
5 Probabilistic Analysis and Randomized Algorithms

This chapter introduces probabilistic analysis and randomized algorithms. If you are unfamiliar with the basics of probability theory, you should read Appendix C, which reviews this material. We shall revisit probabilistic analysis and randomized algorithms several times throughout this book.

5.1 The hiring problem

Suppose that you need to hire a new office assistant. Your previous attempts at hiring have been unsuccessful, and you decide to use an employment agency. The employment agency sends you one candidate each day. You interview that person and then decide either to hire that person or not. You must pay the employment agency a small fee to interview an applicant. To actually hire an applicant is more costly, however, since you must fire your current office assistant and pay a substantial hiring fee to the employment agency. You are committed to having, at all times, the best possible person for the job. Therefore, you decide that, after interviewing each applicant, if that applicant is better qualified than the current office assistant, you will fire the current office assistant and hire the new applicant. You are willing to pay the resulting price of this strategy, but you wish to estimate what that price will be.

The procedure HIRE-ASSISTANT, given below, expresses this strategy for hiring in pseudocode. It assumes that the candidates for the office assistant job are numbered 1 through n . The procedure assumes that you are able to, after interviewing candidate i , determine whether candidate i is the best candidate you have seen so far. To initialize, the procedure creates a dummy candidate, numbered 0, who is less qualified than each of the other candidates.

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  $i$ 

```

The cost model for this problem differs from the model described in Chapter 2. We focus not on the running time of HIRE-ASSISTANT, but instead on the costs incurred by interviewing and hiring. On the surface, analyzing the cost of this algorithm may seem very different from analyzing the running time of, say, merge sort. The analytical techniques used, however, are identical whether we are analyzing cost or running time. In either case, we are counting the number of times certain basic operations are executed.

Interviewing has a low cost, say c_i , whereas hiring is expensive, costing c_h . Letting m be the number of people hired, the total cost associated with this algorithm is $O(c_i n + c_h m)$. No matter how many people we hire, we always interview n candidates and thus always incur the cost $c_i n$ associated with interviewing. We therefore concentrate on analyzing $c_h m$, the hiring cost. This quantity varies with each run of the algorithm.

This scenario serves as a model for a common computational paradigm. We often need to find the maximum or minimum value in a sequence by examining each element of the sequence and maintaining a current “winner.” The hiring problem models how often we update our notion of which element is currently winning.

Worst-case analysis

In the worst case, we actually hire every candidate that we interview. This situation occurs if the candidates come in strictly increasing order of quality, in which case we hire n times, for a total hiring cost of $O(c_h n)$.

Of course, the candidates do not always come in increasing order of quality. In fact, we have no idea about the order in which they arrive, nor do we have any control over this order. Therefore, it is natural to ask what we expect to happen in a typical or average case.

Probabilistic analysis

Probabilistic analysis is the use of probability in the analysis of problems. Most commonly, we use probabilistic analysis to analyze the running time of an algorithm. Sometimes we use it to analyze other quantities, such as the hiring cost

in procedure HIRE-ASSISTANT. 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*.

We must be very careful in deciding on the distribution of inputs. For some problems, we may reasonably assume something about the set of all possible inputs, and then we can use probabilistic analysis as a technique for designing an efficient algorithm and as a means for gaining insight into a problem. For other problems, we cannot describe a reasonable input distribution, and in these cases we cannot use probabilistic analysis.

For the hiring problem, we can assume that the applicants come in a random order. What does that mean for this problem? We assume that we can compare any two candidates and decide which one is better qualified; that is, there is a total order on the candidates. (See Appendix B for the definition of a total order.) Thus, we can rank each candidate with a unique number from 1 through n , using $rank(i)$ to denote the rank of applicant i , and adopt the convention that a higher rank corresponds to a better qualified applicant. The ordered list $\langle rank(1), rank(2), \dots, rank(n) \rangle$ is a permutation of the list $\langle 1, 2, \dots, n \rangle$. Saying that the applicants come in a random order is equivalent to saying that this list of ranks is equally likely to be any one of the $n!$ permutations of the numbers 1 through n . Alternatively, we say that the ranks form a *uniform random permutation*; that is, each of the possible $n!$ permutations appears with equal probability.

Section 5.2 contains a probabilistic analysis of the hiring problem.

Randomized algorithms

In order to use probabilistic analysis, we need to know something about the distribution of the inputs. In many cases, we know very little about the input distribution. Even if we do know something about the distribution, we may not be able to model this knowledge computationally. Yet we often can use probability and randomness as a tool for algorithm design and analysis, by making the behavior of part of the algorithm random.

In the hiring problem, it may seem as if the candidates are being presented to us in a random order, but we have no way of knowing whether or not they really are. Thus, in order to develop a randomized algorithm for the hiring problem, we must have greater control over the order in which we interview the candidates. We will, therefore, change the model slightly. We say that the employment agency has n candidates, and they send us a list of the candidates in advance. On each day, we choose, randomly, which candidate to interview. Although we know nothing about

the candidates (besides their names), we have made a significant change. Instead of relying on a guess that the candidates come to us in a random order, we have instead gained control of the process and enforced a random order.

More generally, we call an algorithm *randomized* if its behavior is determined not only by its input but also by values produced by a *random-number generator*. We shall assume that we have at our disposal a random-number generator `RANDOM`. A call to `RANDOM(a, b)` returns an integer between a and b , inclusive, with each such integer being equally likely. For example, `RANDOM(0, 1)` produces 0 with probability $1/2$, and it produces 1 with probability $1/2$. A call to `RANDOM(3, 7)` returns either 3, 4, 5, 6, or 7, each with probability $1/5$. Each integer returned by `RANDOM` is independent of the integers returned on previous calls. You may imagine `RANDOM` as rolling a $(b - a + 1)$ -sided die to obtain its output. (In practice, most programming environments offer a *pseudorandom-number generator*: a deterministic algorithm returning numbers that “look” statistically random.)

When analyzing the running time of a randomized algorithm, we take the expectation of the running time over the distribution of values returned by the random number generator. We distinguish these algorithms from those in which the input is random by referring to the running time of a randomized algorithm as an *expected running time*. In general, we discuss the average-case running time when the probability distribution is over the inputs to the algorithm, and we discuss the expected running time when the algorithm itself makes random choices.

Exercises

5.1-1

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

5.1-2 ★

Describe an implementation of the procedure `RANDOM(a, b)` that only makes calls to `RANDOM(0, 1)`. What is the expected running time of your procedure, as a function of a and b ?

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 `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 ?

5.2 Indicator random variables

In order to analyze many algorithms, including the hiring problem, we use indicator random variables. Indicator random variables provide a convenient method for converting between probabilities and expectations. Suppose we are given a sample space S and an event A . Then the *indicator random variable* $I\{A\}$ associated with event A is defined as

$$I\{A\} = \begin{cases} 1 & \text{if } A \text{ occurs,} \\ 0 & \text{if } A \text{ does not occur.} \end{cases} \quad (5.1)$$

As a simple example, let us determine the expected number of heads that we obtain when flipping a fair coin. Our sample space is $S = \{H, T\}$, with $\Pr\{H\} = \Pr\{T\} = 1/2$. We can then define an indicator random variable X_H , associated with the coin coming up heads, which is the event H . This variable counts the number of heads obtained in this flip, and it is 1 if the coin comes up heads and 0 otherwise. We write

$$\begin{aligned} X_H &= I\{H\} \\ &= \begin{cases} 1 & \text{if } H \text{ occurs,} \\ 0 & \text{if } T \text{ occurs.} \end{cases} \end{aligned}$$

The expected number of heads obtained in one flip of the coin is simply the expected value of our indicator variable X_H :

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

Thus the expected number of heads obtained by one flip of a fair coin is $1/2$. As the following lemma shows, the expected value of an indicator random variable associated with an event A is equal to the probability that A occurs.

Lemma 5.1

Given a sample space S and an event A in the sample space S , let $X_A = I\{A\}$. Then $E[X_A] = \Pr\{A\}$.

Proof By the definition of an indicator random variable from equation (5.1) and the definition of expected value, we have

$$\begin{aligned} E[X_A] &= E[I\{A\}] \\ &= 1 \cdot \Pr\{A\} + 0 \cdot \Pr\{\bar{A}\} \\ &= \Pr\{A\} , \end{aligned}$$

where \bar{A} denotes $S - A$, the complement of A . ■

Although indicator random variables may seem cumbersome for an application such as counting the expected number of heads on a flip of a single coin, they are useful for analyzing situations in which we perform repeated random trials. For example, indicator random variables give us a simple way to arrive at the result of equation (C.37). In this equation, we compute the number of heads in n coin flips by considering separately the probability of obtaining 0 heads, 1 head, 2 heads, etc. The simpler method proposed in equation (C.38) instead uses indicator random variables implicitly. Making this argument more explicit, we let X_i be the indicator random variable associated with the event in which the i th flip comes up heads: $X_i = I\{\text{the } i\text{th flip results in the event } H\}$. Let X be the random variable denoting the total number of heads in the n coin flips, so that

$$X = \sum_{i=1}^n X_i .$$

We wish to compute the expected number of heads, and so we take the expectation of both sides of the above equation to obtain

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

The above equation gives the expectation of the sum of n indicator random variables. By Lemma 5.1, we can easily compute the expectation of each of the random variables. By equation (C.21)—linearity of expectation—it is easy to compute the expectation of the sum: it equals the sum of the expectations of the n random variables. Linearity of expectation makes the use of indicator random variables a powerful analytical technique; it applies even when there is dependence among the random variables. We now can easily compute the expected number of heads:

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

Thus, compared to the method used in equation (C.37), indicator random variables greatly simplify the calculation. We shall use indicator random variables throughout this book.

Analysis of the hiring problem using indicator random variables

Returning to the hiring problem, we now wish to compute the expected number of times that we hire a new office assistant. In order to use a probabilistic analysis, we assume that the candidates arrive in a random order, as discussed in the previous section. (We shall see in Section 5.3 how to remove this assumption.) Let X be the random variable whose value equals the number of times we hire a new office assistant. We could then apply the definition of expected value from equation (C.20) to obtain

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

but this calculation would be cumbersome. We shall instead use indicator random variables to greatly simplify the calculation.

To use indicator random variables, instead of computing $E[X]$ by defining one variable associated with the number of times we hire a new office assistant, we define n variables related to whether or not each particular candidate is hired. In particular, we let X_i be the indicator random variable associated with the event in which the i th candidate is hired. Thus,

$$\begin{aligned}
X_i &= I\{\text{candidate } i \text{ is hired}\} \\
&= \begin{cases} 1 & \text{if candidate } i \text{ is hired,} \\ 0 & \text{if candidate } i \text{ is not hired,} \end{cases}
\end{aligned}$$

and

$$X = X_1 + X_2 + \cdots + X_n. \tag{5.2}$$

By Lemma 5.1, we have that

$$E[X_i] = \Pr\{\text{candidate } i \text{ is hired}\} ,$$

and we must therefore compute the probability that lines 5–6 of HIRE-ASSISTANT are executed.

Candidate i is hired, in line 6, exactly when candidate i is better than each of candidates 1 through $i - 1$. Because we have assumed that the candidates arrive in a random order, the first i candidates have appeared in a random order. Any one of these first i candidates is equally likely to be the best-qualified so far. Candidate i has a probability of $1/i$ of being better qualified than candidates 1 through $i - 1$ and thus a probability of $1/i$ of being hired. By Lemma 5.1, we conclude that

$$E[X_i] = 1/i . \quad (5.3)$$

Now we can compute $E[X]$:

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^n X_i\right] \quad (\text{by equation (5.2)}) \\ &= \sum_{i=1}^n E[X_i] \quad (\text{by linearity of expectation}) \\ &= \sum_{i=1}^n 1/i \quad (\text{by equation (5.3)}) \\ &= \ln n + O(1) \quad (\text{by equation (A.7)}) . \end{aligned} \quad (5.5)$$

Even though we interview n people, we actually hire only approximately $\ln n$ of them, on average. We summarize this result in the following lemma.

Lemma 5.2

Assuming that the candidates are presented in a random order, algorithm HIRE-ASSISTANT has an average-case total hiring cost of $O(c_h \ln n)$.

Proof The bound follows immediately from our definition of the hiring cost and equation (5.5), which shows that the expected number of hires is approximately $\ln n$. ■

The average-case hiring cost is a significant improvement over the worst-case hiring cost of $O(c_h n)$.