot p - s \\ \hline h_x \cdot p - h_y \cdot p \end{array}$$

$$\text{Dif}(x,y) = \text{Number}(x) - \text{Number}(y) = h_x \cdot p - h_y \cdot p = (h_x - h_y) \cdot p.$$

你能得出什么有用的结论吗？

A prime p is bad for (x, y) if and only if p divides Dif(x, y)

Bad Primes数量的上限

显然 $\mathrm{Dif}(x,y) = \mathrm{Number}(x) - \mathrm{Number}(y) < 2^n$

你还记得“算术基本定理”吗？

each positive integer larger than 1 can be unambiguously expressed as a product of primes.

$$\begin{aligned}\mathrm{Dif}(x,y) = p_1^{i_1} \cdot p_2^{i_2} \cdot p_3^{i_3} \cdot \ldots \cdot p_k^{i_k} &\geq p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_k \\ &> 1 \cdot 2 \cdot 3 \cdot \ldots \cdot k \\ &= k!\end{aligned}$$

Since $n! > 2^n$ for $n \geq 4$, k must be smaller than n and in this way we have obtained the stated aim that

$$k \leq n-1,$$