Chapter 5 Probabilistic Analysis and Randomized Algorithm

5.1 The hiring problem

Exercise 5.1-1

Show that the assumption that we are always able to determine which candidate is best, in line 4 of procedure HireAssistant, implies that we know a total order on the ranks of the candidates.

Exercise 5.1-2 ★

Describe an implementation of the procedure $\text{Random}(a, b)$ that only makes calls to $\text{Random}(0, 1)$. What is the expected running time of your procedure, as a function of a and b?

Answer:

(See `+rndm/randint.m`.)

$\Theta(\lg(b-a))$

Exercise 5.1-3 ★

Suppose that you want to output 0 with probability $1/2$ and 1 with probability $1/2$. At your disposal is a procedure BiasedRandom, 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 BiasedRandom 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?

Answer:*

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

(See `+rndm/randbit.m`.)

To see that UnbiasedRandom 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 BiasedRandom 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 UnbiasedRandom 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 UnbiasedRandom 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.32) ($\text{E}[X] = 1/p$), the expected number of trials for this scenario is $1/(2p(1-p))$. Thus, the expected running time of UnbiasedRandom is $\Theta(1/(2p(1-p)))$.

5.2 Indicator random variables

Exercises 5.2-1

In HireAssistant, assuming that the candidates are presented in a random order, what is the probability that you hire exactly one time? What is the probability that you hire exactly n times?

Answer:*

Since HireAssistant 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 $1/n$.

HireAssistant 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 $1/n!$.

Exercises 5.2-2

In HireAssistant, assuming that the candidates are presented in a random order, what is the probability that you hire exactly twice?

Answer:*

We make three observations:

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

Therefore, in order for HireAssistant to hire exactly twice, candidate 1 must have rank $i \leq n-1$ 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, $\Pr\{F|E_i\} = 1/(n-i)$.

Our final event is A, which occurs when HireAssistant 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) ($\Pr\{A|B\} = \Pr\{A \cap B\} / \Pr\{B\}$),

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

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.

Exercises 5.2-3

Use indicator random variables to compute the expected value of the sum of n dice.

Answer:

$3.5n$

Exercises 5.2-4

Use indicator random variables to solve the following problem, which is known as the ***hat-check problem***. Each of n customers gives a hat to a hat-check person at a restaurant. The hat-check person gives the hats back to the customers in a random order. What is the expected number of customers who get back their own hat?

Answer:*

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 $\pi(i) = i$.) We 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 or her own hat. In other words, $\Pr\{X_i = 1\} = 1/n$, which, by Lemma 5.1, (Given a sample space S, let $X_A = \mathrm{I}\{A\}$. Then $\mathrm{E}[X_A] = \Pr\{A\}$.) 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 $X_0 = 1$, 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.

Exercises 5.2-5

Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A. (See Problem 2-4 for more on inversions.) Suppose that the elements of A form a uniform random permutation of $\langle 1, 2, \ldots, n \rangle$. Use indicator random variables to compute the expected number of inversions.

Answer:*

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

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

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

5.3 Randomized algorithms

Exercise 5.3-1

Professor Marceau objects to the loop invariant used in the proof of Lemma 5.5. He questions whether it is true prior to the first iteration. He reasons that we could just as easily declare that an empty subarray contains no 0-permutations. Therefore the probability that an empty subarray

contains a 0-permutation should be 0, thus invalidating the loop invariant prior to the first iteration. Rewrite the procedure RandomizeInPlace so that its associated loop invariant applies to a nonempty subarray prior to the first iteration, and modify the proof of Lemma 5.5 for your procedure.

Answer:*

Here's the rewritten procedure:

```
function a = randomize(a)

    n = length(a);

    j = randi([1, n]);

    [a(1), a(j)] = deal(a(j), a(1));

    for i = 2:n

        j = randi([i, n]);

        [a(i), a(j)] = deal(a(j), a(i));

    end

end
```

The loop invariant becomes

- **Loop invariant:** Just prior to the iteration of the $\textbf{for}$ 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 partis for the subarray $A[1..1]$, which contains any 1-permutation with probability $(n-1)!/n! = 1/n$.

Exercise 5.3-2

Professor Kelp decides to write a procedure that produces at random any permutation besides the identity permutation. He proposes the following procedure:

```
function a = permute_without_identity(a)

    n = length(a);

    for i = 1:n-1

        j = randi([i+1, n]);

        [a(i), a(j)] = deal(a(j), a(i));

    end

end
```

Does this code do what Professor Kelp intends?

Answer:*

Although $\text{PermuteWithoutIdentity}$ 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 $\textbf{for}$ 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, $\text{PermuteWithoutIdentity}$ can produce only $2 \cdot 1 = 2$ possible permutations, rather than the 5 that are required.

Exercise 5.3-3

Suppose that instead of swapping element $A[i]$ with a random element from the subarray $A[i..n]$, we swapped it with a random element from anywhere in the array:

```
function a = permute_with_all(a)

    n = length(a);

    for i = 1:n

        j = randi([1, n]);

        [a(i), a(j)] = deal(a(j), a(i));

    end

end
```

Does this code produce a uniform random permutation? Why or why not?

Answer:*

The PermuteWithAll procedure does not produce a uniform random permutation. Consider the permutations it produces when $n = 3$. The procedure makes 3 calls to Random, each of which returns one of 3 values, and so calling PermuteWithAll has 27 possible outcomes. Since there are $3! = 6$ permutations, if PermuteWithAll 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 PermuteWithAll, 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 sum to 1, none are equal to $1/6$.

Exercise 5.3-4

Professor Armstrong suggests the following procedure for generating a uniform random permutation:

(See `+random/permute_cyclic.m`).

Show that each element $A[i]$ has a $1/n$ probability of winding up in any particular position in B. Then show that Professor Armstrong is mistaken by showing that the resulting permutation is not uniformly random.

Answer:*

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

Exercise 5.3-5 ★

Prove that in the array P in procedure PermuteBySorting, the probability that all elements are unique is at least $1 - 1/n$.

Answer:

$$\begin{aligned}
\Pr\left\{\bigcup_{i<j} X_{ij}\right\} &\le \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \Pr\{X_{ij}\} \\
&= \frac{n(n-1)}{2}\cdot\frac{1}{n^3} \\
&= \frac{n-1}{2n^2} \\
&< \frac{1}{n}
\end{aligned}$$

Exercise 5.3-6

Explain how to implement the algorithm PermuteBySorting to handle the case in which two or more priorities are identical. That is, your algorithm should produce a uniform random permutation, even if two or more priorities are identical.

Answer:

(See `+rndm/permute.m`.)

Exercise 5.3-7

Suppose we want to create a ***random sample*** of the set $\{1, 2, 3, \ldots, n\}$, that is, an m-element subset S, where $0 \le m \le n$, such that each m-subset is equally likely to be created. One way would be to set $A[i] = i$ for $i = 1, 2, 3, \ldots, n$, call $\text{RandomizeInPlace}(A)$, and then take just the first m array elements. This method would make n calls to the Random procedure. If n is much larger than m, we can create a random sample with fewer calls to Random. Show that the following recursive procedure returns a random m-subset S of $\{1, 2, 3, \ldots, n\}$, in which each m-subset is equally likely, while making only m calls to Random:

(See `+rndm/random_sample.m`.)

Answer:*

Since each recursive call reduces m by 1 and makes only one call to Random, it's easy to see that there are a total of m calls to Random. Moreover, since each recursive call adds exactly one element to the set, it's easy to see that the resulting set S contains exactly m elements.

Because the elements of set S are chosen independently of each other, it suffices to show that each of the n values appears in S with probability m/n. We use an inductive proof. The inductive hypothesis is that a call to $\text{RandomSubset}(m, n)$ returns a set S of m elements, each appearing with probability m/n. The base cases are for $m = 0$ and $m = 1$. When $m = 0$, the returned set is empty, and so it contains each element with probability 0. When $m = 1$, the returned set has one element, and it is equally likely to be any number in $\{1, 2, 3, \ldots, n\}$.

For the inductive step, we assume that the call $\text{RandomSubset}(m-1, n-1)$ returns a set S' of $m-1$ elements in which each value in $\{1, 2, 3, \ldots, n-1\}$ occurs with probability $(m-1)/(n-1)$. After the line $i = \text{Random}(1, n)$, i is equally likely to be any value in $\{1, 2, 3, \ldots, n\}$. We consider separately the probabilities that S contains $j < n$ and that S contains n. Let R_j be the event that the call $\text{Random}(1, n)$ returns j, so that $\Pr\{R_j\} = 1/n$.

For $j < n$, the event that $j \in S$ is the union of two disjoint events:

- $j \in S'$, and
- $j \notin S'$ and R_j (these events are independent),

Thus,

$$\begin{aligned}
\Pr\{j \in S\} &= \Pr\{j \in S'\} + \Pr\{j \notin S' \text{ and } R_j\} && \text{(the events are disjoint)} \\
&= \frac{m-1}{n-1} + \left(1 - \frac{m-1}{n-1}\right) \cdot \frac{1}{n} && \text{(by the inductive hypothesis)} \\
&= \frac{m-1}{n-1} + \left(\frac{n-1}{n-1} - \frac{m-1}{n-1}\right) \cdot \frac{1}{n} \\
&= \frac{m-1}{n-1} \cdot \frac{n}{n} + \frac{n-m}{n-1} \cdot \frac{1}{n} \\
&= \frac{(m-1)n + (n-m)}{(n-1)n} \\
&= \frac{mn - n + n - m}{(n-1)n} \\
&= \frac{m(n-1)}{(n-1)n} \\
&= \frac{m}{n}.
\end{aligned}$$

The event that $n \in S$ is also the union of two disjoint events:

- R_n, and
- R_j and $j \in S'$ for some $j < n$ (these events are independent).

Thus,

$$\begin{aligned}
\Pr\{n \in S\} &= \Pr\{R_n\} + \Pr\{R_j \text{ and } j \in S' \text{ for some } j < n\} && \text{(the events are disjoint)} \\
&= \frac{1}{n} + \frac{n-1}{n} \cdot \frac{m-1}{n-1} && \text{(by the inductive hypothesis)} \\
&= \frac{1}{n} \cdot \frac{n-1}{n-1} + \frac{n-1}{n} \cdot \frac{m-1}{n-1} \\
&= \frac{n - 1 + nm - n - m + 1}{n(n-1)} \\
&= \frac{nm - m}{n(n-1)} \\
&= \frac{m(n-1)}{n(n-1)} \\
&= \frac{m}{n}.
\end{aligned}$$

5.4 Probabilistic analysis and further uses of indicator random variables ★

Exercise 5.4-1

How many people must there be in a room before the probability that someone has the same birthday as you do is at least $1/2$? How many people must there be before the probability that at

least two people have a birthday on July 4 is greater than $1/2$?

Answer:

$$\begin{aligned} 1 - \left(\frac{364}{365}\right)^k &\geq \frac{1}{2} \\ k &\geq 253 \end{aligned}$$

$$\begin{aligned} 1 - \left(\frac{364}{365}\right)^k - k \cdot \frac{1}{365}\left(\frac{364}{365}\right)^{k-1} &\geq \frac{1}{2} \\ k &\geq 612 \end{aligned}$$

Exercise 5.4-2

Suppose that we toss balls into b bins until some bin contains two balls. Each toss is independent, and each ball is equally likely to end up in any bin. What is the expected number of ball tosses?

Answer:

$$\mathrm{E}[X] = \sum_{i=2}^{b+1} i \cdot \frac{b!(i-1)}{b^i(b-i+1)!}$$

Exercise 5.4-3 ★

For the analysis of the birthday paradox, is it important that the birthdays be mutually independent, or is pairwise independence sufficient? Justify your answer.

Answer:

Pairwise independence is sufficient.

Exercise 5.4-4 ★

How many people should be invited to a party in order to make it likely that there are three people with the same birthday?

Answer:

$$\begin{aligned} \mathrm{E}[X] &= \sum_{i=1}^{m-2} \sum_{j=i+1}^{m-1} \sum_{k=j+1}^{m} \mathrm{E}[X_{ijk}] \\ &= \binom{m}{3} \frac{1}{365^2} \\ &\geq 1 \\ m &\geq 94 \end{aligned}$$

Exercise 5.4-5 ★

What is the probability that a k-string over a set of size n forms a k-permutation? How does this question relate to the birthday paradox?

Answer:

$$\frac{n!}{n^k(n-k)!}$$

Exercise 5.4-6 ★

Suppose that n balls are tossed into n bins, where each toss is independent and the ball is equally likely to end up in any bin. What is the expected number of empty bins? What is the expected number of bins with exactly one ball?

Answer:*

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 (3.14) ($\lim_{n\to\infty}(1 + x/n)^n = e^n$), 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

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

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

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

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

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

Exercise 5.4-7 ★

Sharpen the lower bound on streak length by showing that in n flips of a fair coin, the probability is less than $1/n$ that no streak longer than $\lg n - 2\lg\lg n$ consecutive heads occurs.

Answer:

$$\begin{aligned}\Pr\{A_{i,\lfloor \lg n - 2\lg\lg n\rfloor}\} &= \left(\frac{1}{2}\right)^{\lfloor \lg n - 2\lg\lg n\rfloor} \\ &\geq \frac{\lg^2 n}{n}\end{aligned}$$

$$\begin{aligned}\left(1 - \frac{\lg^2 n}{n}\right)^{\lfloor n/\lfloor \lg n - 2\lg\lg n\rfloor\rfloor} &\leq \left(1 - \frac{\lg^2 n}{n}\right)^{n/(\lg n - 2\lg\lg n) - 1} \\ &\leq e^{-(n/(\lg n - 2\lg\lg n) - 1)(\lg^2 n)/n} \\ &\leq e^{-\log n} \\ &= \frac{1}{n}\end{aligned}$$

Problems

Problem 5-1 Probabilistic counting

With a b-bit counter, we can ordinarily only count up to $2^b - 1$. With R. Morris's ***probabilistic counting***, we can count up to a much larger value at the expense of some loss of precision.

We let a counter value of i represent a count of n_i for $i = 0, 1, \ldots, 2^b - 1$, where the n_i form an increasing sequence of nonnegative values. We assume that the initial value of the counter is 0, representing a count of $n_0 = 0$. The $\mathrm{Increment}$ operation works on a counter containing the value i in a probabilistic manner. If $i = 2^b - 1$, then the operation reports an overflow error. Otherwise, the $\mathrm{Increment}$ operation increases the counter by 1 with probability $1/(n_{i+1} - n_i)$, and it leaves the counter unchanged with probability $1 - 1/(n_{i+1} - n_i)$.

If we select $n_i = i$ for all $i \geq 0$, then the counter is an ordinary one. More interesting situations arise if we select, say, $n_i = 2^{i-1}$ for $i \geq 0$ or $n_i = F_i$ (the ith Fibonacci number—see Section 3.2).

For this problem, assume that $n_{2^b - 1}$ is large enough that the probability of an overflow error is negligible.

a. Show that the expected value represented by the counter after n $\mathrm{Increment}$ operations have been performed is exactly n.

b. The analysis of the variance of the count represented by the counter depends on the sequence of the n_i. Let us consider a simple case: $n_i = 100i$ for all $i \geq 0$. Estimate the variance in the value represented by the register after n Increment operations have been performed.

Answer:*

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 $V_n = X_1 + X_2 + \ldots + X_n$. 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

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

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:

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

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

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

Thus,

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

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.29) ($\mathrm{Var}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$), $\mathrm{Var}[V_n] = \mathrm{Var}[X_1] + \mathrm{Var}[X_2] + \cdots + \mathrm{Var}[X_n]$.

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 $\mathrm{Increment}$ operation is 0, and with probability $1/100$, the value represented increases by 100. Thus, by equation (C.27) ($\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}^2[X]$),

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

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

Problem 5-2 Searching an unsorted array

This problem examines three algorithms for searching for a value x in an unsorted array A consisting of n elements.

Consider the following randomized strategy: pick a random index i into A. If $A[i] = x$, then we terminate; otherwise, we continue the search by picking a new random index into A. We continue picking random indices into A until we find an index j such that $A[j] = x$ or until we have checked every element of A. Note that we pick from the whole set of indices each time, so that we may examine a given element more than once.

a. Write pseudocode for a procedure $\mathrm{RandomSearch}$ to implement the strategy above. Be sure that your algorithm terminates when all indices into A have been picked.

b. Suppose that there is exactly one index i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we find x and $\mathrm{RandomSearch}$ terminates?

c. Generalizing your solution to part (b), suppose that there are $k \geq 1$ indices i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we find x and $\mathrm{RandomSearch}$ terminates? Your answer should be a function of n and k.

d. Suppose that there are no indices i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we have checked all elements of A and $\mathrm{RandomSearch}$ terminates?

Now consider a deterministic linear search algorithm, which we refer to as $\mathrm{DeterministicSearch}$. Specifically, the algorithm searches A for x in order, considering $A[1], A[2], A[3], \ldots, A[n]$ until either

it finds $A[i] = x$ or it reaches the end of the array. Assume that all possible permutations of the input array are equally likely.

e. Suppose that there is exactly one index i such that $A[i] = x$. What is the average-case running time of $\text{DeterministicSearch}$? What is the worstcase running time of $\text{DeterministicSearch}$?

f. Generalizing your solution to part (e), suppose that there are $k \geq 1$ indices i such that $A[i] = x$. What is the average-case running time of $\text{DeterministicSearch}$? What is the worst-case running time of $\text{DeterministicSearch}$? Your answer should be a function of n and k.

g. Suppose that there are no indices i such that $A[i] = x$. What is the average-case running time of $\text{DeterministicSearch}$? What is the worst-case running time of $\text{DeterministicSearch}$?

Finally, consider a randomized algorithm ScrambleSearch that works by first randomly permuting the input array and then running the deterministic linear search given above on the resulting permuted array.

h. Letting k be the number of indices i such that $A[i] = x$, give the worst-case and expected running times of ScrambleSearch for the cases in which $k = 0$ and $k = 1$. Generalize your solution to handle the case in which $k \geq 1$.

i. Which of the three searching algorithms would you use? Explain your answer.

Answer:

a.

(See `+srch/random_search.m`.)

b.

n

c.

$\frac{n}{k}$

d.

$n(\ln n + O(1))$

e.

$(n+1)/2$
n

f.

$$\begin{aligned} \mathrm{E}[X] &= \sum_{i=1}^{n} \mathrm{E}[X_i] \\ &= \sum_{A[i]=x} \mathrm{E}[X_i] + \sum_{A[i]\neq x} \mathrm{E}[X_i] \\ &= 1 + (n-k) \cdot \frac{1}{k+1} \\ &= \frac{n+1}{k+1} \\ \max(X) &= n - k + 1 \end{aligned}$$

g.

n

n

h.

All same with deterministic search.

i.

Deterministic search is better.