of S, which are all initially uncovered. Let k be the least index such that $u_k = 0$, so that every element in S is covered by at least one of the sets $S_1, S_2, \ldots, S_k$ and some element in S is uncovered by $S_1 \cup S_2 \cup \cdots \cup S_{k-1}$. Then, $u_{i-1} \geq u_i$, and $u_{i-1} - u_i$ elements of S are covered for the first time by S_i, for $i = 1, 2, \ldots, k$. Thus,

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{1}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|} .$$

Observe that

$$\begin{aligned} |S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})| &\geq |S - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})| \\ &= u_{i-1} , \end{aligned}$$

because the greedy choice of S_i guarantees that S cannot cover more new elements than S_i does (otherwise, the algorithm would have chosen S instead of S_i). Consequently, we obtain

$$\sum_{x \in S} c_x \leq \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}} .$$

We now bound this quantity as follows:

$$\begin{aligned} \sum_{x \in S} c_x &\leq \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}} \\ &= \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{u_{i-1}} \\ &\leq \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{j} && \text{(because } j \leq u_{i-1}\text{)} \\ &= \sum_{i=1}^{k} \left(\sum_{j=1}^{u_{i-1}} \frac{1}{j} - \sum_{j=1}^{u_i} \frac{1}{j} \right) \\ &= \sum_{i=1}^{k} (H(u_{i-1}) - H(u_i)) \\ &= H(u_0) - H(u_k) && \text{(because the sum telescopes)} \\ &= H(u_0) - H(0) \\ &= H(u_0) && \text{(because } H(0) = 0\text{)} \\ &= H(|S|) , \end{aligned}$$

which completes the proof of inequality (35.12). ■

Corollary 35.5
GREEDY-SET-COVER is a polynomial-time $(\ln|X|+1)$-approximation algorithm.

Proof Use inequality (A.14) and Theorem 35.4. ■

In some applications, $\max\{|S| : S \in \mathcal{F}\}$ is a small constant, and so the solution returned by GREEDY-SET-COVER is at most a small constant times larger than optimal. One such application occurs when this heuristic finds an approximate vertex cover for a graph whose vertices have degree at most 3. In this case, the solution found by GREEDY-SET-COVER is not more than $H(3) = 11/6$ times as large as an optimal solution, a performance guarantee that is slightly better than that of APPROX-VERTEX-COVER.

Exercises

35.3-1
Consider each of the following words as a set of letters: {`arid`, `dash`, `drain`, `heard`, `lost`, `nose`, `shun`, `slate`, `snare`, `thread`}. Show which set cover GREEDY-SET-COVER produces when we break ties in favor of the word that appears first in the dictionary.

35.3-2
Show that the decision version of the set-covering problem is NP-complete by reducing it from the vertex-cover problem.

35.3-3
Show how to implement GREEDY-SET-COVER in such a way that it runs in time $O\left(\sum_{S\in\mathcal{F}}|S|\right)$.

35.3-4
Show that the following weaker form of Theorem 35.4 is trivially true:

$$|\mathcal{C}| \le |\mathcal{C}^*| \max\{|S| : S \in \mathcal{F}\} .$$

35.3-5
GREEDY-SET-COVER can return a number of different solutions, depending on how we break ties in line 4. Give a procedure BAD-SET-COVER-INSTANCE(n) that returns an n-element instance of the set-covering problem for which, depending on how we break ties in line 4, GREEDY-SET-COVER can return a number of different solutions that is exponential in n.

35.4 Randomization and linear programming

In this section, we study two useful techniques for designing approximation algorithms: randomization and linear programming. We shall give a simple randomized algorithm for an optimization version of 3-CNF satisfiability, and then we shall use linear programming to help design an approximation algorithm for a weighted version of the vertex-cover problem. This section only scratches the surface of these two powerful techniques. The chapter notes give references for further study of these areas.

A randomized approximation algorithm for MAX-3-CNF satisfiability

Just as some randomized algorithms compute exact solutions, some randomized algorithms compute approximate solutions. We say that a randomized algorithm for a problem has an ***approximation ratio*** of $\rho(n)$ if, for any input of size n, the *expected* cost C of the solution produced by the randomized algorithm is within a factor of $\rho(n)$ of the cost C^* of an optimal solution:

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \le \rho(n) \; . \tag{35.13}$$

We call a randomized algorithm that achieves an approximation ratio of $\rho(n)$ a ***randomized $\rho(n)$-approximation algorithm.*** In other words, a randomized approximation algorithm is like a deterministic approximation algorithm, except that the approximation ratio is for an expected cost.

A particular instance of 3-CNF satisfiability, as defined in Section 34.4, may or may not be satisfiable. In order to be satisfiable, there must exist an assignment of the variables so that every clause evaluates to 1. If an instance is not satisfiable, we may want to compute how "close" to satisfiable it is, that is, we may wish to find an assignment of the variables that satisfies as many clauses as possible. We call the resulting maximization problem ***MAX-3-CNF satisfiability***. The input to MAX-3-CNF satisfiability is the same as for 3-CNF satisfiability, and the goal is to return an assignment of the variables that maximizes the number of clauses evaluating to 1. We now show that randomly setting each variable to 1 with probability $1/2$ and to 0 with probability $1/2$ yields a randomized $8/7$-approximation algorithm. According to the definition of 3-CNF satisfiability from Section 34.4, we require each clause to consist of exactly three distinct literals. We further assume that no clause contains both a variable and its negation. (Exercise 35.4-1 asks you to remove this last assumption.)

Theorem 35.6
Given an instance of MAX-3-CNF satisfiability with n variables $x_1, x_2, \ldots, x_n$ and m clauses, the randomized algorithm that independently sets each variable to 1 with probability $1/2$ and to 0 with probability $1/2$ is a randomized $8/7$-approximation algorithm.

Proof Suppose that we have independently set each variable to 1 with probability $1/2$ and to 0 with probability $1/2$. For $i = 1, 2, \ldots, m$, we define the indicator random variable

$$Y_i = \mathrm{I}\{\text{clause } i \text{ is satisfied}\} ,$$

so that $Y_i = 1$ as long as we have set at least one of the literals in the ith clause to 1. Since no literal appears more than once in the same clause, and since we have assumed that no variable and its negation appear in the same clause, the settings of the three literals in each clause are independent. A clause is not satisfied only if all three of its literals are set to 0, and so $\Pr\{\text{clause } i \text{ is not satisfied}\} = (1/2)^3 = 1/8$. Thus, we have $\Pr\{\text{clause } i \text{ is satisfied}\} = 1 - 1/8 = 7/8$, and by Lemma 5.1, we have $\mathrm{E}[Y_i] = 7/8$. Let Y be the number of satisfied clauses overall, so that $Y = Y_1 + Y_2 + \cdots + Y_m$. Then, we have

$$\begin{aligned}
\mathrm{E}[Y] &= \mathrm{E}\left[\sum_{i=1}^{m} Y_i\right] \\
&= \sum_{i=1}^{m} \mathrm{E}[Y_i] \qquad \text{(by linearity of expectation)} \\
&= \sum_{i=1}^{m} 7/8 \\
&= 7m/8 .
\end{aligned}$$

Clearly, m is an upper bound on the number of satisfied clauses, and hence the approximation ratio is at most $m/(7m/8) = 8/7$. ■

Approximating weighted vertex cover using linear programming

In the ***minimum-weight vertex-cover problem***, we are given an undirected graph $G = (V, E)$ in which each vertex $v \in V$ has an associated positive weight $w(v)$. For any vertex cover $V' \subseteq V$, we define the weight of the vertex cover $w(V') = \sum_{v \in V'} w(v)$. The goal is to find a vertex cover of minimum weight.

We cannot apply the algorithm used for unweighted vertex cover, nor can we use a random solution; both methods may return solutions that are far from optimal. We shall, however, compute a lower bound on the weight of the minimum-weight

vertex cover, by using a linear program. We shall then "round" this solution and use it to obtain a vertex cover.

Suppose that we associate a variable $x(v)$ with each vertex $v \in V$, and let us require that $x(v)$ equals either 0 or 1 for each $v \in V$. We put v into the vertex cover if and only if $x(v) = 1$. Then, we can write the constraint that for any edge (u, v), at least one of u and v must be in the vertex cover as $x(u) + x(v) \geq 1$. This view gives rise to the following ***0-1 integer program*** for finding a minimum-weight vertex cover:

$$\text{minimize} \quad \sum_{v \in V} w(v)\, x(v) \tag{35.14}$$

subject to

$$\begin{aligned} x(u) + x(v) &\geq 1 && \text{for each } (u, v) \in E && (35.15)\\ x(v) &\in \{0, 1\} && \text{for each } v \in V\ . && (35.16) \end{aligned}$$

In the special case in which all the weights $w(v)$ are equal to 1, this formulation is the optimization version of the NP-hard vertex-cover problem. Suppose, however, that we remove the constraint that $x(v) \in \{0, 1\}$ and replace it by $0 \leq x(v) \leq 1$. We then obtain the following linear program, which is known as the ***linear-programming relaxation***:

$$\text{minimize} \quad \sum_{v \in V} w(v)\, x(v) \tag{35.17}$$

subject to

$$\begin{aligned} x(u) + x(v) &\geq 1 && \text{for each } (u, v) \in E && (35.18)\\ x(v) &\leq 1 && \text{for each } v \in V && (35.19)\\ x(v) &\geq 0 && \text{for each } v \in V\ . && (35.20) \end{aligned}$$

Any feasible solution to the 0-1 integer program in lines (35.14)–(35.16) is also a feasible solution to the linear program in lines (35.17)–(35.20). Therefore, the value of an optimal solution to the linear program gives a lower bound on the value of an optimal solution to the 0-1 integer program, and hence a lower bound on the optimal weight in the minimum-weight vertex-cover problem.

The following procedure uses the solution to the linear-programming relaxation to construct an approximate solution to the minimum-weight vertex-cover problem:

APPROX-MIN-WEIGHT-VC(G, w)

```
1  C = ∅
2  compute x̄, an optimal solution to the linear program in lines (35.17)–(35.20)
3  for each v ∈ V
4      if x̄(v) ≥ 1/2
5          C = C ∪ {v}
6  return C
```

The APPROX-MIN-WEIGHT-VC procedure works as follows. Line 1 initializes the vertex cover to be empty. Line 2 formulates the linear program in lines (35.17)–(35.20) and then solves this linear program. An optimal solution gives each vertex v an associated value $\bar{x}(v)$, where $0 \le \bar{x}(v) \le 1$. We use this value to guide the choice of which vertices to add to the vertex cover C in lines 3–5. If $\bar{x}(v) \ge 1/2$, we add v to C; otherwise we do not. In effect, we are "rounding" each fractional variable in the solution to the linear program to 0 or 1 in order to obtain a solution to the 0-1 integer program in lines (35.14)–(35.16). Finally, line 6 returns the vertex cover C.

Theorem 35.7
Algorithm APPROX-MIN-WEIGHT-VC is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.

Proof Because there is a polynomial-time algorithm to solve the linear program in line 2, and because the **for** loop of lines 3–5 runs in polynomial time, APPROX-MIN-WEIGHT-VC is a polynomial-time algorithm.

Now we show that APPROX-MIN-WEIGHT-VC is a 2-approximation algorithm. Let C^* be an optimal solution to the minimum-weight vertex-cover problem, and let z^* be the value of an optimal solution to the linear program in lines (35.17)–(35.20). Since an optimal vertex cover is a feasible solution to the linear program, z^* must be a lower bound on $w(C^*)$, that is,

$$z^* \le w(C^*) . \tag{35.21}$$

Next, we claim that by rounding the fractional values of the variables $\bar{x}(v)$, we produce a set C that is a vertex cover and satisfies $w(C) \le 2z^*$. To see that C is a vertex cover, consider any edge $(u, v) \in E$. By constraint (35.18), we know that $x(u) + x(v) \ge 1$, which implies that at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2$. Therefore, at least one of u and v is included in the vertex cover, and so every edge is covered.

Now, we consider the weight of the cover. We have

$$
\begin{aligned}
z^* &= \sum_{v \in V} w(v)\,\bar{x}(v) \\
&\geq \sum_{v \in V : \bar{x}(v) \geq 1/2} w(v)\,\bar{x}(v) \\
&\geq \sum_{v \in V : \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} \\
&= \sum_{v \in C} w(v) \cdot \frac{1}{2} \\
&= \frac{1}{2} \sum_{v \in C} w(v) \\
&= \frac{1}{2} w(C) \,. \qquad (35.22)
\end{aligned}
$$

Combining inequalities (35.21) and (35.22) gives

$$w(C) \leq 2z^* \leq 2w(C^*) \,,$$

and hence APPROX-MIN-WEIGHT-VC is a 2-approximation algorithm. ■

Exercises

35.4-1

Show that even if we allow a clause to contain both a variable and its negation, randomly setting each variable to 1 with probability $1/2$ and to 0 with probability $1/2$ still yields a randomized $8/7$-approximation algorithm.

35.4-2

The ***MAX-CNF satisfiability problem*** is like the MAX-3-CNF satisfiability problem, except that it does not restrict each clause to have exactly 3 literals. Give a randomized 2-approximation algorithm for the MAX-CNF satisfiability problem.

35.4-3

In the MAX-CUT problem, we are given an unweighted undirected graph $G = (V, E)$. We define a cut $(S, V - S)$ as in Chapter 23 and the ***weight*** of a cut as the number of edges crossing the cut. The goal is to find a cut of maximum weight. Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V - S$ with probability $1/2$. Show that this algorithm is a randomized 2-approximation algorithm.

35.4-4

Show that the constraints in line (35.19) are redundant in the sense that if we remove them from the linear program in lines (35.17)–(35.20), any optimal solution to the resulting linear program must satisfy $x(v) \leq 1$ for each $v \in V$.

35.5 The subset-sum problem

Recall from Section 34.5.5 that an instance of the subset-sum problem is a pair (S, t), where S is a set $\{x_1, x_2, \ldots, x_n\}$ of positive integers and t is a positive integer. This decision problem asks whether there exists a subset of S that adds up exactly to the target value t. As we saw in Section 34.5.5, this problem is NP-complete.

The optimization problem associated with this decision problem arises in practical applications. In the optimization problem, we wish to find a subset of $\{x_1, x_2, \ldots, x_n\}$ whose sum is as large as possible but not larger than t. For example, we may have a truck that can carry no more than t pounds, and n different boxes to ship, the ith of which weighs x_i pounds. We wish to fill the truck with as heavy a load as possible without exceeding the given weight limit.

In this section, we present an exponential-time algorithm that computes the optimal value for this optimization problem, and then we show how to modify the algorithm so that it becomes a fully polynomial-time approximation scheme. (Recall that a fully polynomial-time approximation scheme has a running time that is polynomial in $1/\epsilon$ as well as in the size of the input.)

An exponential-time exact algorithm

Suppose that we computed, for each subset S' of S, the sum of the elements in S', and then we selected, among the subsets whose sum does not exceed t, the one whose sum was closest to t. Clearly this algorithm would return the optimal solution, but it could take exponential time. To implement this algorithm, we could use an iterative procedure that, in iteration i, computes the sums of all subsets of $\{x_1, x_2, \ldots, x_i\}$, using as a starting point the sums of all subsets of $\{x_1, x_2, \ldots, x_{i-1}\}$. In doing so, we would realize that once a particular subset S' had a sum exceeding t, there would be no reason to maintain it, since no superset of S' could be the optimal solution. We now give an implementation of this strategy.

The procedure EXACT-SUBSET-SUM takes an input set $S = \{x_1, x_2, \ldots, x_n\}$ and a target value t; we'll see its pseudocode in a moment. This procedure it-

eratively computes L_i, the list of sums of all subsets of $\{x_1, \ldots, x_i\}$ that do not exceed t, and then it returns the maximum value in L_n.

If L is a list of positive integers and x is another positive integer, then we let $L + x$ denote the list of integers derived from L by increasing each element of L by x. For example, if $L = \langle 1, 2, 3, 5, 9 \rangle$, then $L + 2 = \langle 3, 4, 5, 7, 11 \rangle$. We also use this notation for sets, so that

$$S + x = \{s + x : s \in S\} .$$

We also use an auxiliary procedure MERGE-LISTS(L, L'), which returns the sorted list that is the merge of its two sorted input lists L and L' with duplicate values removed. Like the MERGE procedure we used in merge sort (Section 2.3.1), MERGE-LISTS runs in time $O(|L| + |L'|)$. We omit the pseudocode for MERGE-LISTS.

```
EXACT-SUBSET-SUM(S, t)
1  n = |S|
2  L_0 = ⟨0⟩
3  for i = 1 to n
4       L_i = MERGE-LISTS(L_{i-1}, L_{i-1} + x_i)
5       remove from L_i every element that is greater than t
6  return the largest element in L_n
```

To see how EXACT-SUBSET-SUM works, let P_i denote the set of all values obtained by selecting a (possibly empty) subset of $\{x_1, x_2, \ldots, x_i\}$ and summing its members. For example, if $S = \{1, 4, 5\}$, then

$$\begin{aligned} P_1 &= \{0, 1\} , \\ P_2 &= \{0, 1, 4, 5\} , \\ P_3 &= \{0, 1, 4, 5, 6, 9, 10\} . \end{aligned}$$

Given the identity

$$P_i = P_{i-1} \cup (P_{i-1} + x_i) , \tag{35.23}$$

we can prove by induction on i (see Exercise 35.5-1) that the list L_i is a sorted list containing every element of P_i whose value is not more than t. Since the length of L_i can be as much as 2^i, EXACT-SUBSET-SUM is an exponential-time algorithm in general, although it is a polynomial-time algorithm in the special cases in which t is polynomial in $|S|$ or all the numbers in S are bounded by a polynomial in $|S|$.

A fully polynomial-time approximation scheme

We can derive a fully polynomial-time approximation scheme for the subset-sum problem by "trimming" each list L_i after it is created. The idea behind trimming is

that if two values in L are close to each other, then since we want just an approximate solution, we do not need to maintain both of them explicitly. More precisely, we use a trimming parameter δ such that $0 < \delta < 1$. When we ***trim*** a list L by δ, we remove as many elements from L as possible, in such a way that if L' is the result of trimming L, then for every element y that was removed from L, there is an element z still in L' that approximates y, that is,

$$\frac{y}{1+\delta} \le z \le y \; . \tag{35.24}$$

We can think of such a z as "representing" y in the new list L'. Each removed element y is represented by a remaining element z satisfying inequality (35.24). For example, if $\delta = 0.1$ and

$$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle \; ,$$

then we can trim L to obtain

$$L' = \langle 10, 12, 15, 20, 23, 29 \rangle \; ,$$

where the deleted value 11 is represented by 10, the deleted values 21 and 22 are represented by 20, and the deleted value 24 is represented by 23. Because every element of the trimmed version of the list is also an element of the original version of the list, trimming can dramatically decrease the number of elements kept while keeping a close (and slightly smaller) representative value in the list for each deleted element.

The following procedure trims list $L = \langle y_1, y_2, \ldots, y_m \rangle$ in time $\Theta(m)$, given L and δ, and assuming that L is sorted into monotonically increasing order. The output of the procedure is a trimmed, sorted list.

```
TRIM(L, δ)
1  let m be the length of L
2  L' = ⟨y_1⟩
3  last = y_1
4  for i = 2 to m
5      if y_i > last · (1 + δ)        // y_i ≥ last because L is sorted
6          append y_i onto the end of L'
7          last = y_i
8  return L'
```

The procedure scans the elements of L in monotonically increasing order. A number is appended onto the returned list L' only if it is the first element of L or if it cannot be represented by the most recent number placed into L'.

Given the procedure TRIM, we can construct our approximation scheme as follows. This procedure takes as input a set $S = \{x_1, x_2, \ldots, x_n\}$ of n integers (in arbitrary order), a target integer t, and an "approximation parameter" ϵ, where

$$0 < \epsilon < 1 \,. \tag{35.25}$$

It returns a value z whose value is within a $1 + \epsilon$ factor of the optimal solution.

APPROX-SUBSET-SUM(S, t, ϵ)

```
1  n = |S|
2  L_0 = ⟨0⟩
3  for i = 1 to n
4       L_i = MERGE-LISTS(L_{i-1}, L_{i-1} + x_i)
5       L_i = TRIM(L_i, ε/2n)
6       remove from L_i every element that is greater than t
7  let z* be the largest value in L_n
8  return z*
```

Line 2 initializes the list L_0 to be the list containing just the element 0. The **for** loop in lines 3–6 computes L_i as a sorted list containing a suitably trimmed version of the set P_i, with all elements larger than t removed. Since we create L_i from L_{i-1}, we must ensure that the repeated trimming doesn't introduce too much compounded inaccuracy. In a moment, we shall see that APPROX-SUBSET-SUM returns a correct approximation if one exists.

As an example, suppose we have the instance

$$S = \langle 104, 102, 201, 101 \rangle$$

with $t = 308$ and $\epsilon = 0.40$. The trimming parameter δ is $\epsilon/8 = 0.05$. APPROX-SUBSET-SUM computes the following values on the indicated lines:

line 2: $L_0 = \langle 0 \rangle$,

line 4: $L_1 = \langle 0, 104 \rangle$,
line 5: $L_1 = \langle 0, 104 \rangle$,
line 6: $L_1 = \langle 0, 104 \rangle$,

line 4: $L_2 = \langle 0, 102, 104, 206 \rangle$,
line 5: $L_2 = \langle 0, 102, 206 \rangle$,
line 6: $L_2 = \langle 0, 102, 206 \rangle$,

line 4: $L_3 = \langle 0, 102, 201, 206, 303, 407 \rangle$,
line 5: $L_3 = \langle 0, 102, 201, 303, 407 \rangle$,
line 6: $L_3 = \langle 0, 102, 201, 303 \rangle$,

line 4: $L_4 = \langle 0, 101, 102, 201, 203, 302, 303, 404 \rangle$,
line 5: $L_4 = \langle 0, 101, 201, 302, 404 \rangle$,
line 6: $L_4 = \langle 0, 101, 201, 302 \rangle$.

The algorithm returns $z^* = 302$ as its answer, which is well within $\epsilon = 40\%$ of the optimal answer $307 = 104 + 102 + 101$; in fact, it is within 2%.

Theorem 35.8
APPROX-SUBSET-SUM is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof The operations of trimming L_i in line 5 and removing from L_i every element that is greater than t maintain the property that every element of L_i is also a member of P_i. Therefore, the value z^* returned in line 8 is indeed the sum of some subset of S. Let $y^* \in P_n$ denote an optimal solution to the subset-sum problem. Then, from line 6, we know that $z^* \le y^*$. By inequality (35.1), we need to show that $y^*/z^* \le 1 + \epsilon$. We must also show that the running time of this algorithm is polynomial in both $1/\epsilon$ and the size of the input.

As Exercise 35.5-2 asks you to show, for every element y in P_i that is at most t, there exists an element $z \in L_i$ such that

$$\frac{y}{(1+\epsilon/2n)^i} \le z \le y \; . \tag{35.26}$$

Inequality (35.26) must hold for $y^* \in P_n$, and therefore there exists an element $z \in L_n$ such that

$$\frac{y^*}{(1+\epsilon/2n)^n} \le z \le y^* \; ,$$

and thus

$$\frac{y^*}{z} \le \left(1 + \frac{\epsilon}{2n}\right)^n \; . \tag{35.27}$$

Since there exists an element $z \in L_n$ fulfilling inequality (35.27), the inequality must hold for z^*, which is the largest value in L_n; that is,

$$\frac{y^*}{z^*} \le \left(1 + \frac{\epsilon}{2n}\right)^n \; . \tag{35.28}$$

Now, we show that $y^*/z^* \le 1 + \epsilon$. We do so by showing that $(1 + \epsilon/2n)^n \le 1 + \epsilon$. By equation (3.14), we have $\lim_{n\to\infty}(1 + \epsilon/2n)^n = e^{\epsilon/2}$. Exercise 35.5-3 asks you to show that

$$\frac{d}{dn}\left(1 + \frac{\epsilon}{2n}\right)^n > 0 \; . \tag{35.29}$$

Therefore, the function $(1 + \epsilon/2n)^n$ increases with n as it approaches its limit of $e^{\epsilon/2}$, and we have

$$\begin{aligned}\left(1+\frac{\epsilon}{2n}\right)^n &\le e^{\epsilon/2} \\ &\le 1+\epsilon/2+(\epsilon/2)^2 \quad \text{(by inequality (3.13))} \\ &\le 1+\epsilon \quad \text{(by inequality (35.25))} \ . \end{aligned} \tag{35.30}$$

Combining inequalities (35.28) and (35.30) completes the analysis of the approximation ratio.

To show that APPROX-SUBSET-SUM is a fully polynomial-time approximation scheme, we derive a bound on the length of L_i. After trimming, successive elements z and z' of L_i must have the relationship $z'/z > 1+\epsilon/2n$. That is, they must differ by a factor of at least $1+\epsilon/2n$. Each list, therefore, contains the value 0, possibly the value 1, and up to $\lfloor \log_{1+\epsilon/2n} t \rfloor$ additional values. The number of elements in each list L_i is at most

$$\begin{aligned}\log_{1+\epsilon/2n} t + 2 &= \frac{\ln t}{\ln(1+\epsilon/2n)} + 2 \\ &\le \frac{2n(1+\epsilon/2n)\ln t}{\epsilon} + 2 \quad \text{(by inequality (3.17))} \\ &< \frac{3n \ln t}{\epsilon} + 2 \quad \text{(by inequality (35.25))} \ . \end{aligned}$$

This bound is polynomial in the size of the input—which is the number of bits $\lg t$ needed to represent t plus the number of bits needed to represent the set S, which is in turn polynomial in n—and in $1/\epsilon$. Since the running time of APPROX-SUBSET-SUM is polynomial in the lengths of the L_i, we conclude that APPROX-SUBSET-SUM is a fully polynomial-time approximation scheme. ■

Exercises

35.5-1

Prove equation (35.23). Then show that after executing line 5 of EXACT-SUBSET-SUM, L_i is a sorted list containing every element of P_i whose value is not more than t.

35.5-2

Using induction on i, prove inequality (35.26).

35.5-3

Prove inequality (35.29).

35.5-4

How would you modify the approximation scheme presented in this section to find a good approximation to the smallest value not less than t that is a sum of some subset of the given input list?

35.5-5

Modify the APPROX-SUBSET-SUM procedure to also return the subset of S that sums to the value z^*.

Problems

35-1 Bin packing

Suppose that we are given a set of n objects, where the size s_i of the ith object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1.

a. Prove that the problem of determining the minimum number of bins required is NP-hard. (*Hint:* Reduce from the subset-sum problem.)

The ***first-fit*** heuristic takes each object in turn and places it into the first bin that can accommodate it. Let $S = \sum_{i=1}^{n} s_i$.

b. Argue that the optimal number of bins required is at least $\lceil S \rceil$.

c. Argue that the first-fit heuristic leaves at most one bin less than half full.

d. Prove that the number of bins used by the first-fit heuristic is never more than $\lceil 2S \rceil$.

e. Prove an approximation ratio of 2 for the first-fit heuristic.

f. Give an efficient implementation of the first-fit heuristic, and analyze its running time.

35-2 Approximating the size of a maximum clique

Let $G = (V, E)$ be an undirected graph. For any $k \geq 1$, define $G^{(k)}$ to be the undirected graph $(V^{(k)}, E^{(k)})$, where $V^{(k)}$ is the set of all ordered k-tuples of vertices from V and $E^{(k)}$ is defined so that $(v_1, v_2, \ldots, v_k)$ is adjacent to $(w_1, w_2, \ldots, w_k)$ if and only if for $i = 1, 2, \ldots, k$, either vertex v_i is adjacent to w_i in G, or else $v_i = w_i$.

a. Prove that the size of the maximum clique in $G^{(k)}$ is equal to the kth power of the size of the maximum clique in G.

b. Argue that if there is an approximation algorithm that has a constant approximation ratio for finding a maximum-size clique, then there is a polynomial-time approximation scheme for the problem.

35-3 Weighted set-covering problem

Suppose that we generalize the set-covering problem so that each set S_i in the family $\mathcal{F}$ has an associated weight w_i and the weight of a cover $\mathcal{C}$ is $\sum_{S_i \in \mathcal{C}} w_i$. We wish to determine a minimum-weight cover. (Section 35.3 handles the case in which $w_i = 1$ for all i.)

Show how to generalize the greedy set-covering heuristic in a natural manner to provide an approximate solution for any instance of the weighted set-covering problem. Show that your heuristic has an approximation ratio of $H(d)$, where d is the maximum size of any set S_i.

35-4 Maximum matching

Recall that for an undirected graph G, a matching is a set of edges such that no two edges in the set are incident on the same vertex. In Section 26.3, we saw how to find a maximum matching in a bipartite graph. In this problem, we will look at matchings in undirected graphs in general (i.e., the graphs are not required to be bipartite).

a. A ***maximal matching*** is a matching that is not a proper subset of any other matching. Show that a maximal matching need not be a maximum matching by exhibiting an undirected graph G and a maximal matching M in G that is not a maximum matching. (*Hint:* You can find such a graph with only four vertices.)

b. Consider an undirected graph $G = (V, E)$. Give an $O(E)$-time greedy algorithm to find a maximal matching in G.

In this problem, we shall concentrate on a polynomial-time approximation algorithm for maximum matching. Whereas the fastest known algorithm for maximum matching takes superlinear (but polynomial) time, the approximation algorithm here will run in linear time. You will show that the linear-time greedy algorithm for maximal matching in part (b) is a 2-approximation algorithm for maximum matching.

c. Show that the size of a maximum matching in G is a lower bound on the size of any vertex cover for G.

d. Consider a maximal matching M in $G = (V, E)$. Let

$$T = \{v \in V : \text{some edge in } M \text{ is incident on } v\} .$$

What can you say about the subgraph of G induced by the vertices of G that are not in T?

e. Conclude from part (d) that $2|M|$ is the size of a vertex cover for G.

f. Using parts (c) and (e), prove that the greedy algorithm in part (b) is a 2-approximation algorithm for maximum matching.

35-5 Parallel machine scheduling

In the ***parallel-machine-scheduling problem***, we are given n jobs, $J_1, J_2, \ldots, J_n$, where each job J_k has an associated nonnegative processing time of p_k. We are also given m identical machines, $M_1, M_2, \ldots, M_m$. Any job can run on any machine. A ***schedule*** specifies, for each job J_k, the machine on which it runs and the time period during which it runs. Each job J_k must run on some machine M_i for p_k consecutive time units, and during that time period no other job may run on M_i. Let C_k denote the ***completion time*** of job J_k, that is, the time at which job J_k completes processing. Given a schedule, we define $C_{\max} = \max_{1 \le j \le n} C_j$ to be the ***makespan*** of the schedule. The goal is to find a schedule whose makespan is minimum.

For example, suppose that we have two machines M_1 and M_2 and that we have four jobs J_1, J_2, J_3, J_4, with $p_1 = 2$, $p_2 = 12$, $p_3 = 4$, and $p_4 = 5$. Then one possible schedule runs, on machine M_1, job J_1 followed by job J_2, and on machine M_2, it runs job J_4 followed by job J_3. For this schedule, $C_1 = 2$, $C_2 = 14$, $C_3 = 9$, $C_4 = 5$, and $C_{\max} = 14$. An optimal schedule runs J_2 on machine M_1, and it runs jobs J_1, J_3, and J_4 on machine M_2. For this schedule, $C_1 = 2$, $C_2 = 12$, $C_3 = 6$, $C_4 = 11$, and $C_{\max} = 12$.

Given a parallel-machine-scheduling problem, we let $C^*_{\max}$ denote the makespan of an optimal schedule.

a. Show that the optimal makespan is at least as large as the greatest processing time, that is,

$$C^*_{\max} \ge \max_{1 \le k \le n} p_k .$$

b. Show that the optimal makespan is at least as large as the average machine load, that is,

$$C^*_{\max} \ge \frac{1}{m} \sum_{1 \le k \le n} p_k .$$

Suppose that we use the following greedy algorithm for parallel machine scheduling: whenever a machine is idle, schedule any job that has not yet been scheduled.

c. Write pseudocode to implement this greedy algorithm. What is the running time of your algorithm?

d. For the schedule returned by the greedy algorithm, show that

$$C_{\max} \le \frac{1}{m} \sum_{1 \le k \le n} p_k + \max_{1 \le k \le n} p_k \ .$$

Conclude that this algorithm is a polynomial-time 2-approximation algorithm.

35-6 Approximating a maximum spanning tree

Let $G = (V, E)$ be an undirected graph with distinct edge weights $w(u, v)$ on each edge $(u, v) \in E$. For each vertex $v \in V$, let $\max(v) = \max_{(u,v) \in E} \{w(u, v)\}$ be the maximum-weight edge incident on that vertex. Let $S_G = \{\max(v) : v \in V\}$ be the set of maximum-weight edges incident on each vertex, and let T_G be the maximum-weight spanning tree of G, that is, the spanning tree of maximum total weight. For any subset of edges $E' \subseteq E$, define $w(E') = \sum_{(u,v) \in E'} w(u, v)$.

a. Give an example of a graph with at least 4 vertices for which $S_G = T_G$.

b. Give an example of a graph with at least 4 vertices for which $S_G \neq T_G$.

c. Prove that $S_G \subseteq T_G$ for any graph G.

d. Prove that $w(T_G) \ge w(S_G)/2$ for any graph G.

e. Give an $O(V + E)$-time algorithm to compute a 2-approximation to the maximum spanning tree.

35-7 An approximation algorithm for the 0-1 knapsack problem

Recall the knapsack problem from Section 16.2. There are n items, where the ith item is worth v_i dollars and weighs w_i pounds. We are also given a knapsack that can hold at most W pounds. Here, we add the further assumptions that each weight w_i is at most W and that the items are indexed in monotonically decreasing order of their values: $v_1 \ge v_2 \ge \cdots \ge v_n$.

In the 0-1 knapsack problem, we wish to find a subset of the items whose total weight is at most W and whose total value is maximum. The fractional knapsack problem is like the 0-1 knapsack problem, except that we are allowed to take a fraction of each item, rather than being restricted to taking either all or none of

each item. If we take a fraction x_i of item i, where $0 \leq x_i \leq 1$, we contribute $x_i w_i$ to the weight of the knapsack and receive value $x_i v_i$. Our goal is to develop a polynomial-time 2-approximation algorithm for the 0-1 knapsack problem.

In order to design a polynomial-time algorithm, we consider restricted instances of the 0-1 knapsack problem. Given an instance I of the knapsack problem, we form restricted instances I_j, for $j = 1, 2, \ldots, n$, by removing items $1, 2, \ldots, j-1$ and requiring the solution to include item j (all of item j in both the fractional and 0-1 knapsack problems). No items are removed in instance I_1. For instance I_j, let P_j denote an optimal solution to the 0-1 problem and Q_j denote an optimal solution to the fractional problem.

a. Argue that an optimal solution to instance I of the 0-1 knapsack problem is one of $\{P_1, P_2, \ldots, P_n\}$.

b. Prove that we can find an optimal solution Q_j to the fractional problem for instance I_j by including item j and then using the greedy algorithm in which at each step, we take as much as possible of the unchosen item in the set $\{j+1, j+2, \ldots, n\}$ with maximum value per pound v_i/w_i.

c. Prove that we can always construct an optimal solution Q_j to the fractional problem for instance I_j that includes at most one item fractionally. That is, for all items except possibly one, we either include all of the item or none of the item in the knapsack.

d. Given an optimal solution Q_j to the fractional problem for instance I_j, form solution R_j from Q_j by deleting any fractional items from Q_j. Let $v(S)$ denote the total value of items taken in a solution S. Prove that $v(R_j) \geq v(Q_j)/2 \geq v(P_j)/2$.

e. Give a polynomial-time algorithm that returns a maximum-value solution from the set $\{R_1, R_2, \ldots, R_n\}$, and prove that your algorithm is a polynomial-time 2-approximation algorithm for the 0-1 knapsack problem.

Chapter notes

Although methods that do not necessarily compute exact solutions have been known for thousands of years (for example, methods to approximate the value of π), the notion of an approximation algorithm is much more recent. Hochbaum [172] credits Garey, Graham, and Ullman [128] and Johnson [190] with formalizing the concept of a polynomial-time approximation algorithm. The first such algorithm is often credited to Graham [149].

Since this early work, thousands of approximation algorithms have been designed for a wide range of problems, and there is a wealth of literature on this field. Recent texts by Ausiello et al. [26], Hochbaum [172], and Vazirani [345] deal exclusively with approximation algorithms, as do surveys by Shmoys [315] and Klein and Young [207]. Several other texts, such as Garey and Johnson [129] and Papadimitriou and Steiglitz [271], have significant coverage of approximation algorithms as well. Lawler, Lenstra, Rinnooy Kan, and Shmoys [225] provide an extensive treatment of approximation algorithms for the traveling-salesman problem.

Papadimitriou and Steiglitz attribute the algorithm APPROX-VERTEX-COVER to F. Gavril and M. Yannakakis. The vertex-cover problem has been studied extensively (Hochbaum [172] lists 16 different approximation algorithms for this problem), but all the approximation ratios are at least $2 - o(1)$.

The algorithm APPROX-TSP-TOUR appears in a paper by Rosenkrantz, Stearns, and Lewis [298]. Christofides improved on this algorithm and gave a $3/2$-approximation algorithm for the traveling-salesman problem with the triangle inequality. Arora [22] and Mitchell [257] have shown that if the points are in the euclidean plane, there is a polynomial-time approximation scheme. Theorem 35.3 is due to Sahni and Gonzalez [301].

The analysis of the greedy heuristic for the set-covering problem is modeled after the proof published by Chvátal [68] of a more general result; the basic result as presented here is due to Johnson [190] and Lovász [238].

The algorithm APPROX-SUBSET-SUM and its analysis are loosely modeled after related approximation algorithms for the knapsack and subset-sum problems by Ibarra and Kim [187].

Problem 35-7 is a combinatorial version of a more general result on approximating knapsack-type integer programs by Bienstock and McClosky [45].

The randomized algorithm for MAX-3-CNF satisfiability is implicit in the work of Johnson [190]. The weighted vertex-cover algorithm is by Hochbaum [171]. Section 35.4 only touches on the power of randomization and linear programming in the design of approximation algorithms. A combination of these two ideas yields a technique called "randomized rounding," which formulates a problem as an integer linear program, solves the linear-programming relaxation, and interprets the variables in the solution as probabilities. These probabilities then help guide the solution of the original problem. This technique was first used by Raghavan and Thompson [290], and it has had many subsequent uses. (See Motwani, Naor, and Raghavan [261] for a survey.) Several other notable recent ideas in the field of approximation algorithms include the primal-dual method (see Goemans and Williamson [135] for a survey), finding sparse cuts for use in divide-and-conquer algorithms [229], and the use of semidefinite programming [134].

As mentioned in the chapter notes for Chapter 34, recent results in probabilistically checkable proofs have led to lower bounds on the approximability of many problems, including several in this chapter. In addition to the references there, the chapter by Arora and Lund [23] contains a good description of the relationship between probabilistically checkable proofs and the hardness of approximating various problems.

VIII Appendix: Mathematical Background

Introduction

When we analyze algorithms, we often need to draw upon a body of mathematical tools. Some of these tools are as simple as high-school algebra, but others may be new to you. In Part I, we saw how to manipulate asymptotic notations and solve recurrences. This appendix comprises a compendium of several other concepts and methods we use to analyze algorithms. As noted in the introduction to Part I, you may have seen much of the material in this appendix before having read this book (although the specific notational conventions we use might occasionally differ from those you have seen elsewhere). Hence, you should treat this appendix as reference material. As in the rest of this book, however, we have included exercises and problems, in order for you to improve your skills in these areas.

Appendix A offers methods for evaluating and bounding summations, which occur frequently in the analysis of algorithms. Many of the formulas here appear in any calculus text, but you will find it convenient to have these methods compiled in one place.

Appendix B contains basic definitions and notations for sets, relations, functions, graphs, and trees. It also gives some basic properties of these mathematical objects.

Appendix C begins with elementary principles of counting: permutations, combinations, and the like. The remainder contains definitions and properties of basic probability. Most of the algorithms in this book require no probability for their analysis, and thus you can easily omit the latter sections of the chapter on a first reading, even without skimming them. Later, when you encounter a probabilistic analysis that you want to understand better, you will find Appendix C well organized for reference purposes.

Appendix D defines matrices, their operations, and some of their basic properties. You have probably seen most of this material already if you have taken a course in linear algebra, but you might find it helpful to have one place to look for our notation and definitions.

A Summations

When an algorithm contains an iterative control construct such as a **while** or **for** loop, we can express its running time as the sum of the times spent on each execution of the body of the loop. For example, we found in Section 2.2 that the jth iteration of insertion sort took time proportional to j in the worst case. By adding up the time spent on each iteration, we obtained the summation (or series)

$$\sum_{j=2}^{n} j \ .$$

When we evaluated this summation, we attained a bound of $\Theta(n^2)$ on the worst-case running time of the algorithm. This example illustrates why you should know how to manipulate and bound summations.

Section A.1 lists several basic formulas involving summations. Section A.2 offers useful techniques for bounding summations. We present the formulas in Section A.1 without proof, though proofs for some of them appear in Section A.2 to illustrate the methods of that section. You can find most of the other proofs in any calculus text.

A.1 Summation formulas and properties

Given a sequence $a_1, a_2, \ldots, a_n$ of numbers, where n is a nonnegative integer, we can write the finite sum $a_1 + a_2 + \cdots + a_n$ as

$$\sum_{k=1}^{n} a_k \ .$$

If $n = 0$, the value of the summation is defined to be 0. The value of a finite series is always well defined, and we can add its terms in any order.

Given an infinite sequence $a_1, a_2, \ldots$ of numbers, we can write the infinite sum $a_1 + a_2 + \cdots$ as

$$\sum_{k=1}^{\infty} a_k \ ,$$

which we interpret to mean

$$\lim_{n\to\infty} \sum_{k=1}^{n} a_k \ .$$

If the limit does not exist, the series ***diverges***; otherwise, it ***converges***. The terms of a convergent series cannot always be added in any order. We can, however, rearrange the terms of an ***absolutely convergent series***, that is, a series $\sum_{k=1}^{\infty} a_k$ for which the series $\sum_{k=1}^{\infty} |a_k|$ also converges.

Linearity

For any real number c and any finite sequences $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$,

$$\sum_{k=1}^{n} (ca_k + b_k) = c \sum_{k=1}^{n} a_k + \sum_{k=1}^{n} b_k \ .$$

The linearity property also applies to infinite convergent series.

We can exploit the linearity property to manipulate summations incorporating asymptotic notation. For example,

$$\sum_{k=1}^{n} \Theta(f(k)) = \Theta\left(\sum_{k=1}^{n} f(k)\right) \ .$$

In this equation, the Θ-notation on the left-hand side applies to the variable k, but on the right-hand side, it applies to n. We can also apply such manipulations to infinite convergent series.

Arithmetic series

The summation

$$\sum_{k=1}^{n} k = 1 + 2 + \cdots + n \ ,$$

is an ***arithmetic series*** and has the value

$$\sum_{k=1}^{n} k = \frac{1}{2} n(n+1) \tag{A.1}$$

$$= \Theta(n^2) \ . \tag{A.2}$$

Sums of squares and cubes

We have the following summations of squares and cubes:

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}, \tag{A.3}$$

$$\sum_{k=0}^{n} k^3 = \frac{n^2(n+1)^2}{4}. \tag{A.4}$$

Geometric series

For real $x \neq 1$, the summation

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

is a ***geometric*** or ***exponential series*** and has the value

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1}. \tag{A.5}$$

When the summation is infinite and $|x| < 1$, we have the infinite decreasing geometric series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}. \tag{A.6}$$

Harmonic series

For positive integers n, the nth ***harmonic number*** is

$$\begin{aligned} H_n &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} \\ &= \sum_{k=1}^{n} \frac{1}{k} \\ &= \ln n + O(1). \end{aligned} \tag{A.7}$$

(We shall prove a related bound in Section A.2.)

Integrating and differentiating series

By integrating or differentiating the formulas above, additional formulas arise. For example, by differentiating both sides of the infinite geometric series (A.6) and multiplying by x, we get

$$\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2} \tag{A.8}$$

for $|x| < 1$.

Telescoping series

For any sequence $a_0, a_1, \ldots, a_n$,

$$\sum_{k=1}^{n} (a_k - a_{k-1}) = a_n - a_0 \,, \tag{A.9}$$

since each of the terms $a_1, a_2, \ldots, a_{n-1}$ is added in exactly once and subtracted out exactly once. We say that the sum ***telescopes***. Similarly,

$$\sum_{k=0}^{n-1} (a_k - a_{k+1}) = a_0 - a_n \,.$$

As an example of a telescoping sum, consider the series

$$\sum_{k=1}^{n-1} \frac{1}{k(k+1)} \,.$$

Since we can rewrite each term as

$$\frac{1}{k(k+1)} = \frac{1}{k} - \frac{1}{k+1} \,,$$

we get

$$\begin{aligned} \sum_{k=1}^{n-1} \frac{1}{k(k+1)} &= \sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1} \right) \\ &= 1 - \frac{1}{n} \,. \end{aligned}$$

Products

We can write the finite product $a_1 a_2 \cdots a_n$ as

$$\prod_{k=1}^{n} a_k \,.$$

If $n = 0$, the value of the product is defined to be 1. We can convert a formula with a product to a formula with a summation by using the identity

$$\lg \left(\prod_{k=1}^{n} a_k \right) = \sum_{k=1}^{n} \lg a_k \,.$$

Exercises

A.1-1
Find a simple formula for $\sum_{k=1}^{n}(2k-1)$.

A.1-2 ⋆
Show that $\sum_{k=1}^{n} 1/(2k-1) = \ln(\sqrt{n}) + O(1)$ by manipulating the harmonic series.

A.1-3
Show that $\sum_{k=0}^{\infty} k^2 x^k = x(1+x)/(1-x)^3$ for $0 < |x| < 1$.

A.1-4 ⋆
Show that $\sum_{k=0}^{\infty}(k-1)/2^k = 0$.

A.1-5 ⋆
Evaluate the sum $\sum_{k=1}^{\infty}(2k+1)x^{2k}$.

A.1-6
Prove that $\sum_{k=1}^{n} O(f_k(i)) = O\bigl(\sum_{k=1}^{n} f_k(i)\bigr)$ by using the linearity property of summations.

A.1-7
Evaluate the product $\prod_{k=1}^{n} 2 \cdot 4^k$.

A.1-8 ⋆
Evaluate the product $\prod_{k=2}^{n}(1 - 1/k^2)$.

A.2 Bounding summations

We have many techniques at our disposal for bounding the summations that describe the running times of algorithms. Here are some of the most frequently used methods.

Mathematical induction

The most basic way to evaluate a series is to use mathematical induction. As an example, let us prove that the arithmetic series $\sum_{k=1}^{n} k$ evaluates to $\frac{1}{2}n(n+1)$. We can easily verify this assertion for $n = 1$. We make the inductive assumption that

it holds for n, and we prove that it holds for $n + 1$. We have

$$\begin{aligned}\sum_{k=1}^{n+1} k &= \sum_{k=1}^{n} k + (n+1) \\ &= \frac{1}{2}n(n+1) + (n+1) \\ &= \frac{1}{2}(n+1)(n+2) .\end{aligned}$$

You don't always need to guess the exact value of a summation in order to use mathematical induction. Instead, you can use induction to prove a bound on a summation. As an example, let us prove that the geometric series $\sum_{k=0}^{n} 3^k$ is $O(3^n)$. More specifically, let us prove that $\sum_{k=0}^{n} 3^k \le c3^n$ for some constant c. For the initial condition $n = 0$, we have $\sum_{k=0}^{0} 3^k = 1 \le c \cdot 1$ as long as $c \ge 1$. Assuming that the bound holds for n, let us prove that it holds for $n + 1$. We have

$$\begin{aligned}\sum_{k=0}^{n+1} 3^k &= \sum_{k=0}^{n} 3^k + 3^{n+1} \\ &\le c3^n + 3^{n+1} && \text{(by the inductive hypothesis)} \\ &= \left(\frac{1}{3} + \frac{1}{c}\right) c3^{n+1} \\ &\le c3^{n+1}\end{aligned}$$

as long as $(1/3 + 1/c) \le 1$ or, equivalently, $c \ge 3/2$. Thus, $\sum_{k=0}^{n} 3^k = O(3^n)$, as we wished to show.

We have to be careful when we use asymptotic notation to prove bounds by induction. Consider the following fallacious proof that $\sum_{k=1}^{n} k = O(n)$. Certainly, $\sum_{k=1}^{1} k = O(1)$. Assuming that the bound holds for n, we now prove it for $n + 1$:

$$\begin{aligned}\sum_{k=1}^{n+1} k &= \sum_{k=1}^{n} k + (n+1) \\ &= O(n) + (n+1) && \Longleftarrow \textit{wrong!!} \\ &= O(n+1) .\end{aligned}$$

The bug in the argument is that the "constant" hidden by the "big-oh" grows with n and thus is not constant. We have not shown that the same constant works for *all* n.

Bounding the terms

We can sometimes obtain a good upper bound on a series by bounding each term of the series, and it often suffices to use the largest term to bound the others. For

example, a quick upper bound on the arithmetic series (A.1) is

$$\begin{aligned}\sum_{k=1}^{n} k &\leq \sum_{k=1}^{n} n \\ &= n^2 .\end{aligned}$$

In general, for a series $\sum_{k=1}^{n} a_k$, if we let $a_{\max} = \max_{1 \leq k \leq n} a_k$, then

$$\sum_{k=1}^{n} a_k \leq n \cdot a_{\max} .$$

The technique of bounding each term in a series by the largest term is a weak method when the series can in fact be bounded by a geometric series. Given the series $\sum_{k=0}^{n} a_k$, suppose that $a_{k+1}/a_k \leq r$ for all $k \geq 0$, where $0 < r < 1$ is a constant. We can bound the sum by an infinite decreasing geometric series, since $a_k \leq a_0 r^k$, and thus

$$\begin{aligned}\sum_{k=0}^{n} a_k &\leq \sum_{k=0}^{\infty} a_0 r^k \\ &= a_0 \sum_{k=0}^{\infty} r^k \\ &= a_0 \frac{1}{1-r} .\end{aligned}$$

We can apply this method to bound the summation $\sum_{k=1}^{\infty} (k/3^k)$. In order to start the summation at $k = 0$, we rewrite it as $\sum_{k=0}^{\infty} ((k+1)/3^{k+1})$. The first term ($a_0$) is $1/3$, and the ratio (r) of consecutive terms is

$$\begin{aligned}\frac{(k+2)/3^{k+2}}{(k+1)/3^{k+1}} &= \frac{1}{3} \cdot \frac{k+2}{k+1} \\ &\leq \frac{2}{3}\end{aligned}$$

for all $k \geq 0$. Thus, we have

$$\begin{aligned}\sum_{k=1}^{\infty} \frac{k}{3^k} &= \sum_{k=0}^{\infty} \frac{k+1}{3^{k+1}} \\ &\leq \frac{1}{3} \cdot \frac{1}{1-2/3} \\ &= 1 .\end{aligned}$$

A common bug in applying this method is to show that the ratio of consecutive terms is less than 1 and then to assume that the summation is bounded by a geometric series. An example is the infinite harmonic series, which diverges since

$$\begin{aligned}\sum_{k=1}^{\infty}\frac{1}{k} &= \lim_{n\to\infty}\sum_{k=1}^{n}\frac{1}{k}\\ &= \lim_{n\to\infty}\Theta(\lg n)\\ &= \infty\,.\end{aligned}$$

The ratio of the $(k+1)$st and kth terms in this series is $k/(k+1) < 1$, but the series is not bounded by a decreasing geometric series. To bound a series by a geometric series, we must show that there is an $r < 1$, which is a *constant*, such that the ratio of all pairs of consecutive terms never exceeds r. In the harmonic series, no such r exists because the ratio becomes arbitrarily close to 1.

Splitting summations

One way to obtain bounds on a difficult summation is to express the series as the sum of two or more series by partitioning the range of the index and then to bound each of the resulting series. For example, suppose we try to find a lower bound on the arithmetic series $\sum_{k=1}^{n} k$, which we have already seen has an upper bound of n^2. We might attempt to bound each term in the summation by the smallest term, but since that term is 1, we get a lower bound of n for the summation—far off from our upper bound of n^2.

We can obtain a better lower bound by first splitting the summation. Assume for convenience that n is even. We have

$$\begin{aligned}\sum_{k=1}^{n} k &= \sum_{k=1}^{n/2} k + \sum_{k=n/2+1}^{n} k\\ &\geq \sum_{k=1}^{n/2} 0 + \sum_{k=n/2+1}^{n} (n/2)\\ &= (n/2)^2\\ &= \Omega(n^2)\,,\end{aligned}$$

which is an asymptotically tight bound, since $\sum_{k=1}^{n} k = O(n^2)$.

For a summation arising from the analysis of an algorithm, we can often split the summation and ignore a constant number of the initial terms. Generally, this technique applies when each term a_k in a summation $\sum_{k=0}^{n} a_k$ is independent of n.

Then for any constant $k_0 > 0$, we can write

$$\begin{aligned}\sum_{k=0}^{n} a_k &= \sum_{k=0}^{k_0-1} a_k + \sum_{k=k_0}^{n} a_k \\ &= \Theta(1) + \sum_{k=k_0}^{n} a_k \ ,\end{aligned}$$

since the initial terms of the summation are all constant and there are a constant number of them. We can then use other methods to bound $\sum_{k=k_0}^{n} a_k$. This technique applies to infinite summations as well. For example, to find an asymptotic upper bound on

$$\sum_{k=0}^{\infty} \frac{k^2}{2^k} \ ,$$

we observe that the ratio of consecutive terms is

$$\begin{aligned}\frac{(k+1)^2/2^{k+1}}{k^2/2^k} &= \frac{(k+1)^2}{2k^2} \\ &\le \frac{8}{9}\end{aligned}$$

if $k \ge 3$. Thus, the summation can be split into

$$\begin{aligned}\sum_{k=0}^{\infty} \frac{k^2}{2^k} &= \sum_{k=0}^{2} \frac{k^2}{2^k} + \sum_{k=3}^{\infty} \frac{k^2}{2^k} \\ &\le \sum_{k=0}^{2} \frac{k^2}{2^k} + \frac{9}{8} \sum_{k=0}^{\infty} \left(\frac{8}{9}\right)^k \\ &= O(1) \ ,\end{aligned}$$

since the first summation has a constant number of terms and the second summation is a decreasing geometric series.

The technique of splitting summations can help us determine asymptotic bounds in much more difficult situations. For example, we can obtain a bound of $O(\lg n)$ on the harmonic series (A.7):

$$H_n = \sum_{k=1}^{n} \frac{1}{k} \ .$$

We do so by splitting the range 1 to n into $\lfloor \lg n \rfloor + 1$ pieces and upper-bounding the contribution of each piece by 1. For $i = 0, 1, \ldots, \lfloor \lg n \rfloor$, the ith piece consists

of the terms starting at $1/2^i$ and going up to but not including $1/2^{i+1}$. The last piece might contain terms not in the original harmonic series, and thus we have

$$\begin{aligned}\sum_{k=1}^{n}\frac{1}{k} &\le \sum_{i=0}^{\lfloor \lg n\rfloor}\sum_{j=0}^{2^i-1}\frac{1}{2^i+j}\\ &\le \sum_{i=0}^{\lfloor \lg n\rfloor}\sum_{j=0}^{2^i-1}\frac{1}{2^i}\\ &= \sum_{i=0}^{\lfloor \lg n\rfloor}1\\ &\le \lg n+1\ . \end{aligned} \tag{A.10}$$

Approximation by integrals

When a summation has the form $\sum_{k=m}^{n} f(k)$, where $f(k)$ is a monotonically increasing function, we can approximate it by integrals:

$$\int_{m-1}^{n} f(x)\,dx \le \sum_{k=m}^{n} f(k) \le \int_{m}^{n+1} f(x)\,dx\ . \tag{A.11}$$

Figure A.1 justifies this approximation. The summation is represented as the area of the rectangles in the figure, and the integral is the shaded region under the curve. When $f(k)$ is a monotonically decreasing function, we can use a similar method to provide the bounds

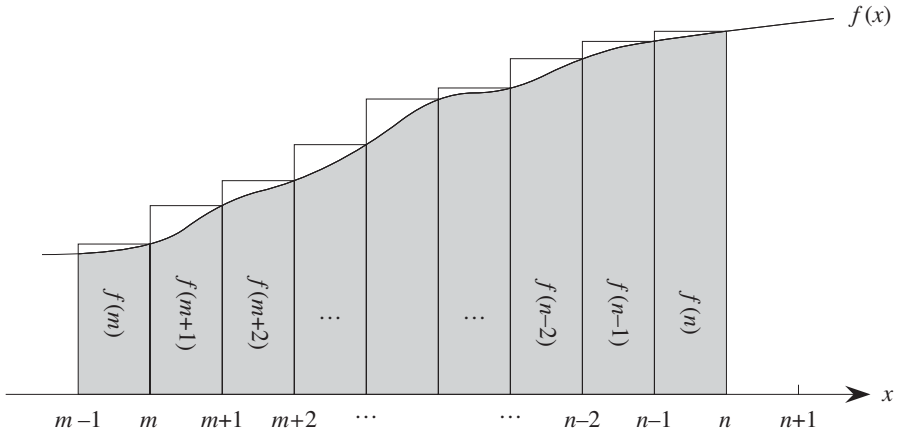
$$\int_{m}^{n+1} f(x)\,dx \le \sum_{k=m}^{n} f(k) \le \int_{m-1}^{n} f(x)\,dx\ . \tag{A.12}$$

The integral approximation (A.12) gives a tight estimate for the nth harmonic number. For a lower bound, we obtain

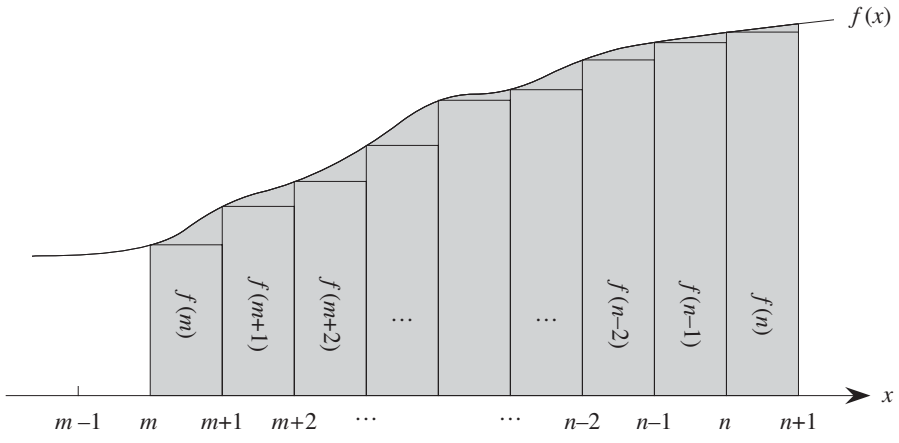
$$\begin{aligned}\sum_{k=1}^{n}\frac{1}{k} &\ge \int_{1}^{n+1}\frac{dx}{x}\\ &= \ln(n+1)\ . \end{aligned} \tag{A.13}$$

For the upper bound, we derive the inequality

$$\begin{aligned}\sum_{k=2}^{n}\frac{1}{k} &\le \int_{1}^{n}\frac{dx}{x}\\ &= \ln n\ , \end{aligned}$$



(a)



(b)

Figure A.1 Approximation of $\sum_{k=m}^{n} f(k)$ by integrals. The area of each rectangle is shown within the rectangle, and the total rectangle area represents the value of the summation. The integral is represented by the shaded area under the curve. By comparing areas in **(a)**, we get $\int_{m-1}^{n} f(x)\,dx \leq \sum_{k=m}^{n} f(k)$, and then by shifting the rectangles one unit to the right, we get $\sum_{k=m}^{n} f(k) \leq \int_{m}^{n+1} f(x)\,dx$ in **(b)**.

which yields the bound

$$\sum_{k=1}^{n} \frac{1}{k} \le \ln n + 1 \ . \tag{A.14}$$

Exercises

A.2-1
Show that $\sum_{k=1}^{n} 1/k^2$ is bounded above by a constant.

A.2-2
Find an asymptotic upper bound on the summation

$$\sum_{k=0}^{\lfloor \lg n \rfloor} \lceil n/2^k \rceil \ .$$

A.2-3
Show that the nth harmonic number is $\Omega(\lg n)$ by splitting the summation.

A.2-4
Approximate $\sum_{k=1}^{n} k^3$ with an integral.

A.2-5
Why didn't we use the integral approximation (A.12) directly on $\sum_{k=1}^{n} 1/k$ to obtain an upper bound on the nth harmonic number?

Problems

A-1 Bounding summations
Give asymptotically tight bounds on the following summations. Assume that $r \ge 0$ and $s \ge 0$ are constants.

a. $\displaystyle\sum_{k=1}^{n} k^r$.

b. $\displaystyle\sum_{k=1}^{n} \lg^s k$.

c. $\displaystyle\sum_{k=1}^{n} k^r \lg^s k$.

Appendix notes

Knuth [209] provides an excellent reference for the material presented here. You can find basic properties of series in any good calculus book, such as Apostol [18] or Thomas et al. [334].

B Sets, Etc.

Many chapters of this book touch on the elements of discrete mathematics. This appendix reviews more completely the notations, definitions, and elementary properties of sets, relations, functions, graphs, and trees. If you are already well versed in this material, you can probably just skim this chapter.

B.1 Sets

A ***set*** is a collection of distinguishable objects, called its ***members*** or ***elements***. If an object x is a member of a set S, we write $x \in S$ (read "x is a member of S" or, more briefly, "x is in S"). If x is not a member of S, we write $x \notin S$. We can describe a set by explicitly listing its members as a list inside braces. For example, we can define a set S to contain precisely the numbers 1, 2, and 3 by writing $S = \{1, 2, 3\}$. Since 2 is a member of the set S, we can write $2 \in S$, and since 4 is not a member, we have $4 \notin S$. A set cannot contain the same object more than once,[1] and its elements are not ordered. Two sets A and B are ***equal***, written $A = B$, if they contain the same elements. For example, $\{1, 2, 3, 1\} = \{1, 2, 3\} = \{3, 2, 1\}$.

We adopt special notations for frequently encountered sets:

- $\emptyset$ denotes the ***empty set***, that is, the set containing no members.
- $\mathbb{Z}$ denotes the set of ***integers***, that is, the set $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
- $\mathbb{R}$ denotes the set of ***real numbers***.
- $\mathbb{N}$ denotes the set of ***natural numbers***, that is, the set $\{0, 1, 2, \ldots\}$.[2]

[1] A variation of a set, which can contain the same object more than once, is called a ***multiset***.

[2] Some authors start the natural numbers with 1 instead of 0. The modern trend seems to be to start with 0.

If all the elements of a set A are contained in a set B, that is, if $x \in A$ implies $x \in B$, then we write $A \subseteq B$ and say that A is a ***subset*** of B. A set A is a ***proper subset*** of B, written $A \subset B$, if $A \subseteq B$ but $A \neq B$. (Some authors use the symbol "$\subset$" to denote the ordinary subset relation, rather than the proper-subset relation.) For any set A, we have $A \subseteq A$. For two sets A and B, we have $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$. For any three sets A, B, and C, if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$. For any set A, we have $\emptyset \subseteq A$.

We sometimes define sets in terms of other sets. Given a set A, we can define a set $B \subseteq A$ by stating a property that distinguishes the elements of B. For example, we can define the set of even integers by $\{x : x \in \mathbb{Z} \text{ and } x/2 \text{ is an integer}\}$. The colon in this notation is read "such that." (Some authors use a vertical bar in place of the colon.)

Given two sets A and B, we can also define new sets by applying ***set operations***:

- The ***intersection*** of sets A and B is the set

  $$A \cap B = \{x : x \in A \text{ and } x \in B\} .$$

- The ***union*** of sets A and B is the set

  $$A \cup B = \{x : x \in A \text{ or } x \in B\} .$$

- The ***difference*** between two sets A and B is the set

  $$A - B = \{x : x \in A \text{ and } x \notin B\} .$$

Set operations obey the following laws:

Empty set laws:

$$\begin{aligned} A \cap \emptyset &= \emptyset , \\ A \cup \emptyset &= A . \end{aligned}$$

Idempotency laws:

$$\begin{aligned} A \cap A &= A , \\ A \cup A &= A . \end{aligned}$$

Commutative laws:

$$\begin{aligned} A \cap B &= B \cap A , \\ A \cup B &= B \cup A . \end{aligned}$$

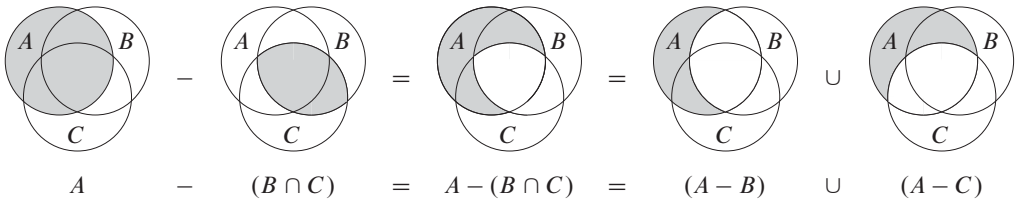


Figure B.1 A Venn diagram illustrating the first of DeMorgan's laws (B.2). Each of the sets A, B, and C is represented as a circle.

Associative laws:

$$\begin{aligned} A \cap (B \cap C) &= (A \cap B) \cap C \ , \\ A \cup (B \cup C) &= (A \cup B) \cup C \ . \end{aligned}$$

Distributive laws:

$$\begin{aligned} A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \ , \\ A \cup (B \cap C) &= (A \cup B) \cap (A \cup C) \ . \end{aligned} \tag{B.1}$$

Absorption laws:

$$\begin{aligned} A \cap (A \cup B) &= A \ , \\ A \cup (A \cap B) &= A \ . \end{aligned}$$

DeMorgan's laws:

$$\begin{aligned} A - (B \cap C) &= (A - B) \cup (A - C) \ , \\ A - (B \cup C) &= (A - B) \cap (A - C) \ . \end{aligned} \tag{B.2}$$

Figure B.1 illustrates the first of DeMorgan's laws, using a ***Venn diagram***: a graphical picture in which sets are represented as regions of the plane.

Often, all the sets under consideration are subsets of some larger set U called the ***universe***. For example, if we are considering various sets made up only of integers, the set $\mathbb{Z}$ of integers is an appropriate universe. Given a universe U, we define the ***complement*** of a set A as $\overline{A} = U - A = \{x : x \in U \text{ and } x \notin A\}$. For any set $A \subseteq U$, we have the following laws:

$$\begin{aligned} \overline{\overline{A}} &= A \ , \\ A \cap \overline{A} &= \emptyset \ , \\ A \cup \overline{A} &= U \ . \end{aligned}$$

We can rewrite DeMorgan's laws (B.2) with set complements. For any two sets $B, C \subseteq U$, we have

$$\begin{aligned}\overline{B \cap C} &= \overline{B} \cup \overline{C} \,, \\ \overline{B \cup C} &= \overline{B} \cap \overline{C} \,.\end{aligned}$$

Two sets A and B are ***disjoint*** if they have no elements in common, that is, if $A \cap B = \emptyset$. A collection $\mathcal{S} = \{S_i\}$ of nonempty sets forms a ***partition*** of a set S if

- the sets are ***pairwise disjoint***, that is, $S_i, S_j \in \mathcal{S}$ and $i \neq j$ imply $S_i \cap S_j = \emptyset$, and
- their union is S, that is,

$$S = \bigcup_{S_i \in \mathcal{S}} S_i \,.$$

In other words, $\mathcal{S}$ forms a partition of S if each element of S appears in exactly one $S_i \in \mathcal{S}$.

The number of elements in a set is the ***cardinality*** (or ***size***) of the set, denoted $|S|$. Two sets have the same cardinality if their elements can be put into a one-to-one correspondence. The cardinality of the empty set is $|\emptyset| = 0$. If the cardinality of a set is a natural number, we say the set is ***finite***; otherwise, it is ***infinite***. An infinite set that can be put into a one-to-one correspondence with the natural numbers $\mathbb{N}$ is ***countably infinite***; otherwise, it is ***uncountable***. For example, the integers $\mathbb{Z}$ are countable, but the reals $\mathbb{R}$ are uncountable.

For any two finite sets A and B, we have the identity

$$|A \cup B| = |A| + |B| - |A \cap B| \,, \tag{B.3}$$

from which we can conclude that

$$|A \cup B| \leq |A| + |B| \,.$$

If A and B are disjoint, then $|A \cap B| = 0$ and thus $|A \cup B| = |A| + |B|$. If $A \subseteq B$, then $|A| \leq |B|$.

A finite set of n elements is sometimes called an ***n-set***. A 1-set is called a ***singleton***. A subset of k elements of a set is sometimes called a ***k-subset***.

We denote the set of all subsets of a set S, including the empty set and S itself, by 2^S; we call 2^S the ***power set*** of S. For example, $2^{\{a,b\}} = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$. The power set of a finite set S has cardinality $2^{|S|}$ (see Exercise B.1-5).

We sometimes care about setlike structures in which the elements are ordered. An ***ordered pair*** of two elements a and b is denoted (a, b) and is defined formally as the set $(a, b) = \{a, \{a, b\}\}$. Thus, the ordered pair (a, b) is *not* the same as the ordered pair (b, a).

The ***Cartesian product*** of two sets A and B, denoted $A \times B$, is the set of all ordered pairs such that the first element of the pair is an element of A and the second is an element of B. More formally,

$$A \times B = \{(a, b) : a \in A \text{ and } b \in B\} .$$

For example, $\{a, b\} \times \{a, b, c\} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$. When A and B are finite sets, the cardinality of their Cartesian product is

$$|A \times B| = |A| \cdot |B| . \tag{B.4}$$

The Cartesian product of n sets $A_1, A_2, \ldots, A_n$ is the set of ***n-tuples***

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) : a_i \in A_i \text{ for } i = 1, 2, \ldots, n\} ,$$

whose cardinality is

$$|A_1 \times A_2 \times \cdots \times A_n| = |A_1| \cdot |A_2| \cdots |A_n|$$

if all sets are finite. We denote an n-fold Cartesian product over a single set A by the set

$$A^n = A \times A \times \cdots \times A ,$$

whose cardinality is $|A^n| = |A|^n$ if A is finite. We can also view an n-tuple as a finite sequence of length n (see page 1166).

Exercises

B.1-1
Draw Venn diagrams that illustrate the first of the distributive laws (B.1).

B.1-2
Prove the generalization of DeMorgan's laws to any finite collection of sets:

$$\begin{aligned} \overline{A_1 \cap A_2 \cap \cdots \cap A_n} &= \overline{A_1} \cup \overline{A_2} \cup \cdots \cup \overline{A_n} , \\ \overline{A_1 \cup A_2 \cup \cdots \cup A_n} &= \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_n} . \end{aligned}$$

B.1-3 ⋆
Prove the generalization of equation (B.3), which is called the ***principle of inclusion and exclusion***:

$$
\begin{aligned}
&|A_1 \cup A_2 \cup \cdots \cup A_n| = \\
&\quad |A_1| + |A_2| + \cdots + |A_n| \\
&\quad - |A_1 \cap A_2| - |A_1 \cap A_3| - \cdots && \text{(all pairs)} \\
&\quad + |A_1 \cap A_2 \cap A_3| + \cdots && \text{(all triples)} \\
&\qquad\qquad \vdots \\
&\quad + (-1)^{n-1} |A_1 \cap A_2 \cap \cdots \cap A_n| \ .
\end{aligned}
$$

B.1-4
Show that the set of odd natural numbers is countable.

B.1-5
Show that for any finite set S, the power set 2^S has $2^{|S|}$ elements (that is, there are $2^{|S|}$ distinct subsets of S).

B.1-6
Give an inductive definition for an n-tuple by extending the set-theoretic definition for an ordered pair.

B.2 Relations

A ***binary relation*** R on two sets A and B is a subset of the Cartesian product $A \times B$. If $(a, b) \in R$, we sometimes write $a\ R\ b$. When we say that R is a binary relation on a set A, we mean that R is a subset of $A \times A$. For example, the "less than" relation on the natural numbers is the set $\{(a, b) : a, b \in \mathbb{N} \text{ and } a < b\}$. An n-ary relation on sets $A_1, A_2, \ldots, A_n$ is a subset of $A_1 \times A_2 \times \cdots \times A_n$.

A binary relation $R \subseteq A \times A$ is ***reflexive*** if

$a\ R\ a$

for all $a \in A$. For example, "$=$" and "$\leq$" are reflexive relations on $\mathbb{N}$, but "$<$" is not. The relation R is ***symmetric*** if

$a\ R\ b$ implies $b\ R\ a$

for all $a, b \in A$. For example, "$=$" is symmetric, but "$<$" and "$\leq$" are not. The relation R is ***transitive*** if

$a\ R\ b$ and $b\ R\ c$ imply $a\ R\ c$

for all $a, b, c \in A$. For example, the relations "$<$," "$\leq$," and "$=$" are transitive, but the relation $R = \{(a, b) : a, b \in \mathbb{N} \text{ and } a = b - 1\}$ is not, since $3\ R\ 4$ and $4\ R\ 5$ do not imply $3\ R\ 5$.

A relation that is reflexive, symmetric, and transitive is an ***equivalence relation***. For example, "$=$" is an equivalence relation on the natural numbers, but "$<$" is not. If R is an equivalence relation on a set A, then for $a \in A$, the ***equivalence class*** of a is the set $[a] = \{b \in A : a\ R\ b\}$, that is, the set of all elements equivalent to a. For example, if we define $R = \{(a, b) : a, b \in \mathbb{N} \text{ and } a + b \text{ is an even number}\}$, then R is an equivalence relation, since $a + a$ is even (reflexive), $a + b$ is even implies $b + a$ is even (symmetric), and $a + b$ is even and $b + c$ is even imply $a + c$ is even (transitive). The equivalence class of 4 is $[4] = \{0, 2, 4, 6, \ldots\}$, and the equivalence class of 3 is $[3] = \{1, 3, 5, 7, \ldots\}$. A basic theorem of equivalence classes is the following.

Theorem B.1 (An equivalence relation is the same as a partition)
The equivalence classes of any equivalence relation R on a set A form a partition of A, and any partition of A determines an equivalence relation on A for which the sets in the partition are the equivalence classes.

Proof For the first part of the proof, we must show that the equivalence classes of R are nonempty, pairwise-disjoint sets whose union is A. Because R is reflexive, $a \in [a]$, and so the equivalence classes are nonempty; moreover, since every element $a \in A$ belongs to the equivalence class $[a]$, the union of the equivalence classes is A. It remains to show that the equivalence classes are pairwise disjoint, that is, if two equivalence classes $[a]$ and $[b]$ have an element c in common, then they are in fact the same set. Suppose that $a\ R\ c$ and $b\ R\ c$. By symmetry, $c\ R\ b$, and by transitivity, $a\ R\ b$. Thus, for any arbitrary element $x \in [a]$, we have $x\ R\ a$ and, by transitivity, $x\ R\ b$, and thus $[a] \subseteq [b]$. Similarly, $[b] \subseteq [a]$, and thus $[a] = [b]$.

For the second part of the proof, let $\mathcal{A} = \{A_i\}$ be a partition of A, and define $R = \{(a, b) : \text{there exists } i \text{ such that } a \in A_i \text{ and } b \in A_i\}$. We claim that R is an equivalence relation on A. Reflexivity holds, since $a \in A_i$ implies $a\ R\ a$. Symmetry holds, because if $a\ R\ b$, then a and b are in the same set A_i, and hence $b\ R\ a$. If $a\ R\ b$ and $b\ R\ c$, then all three elements are in the same set A_i, and thus $a\ R\ c$ and transitivity holds. To see that the sets in the partition are the equivalence classes of R, observe that if $a \in A_i$, then $x \in [a]$ implies $x \in A_i$, and $x \in A_i$ implies $x \in [a]$. ■

A binary relation R on a set A is ***antisymmetric*** if

$a\ R\ b$ and $b\ R\ a$ imply $a = b$.

For example, the "$\leq$" relation on the natural numbers is antisymmetric, since $a \leq b$ and $b \leq a$ imply $a = b$. A relation that is reflexive, antisymmetric, and transitive is a ***partial order***, and we call a set on which a partial order is defined a ***partially ordered set***. For example, the relation "is a descendant of" is a partial order on the set of all people (if we view individuals as being their own descendants).

In a partially ordered set A, there may be no single "maximum" element a such that $b\ R\ a$ for all $b \in A$. Instead, the set may contain several ***maximal*** elements a such that for no $b \in A$, where $b \neq a$, is it the case that $a\ R\ b$. For example, a collection of different-sized boxes may contain several maximal boxes that don't fit inside any other box, yet it has no single "maximum" box into which any other box will fit.[3]

A relation R on a set A is a ***total relation*** if for all $a, b \in A$, we have $a\ R\ b$ or $b\ R\ a$ (or both), that is, if every pairing of elements of A is related by R. A partial order that is also a total relation is a ***total order*** or ***linear order***. For example, the relation "$\leq$" is a total order on the natural numbers, but the "is a descendant of" relation is not a total order on the set of all people, since there are individuals neither of whom is descended from the other. A total relation that is transitive, but not necessarily reflexive and antisymmetric, is a ***total preorder***.

Exercises

B.2-1

Prove that the subset relation "$\subseteq$" on all subsets of $\mathbb{Z}$ is a partial order but not a total order.

B.2-2

Show that for any positive integer n, the relation "equivalent modulo n" is an equivalence relation on the integers. (We say that $a \equiv b \pmod{n}$ if there exists an integer q such that $a - b = qn$.) Into what equivalence classes does this relation partition the integers?

B.2-3

Give examples of relations that are

a. reflexive and symmetric but not transitive,

b. reflexive and transitive but not symmetric,

c. symmetric and transitive but not reflexive.

---

[3] To be precise, in order for the "fit inside" relation to be a partial order, we need to view a box as fitting inside itself.

B.2-4
Let S be a finite set, and let R be an equivalence relation on $S \times S$. Show that if in addition R is antisymmetric, then the equivalence classes of S with respect to R are singletons.

B.2-5
Professor Narcissus claims that if a relation R is symmetric and transitive, then it is also reflexive. He offers the following proof. By symmetry, $a\ R\ b$ implies $b\ R\ a$. Transitivity, therefore, implies $a\ R\ a$. Is the professor correct?

B.3 Functions

Given two sets A and B, a ***function*** f is a binary relation on A and B such that for all $a \in A$, there exists precisely one $b \in B$ such that $(a, b) \in f$. The set A is called the ***domain*** of f, and the set B is called the ***codomain*** of f. We sometimes write $f : A \to B$; and if $(a, b) \in f$, we write $b = f(a)$, since b is uniquely determined by the choice of a.

Intuitively, the function f assigns an element of B to each element of A. No element of A is assigned two different elements of B, but the same element of B can be assigned to two different elements of A. For example, the binary relation

$$f = \{(a, b) : a, b \in \mathbb{N} \text{ and } b = a \bmod 2\}$$

is a function $f : \mathbb{N} \to \{0, 1\}$, since for each natural number a, there is exactly one value b in $\{0, 1\}$ such that $b = a \bmod 2$. For this example, $0 = f(0)$, $1 = f(1)$, $0 = f(2)$, etc. In contrast, the binary relation

$$g = \{(a, b) : a, b \in \mathbb{N} \text{ and } a + b \text{ is even}\}$$

is not a function, since $(1, 3)$ and $(1, 5)$ are both in g, and thus for the choice $a = 1$, there is not precisely one b such that $(a, b) \in g$.

Given a function $f : A \to B$, if $b = f(a)$, we say that a is the ***argument*** of f and that b is the ***value*** of f at a. We can define a function by stating its value for every element of its domain. For example, we might define $f(n) = 2n$ for $n \in \mathbb{N}$, which means $f = \{(n, 2n) : n \in \mathbb{N}\}$. Two functions f and g are ***equal*** if they have the same domain and codomain and if, for all a in the domain, $f(a) = g(a)$.

A ***finite sequence*** of length n is a function f whose domain is the set of n integers $\{0, 1, \ldots, n - 1\}$. We often denote a finite sequence by listing its values: $\langle f(0), f(1), \ldots, f(n - 1)\rangle$. An ***infinite sequence*** is a function whose domain is the set $\mathbb{N}$ of natural numbers. For example, the Fibonacci sequence, defined by recurrence (3.22), is the infinite sequence $\langle 0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots\rangle$.

When the domain of a function f is a Cartesian product, we often omit the extra parentheses surrounding the argument of f. For example, if we had a function $f : A_1 \times A_2 \times \cdots \times A_n \to B$, we would write $b = f(a_1, a_2, \ldots, a_n)$ instead of $b = f((a_1, a_2, \ldots, a_n))$. We also call each a_i an ***argument*** to the function f, though technically the (single) argument to f is the n-tuple $(a_1, a_2, \ldots, a_n)$.

If $f : A \to B$ is a function and $b = f(a)$, then we sometimes say that b is the ***image*** of a under f. The image of a set $A' \subseteq A$ under f is defined by

$$f(A') = \{b \in B : b = f(a) \text{ for some } a \in A'\} .$$

The ***range*** of f is the image of its domain, that is, $f(A)$. For example, the range of the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = 2n$ is $f(\mathbb{N}) = \{m : m = 2n \text{ for some } n \in \mathbb{N}\}$, in other words, the set of nonnegative even integers.

A function is a ***surjection*** if its range is its codomain. For example, the function $f(n) = \lfloor n/2 \rfloor$ is a surjective function from $\mathbb{N}$ to $\mathbb{N}$, since every element in $\mathbb{N}$ appears as the value of f for some argument. In contrast, the function $f(n) = 2n$ is not a surjective function from $\mathbb{N}$ to $\mathbb{N}$, since no argument to f can produce 3 as a value. The function $f(n) = 2n$ is, however, a surjective function from the natural numbers to the even numbers. A surjection $f : A \to B$ is sometimes described as mapping A ***onto*** B. When we say that f is onto, we mean that it is surjective.

A function $f : A \to B$ is an ***injection*** if distinct arguments to f produce distinct values, that is, if $a \neq a'$ implies $f(a) \neq f(a')$. For example, the function $f(n) = 2n$ is an injective function from $\mathbb{N}$ to $\mathbb{N}$, since each even number b is the image under f of at most one element of the domain, namely $b/2$. The function $f(n) = \lfloor n/2 \rfloor$ is not injective, since the value 1 is produced by two arguments: 2 and 3. An injection is sometimes called a ***one-to-one*** function.

A function $f : A \to B$ is a ***bijection*** if it is injective and surjective. For example, the function $f(n) = (-1)^n \lceil n/2 \rceil$ is a bijection from $\mathbb{N}$ to $\mathbb{Z}$:

$$\begin{aligned} 0 &\to 0 , \\ 1 &\to -1 , \\ 2 &\to 1 , \\ 3 &\to -2 , \\ 4 &\to 2 , \\ &\vdots \end{aligned}$$

The function is injective, since no element of $\mathbb{Z}$ is the image of more than one element of $\mathbb{N}$. It is surjective, since every element of $\mathbb{Z}$ appears as the image of some element of $\mathbb{N}$. Hence, the function is bijective. A bijection is sometimes called a ***one-to-one correspondence***, since it pairs elements in the domain and codomain. A bijection from a set A to itself is sometimes called a ***permutation***.

When a function f is bijective, we define its ***inverse*** f^{-1} as

$$f^{-1}(b) = a \text{ if and only if } f(a) = b .$$

For example, the inverse of the function $f(n) = (-1)^n \lceil n/2 \rceil$ is

$$f^{-1}(m) = \begin{cases} 2m & \text{if } m \geq 0\,, \\ -2m-1 & \text{if } m < 0\,. \end{cases}$$

Exercises

B.3-1
Let A and B be finite sets, and let $f : A \to B$ be a function. Show that

a. if f is injective, then $|A| \leq |B|$;

b. if f is surjective, then $|A| \geq |B|$.

B.3-2
Is the function $f(x) = x + 1$ bijective when the domain and the codomain are $\mathbb{N}$? Is it bijective when the domain and the codomain are $\mathbb{Z}$?

B.3-3
Give a natural definition for the inverse of a binary relation such that if a relation is in fact a bijective function, its relational inverse is its functional inverse.

B.3-4 ⋆
Give a bijection from $\mathbb{Z}$ to $\mathbb{Z} \times \mathbb{Z}$.

B.4 Graphs

This section presents two kinds of graphs: directed and undirected. Certain definitions in the literature differ from those given here, but for the most part, the differences are slight. Section 22.1 shows how we can represent graphs in computer memory.

A ***directed graph*** (or ***digraph***) G is a pair (V, E), where V is a finite set and E is a binary relation on V. The set V is called the ***vertex set*** of G, and its elements are called ***vertices*** (singular: ***vertex***). The set E is called the ***edge set*** of G, and its elements are called ***edges***. Figure B.2(a) is a pictorial representation of a directed graph on the vertex set $\{1, 2, 3, 4, 5, 6\}$. Vertices are represented by circles in the figure, and edges are represented by arrows. Note that ***self-loops***—edges from a vertex to itself—are possible.

In an ***undirected graph*** $G = (V, E)$, the edge set E consists of *unordered* pairs of vertices, rather than ordered pairs. That is, an edge is a set $\{u, v\}$, where

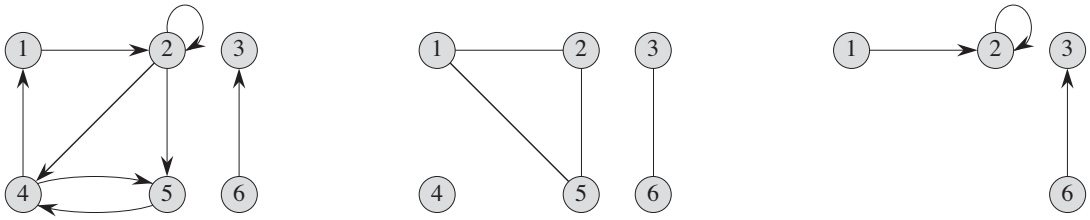


Figure B.2 Directed and undirected graphs. **(a)** A directed graph $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(1, 2), (2, 2), (2, 4), (2, 5), (4, 1), (4, 5), (5, 4), (6, 3)\}$. The edge $(2, 2)$ is a self-loop. **(b)** An undirected graph $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(1, 2), (1, 5), (2, 5), (3, 6)\}$. The vertex 4 is isolated. **(c)** The subgraph of the graph in part (a) induced by the vertex set $\{1, 2, 3, 6\}$.

$u, v \in V$ and $u \neq v$. By convention, we use the notation (u, v) for an edge, rather than the set notation $\{u, v\}$, and we consider (u, v) and (v, u) to be the same edge. In an undirected graph, self-loops are forbidden, and so every edge consists of two distinct vertices. Figure B.2(b) is a pictorial representation of an undirected graph on the vertex set $\{1, 2, 3, 4, 5, 6\}$.

Many definitions for directed and undirected graphs are the same, although certain terms have slightly different meanings in the two contexts. If (u, v) is an edge in a directed graph $G = (V, E)$, we say that (u, v) is ***incident from*** or ***leaves*** vertex u and is ***incident to*** or ***enters*** vertex v. For example, the edges leaving vertex 2 in Figure B.2(a) are $(2, 2)$, $(2, 4)$, and $(2, 5)$. The edges entering vertex 2 are $(1, 2)$ and $(2, 2)$. If (u, v) is an edge in an undirected graph $G = (V, E)$, we say that (u, v) is ***incident on*** vertices u and v. In Figure B.2(b), the edges incident on vertex 2 are $(1, 2)$ and $(2, 5)$.

If (u, v) is an edge in a graph $G = (V, E)$, we say that vertex v is ***adjacent*** to vertex u. When the graph is undirected, the adjacency relation is symmetric. When the graph is directed, the adjacency relation is not necessarily symmetric. If v is adjacent to u in a directed graph, we sometimes write $u \rightarrow v$. In parts (a) and (b) of Figure B.2, vertex 2 is adjacent to vertex 1, since the edge $(1, 2)$ belongs to both graphs. Vertex 1 is *not* adjacent to vertex 2 in Figure B.2(a), since the edge $(2, 1)$ does not belong to the graph.

The ***degree*** of a vertex in an undirected graph is the number of edges incident on it. For example, vertex 2 in Figure B.2(b) has degree 2. A vertex whose degree is 0, such as vertex 4 in Figure B.2(b), is ***isolated***. In a directed graph, the ***out-degree*** of a vertex is the number of edges leaving it, and the ***in-degree*** of a vertex is the number of edges entering it. The ***degree*** of a vertex in a directed graph is its in-

degree plus its out-degree. Vertex 2 in Figure B.2(a) has in-degree 2, out-degree 3, and degree 5.

A ***path*** of ***length*** k from a vertex u to a vertex u' in a graph $G = (V, E)$ is a sequence $\langle v_0, v_1, v_2, \ldots, v_k \rangle$ of vertices such that $u = v_0$, $u' = v_k$, and $(v_{i-1}, v_i) \in E$ for $i = 1, 2, \ldots, k$. The length of the path is the number of edges in the path. The path ***contains*** the vertices $v_0, v_1, \ldots, v_k$ and the edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$. (There is always a 0-length path from u to u.) If there is a path p from u to u', we say that u' is ***reachable*** from u via p, which we sometimes write as $u \overset{p}{\rightsquigarrow} u'$ if G is directed. A path is ***simple***[4] if all vertices in the path are distinct. In Figure B.2(a), the path $\langle 1, 2, 5, 4 \rangle$ is a simple path of length 3. The path $\langle 2, 5, 4, 5 \rangle$ is not simple.

A ***subpath*** of path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is a contiguous subsequence of its vertices. That is, for any $0 \le i \le j \le k$, the subsequence of vertices $\langle v_i, v_{i+1}, \ldots, v_j \rangle$ is a subpath of p.

In a directed graph, a path $\langle v_0, v_1, \ldots, v_k \rangle$ forms a ***cycle*** if $v_0 = v_k$ and the path contains at least one edge. The cycle is ***simple*** if, in addition, $v_1, v_2, \ldots, v_k$ are distinct. A self-loop is a cycle of length 1. Two paths $\langle v_0, v_1, v_2, \ldots, v_{k-1}, v_0 \rangle$ and $\langle v'_0, v'_1, v'_2, \ldots, v'_{k-1}, v'_0 \rangle$ form the same cycle if there exists an integer j such that $v'_i = v_{(i+j) \bmod k}$ for $i = 0, 1, \ldots, k-1$. In Figure B.2(a), the path $\langle 1, 2, 4, 1 \rangle$ forms the same cycle as the paths $\langle 2, 4, 1, 2 \rangle$ and $\langle 4, 1, 2, 4 \rangle$. This cycle is simple, but the cycle $\langle 1, 2, 4, 5, 4, 1 \rangle$ is not. The cycle $\langle 2, 2 \rangle$ formed by the edge $(2, 2)$ is a self-loop. A directed graph with no self-loops is ***simple***. In an undirected graph, a path $\langle v_0, v_1, \ldots, v_k \rangle$ forms a ***cycle*** if $k \ge 3$ and $v_0 = v_k$; the cycle is ***simple*** if $v_1, v_2, \ldots, v_k$ are distinct. For example, in Figure B.2(b), the path $\langle 1, 2, 5, 1 \rangle$ is a simple cycle. A graph with no cycles is ***acyclic***.

An undirected graph is ***connected*** if every vertex is reachable from all other vertices. The ***connected components*** of a graph are the equivalence classes of vertices under the "is reachable from" relation. The graph in Figure B.2(b) has three connected components: $\{1, 2, 5\}$, $\{3, 6\}$, and $\{4\}$. Every vertex in $\{1, 2, 5\}$ is reachable from every other vertex in $\{1, 2, 5\}$. An undirected graph is connected if it has exactly one connected component. The edges of a connected component are those that are incident on only the vertices of the component; in other words, edge (u, v) is an edge of a connected component only if both u and v are vertices of the component.

A directed graph is ***strongly connected*** if every two vertices are reachable from each other. The ***strongly connected components*** of a directed graph are the equiv-

[4] Some authors refer to what we call a path as a "walk" and to what we call a simple path as just a "path." We use the terms "path" and "simple path" throughout this book in a manner consistent with their definitions.

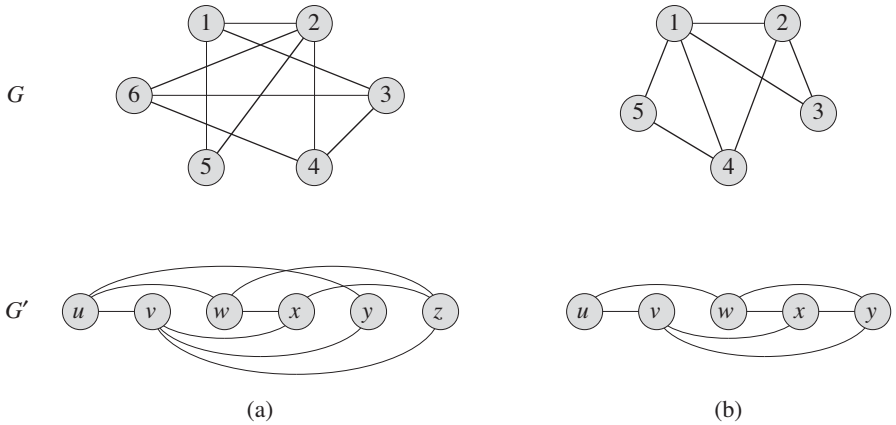


Figure B.3 **(a)** A pair of isomorphic graphs. The vertices of the top graph are mapped to the vertices of the bottom graph by $f(1) = u$, $f(2) = v$, $f(3) = w$, $f(4) = x$, $f(5) = y$, $f(6) = z$. **(b)** Two graphs that are not isomorphic, since the top graph has a vertex of degree 4 and the bottom graph does not.

alence classes of vertices under the "are mutually reachable" relation. A directed graph is strongly connected if it has only one strongly connected component. The graph in Figure B.2(a) has three strongly connected components: $\{1, 2, 4, 5\}$, $\{3\}$, and $\{6\}$. All pairs of vertices in $\{1, 2, 4, 5\}$ are mutually reachable. The vertices $\{3, 6\}$ do not form a strongly connected component, since vertex 6 cannot be reached from vertex 3.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are ***isomorphic*** if there exists a bijection $f : V \to V'$ such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in E'$. In other words, we can relabel the vertices of G to be vertices of G', maintaining the corresponding edges in G and G'. Figure B.3(a) shows a pair of isomorphic graphs G and G' with respective vertex sets $V = \{1, 2, 3, 4, 5, 6\}$ and $V' = \{u, v, w, x, y, z\}$. The mapping from V to V' given by $f(1) = u$, $f(2) = v$, $f(3) = w$, $f(4) = x$, $f(5) = y$, $f(6) = z$ provides the required bijective function. The graphs in Figure B.3(b) are not isomorphic. Although both graphs have 5 vertices and 7 edges, the top graph has a vertex of degree 4 and the bottom graph does not.

We say that a graph $G' = (V', E')$ is a ***subgraph*** of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Given a set $V' \subseteq V$, the subgraph of G ***induced*** by V' is the graph $G' = (V', E')$, where

$$E' = \{(u, v) \in E : u, v \in V'\} \ .$$

The subgraph induced by the vertex set $\{1, 2, 3, 6\}$ in Figure B.2(a) appears in Figure B.2(c) and has the edge set $\{(1, 2), (2, 2), (6, 3)\}$.

Given an undirected graph $G = (V, E)$, the ***directed version*** of G is the directed graph $G' = (V, E')$, where $(u, v) \in E'$ if and only if $(u, v) \in E$. That is, we replace each undirected edge (u, v) in G by the two directed edges (u, v) and (v, u) in the directed version. Given a directed graph $G = (V, E)$, the ***undirected version*** of G is the undirected graph $G' = (V, E')$, where $(u, v) \in E'$ if and only if $u \neq v$ and $(u, v) \in E$. That is, the undirected version contains the edges of G "with their directions removed" and with self-loops eliminated. (Since (u, v) and (v, u) are the same edge in an undirected graph, the undirected version of a directed graph contains it only once, even if the directed graph contains both edges (u, v) and (v, u).) In a directed graph $G = (V, E)$, a ***neighbor*** of a vertex u is any vertex that is adjacent to u in the undirected version of G. That is, v is a neighbor of u if $u \neq v$ and either $(u, v) \in E$ or $(v, u) \in E$. In an undirected graph, u and v are neighbors if they are adjacent.

Several kinds of graphs have special names. A ***complete graph*** is an undirected graph in which every pair of vertices is adjacent. A ***bipartite graph*** is an undirected graph $G = (V, E)$ in which V can be partitioned into two sets V_1 and V_2 such that $(u, v) \in E$ implies either $u \in V_1$ and $v \in V_2$ or $u \in V_2$ and $v \in V_1$. That is, all edges go between the two sets V_1 and V_2. An acyclic, undirected graph is a ***forest***, and a connected, acyclic, undirected graph is a ***(free) tree*** (see Section B.5). We often take the first letters of "directed acyclic graph" and call such a graph a ***dag***.

There are two variants of graphs that you may occasionally encounter. A ***multigraph*** is like an undirected graph, but it can have both multiple edges between vertices and self-loops. A ***hypergraph*** is like an undirected graph, but each ***hyperedge***, rather than connecting two vertices, connects an arbitrary subset of vertices. Many algorithms written for ordinary directed and undirected graphs can be adapted to run on these graphlike structures.

The ***contraction*** of an undirected graph $G = (V, E)$ by an edge $e = (u, v)$ is a graph $G' = (V', E')$, where $V' = V - \{u, v\} \cup \{x\}$ and x is a new vertex. The set of edges E' is formed from E by deleting the edge (u, v) and, for each vertex w incident on u or v, deleting whichever of (u, w) and (v, w) is in E and adding the new edge (x, w). In effect, u and v are "contracted" into a single vertex.

Exercises

B.4-1

Attendees of a faculty party shake hands to greet each other, and each professor remembers how many times he or she shook hands. At the end of the party, the department head adds up the number of times that each professor shook hands.

Show that the result is even by proving the ***handshaking lemma***: if $G = (V, E)$ is an undirected graph, then

$$\sum_{v \in V} \text{degree}(v) = 2|E| \ .$$

B.4-2
Show that if a directed or undirected graph contains a path between two vertices u and v, then it contains a simple path between u and v. Show that if a directed graph contains a cycle, then it contains a simple cycle.

B.4-3
Show that any connected, undirected graph $G = (V, E)$ satisfies $|E| \geq |V| - 1$.

B.4-4
Verify that in an undirected graph, the "is reachable from" relation is an equivalence relation on the vertices of the graph. Which of the three properties of an equivalence relation hold in general for the "is reachable from" relation on the vertices of a directed graph?

B.4-5
What is the undirected version of the directed graph in Figure B.2(a)? What is the directed version of the undirected graph in Figure B.2(b)?

B.4-6 ⋆
Show that we can represent a hypergraph by a bipartite graph if we let incidence in the hypergraph correspond to adjacency in the bipartite graph. (*Hint:* Let one set of vertices in the bipartite graph correspond to vertices of the hypergraph, and let the other set of vertices of the bipartite graph correspond to hyperedges.)

B.5 Trees

As with graphs, there are many related, but slightly different, notions of trees. This section presents definitions and mathematical properties of several kinds of trees. Sections 10.4 and 22.1 describe how we can represent trees in computer memory.

B.5.1 Free trees

As defined in Section B.4, a ***free tree*** is a connected, acyclic, undirected graph. We often omit the adjective "free" when we say that a graph is a tree. If an undirected graph is acyclic but possibly disconnected, it is a ***forest***. Many algorithms that work

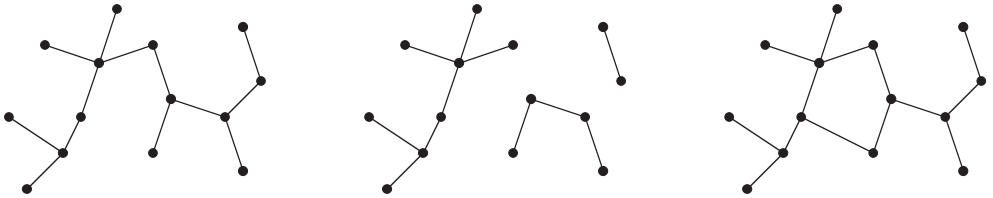


Figure B.4 **(a)** A free tree. **(b)** A forest. **(c)** A graph that contains a cycle and is therefore neither a tree nor a forest.

for trees also work for forests. Figure B.4(a) shows a free tree, and Figure B.4(b) shows a forest. The forest in Figure B.4(b) is not a tree because it is not connected. The graph in Figure B.4(c) is connected but neither a tree nor a forest, because it contains a cycle.

The following theorem captures many important facts about free trees.

Theorem B.2 (Properties of free trees)
Let $G = (V, E)$ be an undirected graph. The following statements are equivalent.

1. G is a free tree.
2. Any two vertices in G are connected by a unique simple path.
3. G is connected, but if any edge is removed from E, the resulting graph is disconnected.
4. G is connected, and $|E| = |V| - 1$.
5. G is acyclic, and $|E| = |V| - 1$.
6. G is acyclic, but if any edge is added to E, the resulting graph contains a cycle.

Proof (1) $\Rightarrow$ (2): Since a tree is connected, any two vertices in G are connected by at least one simple path. Suppose, for the sake of contradiction, that vertices u and v are connected by two distinct simple paths p_1 and p_2, as shown in Figure B.5. Let w be the vertex at which the paths first diverge; that is, w is the first vertex on both p_1 and p_2 whose successor on p_1 is x and whose successor on p_2 is y, where $x \neq y$. Let z be the first vertex at which the paths reconverge; that is, z is the first vertex following w on p_1 that is also on p_2. Let p' be the subpath of p_1 from w through x to z, and let p'' be the subpath of p_2 from w through y to z. Paths p' and p'' share no vertices except their endpoints. Thus, the path obtained by concatenating p' and the reverse of p'' is a cycle, which contradicts our assumption

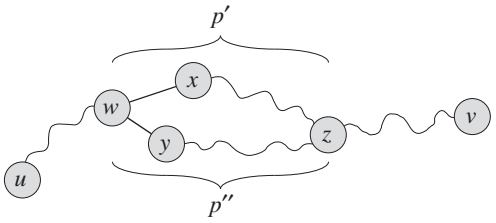


Figure B.5 A step in the proof of Theorem B.2: if (1) G is a free tree, then (2) any two vertices in G are connected by a unique simple path. Assume for the sake of contradiction that vertices u and v are connected by two distinct simple paths p_1 and p_2. These paths first diverge at vertex w, and they first reconverge at vertex z. The path p' concatenated with the reverse of the path p'' forms a cycle, which yields the contradiction.

that G is a tree. Thus, if G is a tree, there can be at most one simple path between two vertices.

(2) $\Rightarrow$ (3): If any two vertices in G are connected by a unique simple path, then G is connected. Let (u, v) be any edge in E. This edge is a path from u to v, and so it must be the unique path from u to v. If we remove (u, v) from G, there is no path from u to v, and hence its removal disconnects G.

(3) $\Rightarrow$ (4): By assumption, the graph G is connected, and by Exercise B.4-3, we have $|E| \geq |V| - 1$. We shall prove $|E| \leq |V| - 1$ by induction. A connected graph with $n = 1$ or $n = 2$ vertices has $n - 1$ edges. Suppose that G has $n \geq 3$ vertices and that all graphs satisfying (3) with fewer than n vertices also satisfy $|E| \leq |V| - 1$. Removing an arbitrary edge from G separates the graph into $k \geq 2$ connected components (actually $k = 2$). Each component satisfies (3), or else G would not satisfy (3). If we view each connected component V_i, with edge set E_i, as its own free tree, then because each component has fewer than $|V|$ vertices, by the inductive hypothesis we have $|E_i| \leq |V_i| - 1$. Thus, the number of edges in all components combined is at most $|V| - k \leq |V| - 2$. Adding in the removed edge yields $|E| \leq |V| - 1$.

(4) $\Rightarrow$ (5): Suppose that G is connected and that $|E| = |V| - 1$. We must show that G is acyclic. Suppose that G has a cycle containing k vertices $v_1, v_2, \ldots, v_k$, and without loss of generality assume that this cycle is simple. Let $G_k = (V_k, E_k)$ be the subgraph of G consisting of the cycle. Note that $|V_k| = |E_k| = k$. If $k < |V|$, there must be a vertex $v_{k+1} \in V - V_k$ that is adjacent to some vertex $v_i \in V_k$, since G is connected. Define $G_{k+1} = (V_{k+1}, E_{k+1})$ to be the subgraph of G with $V_{k+1} = V_k \cup \{v_{k+1}\}$ and $E_{k+1} = E_k \cup \{(v_i, v_{k+1})\}$. Note that $|V_{k+1}| = |E_{k+1}| = k + 1$. If $k + 1 < |V|$, we can continue, defining G_{k+2} in the same manner, and so forth, until we obtain $G_n = (V_n, E_n)$, where $n = |V|$,

$V_n = V$, and $|E_n| = |V_n| = |V|$. Since G_n is a subgraph of G, we have $E_n \subseteq E$, and hence $|E| \geq |V|$, which contradicts the assumption that $|E| = |V| - 1$. Thus, G is acyclic.

(5) $\Rightarrow$ (6): Suppose that G is acyclic and that $|E| = |V| - 1$. Let k be the number of connected components of G. Each connected component is a free tree by definition, and since (1) implies (5), the sum of all edges in all connected components of G is $|V| - k$. Consequently, we must have $k = 1$, and G is in fact a tree. Since (1) implies (2), any two vertices in G are connected by a unique simple path. Thus, adding any edge to G creates a cycle.

(6) $\Rightarrow$ (1): Suppose that G is acyclic but that adding any edge to E creates a cycle. We must show that G is connected. Let u and v be arbitrary vertices in G. If u and v are not already adjacent, adding the edge (u, v) creates a cycle in which all edges but (u, v) belong to G. Thus, the cycle minus edge (u, v) must contain a path from u to v, and since u and v were chosen arbitrarily, G is connected. ■

B.5.2 Rooted and ordered trees

A ***rooted tree*** is a free tree in which one of the vertices is distinguished from the others. We call the distinguished vertex the ***root*** of the tree. We often refer to a vertex of a rooted tree as a ***node***[5] of the tree. Figure B.6(a) shows a rooted tree on a set of 12 nodes with root 7.

Consider a node x in a rooted tree T with root r. We call any node y on the unique simple path from r to x an ***ancestor*** of x. If y is an ancestor of x, then x is a ***descendant*** of y. (Every node is both an ancestor and a descendant of itself.) If y is an ancestor of x and $x \neq y$, then y is a ***proper ancestor*** of x and x is a ***proper descendant*** of y. The ***subtree rooted at x*** is the tree induced by descendants of x, rooted at x. For example, the subtree rooted at node 8 in Figure B.6(a) contains nodes 8, 6, 5, and 9.

If the last edge on the simple path from the root r of a tree T to a node x is (y, x), then y is the ***parent*** of x, and x is a ***child*** of y. The root is the only node in T with no parent. If two nodes have the same parent, they are ***siblings***. A node with no children is a ***leaf*** or ***external node***. A nonleaf node is an ***internal node***.

---

[5]The term "node" is often used in the graph theory literature as a synonym for "vertex." We reserve the term "node" to mean a vertex of a rooted tree.

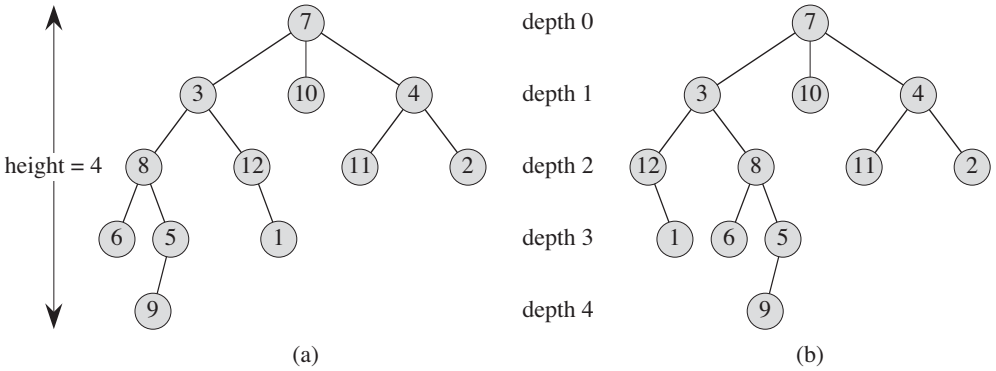


Figure B.6 Rooted and ordered trees. **(a)** A rooted tree with height 4. The tree is drawn in a standard way: the root (node 7) is at the top, its children (nodes with depth 1) are beneath it, their children (nodes with depth 2) are beneath them, and so forth. If the tree is ordered, the relative left-to-right order of the children of a node matters; otherwise it doesn't. **(b)** Another rooted tree. As a rooted tree, it is identical to the tree in (a), but as an ordered tree it is different, since the children of node 3 appear in a different order.

The number of children of a node x in a rooted tree T equals the ***degree*** of x.[6] The length of the simple path from the root r to a node x is the ***depth*** of x in T. A ***level*** of a tree consists of all nodes at the same depth. The ***height*** of a node in a tree is the number of edges on the longest simple downward path from the node to a leaf, and the height of a tree is the height of its root. The height of a tree is also equal to the largest depth of any node in the tree.

An ***ordered tree*** is a rooted tree in which the children of each node are ordered. That is, if a node has k children, then there is a first child, a second child, ..., and a kth child. The two trees in Figure B.6 are different when considered to be ordered trees, but the same when considered to be just rooted trees.

B.5.3 Binary and positional trees

We define binary trees recursively. A ***binary tree*** T is a structure defined on a finite set of nodes that either

- contains no nodes, or

[6] Notice that the degree of a node depends on whether we consider T to be a rooted tree or a free tree. The degree of a vertex in a free tree is, as in any undirected graph, the number of adjacent vertices. In a rooted tree, however, the degree is the number of children—the parent of a node does not count toward its degree.

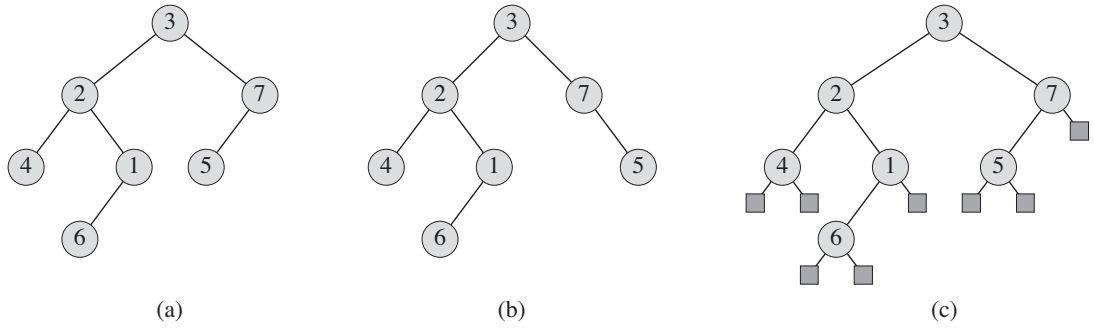


Figure B.7 Binary trees. **(a)** A binary tree drawn in a standard way. The left child of a node is drawn beneath the node and to the left. The right child is drawn beneath and to the right. **(b)** A binary tree different from the one in (a). In (a), the left child of node 7 is 5 and the right child is absent. In (b), the left child of node 7 is absent and the right child is 5. As ordered trees, these trees are the same, but as binary trees, they are distinct. **(c)** The binary tree in (a) represented by the internal nodes of a full binary tree: an ordered tree in which each internal node has degree 2. The leaves in the tree are shown as squares.

- is composed of three disjoint sets of nodes: a ***root*** node, a binary tree called its ***left subtree***, and a binary tree called its ***right subtree***.

The binary tree that contains no nodes is called the ***empty tree*** or ***null tree***, sometimes denoted NIL. If the left subtree is nonempty, its root is called the ***left child*** of the root of the entire tree. Likewise, the root of a nonnull right subtree is the ***right child*** of the root of the entire tree. If a subtree is the null tree NIL, we say that the child is ***absent*** or ***missing***. Figure B.7(a) shows a binary tree.

A binary tree is not simply an ordered tree in which each node has degree at most 2. For example, in a binary tree, if a node has just one child, the position of the child—whether it is the ***left child*** or the ***right child***—matters. In an ordered tree, there is no distinguishing a sole child as being either left or right. Figure B.7(b) shows a binary tree that differs from the tree in Figure B.7(a) because of the position of one node. Considered as ordered trees, however, the two trees are identical.

We can represent the positioning information in a binary tree by the internal nodes of an ordered tree, as shown in Figure B.7(c). The idea is to replace each missing child in the binary tree with a node having no children. These leaf nodes are drawn as squares in the figure. The tree that results is a ***full binary tree***: each node is either a leaf or has degree exactly 2. There are no degree-1 nodes. Consequently, the order of the children of a node preserves the position information.

We can extend the positioning information that distinguishes binary trees from ordered trees to trees with more than 2 children per node. In a ***positional tree***, the

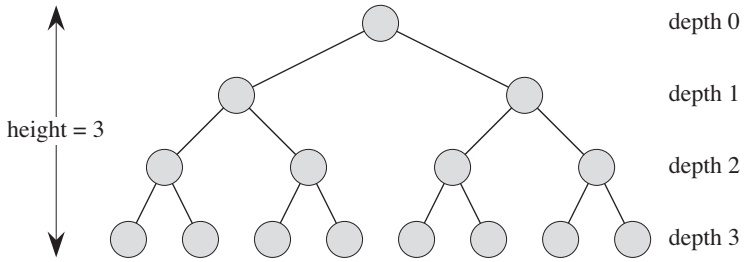


Figure B.8 A complete binary tree of height 3 with 8 leaves and 7 internal nodes.

children of a node are labeled with distinct positive integers. The ith child of a node is ***absent*** if no child is labeled with integer i. A ***k-ary*** tree is a positional tree in which for every node, all children with labels greater than k are missing. Thus, a binary tree is a k-ary tree with $k = 2$.

A ***complete k-ary tree*** is a k-ary tree in which all leaves have the same depth and all internal nodes have degree k. Figure B.8 shows a complete binary tree of height 3. How many leaves does a complete k-ary tree of height h have? The root has k children at depth 1, each of which has k children at depth 2, etc. Thus, the number of leaves at depth h is k^h. Consequently, the height of a complete k-ary tree with n leaves is $\log_k n$. The number of internal nodes of a complete k-ary tree of height h is

$$\begin{aligned} 1 + k + k^2 + \cdots + k^{h-1} &= \sum_{i=0}^{h-1} k^i \\ &= \frac{k^h - 1}{k - 1} \end{aligned}$$

by equation (A.5). Thus, a complete binary tree has $2^h - 1$ internal nodes.

Exercises

B.5-1

Draw all the free trees composed of the three vertices x, y, and z. Draw all the rooted trees with nodes x, y, and z with x as the root. Draw all the ordered trees with nodes x, y, and z with x as the root. Draw all the binary trees with nodes x, y, and z with x as the root.

B.5-2
Let $G = (V, E)$ be a directed acyclic graph in which there is a vertex $v_0 \in V$ such that there exists a unique path from v_0 to every vertex $v \in V$. Prove that the undirected version of G forms a tree.

B.5-3
Show by induction that the number of degree-2 nodes in any nonempty binary tree is 1 fewer than the number of leaves. Conclude that the number of internal nodes in a full binary tree is 1 fewer than the number of leaves.

B.5-4
Use induction to show that a nonempty binary tree with n nodes has height at least $\lfloor \lg n \rfloor$.

B.5-5 ⋆
The ***internal path length*** of a full binary tree is the sum, taken over all internal nodes of the tree, of the depth of each node. Likewise, the ***external path length*** is the sum, taken over all leaves of the tree, of the depth of each leaf. Consider a full binary tree with n internal nodes, internal path length i, and external path length e. Prove that $e = i + 2n$.

B.5-6 ⋆
Let us associate a "weight" $w(x) = 2^{-d}$ with each leaf x of depth d in a binary tree T, and let L be the set of leaves of T. Prove that $\sum_{x \in L} w(x) \leq 1$. (This is known as the ***Kraft inequality***.)

B.5-7 ⋆
Show that if $L \geq 2$, then every binary tree with L leaves contains a subtree having between $L/3$ and $2L/3$ leaves, inclusive.

Problems

B-1 Graph coloring
Given an undirected graph $G = (V, E)$, a ***k-coloring*** of G is a function $c : V \to \{0, 1, \ldots, k-1\}$ such that $c(u) \neq c(v)$ for every edge $(u, v) \in E$. In other words, the numbers $0, 1, \ldots, k-1$ represent the k colors, and adjacent vertices must have different colors.

a. Show that any tree is 2-colorable.

b. Show that the following are equivalent:

1. G is bipartite.
2. G is 2-colorable.
3. G has no cycles of odd length.

c. Let d be the maximum degree of any vertex in a graph G. Prove that we can color G with $d + 1$ colors.

d. Show that if G has $O(|V|)$ edges, then we can color G with $O(\sqrt{|V|})$ colors.

B-2 Friendly graphs

Reword each of the following statements as a theorem about undirected graphs, and then prove it. Assume that friendship is symmetric but not reflexive.

a. Any group of at least two people contains at least two people with the same number of friends in the group.

b. Every group of six people contains either at least three mutual friends or at least three mutual strangers.

c. Any group of people can be partitioned into two subgroups such that at least half the friends of each person belong to the subgroup of which that person is *not* a member.

d. If everyone in a group is the friend of at least half the people in the group, then the group can be seated around a table in such a way that everyone is seated between two friends.

B-3 Bisecting trees

Many divide-and-conquer algorithms that operate on graphs require that the graph be bisected into two nearly equal-sized subgraphs, which are induced by a partition of the vertices. This problem investigates bisections of trees formed by removing a small number of edges. We require that whenever two vertices end up in the same subtree after removing edges, then they must be in the same partition.

a. Show that we can partition the vertices of any n-vertex binary tree into two sets A and B, such that $|A| \leq 3n/4$ and $|B| \leq 3n/4$, by removing a single edge.

b. Show that the constant $3/4$ in part (a) is optimal in the worst case by giving an example of a simple binary tree whose most evenly balanced partition upon removal of a single edge has $|A| = 3n/4$.

c. Show that by removing at most $O(\lg n)$ edges, we can partition the vertices of any n-vertex binary tree into two sets A and B such that $|A| = \lfloor n/2 \rfloor$ and $|B| = \lceil n/2 \rceil$.

Appendix notes

G. Boole pioneered the development of symbolic logic, and he introduced many of the basic set notations in a book published in 1854. Modern set theory was created by G. Cantor during the period 1874–1895. Cantor focused primarily on sets of infinite cardinality. The term "function" is attributed to G. W. Leibniz, who used it to refer to several kinds of mathematical formulas. His limited definition has been generalized many times. Graph theory originated in 1736, when L. Euler proved that it was impossible to cross each of the seven bridges in the city of Königsberg exactly once and return to the starting point.

The book by Harary [160] provides a useful compendium of many definitions and results from graph theory.

C Counting and Probability

This appendix reviews elementary combinatorics and probability theory. If you have a good background in these areas, you may want to skim the beginning of this appendix lightly and concentrate on the later sections. Most of this book's chapters do not require probability, but for some chapters it is essential.

Section C.1 reviews elementary results in counting theory, including standard formulas for counting permutations and combinations. The axioms of probability and basic facts concerning probability distributions form Section C.2. Random variables are introduced in Section C.3, along with the properties of expectation and variance. Section C.4 investigates the geometric and binomial distributions that arise from studying Bernoulli trials. The study of the binomial distribution continues in Section C.5, an advanced discussion of the "tails" of the distribution.

C.1 Counting

Counting theory tries to answer the question "How many?" without actually enumerating all the choices. For example, we might ask, "How many different n-bit numbers are there?" or "How many orderings of n distinct elements are there?" In this section, we review the elements of counting theory. Since some of the material assumes a basic understanding of sets, you might wish to start by reviewing the material in Section B.1.

Rules of sum and product

We can sometimes express a set of items that we wish to count as a union of disjoint sets or as a Cartesian product of sets.

The ***rule of sum*** says that the number of ways to choose one element from one of two *disjoint* sets is the sum of the cardinalities of the sets. That is, if A and B are two finite sets with no members in common, then $|A \cup B| = |A| + |B|$, which

follows from equation (B.3). For example, each position on a car's license plate is a letter or a digit. The number of possibilities for each position is therefore $26 + 10 = 36$, since there are 26 choices if it is a letter and 10 choices if it is a digit.

The ***rule of product*** says that the number of ways to choose an ordered pair is the number of ways to choose the first element times the number of ways to choose the second element. That is, if A and B are two finite sets, then $|A \times B| = |A| \cdot |B|$, which is simply equation (B.4). For example, if an ice-cream parlor offers 28 flavors of ice cream and 4 toppings, the number of possible sundaes with one scoop of ice cream and one topping is $28 \cdot 4 = 112$.

Strings

A ***string*** over a finite set S is a sequence of elements of S. For example, there are 8 binary strings of length 3:

$$000, 001, 010, 011, 100, 101, 110, 111 .$$

We sometimes call a string of length k a ***k-string***. A ***substring*** s' of a string s is an ordered sequence of consecutive elements of s. A ***k-substring*** of a string is a substring of length k. For example, 010 is a 3-substring of 01101001 (the 3-substring that begins in position 4), but 111 is not a substring of 01101001.

We can view a k-string over a set S as an element of the Cartesian product S^k of k-tuples; thus, there are $|S|^k$ strings of length k. For example, the number of binary k-strings is 2^k. Intuitively, to construct a k-string over an n-set, we have n ways to pick the first element; for each of these choices, we have n ways to pick the second element; and so forth k times. This construction leads to the k-fold product $n \cdot n \cdots n = n^k$ as the number of k-strings.

Permutations

A ***permutation*** of a finite set S is an ordered sequence of all the elements of S, with each element appearing exactly once. For example, if $S = \{a, b, c\}$, then S has 6 permutations:

$$abc, acb, bac, bca, cab, cba .$$

There are $n!$ permutations of a set of n elements, since we can choose the first element of the sequence in n ways, the second in $n - 1$ ways, the third in $n - 2$ ways, and so on.

A ***k-permutation*** of S is an ordered sequence of k elements of S, with no element appearing more than once in the sequence. (Thus, an ordinary permutation is an n-permutation of an n-set.) The twelve 2-permutations of the set $\{a, b, c, d\}$ are

$ab, ac, ad, ba, bc, bd, ca, cb, cd, da, db, dc$.

The number of k-permutations of an n-set is

$$n(n-1)(n-2)\cdots(n-k+1) = \frac{n!}{(n-k)!}\,, \tag{C.1}$$

since we have n ways to choose the first element, $n-1$ ways to choose the second element, and so on, until we have selected k elements, the last being a selection from the remaining $n-k+1$ elements.

Combinations

A ***k*-combination** of an n-set S is simply a k-subset of S. For example, the 4-set $\{a, b, c, d\}$ has six 2-combinations:

ab, ac, ad, bc, bd, cd .

(Here we use the shorthand of denoting the 2-subset $\{a, b\}$ by ab, and so on.) We can construct a k-combination of an n-set by choosing k distinct (different) elements from the n-set. The order in which we select the elements does not matter.

We can express the number of k-combinations of an n-set in terms of the number of k-permutations of an n-set. Every k-combination has exactly $k!$ permutations of its elements, each of which is a distinct k-permutation of the n-set. Thus, the number of k-combinations of an n-set is the number of k-permutations divided by $k!$; from equation (C.1), this quantity is

$$\frac{n!}{k!\,(n-k)!}\,. \tag{C.2}$$

For $k = 0$, this formula tells us that the number of ways to choose 0 elements from an n-set is 1 (not 0), since $0! = 1$.

Binomial coefficients

The notation $\binom{n}{k}$ (read "n choose k") denotes the number of k-combinations of an n-set. From equation (C.2), we have

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}\,.$$

This formula is symmetric in k and $n-k$:

$$\binom{n}{k} = \binom{n}{n-k}\,. \tag{C.3}$$

These numbers are also known as ***binomial coefficients***, due to their appearance in the ***binomial expansion***:

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} . \tag{C.4}$$

A special case of the binomial expansion occurs when $x = y = 1$:

$$2^n = \sum_{k=0}^{n} \binom{n}{k} .$$

This formula corresponds to counting the 2^n binary n-strings by the number of 1s they contain: $\binom{n}{k}$ binary n-strings contain exactly k 1s, since we have $\binom{n}{k}$ ways to choose k out of the n positions in which to place the 1s.

Many identities involve binomial coefficients. The exercises at the end of this section give you the opportunity to prove a few.

Binomial bounds

We sometimes need to bound the size of a binomial coefficient. For $1 \le k \le n$, we have the lower bound

$$\begin{aligned}
\binom{n}{k} &= \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1} \\
&= \left(\frac{n}{k}\right)\left(\frac{n-1}{k-1}\right)\cdots\left(\frac{n-k+1}{1}\right) \\
&\ge \left(\frac{n}{k}\right)^k .
\end{aligned}$$

Taking advantage of the inequality $k! \ge (k/e)^k$ derived from Stirling's approximation (3.18), we obtain the upper bounds

$$\begin{aligned}
\binom{n}{k} &= \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1} \\
&\le \frac{n^k}{k!} \\
&\le \left(\frac{en}{k}\right)^k . \qquad \text{(C.5)}
\end{aligned}$$

For all integers k such that $0 \le k \le n$, we can use induction (see Exercise C.1-12) to prove the bound

$$\binom{n}{k} \leq \frac{n^n}{k^k (n-k)^{n-k}} , \tag{C.6}$$

where for convenience we assume that $0^0 = 1$. For $k = \lambda n$, where $0 \leq \lambda \leq 1$, we can rewrite this bound as

$$\begin{aligned} \binom{n}{\lambda n} &\leq \frac{n^n}{(\lambda n)^{\lambda n}((1-\lambda)n)^{(1-\lambda)n}} \\ &= \left(\left(\frac{1}{\lambda}\right)^{\lambda} \left(\frac{1}{1-\lambda}\right)^{1-\lambda}\right)^n \\ &= 2^{n\,H(\lambda)} , \end{aligned}$$

where

$$H(\lambda) = -\lambda \lg \lambda - (1-\lambda) \lg(1-\lambda) \tag{C.7}$$

is the ***(binary) entropy function*** and where, for convenience, we assume that $0 \lg 0 = 0$, so that $H(0) = H(1) = 0$.

Exercises

C.1-1

How many k-substrings does an n-string have? (Consider identical k-substrings at different positions to be different.) How many substrings does an n-string have in total?

C.1-2

An n-input, m-output ***boolean function*** is a function from $\{\text{TRUE}, \text{FALSE}\}^n$ to $\{\text{TRUE}, \text{FALSE}\}^m$. How many n-input, 1-output boolean functions are there? How many n-input, m-output boolean functions are there?

C.1-3

In how many ways can n professors sit around a circular conference table? Consider two seatings to be the same if one can be rotated to form the other.

C.1-4

In how many ways can we choose three distinct numbers from the set $\{1, 2, \ldots, 99\}$ so that their sum is even?

C.1-5
Prove the identity

$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1} \tag{C.8}$$

for $0 < k \leq n$.

C.1-6
Prove the identity

$$\binom{n}{k} = \frac{n}{n-k}\binom{n-1}{k}$$

for $0 \leq k < n$.

C.1-7
To choose k objects from n, you can make one of the objects distinguished and consider whether the distinguished object is chosen. Use this approach to prove that

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

C.1-8
Using the result of Exercise C.1-7, make a table for $n = 0, 1, \ldots, 6$ and $0 \leq k \leq n$ of the binomial coefficients $\binom{n}{k}$ with $\binom{0}{0}$ at the top, $\binom{1}{0}$ and $\binom{1}{1}$ on the next line, and so forth. Such a table of binomial coefficients is called ***Pascal's triangle***.

C.1-9
Prove that

$$\sum_{i=1}^{n} i = \binom{n+1}{2}.$$

C.1-10
Show that for any integers $n \geq 0$ and $0 \leq k \leq n$, the expression $\binom{n}{k}$ achieves its maximum value when $k = \lfloor n/2 \rfloor$ or $k = \lceil n/2 \rceil$.

C.1-11 ⋆
Argue that for any integers $n \geq 0$, $j \geq 0$, $k \geq 0$, and $j + k \leq n$,

$$\binom{n}{j+k} \leq \binom{n}{j}\binom{n-j}{k}. \tag{C.9}$$

Provide both an algebraic proof and an argument based on a method for choosing $j + k$ items out of n. Give an example in which equality does not hold.

C.1-12 ⋆
Use induction on all integers k such that $0 \leq k \leq n/2$ to prove inequality (C.6), and use equation (C.3) to extend it to all integers k such that $0 \leq k \leq n$.

C.1-13 ⋆
Use Stirling's approximation to prove that

$$\binom{2n}{n} = \frac{2^{2n}}{\sqrt{\pi n}}(1 + O(1/n)) \; . \tag{C.10}$$

C.1-14 ⋆
By differentiating the entropy function $H(\lambda)$, show that it achieves its maximum value at $\lambda = 1/2$. What is $H(1/2)$?

C.1-15 ⋆
Show that for any integer $n \geq 0$,

$$\sum_{k=0}^{n} \binom{n}{k} k = n2^{n-1} \; . \tag{C.11}$$

C.2 Probability

Probability is an essential tool for the design and analysis of probabilistic and randomized algorithms. This section reviews basic probability theory.

We define probability in terms of a ***sample space*** S, which is a set whose elements are called ***elementary events***. We can think of each elementary event as a possible outcome of an experiment. For the experiment of flipping two distinguishable coins, with each individual flip resulting in a head (H) or a tail (T), we can view the sample space as consisting of the set of all possible 2-strings over $\{\text{H}, \text{T}\}$:

$$S = \{\text{HH}, \text{HT}, \text{TH}, \text{TT}\} \; .$$

An ***event*** is a subset[1] of the sample space S. For example, in the experiment of flipping two coins, the event of obtaining one head and one tail is $\{\text{HT}, \text{TH}\}$. The event S is called the ***certain event***, and the event $\emptyset$ is called the ***null event***. We say that two events A and B are ***mutually exclusive*** if $A \cap B = \emptyset$. We sometimes treat an elementary event $s \in S$ as the event $\{s\}$. By definition, all elementary events are mutually exclusive.

Axioms of probability

A ***probability distribution*** $\Pr\{\}$ on a sample space S is a mapping from events of S to real numbers satisfying the following ***probability axioms***:

1. $\Pr\{A\} \geq 0$ for any event A.
2. $\Pr\{S\} = 1$.
3. $\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\}$ for any two mutually exclusive events A and B. More generally, for any (finite or countably infinite) sequence of events $A_1, A_2, \ldots$ that are pairwise mutually exclusive,

$$\Pr\left\{\bigcup_i A_i\right\} = \sum_i \Pr\{A_i\} \ .$$

We call $\Pr\{A\}$ the ***probability*** of the event A. We note here that axiom 2 is a normalization requirement: there is really nothing fundamental about choosing 1 as the probability of the certain event, except that it is natural and convenient.

Several results follow immediately from these axioms and basic set theory (see Section B.1). The null event $\emptyset$ has probability $\Pr\{\emptyset\} = 0$. If $A \subseteq B$, then $\Pr\{A\} \leq \Pr\{B\}$. Using $\overline{A}$ to denote the event $S - A$ (the ***complement*** of A), we have $\Pr\{\overline{A}\} = 1 - \Pr\{A\}$. For any two events A and B,

$$\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\} - \Pr\{A \cap B\} \tag{C.12}$$

$$\leq \Pr\{A\} + \Pr\{B\} \ . \tag{C.13}$$

---

[1] For a general probability distribution, there may be some subsets of the sample space S that are not considered to be events. This situation usually arises when the sample space is uncountably infinite. The main requirement for what subsets are events is that the set of events of a sample space be closed under the operations of taking the complement of an event, forming the union of a finite or countable number of events, and taking the intersection of a finite or countable number of events. Most of the probability distributions we shall see are over finite or countable sample spaces, and we shall generally consider all subsets of a sample space to be events. A notable exception is the continuous uniform probability distribution, which we shall see shortly.

In our coin-flipping example, suppose that each of the four elementary events has probability $1/4$. Then the probability of getting at least one head is

$$\begin{aligned}\Pr\{\text{HH}, \text{HT}, \text{TH}\} &= \Pr\{\text{HH}\} + \Pr\{\text{HT}\} + \Pr\{\text{TH}\} \\ &= 3/4 \ .\end{aligned}$$

Alternatively, since the probability of getting strictly less than one head is $\Pr\{\text{TT}\} = 1/4$, the probability of getting at least one head is $1 - 1/4 = 3/4$.

Discrete probability distributions

A probability distribution is ***discrete*** if it is defined over a finite or countably infinite sample space. Let S be the sample space. Then for any event A,

$$\Pr\{A\} = \sum_{s \in A} \Pr\{s\} \ ,$$

since elementary events, specifically those in A, are mutually exclusive. If S is finite and every elementary event $s \in S$ has probability

$$\Pr\{s\} = 1/|S| \ ,$$

then we have the ***uniform probability distribution*** on S. In such a case the experiment is often described as "picking an element of S at random."

As an example, consider the process of flipping a ***fair coin***, one for which the probability of obtaining a head is the same as the probability of obtaining a tail, that is, $1/2$. If we flip the coin n times, we have the uniform probability distribution defined on the sample space $S = \{\text{H}, \text{T}\}^n$, a set of size 2^n. We can represent each elementary event in S as a string of length n over $\{\text{H}, \text{T}\}$, each string occurring with probability $1/2^n$. The event

$$A = \{\text{exactly } k \text{ heads and exactly } n - k \text{ tails occur}\}$$

is a subset of S of size $|A| = \binom{n}{k}$, since $\binom{n}{k}$ strings of length n over $\{\text{H}, \text{T}\}$ contain exactly k H's. The probability of event A is thus $\Pr\{A\} = \binom{n}{k}/2^n$.

Continuous uniform probability distribution

The continuous uniform probability distribution is an example of a probability distribution in which not all subsets of the sample space are considered to be events. The continuous uniform probability distribution is defined over a closed interval $[a, b]$ of the reals, where $a < b$. Our intuition is that each point in the interval $[a, b]$ should be "equally likely." There are an uncountable number of points, however, so if we give all points the same finite, positive probability, we cannot simultaneously satisfy axioms 2 and 3. For this reason, we would like to associate a

probability only with *some* of the subsets of S, in such a way that the axioms are satisfied for these events.

For any closed interval $[c, d]$, where $a \le c \le d \le b$, the ***continuous uniform probability distribution*** defines the probability of the event $[c, d]$ to be

$$\Pr\{[c, d]\} = \frac{d - c}{b - a} .$$

Note that for any point $x = [x, x]$, the probability of x is 0. If we remove the endpoints of an interval $[c, d]$, we obtain the open interval (c, d). Since $[c, d] = [c, c] \cup (c, d) \cup [d, d]$, axiom 3 gives us $\Pr\{[c, d]\} = \Pr\{(c, d)\}$. Generally, the set of events for the continuous uniform probability distribution contains any subset of the sample space $[a, b]$ that can be obtained by a finite or countable union of open and closed intervals, as well as certain more complicated sets.

Conditional probability and independence

Sometimes we have some prior partial knowledge about the outcome of an experiment. For example, suppose that a friend has flipped two fair coins and has told you that at least one of the coins showed a head. What is the probability that both coins are heads? The information given eliminates the possibility of two tails. The three remaining elementary events are equally likely, so we infer that each occurs with probability $1/3$. Since only one of these elementary events shows two heads, the answer to our question is $1/3$.

Conditional probability formalizes the notion of having prior partial knowledge of the outcome of an experiment. The ***conditional probability*** of an event A given that another event B occurs is defined to be

$$\Pr\{A \mid B\} = \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \tag{C.14}$$

whenever $\Pr\{B\} \neq 0$. (We read "$\Pr\{A \mid B\}$" as "the probability of A given B.") Intuitively, since we are given that event B occurs, the event that A also occurs is $A \cap B$. That is, $A \cap B$ is the set of outcomes in which both A and B occur. Because the outcome is one of the elementary events in B, we normalize the probabilities of all the elementary events in B by dividing them by $\Pr\{B\}$, so that they sum to 1. The conditional probability of A given B is, therefore, the ratio of the probability of event $A \cap B$ to the probability of event B. In the example above, A is the event that both coins are heads, and B is the event that at least one coin is a head. Thus, $\Pr\{A \mid B\} = (1/4)/(3/4) = 1/3$.

Two events are ***independent*** if

$$\Pr\{A \cap B\} = \Pr\{A\} \Pr\{B\} , \tag{C.15}$$

which is equivalent, if $\Pr\{B\} \neq 0$, to the condition

$$\Pr\{A \mid B\} = \Pr\{A\} \ .$$

For example, suppose that we flip two fair coins and that the outcomes are independent. Then the probability of two heads is $(1/2)(1/2) = 1/4$. Now suppose that one event is that the first coin comes up heads and the other event is that the coins come up differently. Each of these events occurs with probability $1/2$, and the probability that both events occur is $1/4$; thus, according to the definition of independence, the events are independent—even though you might think that both events depend on the first coin. Finally, suppose that the coins are welded together so that they both fall heads or both fall tails and that the two possibilities are equally likely. Then the probability that each coin comes up heads is $1/2$, but the probability that they both come up heads is $1/2 \neq (1/2)(1/2)$. Consequently, the event that one comes up heads and the event that the other comes up heads are not independent.

A collection $A_1, A_2, \ldots, A_n$ of events is said to be ***pairwise independent*** if

$$\Pr\{A_i \cap A_j\} = \Pr\{A_i\}\Pr\{A_j\}$$

for all $1 \leq i < j \leq n$. We say that the events of the collection are ***(mutually) independent*** if every k-subset $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ of the collection, where $2 \leq k \leq n$ and $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, satisfies

$$\Pr\{A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}\} = \Pr\{A_{i_1}\}\Pr\{A_{i_2}\}\cdots\Pr\{A_{i_k}\} \ .$$

For example, suppose we flip two fair coins. Let A_1 be the event that the first coin is heads, let A_2 be the event that the second coin is heads, and let A_3 be the event that the two coins are different. We have

$$\begin{aligned}
\Pr\{A_1\} &= 1/2 \ , \\
\Pr\{A_2\} &= 1/2 \ , \\
\Pr\{A_3\} &= 1/2 \ , \\
\Pr\{A_1 \cap A_2\} &= 1/4 \ , \\
\Pr\{A_1 \cap A_3\} &= 1/4 \ , \\
\Pr\{A_2 \cap A_3\} &= 1/4 \ , \\
\Pr\{A_1 \cap A_2 \cap A_3\} &= 0 \ .
\end{aligned}$$

Since for $1 \leq i < j \leq 3$, we have $\Pr\{A_i \cap A_j\} = \Pr\{A_i\}\Pr\{A_j\} = 1/4$, the events A_1, A_2, and A_3 are pairwise independent. The events are not mutually independent, however, because $\Pr\{A_1 \cap A_2 \cap A_3\} = 0$ and $\Pr\{A_1\}\Pr\{A_2\}\Pr\{A_3\} = 1/8 \neq 0$.

Bayes's theorem

From the definition of conditional probability (C.14) and the commutative law $A \cap B = B \cap A$, it follows that for two events A and B, each with nonzero probability,

$$\begin{aligned} \Pr\{A \cap B\} &= \Pr\{B\} \Pr\{A \mid B\} \\ &= \Pr\{A\} \Pr\{B \mid A\} . \end{aligned} \tag{C.16}$$

Solving for $\Pr\{A \mid B\}$, we obtain

$$\Pr\{A \mid B\} = \frac{\Pr\{A\} \Pr\{B \mid A\}}{\Pr\{B\}} , \tag{C.17}$$

which is known as ***Bayes's theorem***. The denominator $\Pr\{B\}$ is a normalizing constant, which we can reformulate as follows. Since $B = (B \cap A) \cup (B \cap \overline{A})$, and since $B \cap A$ and $B \cap \overline{A}$ are mutually exclusive events,

$$\begin{aligned} \Pr\{B\} &= \Pr\{B \cap A\} + \Pr\left\{B \cap \overline{A}\right\} \\ &= \Pr\{A\} \Pr\{B \mid A\} + \Pr\left\{\overline{A}\right\} \Pr\left\{B \mid \overline{A}\right\} . \end{aligned}$$

Substituting into equation (C.17), we obtain an equivalent form of Bayes's theorem:

$$\Pr\{A \mid B\} = \frac{\Pr\{A\} \Pr\{B \mid A\}}{\Pr\{A\} \Pr\{B \mid A\} + \Pr\left\{\overline{A}\right\} \Pr\left\{B \mid \overline{A}\right\}} . \tag{C.18}$$

Bayes's theorem can simplify the computing of conditional probabilities. For example, suppose that we have a fair coin and a biased coin that always comes up heads. We run an experiment consisting of three independent events: we choose one of the two coins at random, we flip that coin once, and then we flip it again. Suppose that the coin we have chosen comes up heads both times. What is the probability that it is biased?

We solve this problem using Bayes's theorem. Let A be the event that we choose the biased coin, and let B be the event that the chosen coin comes up heads both times. We wish to determine $\Pr\{A \mid B\}$. We have $\Pr\{A\} = 1/2$, $\Pr\{B \mid A\} = 1$, $\Pr\left\{\overline{A}\right\} = 1/2$, and $\Pr\left\{B \mid \overline{A}\right\} = 1/4$; hence,

$$\begin{aligned} \Pr\{A \mid B\} &= \frac{(1/2) \cdot 1}{(1/2) \cdot 1 + (1/2) \cdot (1/4)} \\ &= 4/5 . \end{aligned}$$

Exercises

C.2-1

Professor Rosencrantz flips a fair coin once. Professor Guildenstern flips a fair coin twice. What is the probability that Professor Rosencrantz obtains more heads than Professor Guildenstern?

C.2-2
Prove ***Boole's inequality***: For any finite or countably infinite sequence of events $A_1, A_2, \ldots$,

$$\Pr\{A_1 \cup A_2 \cup \cdots\} \leq \Pr\{A_1\} + \Pr\{A_2\} + \cdots . \tag{C.19}$$

C.2-3
Suppose we shuffle a deck of 10 cards, each bearing a distinct number from 1 to 10, to mix the cards thoroughly. We then remove three cards, one at a time, from the deck. What is the probability that we select the three cards in sorted (increasing) order?

C.2-4
Prove that

$$\Pr\{A \mid B\} + \Pr\left\{\overline{A} \mid B\right\} = 1 .$$

C.2-5
Prove that for any collection of events $A_1, A_2, \ldots, A_n$,

$$\begin{aligned}\Pr\{A_1 \cap A_2 \cap \cdots \cap A_n\} = \Pr\{A_1\} \cdot \Pr\{A_2 \mid A_1\} \cdot \Pr\{A_3 \mid A_1 \cap A_2\} \cdots \\ \Pr\{A_n \mid A_1 \cap A_2 \cap \cdots \cap A_{n-1}\} .\end{aligned}$$

C.2-6 ⋆
Describe a procedure that takes as input two integers a and b such that $0 < a < b$ and, using fair coin flips, produces as output heads with probability a/b and tails with probability $(b - a)/b$. Give a bound on the expected number of coin flips, which should be $O(1)$. (*Hint:* Represent a/b in binary.)

C.2-7 ⋆
Show how to construct a set of n events that are pairwise independent but such that no subset of $k > 2$ of them is mutually independent.

C.2-8 ⋆
Two events A and B are ***conditionally independent***, given C, if

$$\Pr\{A \cap B \mid C\} = \Pr\{A \mid C\} \cdot \Pr\{B \mid C\} .$$

Give a simple but nontrivial example of two events that are not independent but are conditionally independent given a third event.

C.2-9 ⋆
You are a contestant in a game show in which a prize is hidden behind one of three curtains. You will win the prize if you select the correct curtain. After you

have picked one curtain but before the curtain is lifted, the emcee lifts one of the other curtains, knowing that it will reveal an empty stage, and asks if you would like to switch from your current selection to the remaining curtain. How would your chances change if you switch? (This question is the celebrated ***Monty Hall problem***, named after a game-show host who often presented contestants with just this dilemma.)

C.2-10 ⋆

A prison warden has randomly picked one prisoner among three to go free. The other two will be executed. The guard knows which one will go free but is forbidden to give any prisoner information regarding his status. Let us call the prisoners X, Y, and Z. Prisoner X asks the guard privately which of Y or Z will be executed, arguing that since he already knows that at least one of them must die, the guard won't be revealing any information about his own status. The guard tells X that Y is to be executed. Prisoner X feels happier now, since he figures that either he or prisoner Z will go free, which means that his probability of going free is now $1/2$. Is he right, or are his chances still $1/3$? Explain.

C.3 Discrete random variables

A ***(discrete) random variable*** X is a function from a finite or countably infinite sample space S to the real numbers. It associates a real number with each possible outcome of an experiment, which allows us to work with the probability distribution induced on the resulting set of numbers. Random variables can also be defined for uncountably infinite sample spaces, but they raise technical issues that are unnecessary to address for our purposes. Henceforth, we shall assume that random variables are discrete.

For a random variable X and a real number x, we define the event $X = x$ to be $\{s \in S : X(s) = x\}$; thus,

$$\Pr\{X = x\} = \sum_{s \in S : X(s) = x} \Pr\{s\} \ .$$

The function

$$f(x) = \Pr\{X = x\}$$

is the ***probability density function*** of the random variable X. From the probability axioms, $\Pr\{X = x\} \geq 0$ and $\sum_x \Pr\{X = x\} = 1$.

As an example, consider the experiment of rolling a pair of ordinary, 6-sided dice. There are 36 possible elementary events in the sample space. We assume

that the probability distribution is uniform, so that each elementary event $s \in S$ is equally likely: $\Pr\{s\} = 1/36$. Define the random variable X to be the *maximum* of the two values showing on the dice. We have $\Pr\{X = 3\} = 5/36$, since X assigns a value of 3 to 5 of the 36 possible elementary events, namely, $(1, 3)$, $(2, 3)$, $(3, 3)$, $(3, 2)$, and $(3, 1)$.

We often define several random variables on the same sample space. If X and Y are random variables, the function

$$f(x, y) = \Pr\{X = x \text{ and } Y = y\}$$

is the ***joint probability density function*** of X and Y. For a fixed value y,

$$\Pr\{Y = y\} = \sum_x \Pr\{X = x \text{ and } Y = y\} ,$$

and similarly, for a fixed value x,

$$\Pr\{X = x\} = \sum_y \Pr\{X = x \text{ and } Y = y\} .$$

Using the definition (C.14) of conditional probability, we have

$$\Pr\{X = x \mid Y = y\} = \frac{\Pr\{X = x \text{ and } Y = y\}}{\Pr\{Y = y\}} .$$

We define two random variables X and Y to be ***independent*** if for all x and y, the events $X = x$ and $Y = y$ are independent or, equivalently, if for all x and y, we have $\Pr\{X = x \text{ and } Y = y\} = \Pr\{X = x\}\Pr\{Y = y\}$.

Given a set of random variables defined over the same sample space, we can define new random variables as sums, products, or other functions of the original variables.

Expected value of a random variable

The simplest and most useful summary of the distribution of a random variable is the "average" of the values it takes on. The ***expected value*** (or, synonymously, ***expectation*** or ***mean***) of a discrete random variable X is

$$\mathrm{E}[X] = \sum_x x \cdot \Pr\{X = x\} , \tag{C.20}$$

which is well defined if the sum is finite or converges absolutely. Sometimes the expectation of X is denoted by μ_X or, when the random variable is apparent from context, simply by μ.

Consider a game in which you flip two fair coins. You earn \$3 for each head but lose \$2 for each tail. The expected value of the random variable X representing

your earnings is

$$\begin{aligned} \mathrm{E}[X] &= 6 \cdot \Pr\{2 \text{ H's}\} + 1 \cdot \Pr\{1 \text{ H}, 1 \text{ T}\} - 4 \cdot \Pr\{2 \text{ T's}\} \\ &= 6(1/4) + 1(1/2) - 4(1/4) \\ &= 1 \ . \end{aligned}$$

The expectation of the sum of two random variables is the sum of their expectations, that is,

$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y] \ , \tag{C.21}$$

whenever $\mathrm{E}[X]$ and $\mathrm{E}[Y]$ are defined. We call this property ***linearity of expectation***, and it holds even if X and Y are not independent. It also extends to finite and absolutely convergent summations of expectations. Linearity of expectation is the key property that enables us to perform probabilistic analyses by using indicator random variables (see Section 5.2).

If X is any random variable, any function $g(x)$ defines a new random variable $g(X)$. If the expectation of $g(X)$ is defined, then

$$\mathrm{E}[g(X)] = \sum_x g(x) \cdot \Pr\{X = x\} \ .$$

Letting $g(x) = ax$, we have for any constant a,

$$\mathrm{E}[aX] = a\mathrm{E}[X] \ . \tag{C.22}$$

Consequently, expectations are linear: for any two random variables X and Y and any constant a,

$$\mathrm{E}[aX + Y] = a\mathrm{E}[X] + \mathrm{E}[Y] \ . \tag{C.23}$$

When two random variables X and Y are independent and each has a defined expectation,

$$\begin{aligned} \mathrm{E}[XY] &= \sum_x \sum_y xy \cdot \Pr\{X = x \text{ and } Y = y\} \\ &= \sum_x \sum_y xy \cdot \Pr\{X = x\} \Pr\{Y = y\} \\ &= \left(\sum_x x \cdot \Pr\{X = x\}\right)\left(\sum_y y \cdot \Pr\{Y = y\}\right) \\ &= \mathrm{E}[X]\mathrm{E}[Y] \ . \end{aligned}$$

In general, when n random variables $X_1, X_2, \ldots, X_n$ are mutually independent,

$$\mathrm{E}[X_1 X_2 \cdots X_n] = \mathrm{E}[X_1]\mathrm{E}[X_2] \cdots \mathrm{E}[X_n] \ . \tag{C.24}$$

When a random variable X takes on values from the set of natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$, we have a nice formula for its expectation:

$$\begin{aligned} \mathrm{E}[X] &= \sum_{i=0}^{\infty} i \cdot \Pr\{X = i\} \\ &= \sum_{i=0}^{\infty} i \left(\Pr\{X \geq i\} - \Pr\{X \geq i+1\}\right) \\ &= \sum_{i=1}^{\infty} \Pr\{X \geq i\} \ , \end{aligned} \tag{C.25}$$

since each term $\Pr\{X \geq i\}$ is added in i times and subtracted out $i - 1$ times (except $\Pr\{X \geq 0\}$, which is added in 0 times and not subtracted out at all).

When we apply a convex function $f(x)$ to a random variable X, ***Jensen's inequality*** gives us

$$\mathrm{E}[f(X)] \geq f(\mathrm{E}[X]) \ , \tag{C.26}$$

provided that the expectations exist and are finite. (A function $f(x)$ is ***convex*** if for all x and y and for all $0 \leq \lambda \leq 1$, we have $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$.)

Variance and standard deviation

The expected value of a random variable does not tell us how "spread out" the variable's values are. For example, if we have random variables X and Y for which $\Pr\{X = 1/4\} = \Pr\{X = 3/4\} = 1/2$ and $\Pr\{Y = 0\} = \Pr\{Y = 1\} = 1/2$, then both $\mathrm{E}[X]$ and $\mathrm{E}[Y]$ are $1/2$, yet the actual values taken on by Y are farther from the mean than the actual values taken on by X.

The notion of variance mathematically expresses how far from the mean a random variable's values are likely to be. The ***variance*** of a random variable X with mean $\mathrm{E}[X]$ is

$$\begin{aligned} \mathrm{Var}[X] &= \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] \\ &= \mathrm{E}\left[X^2 - 2X\mathrm{E}[X] + \mathrm{E}^2[X]\right] \\ &= \mathrm{E}\left[X^2\right] - 2\mathrm{E}[X\mathrm{E}[X]] + \mathrm{E}^2[X] \\ &= \mathrm{E}\left[X^2\right] - 2\mathrm{E}^2[X] + \mathrm{E}^2[X] \\ &= \mathrm{E}\left[X^2\right] - \mathrm{E}^2[X] \ . \end{aligned} \tag{C.27}$$

To justify the equality $\mathrm{E}[\mathrm{E}^2[X]] = \mathrm{E}^2[X]$, note that because $\mathrm{E}[X]$ is a real number and not a random variable, so is $\mathrm{E}^2[X]$. The equality $\mathrm{E}[X\mathrm{E}[X]] = \mathrm{E}^2[X]$

follows from equation (C.22), with $a = \mathrm{E}[X]$. Rewriting equation (C.27) yields an expression for the expectation of the square of a random variable:

$$\mathrm{E}\left[X^2\right] = \mathrm{Var}[X] + \mathrm{E}^2[X] \ . \tag{C.28}$$

The variance of a random variable X and the variance of aX are related (see Exercise C.3-10):

$$\mathrm{Var}[aX] = a^2\mathrm{Var}[X] \ .$$

When X and Y are independent random variables,

$$\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] \ .$$

In general, if n random variables $X_1, X_2, \ldots, X_n$ are pairwise independent, then

$$\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{Var}[X_i] \ . \tag{C.29}$$

The ***standard deviation*** of a random variable X is the nonnegative square root of the variance of X. The standard deviation of a random variable X is sometimes denoted σ_X or simply σ when the random variable X is understood from context. With this notation, the variance of X is denoted σ^2.

Exercises

C.3-1

Suppose we roll two ordinary, 6-sided dice. What is the expectation of the sum of the two values showing? What is the expectation of the maximum of the two values showing?

C.3-2

An array $A[1 \ldots n]$ contains n distinct numbers that are randomly ordered, with each permutation of the n numbers being equally likely. What is the expectation of the index of the maximum element in the array? What is the expectation of the index of the minimum element in the array?

C.3-3

A carnival game consists of three dice in a cage. A player can bet a dollar on any of the numbers 1 through 6. The cage is shaken, and the payoff is as follows. If the player's number doesn't appear on any of the dice, he loses his dollar. Otherwise, if his number appears on exactly k of the three dice, for $k = 1, 2, 3$, he keeps his dollar and wins k more dollars. What is his expected gain from playing the carnival game once?

C.3-4
Argue that if X and Y are nonnegative random variables, then

$$\mathrm{E}[\max(X, Y)] \leq \mathrm{E}[X] + \mathrm{E}[Y] .$$

C.3-5 ⋆
Let X and Y be independent random variables. Prove that $f(X)$ and $g(Y)$ are independent for any choice of functions f and g.

C.3-6 ⋆
Let X be a nonnegative random variable, and suppose that $\mathrm{E}[X]$ is well defined. Prove ***Markov's inequality***:

$$\Pr\{X \geq t\} \leq \mathrm{E}[X]/t \tag{C.30}$$

for all $t > 0$.

C.3-7 ⋆
Let S be a sample space, and let X and X' be random variables such that $X(s) \geq X'(s)$ for all $s \in S$. Prove that for any real constant t,

$$\Pr\{X \geq t\} \geq \Pr\{X' \geq t\} .$$

C.3-8
Which is larger: the expectation of the square of a random variable, or the square of its expectation?

C.3-9
Show that for any random variable X that takes on only the values 0 and 1, we have $\mathrm{Var}[X] = \mathrm{E}[X]\,\mathrm{E}[1 - X]$.

C.3-10
Prove that $\mathrm{Var}[aX] = a^2\mathrm{Var}[X]$ from the definition (C.27) of variance.

C.4 The geometric and binomial distributions

We can think of a coin flip as an instance of a ***Bernoulli trial***, which is an experiment with only two possible outcomes: ***success***, which occurs with probability p, and ***failure***, which occurs with probability $q = 1 - p$. When we speak of ***Bernoulli trials*** collectively, we mean that the trials are mutually independent and, unless we specifically say otherwise, that each has the same probability p for success. Two

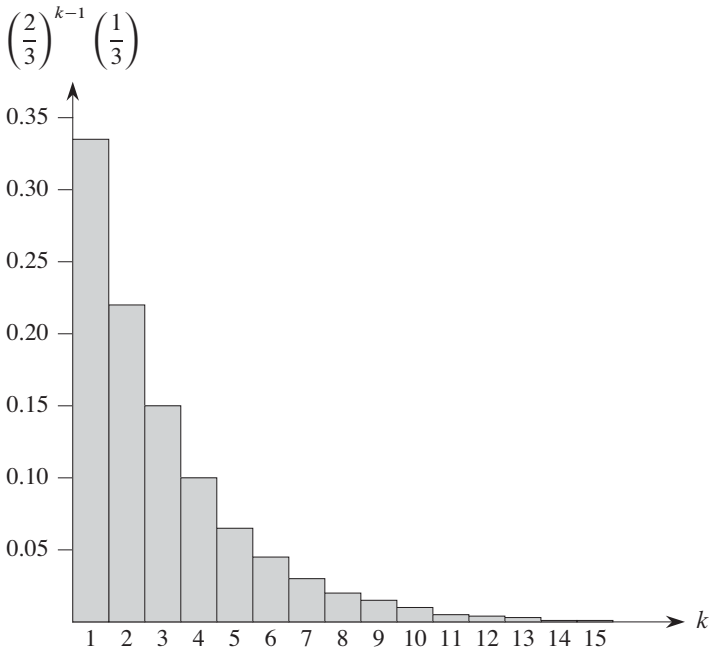


Figure C.1 A geometric distribution with probability $p = 1/3$ of success and a probability $q = 1 - p$ of failure. The expectation of the distribution is $1/p = 3$.

important distributions arise from Bernoulli trials: the geometric distribution and the binomial distribution.

The geometric distribution

Suppose we have a sequence of Bernoulli trials, each with a probability p of success and a probability $q = 1 - p$ of failure. How many trials occur before we obtain a success? Let us define the random variable X be the number of trials needed to obtain a success. Then X has values in the range $\{1, 2, \ldots\}$, and for $k \geq 1$,

$$\Pr\{X = k\} = q^{k-1} p \,, \tag{C.31}$$

since we have $k - 1$ failures before the one success. A probability distribution satisfying equation (C.31) is said to be a ***geometric distribution***. Figure C.1 illustrates such a distribution.

Assuming that $q < 1$, we can calculate the expectation of a geometric distribution using identity (A.8):

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{k=1}^{\infty} k q^{k-1} p \\
&= \frac{p}{q} \sum_{k=0}^{\infty} k q^k \\
&= \frac{p}{q} \cdot \frac{q}{(1-q)^2} \\
&= \frac{p}{q} \cdot \frac{q}{p^2} \\
&= 1/p \, . \qquad \text{(C.32)}
\end{aligned}$$

Thus, on average, it takes $1/p$ trials before we obtain a success, an intuitive result. The variance, which can be calculated similarly, but using Exercise A.1-3, is

$$\mathrm{Var}[X] = q/p^2 \, . \qquad \text{(C.33)}$$

As an example, suppose we repeatedly roll two dice until we obtain either a seven or an eleven. Of the 36 possible outcomes, 6 yield a seven and 2 yield an eleven. Thus, the probability of success is $p = 8/36 = 2/9$, and we must roll $1/p = 9/2 = 4.5$ times on average to obtain a seven or eleven.

The binomial distribution

How many successes occur during n Bernoulli trials, where a success occurs with probability p and a failure with probability $q = 1 - p$? Define the random variable X to be the number of successes in n trials. Then X has values in the range $\{0, 1, \ldots, n\}$, and for $k = 0, 1, \ldots, n$,

$$\Pr\{X = k\} = \binom{n}{k} p^k q^{n-k} \, , \qquad \text{(C.34)}$$

since there are $\binom{n}{k}$ ways to pick which k of the n trials are successes, and the probability that each occurs is $p^k q^{n-k}$. A probability distribution satisfying equation (C.34) is said to be a ***binomial distribution***. For convenience, we define the family of binomial distributions using the notation

$$b(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k} \, . \qquad \text{(C.35)}$$

Figure C.2 illustrates a binomial distribution. The name "binomial" comes from the right-hand side of equation (C.34) being the kth term of the expansion of $(p+q)^n$. Consequently, since $p + q = 1$,

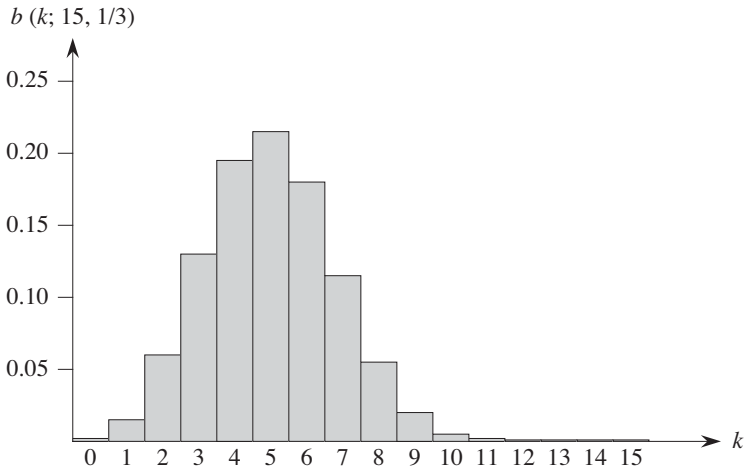


Figure C.2 The binomial distribution $b(k; 15, 1/3)$ resulting from $n = 15$ Bernoulli trials, each with probability $p = 1/3$ of success. The expectation of the distribution is $np = 5$.

$$\sum_{k=0}^{n} b(k; n, p) = 1 , \tag{C.36}$$

as axiom 2 of the probability axioms requires.

We can compute the expectation of a random variable having a binomial distribution from equations (C.8) and (C.36). Let X be a random variable that follows the binomial distribution $b(k; n, p)$, and let $q = 1 - p$. By the definition of expectation, we have

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{k=0}^{n} k \cdot \Pr\{X = k\} \\
&= \sum_{k=0}^{n} k \cdot b(k; n, p) \\
&= \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} \\
&= np \sum_{k=1}^{n} \binom{n-1}{k-1} p^{k-1} q^{n-k} \qquad \text{(by equation (C.8))} \\
&= np \sum_{k=0}^{n-1} \binom{n-1}{k} p^k q^{(n-1)-k}
\end{aligned}$$

$$\begin{aligned} &= np \sum_{k=0}^{n-1} b(k; n-1, p) \\ &= np \qquad \text{(by equation (C.36))} \ . \end{aligned} \tag{C.37}$$

By using the linearity of expectation, we can obtain the same result with substantially less algebra. Let X_i be the random variable describing the number of successes in the ith trial. Then $\mathrm{E}[X_i] = p \cdot 1 + q \cdot 0 = p$, and by linearity of expectation (equation (C.21)), the expected number of successes for n trials is

$$\begin{aligned} \mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] \\ &= \sum_{i=1}^{n} \mathrm{E}[X_i] \\ &= \sum_{i=1}^{n} p \\ &= np \ . \end{aligned} \tag{C.38}$$

We can use the same approach to calculate the variance of the distribution. Using equation (C.27), we have $\mathrm{Var}[X_i] = \mathrm{E}[X_i^2] - \mathrm{E}^2[X_i]$. Since X_i only takes on the values 0 and 1, we have $X_i^2 = X_i$, which implies $\mathrm{E}[X_i^2] = \mathrm{E}[X_i] = p$. Hence,

$$\mathrm{Var}[X_i] = p - p^2 = p(1-p) = pq \ . \tag{C.39}$$

To compute the variance of X, we take advantage of the independence of the n trials; thus, by equation (C.29),

$$\begin{aligned} \mathrm{Var}[X] &= \mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] \\ &= \sum_{i=1}^{n} \mathrm{Var}[X_i] \\ &= \sum_{i=1}^{n} pq \\ &= npq \ . \end{aligned} \tag{C.40}$$

As Figure C.2 shows, the binomial distribution $b(k; n, p)$ increases with k until it reaches the mean np, and then it decreases. We can prove that the distribution always behaves in this manner by looking at the ratio of successive terms:

$$\begin{aligned}\frac{b(k;n,p)}{b(k-1;n,p)} &= \frac{\binom{n}{k}p^k q^{n-k}}{\binom{n}{k-1}p^{k-1}q^{n-k+1}} \\ &= \frac{n!(k-1)!(n-k+1)!p}{k!(n-k)!n!q} \\ &= \frac{(n-k+1)p}{kq} \\ &= 1+\frac{(n+1)p-k}{kq}\,.\end{aligned} \tag{C.41}$$

This ratio is greater than 1 precisely when $(n+1)p-k$ is positive. Consequently, $b(k;n,p) > b(k-1;n,p)$ for $k < (n+1)p$ (the distribution increases), and $b(k;n,p) < b(k-1;n,p)$ for $k > (n+1)p$ (the distribution decreases). If $k = (n+1)p$ is an integer, then $b(k;n,p) = b(k-1;n,p)$, and so the distribution then has two maxima: at $k = (n+1)p$ and at $k-1 = (n+1)p-1 = np-q$. Otherwise, it attains a maximum at the unique integer k that lies in the range $np-q < k < (n+1)p$.

The following lemma provides an upper bound on the binomial distribution.

Lemma C.1
Let $n \geq 0$, let $0 < p < 1$, let $q = 1-p$, and let $0 \leq k \leq n$. Then

$$b(k;n,p) \leq \left(\frac{np}{k}\right)^k \left(\frac{nq}{n-k}\right)^{n-k}\,.$$

Proof Using equation (C.6), we have

$$\begin{aligned}b(k;n,p) &= \binom{n}{k}p^k q^{n-k} \\ &\leq \left(\frac{n}{k}\right)^k \left(\frac{n}{n-k}\right)^{n-k} p^k q^{n-k} \\ &= \left(\frac{np}{k}\right)^k \left(\frac{nq}{n-k}\right)^{n-k}\,.\end{aligned}$$

■

Exercises

C.4-1
Verify axiom 2 of the probability axioms for the geometric distribution.

C.4-2
How many times on average must we flip 6 fair coins before we obtain 3 heads and 3 tails?

C.4-3
Show that $b(k; n, p) = b(n - k; n, q)$, where $q = 1 - p$.

C.4-4
Show that value of the maximum of the binomial distribution $b(k; n, p)$ is approximately $1/\sqrt{2\pi npq}$, where $q = 1 - p$.

C.4-5 ⋆
Show that the probability of no successes in n Bernoulli trials, each with probability $p = 1/n$, is approximately $1/e$. Show that the probability of exactly one success is also approximately $1/e$.

C.4-6 ⋆
Professor Rosencrantz flips a fair coin n times, and so does Professor Guildenstern. Show that the probability that they get the same number of heads is $\binom{2n}{n}/4^n$. (*Hint:* For Professor Rosencrantz, call a head a success; for Professor Guildenstern, call a tail a success.) Use your argument to verify the identity

$$\sum_{k=0}^{n} \binom{n}{k}^2 = \binom{2n}{n} .$$

C.4-7 ⋆
Show that for $0 \le k \le n$,

$$b(k; n, 1/2) \le 2^{n\,H(k/n)-n} ,$$

where $H(x)$ is the entropy function (C.7).

C.4-8 ⋆
Consider n Bernoulli trials, where for $i = 1, 2, \ldots, n$, the ith trial has probability p_i of success, and let X be the random variable denoting the total number of successes. Let $p \ge p_i$ for all $i = 1, 2, \ldots, n$. Prove that for $1 \le k \le n$,

$$\Pr\{X < k\} \ge \sum_{i=0}^{k-1} b(i; n, p) .$$

C.4-9 ⋆
Let X be the random variable for the total number of successes in a set A of n Bernoulli trials, where the ith trial has a probability p_i of success, and let X' be the random variable for the total number of successes in a second set A' of n Bernoulli trials, where the ith trial has a probability $p'_i \ge p_i$ of success. Prove that for $0 \le k \le n$,

$$\Pr\{X' \geq k\} \geq \Pr\{X \geq k\} \ .$$

(*Hint:* Show how to obtain the Bernoulli trials in A' by an experiment involving the trials of A, and use the result of Exercise C.3-7.)

⋆ C.5 The tails of the binomial distribution

The probability of having at least, or at most, k successes in n Bernoulli trials, each with probability p of success, is often of more interest than the probability of having exactly k successes. In this section, we investigate the ***tails*** of the binomial distribution: the two regions of the distribution $b(k; n, p)$ that are far from the mean np. We shall prove several important bounds on (the sum of all terms in) a tail.

We first provide a bound on the right tail of the distribution $b(k; n, p)$. We can determine bounds on the left tail by inverting the roles of successes and failures.

Theorem C.2
Consider a sequence of n Bernoulli trials, where success occurs with probability p. Let X be the random variable denoting the total number of successes. Then for $0 \leq k \leq n$, the probability of at least k successes is

$$\begin{aligned}\Pr\{X \geq k\} &= \sum_{i=k}^{n} b(i; n, p) \\ &\leq \binom{n}{k} p^k \ .\end{aligned}$$

Proof For $S \subseteq \{1, 2, \ldots, n\}$, we let A_S denote the event that the ith trial is a success for every $i \in S$. Clearly $\Pr\{A_S\} = p^k$ if $|S| = k$. We have

$$\begin{aligned}\Pr\{X \geq k\} &= \Pr\{\text{there exists } S \subseteq \{1, 2, \ldots, n\} : |S| = k \text{ and } A_S\} \\ &= \Pr\left\{\bigcup_{S \subseteq \{1,2,\ldots,n\}:|S|=k} A_S\right\} \\ &\leq \sum_{S \subseteq \{1,2,\ldots,n\}:|S|=k} \Pr\{A_S\} \qquad \text{(by inequality (C.19))} \\ &= \binom{n}{k} p^k \ .\end{aligned}$$

■

The following corollary restates the theorem for the left tail of the binomial distribution. In general, we shall leave it to you to adapt the proofs from one tail to the other.

Corollary C.3
Consider a sequence of n Bernoulli trials, where success occurs with probability p. If X is the random variable denoting the total number of successes, then for $0 \le k \le n$, the probability of at most k successes is

$$\begin{aligned}\Pr\{X \le k\} &= \sum_{i=0}^{k} b(i;n,p) \\ &\le \binom{n}{n-k}(1-p)^{n-k} \\ &= \binom{n}{k}(1-p)^{n-k} .\end{aligned}$$

■

Our next bound concerns the left tail of the binomial distribution. Its corollary shows that, far from the mean, the left tail diminishes exponentially.

Theorem C.4
Consider a sequence of n Bernoulli trials, where success occurs with probability p and failure with probability $q = 1 - p$. Let X be the random variable denoting the total number of successes. Then for $0 < k < np$, the probability of fewer than k successes is

$$\begin{aligned}\Pr\{X < k\} &= \sum_{i=0}^{k-1} b(i;n,p) \\ &< \frac{kq}{np-k} b(k;n,p) .\end{aligned}$$

Proof We bound the series $\sum_{i=0}^{k-1} b(i;n,p)$ by a geometric series using the technique from Section A.2, page 1151. For $i = 1, 2, \ldots, k$, we have from equation (C.41),

$$\begin{aligned}\frac{b(i-1;n,p)}{b(i;n,p)} &= \frac{iq}{(n-i+1)p} \\ &< \frac{iq}{(n-i)p} \\ &\le \frac{kq}{(n-k)p} .\end{aligned}$$