\* \(2^{t-2}\) ways to get an odd sum on the first \(t - 1\) variables (by induction), followed by an assignment of 1 to the \(t^{th}\).

The remaining \(2^{t-1}\) assignments give an odd sum, and this completes the induction step.

Once we have \(E[X_t] = 1/2\), we conclude as in part (a): Linearity of expectation gives us \(E[X] = \sum E[X_t] = k/2 \geq c^*/2\).

**Exercises**

1. 3-Coloring is a yes/no question, but we can phrase it as an optimization problem as follows.

   Suppose we are given a graph \(G = (V, E)\), and we want to color each node with one of three colors, even if we aren't necessarily able to give different colors to every pair of adjacent nodes. Rather, we say that an edge \((u, v)\) is satisfied if the colors assigned to \(u\) and \(v\) are different.

   Consider a 3-coloring that maximizes the number of satisfied edges, and let \(c^*\) denote this number. Give a polynomial-time algorithm that produces a 3-coloring that satisfies at least \(\frac{3}{4}c^*\) edges. If you want, your algorithm can be randomized; in this case, the expected number of edges it satisfies should be at least \(\frac{3}{4}c^*\).

2. Consider a county in which 100,000 people vote in an election. There are only two candidates on the ballot: a Democratic candidate (denoted \(D\)) and a Republican candidate (denoted \(R\)). As it happens, this county is heavily Democratic, so 80,000 people go to the polls with the intention of voting for \(D\), and 20,000 go to the polls with the intention of voting for \(R\).

   However, the layout of the ballot is a little confusing, so each voter, independently and with probability \(\frac{3}{10}\), votes for the wrong candidate—that is, the one that he or she didn't intend to vote for. (Remember that in this election, there are only two candidates on the ballot.)

   Let \(X\) denote the random variable equal to the number of votes received by the Democratic candidate \(D\), when the voting is conducted with this process of error. Determine the expected value of \(X\), and give an explanation of your derivation of this value.

3. In Section 13.1, we saw a simple distributed protocol to solve a particular contention-resolution problem. Here is another setting in which randomization can help with contention resolution, through the distributed construction of an independent set.
In terms of the parameters of the graph $G$, give a value of $p$ so that the expected size of the resulting set $S$ is as large as possible. Give a formula for the expected size of $S$ when $p$ is set to this optimal value.

4. A number of peer-to-peer systems on the Internet are based on overlay networks. Rather than using the physical Internet topology as the network on which to perform computation, these systems run protocols by which nodes choose collections of virtual "neighbors" so as to define a higher-level graph whose structure may bear little or no relation to the underlying physical network. Such an overlay network is then used for sharing data and services, and it can be extremely flexible compared with a physical network, which is hard to modify in real time to adapt to changing conditions.

Peer-to-peer networks tend to grow through the arrival of new participants, who join by linking into the existing structure. This growth process has an intrinsic effect on the characteristics of the overall network. Recently, people have investigated simple abstract models for network growth that might provide insight into the way such processes behave, at a qualitative level, in real networks.

Here's a simple example of such a model. The system begins with a single node $v_1$. Nodes then join one at a time; as each node joins, it executes a protocol whereby it forms a directed link to a single other node chosen uniformly at random from those already in the system. More concretely, if the system already contains nodes $v_1, v_2, \ldots, v_{k-1}$ and node $v_k$ wishes to join, it randomly selects one of $v_1, v_2, \ldots, v_{k-1}$ and links to this node.

Suppose we run this process until we have a system consisting of nodes $v_1, v_2, \ldots, v_j$; the random process described above will produce a directed network in which each node other than $v_1$ has exactly one outgoing edge. On the other hand, a node may have multiple incoming links, or none at all. The incoming links to a node $v_j$ reflect all the other nodes whose access into the system is via $v_j$; so if $v_j$ has many incoming links, this can place a large load on it. To keep the system load-balanced, then, we'd like all nodes to have a roughly comparable number of incoming links. That's unlikely to happen here, however, since nodes that join earlier in the process are likely to have more incoming links than nodes that join later. Let's try to quantify this imbalance as follows.

(a) Given the random process described above, what is the expected number of incoming links to node $v_j$ in the resulting network? Give an exact formula in terms of $n$ and $j$, and also try to express this quantity
Fig. 13.6 Towns $T_1, T_2, \ldots, T_n$ need to decide how to share the cost of the cable.

asymptotically (via an expression without large summations) using $\Theta(-)$ notation.

(b) Part (a) makes precise a sense in which the nodes that arrive early carry an "unfair" share of the connections in the network. Another way to quantify the imbalance is to observe that, in a run of this random process, we expect many nodes to end up with no incoming links.

Give a formula for the expected number of nodes with no incoming links in a network grown randomly according to this model.

5. Out in a rural part of the county somewhere, $n$ small towns have decided to get connected to a large Internet switching hub via a high-volume fiber-optic cable. The towns are labeled $T_1, T_2, \ldots, T_n$, and they are all arranged on a single long highway, so that town $T_i$ is $i$ miles from the switching hub (See Figure 13.6).

Now this cable is quite expensive; it costs $k$ dollars per mile, resulting in an overall cost of $kn$ dollars for the whole cable. The towns get together and discuss how to divide up the cost of the cable.

First, one of the towns way out at the far end of the highway makes the following proposal.

Proposal A. Divide the cost evenly among all towns, so each pays $k$ dollars.

There’s some sense in which Proposal A is fair, since it’s as if each town is paying for the mile of cable directly leading up to it.

But one of the towns very close to the switching hub objects, pointing out that the faraway towns are actually benefiting from a large section of the cable, whereas the close-in towns only benefit from a short section of it. So they make the following counterproposal.

Proposal B. Divide the cost so that the contribution of town $T_i$ is proportional to $i$, its distance from the switching hub.

One of the other towns very close to the switching hub points out that there’s another way to do a nonproportional division that is also
6. One of the (many) hard problems that arises in genome mapping can be formulated in the following abstract way. We are given a set of \( n \) markers \( \{\mu_1, \ldots, \mu_n\} \)—these are positions on a chromosome that we are trying to map—and our goal is to output a linear ordering of these markers. The output should be consistent with a set of \( k \) constraints, each specified by a triple \((\mu_i, \mu_j, \mu_k)\), requiring that \( \mu_j \) lie between \( \mu_i \) and \( \mu_k \) in the total ordering that we produce. (Note that this constraint does not specify which of \( \mu_i \) or \( \mu_k \) should come first in the ordering, only that \( \mu_j \) should come between them.)

Now it is not always possible to satisfy all constraints simultaneously, so we wish to produce an ordering that satisfies as many as possible. Unfortunately, deciding whether there is an ordering that satisfies at least \( k' \) of the \( k \) constraints is an NP-complete problem (you don’t have to prove this).

Give a constant \( \alpha > 0 \) (independent of \( n \)) and an algorithm with the following property. If it is possible to satisfy \( k^* \) of the constraints, then the algorithm produces an ordering of markers satisfying at least \( \alpha k^* \) of the constraints. Your algorithm may be randomized; in this case it should produce an ordering for which the expected number of satisfied constraints is at least \( \alpha k^* \).

7. In Section 13.4, we designed an approximation algorithm to within a factor of 7/8 for the MAX 3-SAT Problem, where we assumed that each clause has terms associated with three different variables. In this problem, we will consider the analogous MAX SAT Problem: Given a set of clauses \( C_1, \ldots, C_q \) over a set of variables \( X = \{x_1, \ldots, x_n\} \), find a truth assignment satisfying as many of the clauses as possible. Each clause has at least one term in it, and all the variables in a single clause are distinct, but otherwise we do not make any assumptions on the length of the clauses: There may be clauses that have a lot of variables, and others may have just a single variable.

(a) First consider the randomized approximation algorithm we used for MAX 3-SAT, setting each variable independently to true or false with probability 1/2 each. Show that the expected number of clauses satisfied by this random assignment is at least \( k/2 \), that is, at least half of the clauses are satisfied in expectation. Give an example to show that there are MAX SAT instances such that no assignment satisfies more than half of the clauses.

(b) If we have a clause that consists only of a single term (e.g., a clause consisting just of \( x_i \) or just of \( \overline{x_i} \)), then there is only a single way to satisfy it: We need to set the corresponding variable in the appropriate
way. If we have two clauses such that one consists of just the term $x_i$, and the other consists of just the negated term $\overline{x_i}$, then this is a pretty direct contradiction.

Assume that our instance has no such pair of "conflicting clauses"; that is, for no variable $x_i$ do we have both a clause $C = \{x_i\}$ and a clause $C' = \{\overline{x_i}\}$. Modify the randomized procedure above to improve the approximation factor from $1/2$ to at least $.6$. That is, change the algorithm so that the expected number of clauses satisfied by the process is at least $.6k$.

(c) Give a randomized polynomial-time algorithm for the general MAX SAT Problem, so that the expected number of clauses satisfied by the algorithm is at least a .6 fraction of the maximum possible.

(Note that, by the example in part (a), there are instances where one cannot satisfy more than $k/2$ clauses; the point here is that we'd still like an efficient algorithm that, in expectation, can satisfy a .6 fraction of the maximum that can be satisfied by an optimal assignment.)

8. Let $G = (V, E)$ be an undirected graph with $n$ nodes and $m$ edges. For a subset $X \subseteq V$, we use $G[X]$ to denote the subgraph induced on $X$ — that is, the graph whose node set is $X$ and whose edge set consists of all edges of $G$ for which both ends lie in $X$.

We are given a natural number $k \leq n$ and are interested in finding a set of $k$ nodes that induces a "dense" subgraph of $G$; we'll phrase this concretely as follows. Give a polynomial-time algorithm that produces, for a given natural number $k \leq n$, a set $X \subseteq V$ of $k$ nodes with the property that the induced subgraph $G[X]$ has at least $\frac{mk(k-1)}{n(n-1)}$ edges.

You may give either (a) a deterministic algorithm, or (b) a randomized algorithm that has an expected running time that is polynomial, and that only outputs correct answers.

9. Suppose you're designing strategies for selling items on a popular auction Web site. Unlike other auction sites, this one uses a one-pass auction, in which each bid must be immediately (and irrevocably) accepted or refused. Specifically, the site works as follows.

- First a seller puts up an item for sale.
- Then buyers appear in sequence.
- When buyer $i$ appears, he or she makes a bid $b_i > 0$.
- The seller must decide immediately whether to accept the bid or not.
  If the seller accepts the bid, the item is sold and all future buyers are
turned away. If the seller rejects the bid, buyer \( i \) departs and the bid is withdrawn; and only then does the seller see any future buyers.

Suppose an item is offered for sale, and there are \( n \) buyers, each with a distinct bid. Suppose further that the buyers appear in a random order, and that the seller knows the number \( n \) of buyers. We'd like to design a strategy whereby the seller has a reasonable chance of accepting the highest of the \( n \) bids. By a strategy, we mean a rule by which the seller decides whether to accept each presented bid, based only on the value of \( n \) and the sequence of bids seen so far.

For example, the seller could always accept the first bid presented. This results in the seller accepting the highest of the \( n \) bids with probability only \( 1/n \), since it requires the highest bid to be the first one presented.

Give a strategy under which the seller accepts the highest of the \( n \) bids with probability at least \( 1/4 \), regardless of the value of \( n \). (For simplicity, you may assume that \( n \) is an even number.) Prove that your strategy achieves this probabilistic guarantee.

10. Consider a very simple online auction system that works as follows. There are \( n \) bidding agents; agent \( i \) has a bid \( b_i \), which is a positive natural number. We will assume that all bids \( b_i \) are distinct from one another. The bidding agents appear in an order chosen uniformly at random, each proposes its bid \( b_i \) in turn, and at all times the system maintains a variable \( b^* \) equal to the highest bid seen so far. (Initially \( b^* \) is set to 0.)

What is the expected number of times that \( b^* \) is updated when this process is executed, as a function of the parameters in the problem?

Example. Suppose \( b_1 = 20, b_2 = 25, \) and \( b_3 = 10, \) and the bidders arrive in the order 1, 3, 2. Then \( b^* \) is updated for 1 and 2, but not for 3.

11. Load balancing algorithms for parallel or distributed systems seek to spread out collections of computing jobs over multiple machines. In this way, no one machine becomes a “hot spot.” If some kind of central coordination is possible, then the load can potentially be spread out almost perfectly. But what if the jobs are coming from diverse sources that can’t coordinate? As we saw in Section 13.10, one option is to assign them to machines at random and hope that this randomization will work to prevent imbalances. Clearly, this won’t generally work as well as a perfectly centralized solution, but it can be quite effective. Here we try analyzing some variations and extensions on the simple load balancing heuristic we considered in Section 13.10.
Suppose you have \( k \) machines, and \( k \) jobs show up for processing. Each job is assigned to one of the \( k \) machines independently at random (with each machine equally likely).

(a) Let \( N(k) \) be the expected number of machines that do not receive any jobs, so that \( N(k)/k \) is the expected fraction of machines with nothing to do. What is the value of the limit \( \lim_{k \to \infty} N(k)/k \)? Give a proof of your answer.

(b) Suppose that machines are not able to queue up excess jobs, so if the random assignment of jobs to machines sends more than one job to a machine \( M \), then \( M \) will do the first of the jobs it receives and reject the rest. Let \( R(k) \) be the expected number of rejected jobs; so \( R(k)/k \) is the expected fraction of rejected jobs. What is \( \lim_{k \to \infty} R(k)/k \)? Give a proof of your answer.

(c) Now assume that machines have slightly larger buffers; each machine \( M \) will do the first two jobs it receives, and reject any additional jobs. Let \( R_2(k) \) denote the expected number of rejected jobs under this rule. What is \( \lim_{k \to \infty} R_2(k)/k \)? Give a proof of your answer.

12. Consider the following analogue of Karger’s algorithm for finding minimum \( s-t \) cuts. We will contract edges iteratively using the following randomized procedure. In a given iteration, let \( s \) and \( t \) denote the possibly contracted nodes that contain the original nodes \( s \) and \( t \), respectively. To make sure that \( s \) and \( t \) do not get contracted, at each iteration we delete any edges connecting \( s \) and \( t \) and select a random edge to contract among the remaining edges. Give an example to show that the probability that this method finds a minimum \( s-t \) cut can be exponentially small.

13. Consider a balls-and-bins experiment with \( 2n \) balls but only two bins. As usual, each ball independently selects one of the two bins, both bins equally likely. The expected number of balls in each bin is \( n \). In this problem, we explore the question of how big their difference is likely to be. Let \( X_1 \) and \( X_2 \) denote the number of balls in the two bins, respectively. (\( X_1 \) and \( X_2 \) are random variables.) Prove that for any \( \varepsilon > 0 \) there is a constant \( c > 0 \) such that the probability \( \Pr[X_1 - X_2 \geq c\sqrt{n}] \leq \varepsilon \).

14. Some people designing parallel physical simulations come to you with the following problem. They have a set \( P \) of \( k \) basic processes and want to assign each process to run on one of two machines, \( M_1 \) and \( M_2 \). They are then going to run a sequence of \( n \) jobs, \( J_1, \ldots, J_n \). Each job \( J_i \) is represented by a set \( R_i \subseteq P \) of exactly \( 2n \) basic processes which must be running (each on its assigned machine) while the job is processed. An assignment of basic processes to machines will be called perfectly balanced if, for
Chapter 13  Randomized Algorithms

For $i = 2, 3, 4, \ldots, n$
Set $\beta_i = f(\beta_1, \beta_2, \ldots, \beta_{i-1}, a_i)$
Endfor
Output $\beta$

One could view this as a type of "stream cipher with feedback." One problem with this approach is that, if any bit $a_i$ gets corrupted in transmission, it will corrupt the computed value of $\beta_i$ for all $j \geq i$.

We consider the following problem. A sender $S$ wants to transmit the same (plain-text) message $\beta$ to each of $k$ receivers $R_1, \ldots, R_k$. With each one, he shares a different secret function $f^{(i)}$. Thus he sends a different encrypted message $a^{(i)}$ to each receiver, so that $a^{(i)}$ encrypts to $\beta$ when the above algorithm is run with the function $f^{(i)}$.

Unfortunately, the communication channels are very noisy, so each of the $n$ bits in each of the $k$ transmissions is independently corrupted (i.e., flipped to its complement) with probability $1/2$. Thus no single receiver on his or her own is likely to be able to encrypt the message correctly. Show, however, that if $k$ is large enough as a function of $n$, then the $k$ receivers can jointly reconstruct the plain-text message in the following way. They get together, and without revealing any of the $a^{(i)}$ or the $f^{(i)}$, they interactively run an algorithm that will produce the correct $\beta$ with probability at least 9/10. (How large do you need $k$ to be in your algorithm?)

17. Consider the following simple model of gambling in the presence of bad odds. At the beginning, your net profit is 0. You play for a sequence of $n$ rounds; and in each round, your net profit increases by 1 with probability $1/3$, and decreases by 1 with probability $2/3$.

Show that the expected number of steps in which your net profit is positive can be upper-bounded by an absolute constant, independent of the value of $n$.

18. In this problem, we will consider the following simple randomized algorithm for the Vertex Cover Algorithm.

Start with $S = \emptyset$
While $S$ is not a vertex cover,
Select an edge $e$ not covered by $S$
Select one end of $e$ at random (each end equally likely)
Add the selected node to $S$
Endwhile