Lecture Notes for Chapter 5: Probabilistic Analysis and Randomized Algorithms

[This chapter introduces probabilistic analysis and randomized algorithms. It assumes that the student is familiar with the basic probability material in Appendix C.

The primary goals of these notes are to

- *explain the difference between probabilistic analysis and randomized algorithms,*
- *present the technique of indicator random variables, and*
- *give another example of the analysis of a randomized algorithm (permuting an array in place).*

These notes omit the technique of permuting an array by sorting, and they omit the starred Section 5.4.]

The hiring problem

Scenario:

- You are using an employment agency to hire a new office assistant.
- The agency sends you one candidate each day.
- You interview the candidate and must immediately decide whether or not to hire that person. But if you hire, you must also fire your current office assistant—even if it's someone you have recently hired.
- Cost to interview is c_i per candidate (interview fee paid to agency).
- Cost to hire is c_h per candidate (includes cost to fire current office assistant + hiring fee paid to agency).
- Assume that $c_h > c_i$.
- You are committed to having hired, at all times, the best candidate seen so far. Meaning that whenever you interview a candidate who is better than your current office assistant, you must fire the current office assistant and hire the candidate. Since you must have someone hired at all times, you will always hire the first candidate that you interview.

Goal: Determine what the price of this strategy will be.

Pseudocode to model this scenario: Assumes that the candidates are numbered 1 to n and that after interviewing each candidate, we can determine if it's better than the current office assistant. Uses a dummy candidate 0 that is worse than all others, so that the first candidate is always hired.

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

Cost: If n candidates, and we hire m of them, the cost is $O(nc_i + mc_h)$.

- Have to pay nc_i to interview, no matter how many we hire.
- So we focus on analyzing the hiring cost mc_h.
- mc_h varies with each run—it depends on the order in which we interview the candidates.
- This is a model of a common paradigm: we need to find the maximum or minimum in a sequence by examining each element and maintaining a current "winner." The variable m denotes how many times we change our notion of which element is currently winning.

Worst-case analysis

In the worst case, we hire all n candidates.

This happens if each one is better than all who came before. In other words, if the candidates appear in increasing order of quality.

If we hire all n, then the cost is $O(nc_i + nc_h) = O(nc_h)$ (since $c_h > c_i$).

Probabilistic analysis

In general, we have no control over the order in which candidates appear.

We could assume that they come in a random order:

- Assign a rank to each candidate: $rank(i)$ is a unique integer in the range 1 to n.
- The ordered list $\langle rank(1), rank(2), \ldots, rank(n)\rangle$ is a permutation of the candidate numbers $\langle 1, 2, \ldots, n\rangle$.
- The list of ranks is equally likely to be any one of the $n!$ permutations.
- Equivalently, the ranks form a ***uniform random permutation***: each of the possible $n!$ permutations appears with equal probability.

Essential idea of probabilistic analysis: We must use knowledge of, or make assumptions about, the distribution of inputs.

- The expectation is over this distribution.
- This technique requires that we can make a reasonable characterization of the input distribution.

Randomized algorithms

We might not know the distribution of inputs, or we might not be able to model it computationally.

Instead, we use randomization within the algorithm in order to impose a distribution on the inputs.

For the hiring problem: Change the scenario:

- The employment agency sends us a list of all n candidates in advance.
- On each day, we randomly choose a candidate from the list to interview (but considering only those we have not yet interviewed).
- Instead of relying on the candidates being presented to us in a random order, we take control of the process and enforce a random order.

What makes an algorithm randomized: An algorithm is ***randomized*** if its behavior is determined in part by values produced by a ***random-number generator***.

- $\textsc{Random}(a, b)$ returns an integer r, where $a \le r \le b$ and each of the $b - a + 1$ possible values of r is equally likely.
- In practice, RANDOM is implemented by a ***pseudorandom-number generator***, which is a deterministic method returning numbers that "look" random and pass statistical tests.

Indicator random variables

A simple yet powerful technique for computing the expected value of a random variable.

Helpful in situations in which there may be dependence.

Given a sample space and an event A, we define the ***indicator random variable***

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

Lemma
For an event A, let $X_A = \mathrm{I}\{A\}$. Then $\mathrm{E}[X_A] = \Pr\{A\}$.

Proof Letting $\overline{A}$ be the complement of A, we have

$$\begin{aligned} \mathrm{E}[X_A] &= \mathrm{E}[\mathrm{I}\{A\}] \\ &= 1 \cdot \Pr\{A\} + 0 \cdot \Pr\{\overline{A}\} \quad \text{(definition of expected value)} \\ &= \Pr\{A\} . \end{aligned}$$

■ (lemma)

Simple example: Determine the expected number of heads when we flip a fair coin one time.

- Sample space is $\{H, T\}$.
- $\Pr\{H\} = \Pr\{T\} = 1/2$.
- Define indicator random variable $X_H = \mathrm{I}\{H\}$. X_H counts the number of heads in one flip.
- Since $\Pr\{H\} = 1/2$, lemma says that $\mathrm{E}[X_H] = 1/2$.

Slightly more complicated example: Determine the expected number of heads in n coin flips.

- Let X be a random variable for the number of heads in n flips.
- Could compute $\mathrm{E}[X] = \sum_{k=0}^{n} k \cdot \Pr\{X = k\}$. In fact, this is what the book does in equation (C.36).
- Instead, we'll use indicator random variables.
- For $i = 1, 2, \ldots, n$, define $X_i = \mathrm{I}\{\text{the } i\text{th flip results in event } H\}$.
- Then $X = \sum_{i=1}^{n} X_i$.
- Lemma says that $\mathrm{E}[X_i] = \Pr\{H\} = 1/2$ for $i = 1, 2, \ldots, n$.
- Expected number of heads is $\mathrm{E}[X] = \mathrm{E}[\sum_{i=1}^{n} X_i]$.
- ***Problem:*** We want $\mathrm{E}[\sum_{i=1}^{n} X_i]$. We have only the individual expectations $\mathrm{E}[X_1], \mathrm{E}[X_2], \ldots, \mathrm{E}[X_n]$.
- ***Solution:*** Linearity of expectation says that the expectation of the sum equals the sum of the expectations. Thus,

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

- Linearity of expectation applies even when there is dependence among the random variables. *[Not an issue in this example, but it can be a great help. The hat-check problem of Exercise 5.2-4 is a problem with lots of dependence. See the solution on page 5-10 of this manual.]*

Analysis of the hiring problem

Assume that the candidates arrive in a random order.

Let X be a random variable that equals the number of times we hire a new office assistant.

Define indicator random variables $X_1, X_2, \ldots, X_n$, where

$X_i = \mathrm{I}\{\text{candidate } i \text{ is hired}\}$.

Useful properties:

- $X = X_1 + X_2 + \cdots + X_n$.
- Lemma $\Rightarrow$ $\mathrm{E}[X_i] = \Pr\{\text{candidate } i \text{ is hired}\}$.

We need to compute $\Pr\{\text{candidate } i \text{ is hired}\}$.

- Candidate i is hired if and only if candidate i is better than each of candidates $1, 2, \ldots, i-1$.
- Assumption that the candidates arrive in random order $\Rightarrow$ candidates $1, 2, \ldots, i$ arrive in random order $\Rightarrow$ any one of these first i candidates is equally likely to be the best one so far.
- Thus, $\Pr\{\text{candidate } i \text{ is the best so far}\} = 1/i$.
- Which implies $\mathrm{E}[X_i] = 1/i$.

Now compute $\mathrm{E}[X]$:

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] \\
&= \sum_{i=1}^{n} 1/i \\
&= \ln n + O(1) \quad \text{(equation (A.7): the sum is a harmonic series)} .
\end{aligned}$$

Thus, the expected hiring cost is $O(c_h \ln n)$, which is much better than the worst-case cost of $O(nc_h)$.

Randomized algorithms

Instead of assuming a distribution of the inputs, we impose a distribution.

The hiring problem

For the hiring problem, the algorithm is deterministic:

- For any given input, the number of times we hire a new office assistant will always be the same.
- The number of times we hire a new office assistant depends only on the input.
- In fact, it depends only on the ordering of the candidates' ranks that it is given.
- Some rank orderings will always produce a high hiring cost. Example: $\langle 1, 2, 3, 4, 5, 6\rangle$, where each candidate is hired.
- Some will always produce a low hiring cost. Example: any ordering in which the best candidate is the first one interviewed. Then only the best candidate is hired.
- Some may be in between.

Instead of always interviewing the candidates in the order presented, what if we first randomly permuted this order?

- The randomization is now in the algorithm, not in the input distribution.
- Given a particular input, we can no longer say what its hiring cost will be. Each time we run the algorithm, we can get a different hiring cost.
- In other words, each time we run the algorithm, the execution depends on the random choices made.
- No particular input always elicits worst-case behavior.
- Bad behavior occurs only if we get "unlucky" numbers from the random-number generator.

Pseudocode for randomized hiring problem:

```
RANDOMIZED-HIRE-ASSISTANT(n)
randomly permute the list of candidates
HIRE-ASSISTANT(n)
```

Lemma
The expected hiring cost of RANDOMIZED-HIRE-ASSISTANT is $O(c_h \ln n)$.

Proof After permuting the input array, we have a situation identical to the probabilistic analysis of deterministic HIRE-ASSISTANT. ■

Randomly permuting an array

[The book considers two methods of randomly permuting an n-element array. The first method assigns a random priority in the range 1 to n^3 to each position and then reorders the array elements into increasing priority order. We omit this method from these notes. The second method is better: it works in place (unlike the priority-based method), it runs in linear time without requiring sorting, and it needs fewer random bits (n random numbers in the range 1 to n rather than the range 1 to n^3). We present and analyze the second method in these notes.]

Goal: Produce a uniform random permutation (each of the $n!$ permutations is equally likely to be produced).

Non-goal: Show that for each element $A[i]$, the probability that $A[i]$ moves to position j is $1/n$. (See Exercise 5.3-4, whose solution is on page 5-13 of this manual.)

The following procedure permutes the array $A[1 \,..\, n]$ in place (i.e., no auxiliary array is required).

```
RANDOMIZE-IN-PLACE(A, n)
for i ← 1 to n
    do swap A[i] ↔ A[RANDOM(i, n)]
```

Idea:

- In iteration i, choose $A[i]$ randomly from $A[i \,..\, n]$.
- Will never alter $A[i]$ after iteration i.

Time: $O(1)$ per iteration $\Rightarrow O(n)$ total.

Correctness: Given a set of n elements, a ***k-permutation*** is a sequence containing k of the n elements. There are $n!/(n-k)!$ possible k-permutations.

Lemma

RANDOMIZE-IN-PLACE computes a uniform random permutation.

Proof Use a loop invariant:

> **Loop invariant:** Just prior to the ith iteration of the **for** loop, for each possible $(i-1)$-permutation, subarray $A[1 \,..\, i-1]$ contains this $(i-1)$-permutation with probability $(n-i+1)!/n!$.

Initialization: Just before first iteration, $i = 1$. Loop invariant says that for each possible 0-permutation, subarray $A[1 \,..\, 0]$ contains this 0-permutation with probability $n!/n! = 1$. $A[1 \,..\, 0]$ is an empty subarray, and a 0-permutation has no elements. So, $A[1 \,..\, 0]$ contains any 0-permutation with probability 1.

Maintenance: Assume that just prior to the ith iteration, each possible $(i-1)$-permutation appears in $A[1 \,..\, i-1]$ with probability $(n-i+1)!/n!$. Will show that after the ith iteration, each possible i-permutation appears in $A[1 \,..\, i]$ with probability $(n-i)!/n!$. Incrementing i for the next iteration then maintains the invariant.

Consider a particular i-permutation $\pi = \langle x_1, x_2, \ldots, x_i \rangle$. It consists of an $(i-1)$-permutation $\pi' = \langle x_1, x_2, \ldots, x_{i-1} \rangle$, followed by x_i.

Let E_1 be the event that the algorithm actually puts π' into $A[1 \,..\, i-1]$. By the loop invariant, $\Pr\{E_1\} = (n-i+1)!/n!$.

Let E_2 be the event that the ith iteration puts x_i into $A[i]$.

We get the i-permutation π in $A[1 \,..\, i]$ if and only if both E_1 and E_2 occur $\Rightarrow$ the probability that the algorithm produces π in $A[1 \,..\, i]$ is $\Pr\{E_2 \cap E_1\}$.

Equation (C.14) $\Rightarrow \Pr\{E_2 \cap E_1\} = \Pr\{E_2 \mid E_1\} \Pr\{E_1\}$.

The algorithm chooses x_i randomly from the $n-i+1$ possibilities in $A[i \,..\, n]$ $\Rightarrow \Pr\{E_2 \mid E_1\} = 1/(n-i+1)$. Thus,

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

Termination: At termination, $i = n+1$, so we conclude that $A[1 \,..\, n]$ is a given n-permutation with probability $(n-n)!/n! = 1/n!$. ■ (lemma)

Solutions for Chapter 5: Probabilistic Analysis and Randomized Algorithms

Solution to Exercise 5.1-3

To get an unbiased random bit, given only calls to BIASED-RANDOM, call BIASED-RANDOM twice. Repeatedly do so until the two calls return different values, and when this occurs, return the first of the two bits:

```
UNBIASED-RANDOM
while TRUE
    do
        $x \leftarrow$ BIASED-RANDOM
        $y \leftarrow$ BIASED-RANDOM
        if $x \neq y$
          then return $x$
```

To see that UNBIASED-RANDOM returns 0 and 1 each with probability $1/2$, observe that the probability that a given iteration returns 0 is

$$\Pr\{x = 0 \text{ and } y = 1\} = (1 - p)p ,$$

and the probability that a given iteration returns 1 is

$$\Pr\{x = 1 \text{ and } y = 0\} = p(1 - p) .$$

(We rely on the bits returned by BIASED-RANDOM being independent.) Thus, the probability that a given iteration returns 0 equals the probability that it returns 1. Since there is no other way for UNBIASED-RANDOM to return a value, it returns 0 and 1 each with probability $1/2$.

Assuming that each iteration takes $O(1)$ time, the expected running time of UNBIASED-RANDOM is linear in the expected number of iterations. We can view each iteration as a Bernoulli trial, where "success" means that the iteration returns a value. The probability of success equals the probability that 0 is returned plus the probability that 1 is returned, or $2p(1 - p)$. The number of trials until a success occurs is given by the geometric distribution, and by equation (C.31), the expected number of trials for this scenario is $1/(2p(1 - p))$. Thus, the expected running time of UNBIASED-RANDOM is $\Theta(1/(2p(1 - p)))$.

Solution to Exercise 5.2-1

Since HIRE-ASSISTANT always hires candidate 1, it hires exactly once if and only if no candidates other than candidate 1 are hired. This event occurs when candidate 1 is the best candidate of the n, which occurs with probability $1/n$.

HIRE-ASSISTANT hires n times if each candidate is better than all those who were interviewed (and hired) before. This event occurs precisely when the list of ranks given to the algorithm is $\langle 1, 2, \ldots, n \rangle$, which occurs with probability $1/n!$.

Solution to Exercise 5.2-2

We make three observations:

1. Candidate 1 is always hired.
2. The best candidate, i.e., the one whose rank is n, is always hired.
3. If the best candidate is candidate 1, then that is the only candidate hired.

Therefore, in order for HIRE-ASSISTANT to hire exactly twice, candidate 1 must have rank $i \le n-1$ and all candidates whose ranks are $i+1, i+2, \ldots, n-1$ must be interviewed after the candidate whose rank is n. (When $i = n-1$, this second condition vacuously holds.)

Let E_i be the event in which candidate 1 has rank i; clearly, $\Pr\{E_i\} = 1/n$ for any given value of i.

Letting j denote the position in the interview order of the best candidate, let F be the event in which candidates $2, 3, \ldots, j-1$ have ranks strictly less than the rank of candidate 1. Given that event E_i has occurred, event F occurs when the best candidate is the first one interviewed out of the $n-i$ candidates whose ranks are $i+1, i+2, \ldots, n$. Thus, $\Pr\{F \mid E_i\} = 1/(n-i)$.

Our final event is A, which occurs when HIRE-ASSISTANT hires exactly twice. Noting that the events $E_1, E_2, \ldots, E_n$ are disjoint, we have

$$\begin{aligned} A &= F \cap (E_1 \cup E_2 \cup \cdots \cup E_{n-1}) \\ &= (F \cap E_1) \cup (F \cap E_2) \cup \cdots \cup (F \cap E_{n-1}) \; . \end{aligned}$$

and

$$\Pr\{A\} = \sum_{i=1}^{n-1} \Pr\{F \cap E_i\} \; .$$

By equation (C.14),

$$\begin{aligned} \Pr\{F \cap E_i\} &= \Pr\{F \mid E_i\} \Pr\{E_i\} \\ &= \frac{1}{n-i} \cdot \frac{1}{n} \; , \end{aligned}$$

and so

$$\begin{aligned}
\Pr\{A\} &= \sum_{i=1}^{n-1} \frac{1}{n-i} \cdot \frac{1}{n} \\
&= \frac{1}{n} \sum_{i=1}^{n-1} \frac{1}{n-i} \\
&= \frac{1}{n} \left(\frac{1}{n-1} + \frac{1}{n-2} + \cdots + \frac{1}{1} \right) \\
&= \frac{1}{n} \cdot H_{n-1} ,
\end{aligned}$$

where H_{n-1} is the nth harmonic number.

Solution to Exercise 5.2-4

Another way to think of the hat-check problem is that we want to determine the expected number of fixed points in a random permutation. (A ***fixed point*** of a permutation π is a value i for which $\pi(i) = i$.) One could enumerate all $n!$ permutations, count the total number of fixed points, and divide by $n!$ to determine the average number of fixed points per permutation. This would be a painstaking process, and the answer would turn out to be 1. We can use indicator random variables, however, to arrive at the same answer much more easily.

Define a random variable X that equals the number of customers that get back their own hat, so that we want to compute $\mathrm{E}[X]$.

For $i = 1, 2, \ldots, n$, define the indicator random variable

$X_i = \mathrm{I}\{\text{customer } i \text{ gets back his own hat}\}$.

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

Since the ordering of hats is random, each customer has a probability of $1/n$ of getting back his own hat. In other words, $\Pr\{X_i = 1\} = 1/n$, which, by Lemma 5.1, implies that $\mathrm{E}[X_i] = 1/n$.

Thus,

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] \quad \text{(linearity of expectation)} \\
&= \sum_{i=1}^{n} 1/n \\
&= 1 ,
\end{aligned}$$

and so we expect that exactly 1 customer gets back his own hat.

Note that this is a situation in which the indicator random variables are *not* independent. For example, if $n = 2$ and $X_1 = 1$, then X_2 must also equal 1. Conversely, if $n = 2$ and $X_1 = 0$, then X_2 must also equal 0. Despite the dependence,

$\Pr\{X_i = 1\} = 1/n$ for all i, and linearity of expectation holds. Thus, we can use the technique of indicator random variables even in the presence of dependence.

Solution to Exercise 5.2-5

Let X_{ij} be an indicator random variable for the event where the pair $A[i], A[j]$ for $i < j$ is inverted, i.e., $A[i] > A[j]$. More precisely, we define $X_{ij} = \mathrm{I}\{A[i] > A[j]\}$ for $1 \le i < j \le n$. We have $\Pr\{X_{ij} = 1\} = 1/2$, because given two distinct random numbers, the probability that the first is bigger than the second is 1/2. By Lemma 5.1, $\mathrm{E}[X_{ij}] = 1/2$.

Let X be the the random variable denoting the total number of inverted pairs in the array, so that

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

We want the expected number of inverted pairs, so we take the expectation of both sides of the above equation to obtain

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] .$$

We use linearity of expectation to get

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathrm{E}[X_{ij}] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1/2 \\
&= \binom{n}{2}\frac{1}{2} \\
&= \frac{n(n-1)}{2} \cdot \frac{1}{2} \\
&= \frac{n(n-1)}{4} .
\end{aligned}$$

Thus the expected number of inverted pairs is $n(n-1)/4$.

Solution to Exercise 5.3-1

Here's the rewritten procedure:

```
RANDOMIZE-IN-PLACE(A)
n ← length[A]
swap A[1] ↔ A[RANDOM(1, n)]
for i ← 2 to n
    do swap A[i] ↔ A[RANDOM(i, n)]
```

The loop invariant becomes

> **Loop invariant:** Just prior to the iteration of the **for** loop for each value of $i = 2, \ldots, n$, for each possible $(i-1)$-permutation, the subarray $A[1 \,..\, i-1]$ contains this $(i-1)$-permutation with probability $(n-i+1)!/n!$.

The maintenance and termination parts remain the same. The initialization part is for the subarray $A[1 \,..\, 1]$, which contains any 1-permutation with probability $(n-1)!/n! = 1/n$.

Solution to Exercise 5.3-2

Although PERMUTE-WITHOUT-IDENTITY will not produce the identity permutation, there are other permutations that it fails to produce. For example, consider its operation when $n = 3$, when it should be able to produce the $n! - 1 = 5$ non-identity permutations. The **for** loop iterates for $i = 1$ and $i = 2$. When $i = 1$, the call to RANDOM returns one of two possible values (either 2 or 3), and when $i = 2$, the call to RANDOM returns just one value (3). Thus, there are only $2 \cdot 1 = 2$ possible permutations that PERMUTE-WITHOUT-IDENTITY can produce, rather than the 5 that are required.

Solution to Exercise 5.3-3

The PERMUTE-WITH-ALL procedure does not produce a uniform random permutation. Consider the permutations it produces when $n = 3$. There are 3 calls to RANDOM, each of which returns one of 3 values, and so there are 27 possible outcomes of calling PERMUTE-WITH-ALL. Since there are $3! = 6$ permutations, if PERMUTE-WITH-ALL did produce a uniform random permutation, then each permutation would occur $1/6$ of the time. That would mean that each permutation would have to occur an integer number m times, where $m/27 = 1/6$. No integer m satisfies this condition.

In fact, if we were to work out the possible permutations of $\langle 1, 2, 3 \rangle$ and how often they occur with PERMUTE-WITH-ALL, we would get the following probabilities:

permutation	probability
$\langle 1, 2, 3 \rangle$	$4/27$
$\langle 1, 3, 2 \rangle$	$5/27$
$\langle 2, 1, 3 \rangle$	$5/27$
$\langle 2, 3, 1 \rangle$	$5/27$
$\langle 3, 1, 2 \rangle$	$4/27$
$\langle 3, 2, 1 \rangle$	$4/27$

Although these probabilities add to 1, none are equal to $1/6$.

Solution to Exercise 5.3-4

PERMUTE-BY-CYCLIC chooses *offset* as a random integer in the range $1 \leq \mathit{offset} \leq n$, and then it performs a cyclic rotation of the array. That is, $B[((i + \mathit{offset} - 1) \bmod n) + 1] \leftarrow A[i]$ for $i = 1, 2, \ldots, n$. (The subtraction and addition of 1 in the index calculation is due to the 1-origin indexing. If we had used 0-origin indexing instead, the index calculation would have simplied to $B[(i + \mathit{offset}) \bmod n] \leftarrow A[i]$ for $i = 0, 1, \ldots, n - 1$.)

Thus, once *offset* is determined, so is the entire permutation. Since each value of *offset* occurs with probability $1/n$, each element $A[i]$ has a probability of ending up in position $B[j]$ with probability $1/n$.

This procedure does not produce a uniform random permutation, however, since it can produce only n different permutations. Thus, n permutations occur with probability $1/n$, and the remaining $n! - n$ permutations occur with probability 0.

Solution to Exercise 5.4-6

First we determine the expected number of empty bins. We define a random variable X to be the number of empty bins, so that we want to compute $\mathrm{E}[X]$. Next, for $i = 1, 2, \ldots, n$, we define the indicator random variable $Y_i = \mathrm{I}\{\text{bin } i \text{ is empty}\}$. Thus,

$$X = \sum_{i=1}^{n} Y_i ,$$

and so

$$\begin{aligned} \mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} Y_i\right] \\ &= \sum_{i=1}^{n} \mathrm{E}[Y_i] && \text{(by linearity of expectation)} \\ &= \sum_{i=1}^{n} \Pr\{\text{bin } i \text{ is empty}\} && \text{(by Lemma 5.1)} . \end{aligned}$$

Let us focus on a specific bin, say bin i. We view a toss as a success if it misses bin i and as a failure if it lands in bin i. We have n independent Bernoulli trials, each with probability of success $1 - 1/n$. In order for bin i to be empty, we need n successes in n trials. Using a binomial distribution, therefore, we have that

$$\begin{aligned} \Pr\{\text{bin } i \text{ is empty}\} &= \binom{n}{n}\left(1 - \frac{1}{n}\right)^n \left(\frac{1}{n}\right)^0 \\ &= \left(1 - \frac{1}{n}\right)^n . \end{aligned}$$

Thus,

$$\begin{aligned} \mathrm{E}[X] &= \sum_{i=1}^{n} \left(1 - \frac{1}{n}\right)^n \\ &= n\left(1 - \frac{1}{n}\right)^n . \end{aligned}$$

By equation (3.13), as n approaches ∞, the quantity $(1 - 1/n)^n$ approaches $1/e$, and so $\mathrm{E}[X]$ approaches n/e.

Now we determine the expected number of bins with exactly one ball. We redefine X to be number of bins with exactly one ball, and we redefine Y_i to be $\mathrm{I}\{\text{bin } i \text{ gets exactly one ball}\}$. As before, we find that

$$\mathrm{E}[X] = \sum_{i=1}^{n} \Pr\{\text{bin } i \text{ gets exactly one ball}\} .$$

Again focusing on bin i, we need exactly $n-1$ successes in n independent Bernoulli trials, and so

$$\begin{aligned} \Pr\{\text{bin } i \text{ gets exactly one ball}\} &= \binom{n}{n-1}\left(1 - \frac{1}{n}\right)^{n-1}\left(\frac{1}{n}\right)^1 \\ &= n \cdot \left(1 - \frac{1}{n}\right)^{n-1} \frac{1}{n} \\ &= \left(1 - \frac{1}{n}\right)^{n-1} , \end{aligned}$$

and so

$$\begin{aligned} \mathrm{E}[X] &= \sum_{i=1}^{n} \left(1 - \frac{1}{n}\right)^{n-1} \\ &= n\left(1 - \frac{1}{n}\right)^{n-1} . \end{aligned}$$

Because

$$n\left(1 - \frac{1}{n}\right)^{n-1} = \frac{n\,(1 - \frac{1}{n})^n}{1 - \frac{1}{n}} ,$$

as n approaches ∞, we find that $\mathrm{E}[X]$ approaches

$$\frac{n/e}{1 - 1/n} = \frac{n^2}{e(n-1)} .$$

Solution to Problem 5-1

a. To determine the expected value represented by the counter after n INCREMENT operations, we define some random variables:

- For $j = 1, 2, \ldots, n$, let X_j denote the increase in the value represented by the counter due to the jth INCREMENT operation.
- Let V_n be the value represented by the counter after n INCREMENT operations.

Then $V_n = X_1 + X_2 + \cdots + X_n$. We want to compute $\mathrm{E}[V_n]$. By linearity of expectation,

$$\mathrm{E}[V_n] = \mathrm{E}[X_1 + X_2 + \cdots + X_n] = \mathrm{E}[X_1] + \mathrm{E}[X_2] + \cdots + \mathrm{E}[X_n] \ .$$

We shall show that $\mathrm{E}[X_j] = 1$ for $j = 1, 2, \ldots, n$, which will prove that $\mathrm{E}[V_n] = n$.

We actually show that $\mathrm{E}[X_j] = 1$ in two ways, the second more rigorous than the first:

1. Suppose that at the start of the jth INCREMENT operation, the counter holds the value i, which represents n_i. If the counter increases due to this INCREMENT operation, then the value it represents increases by $n_{i+1} - n_i$. The counter increases with probability $1/(n_{i+1} - n_i)$, and so

$$\begin{aligned}
\mathrm{E}[X_j] &= (0 \cdot \Pr\{\text{counter does not increase}\}) \\
&\qquad + ((n_{i+1} - n_i) \cdot \Pr\{\text{counter increases}\}) \\
&= \left(0 \cdot \left(1 - \frac{1}{n_{i+1} - n_i}\right)\right) + \left((n_{i+1} - n_i) \cdot \frac{1}{n_{i+1} - n_i}\right) \\
&= 1 \ ,
\end{aligned}$$

   and so $\mathrm{E}[X_j] = 1$ regardless of the value held by the counter.

2. Let C_j be the random variable denoting the value held in the counter at the start of the jth INCREMENT operation. Since we can ignore values of C_j greater than $2^b - 1$, we use a formula for conditional expectation:

$$\begin{aligned}
\mathrm{E}[X_j] &= \mathrm{E}[\mathrm{E}[X_j \mid C_j]] \\
&= \sum_{i=0}^{2^b-1} \mathrm{E}[X_j \mid C_j = i] \cdot \Pr\{C_j = i\} \ .
\end{aligned}$$

   To compute $\mathrm{E}[X_j \mid C_j = i]$, we note that

   - $\Pr\{X_j = 0 \mid C_j = i\} = 1 - 1/(n_{i+1} - n_i)$,
   - $\Pr\{X_j = n_{i+1} - n_i \mid C_j = i\} = 1/(n_{i+1} - n_i)$, and
   - $\Pr\{X_j = k \mid C_j = i\} = 0$ for all other k.

   Thus,

$$\begin{aligned}
\mathrm{E}[X_j \mid C_j = i] &= \sum_k k \cdot \Pr\{X_j = k \mid C_j = i\} \\
&= \left(0 \cdot \left(1 - \frac{1}{n_{i+1} - n_i}\right)\right) + \left((n_{i+1} - n_i) \cdot \frac{1}{n_{i+1} - n_i}\right) \\
&= 1 \ .
\end{aligned}$$

   Therefore, noting that

$$\sum_{i=0}^{2^b-1} \Pr\{C_j = i\} = 1 \ ,$$

   we have

$$\begin{aligned}
\mathrm{E}[X_j] &= \sum_{i=0}^{2^b-1} 1 \cdot \Pr\{C_j = i\} \\
&= 1 \ .
\end{aligned}$$

Why is the second way more rigorous than the first? Both ways condition on the value held in the counter, but only the second way incorporates the conditioning into the expression for $\mathrm{E}[X_j]$.

b. Defining V_n and X_j as in part (a), we want to compute $\mathrm{Var}[V_n]$, where $n_i = 100i$. The X_j are pairwise independent, and so by equation (C.28), $\mathrm{Var}[V_n] = \mathrm{Var}[X_1] + \mathrm{Var}[X_2] + \cdots + \mathrm{Var}[X_n]$.

Since $n_i = 100i$, we see that $n_{i+1} - n_i = 100(i+1) - 100i = 100$. Therefore, with probability $99/100$, the increase in the value represented by the counter due to the jth INCREMENT operation is 0, and with probability $1/100$, the value represented increases by 100. Thus, by equation (C.26),

$$\begin{aligned}
\mathrm{Var}[X_j] &= \mathrm{E}\left[X_j^2\right] - \mathrm{E}^2[X_j] \\
&= \left(\left(0^2 \cdot \frac{99}{100}\right) + \left(100^2 \cdot \frac{1}{100}\right)\right) - 1^2 \\
&= 100 - 1 \\
&= 99 .
\end{aligned}$$

Summing up the variances of the X_j gives $\mathrm{Var}[V_n] = 99n$.