Part IV

Randomness

CHAPTER

NINETEEN

RANDOMIZED ALGORITHMS

The algorithms we have seen thus far have been *deterministic*, meaning that they execute the same steps each time they are run and produce the same result. In this unit, we consider how randomness can be applied to computation. We will exploit randomness to design simple, probabilistic algorithms, and we will discuss the tools that are necessary to analyze randomized algorithms.

As a first example, consider a game of *rock-paper-scissors* between two players. In this game, both players simultaneously choose one of three options: rock, paper, or scissors. The game is a tie if both players choose the same option. If they make different choices, then rock beats scissors, scissors beats paper, and paper beats rock:

		Player A		
		Rock	Paper	Scissors
Player B	Rock	Tie	A wins	B wins
	Paper	B wins	Tie	A wins
	Scissors	A wins	B wins	Tie

Suppose we write a program to act as a player in a best-out-of-five game of rock-paper-scissors. If we hardcode a strategy such as playing rock in the first two moves, then paper, then scissors, and then paper, what is the probability that the program wins a game against an opponent? The program is deterministic, so a smart opponent would be able to observe the program's strategy once and defeat it every single time thereafter. If an opponent is able to read the program's source code, then a preliminary observation isn't even necessary. This illustrates the problem with a deterministic strategy – it makes the program's actions predictable and thus easily countered.

What if we change the program to choose a random action in each move, with equal probability for rock, paper, or scissors? Now even if an opponent knows the program's strategy, the opponent cannot know which action the program will take and therefore cannot universally counter that action. In fact, no matter what strategy the opponent uses, the probability the opponent wins an individual round is always $\frac{1}{3}$. Before we see why, we review some basic tools we use to analyze probabilistic outcomes.

19.1 Review of Probability

A *random variable* is a quantity whose value is dependent on the outcome of a random phenomenon. The set of outcomes associated with a random phenomenon is called the *sample space*, and a random variable X can be regarded as a function from the sample space Ω to the set of real numbers, i.e. $X : \Omega \to \mathbb{R}$. Thus, the random variable associates a numerical value with each outcome.

A random variable X can be described with a *probability distribution* – for each possible value a_i, the random variable has a probability p_i of taking on the value a_i:

$$\Pr[X = a_i] = p_i$$

The probability p_i must be in the interval $[0, 1]$, i.e. $0 \le p_i \le 1$. Furthermore, the probabilities p_i over all possible i must add up to 1:

$$\sum_i p_i = 1$$

A random variable X takes on the value a_i when the outcome ω of a random experiment has associated value $X(\omega) = a_i$. Thus, $X = a_i$ is an *event*, a subset of the sample space of an experiment. More specifically, it is the event:

$$(X = a_i) = \{\omega \in \Omega : X(\omega) = a_i\}$$

The probability of an event can be computed as the sum of the probabilities for each outcome in the event. Thus, the probability $\Pr[X = a_i]$ is:

$$\Pr[X = a_i] = \sum_{\omega \in \Omega : X(\omega) = a_i} \Pr[\omega]$$

If each outcome has equal probability $\frac{1}{|\Omega|}$, we get

$$\Pr[X = a_i] = \frac{|\{\omega \in \Omega : X(\omega) = a_i\}|}{|\Omega|}$$

Example 177 Suppose we roll a pair of distinguishable, six-sided dice. Let X be a random variable corresponding to the sum of the values of the dice – if the values are (i, j), then $X = i + j$. The events corresponding to each value of X are as follows:

$$\begin{aligned}
(X = 2) &= \{(1, 1)\}\\
(X = 3) &= \{(1, 2), (2, 1)\}\\
(X = 4) &= \{(1, 3), (2, 2), (3, 1)\}\\
(X = 5) &= \{(1, 4), (2, 3), (3, 2), (4, 1)\}\\
(X = 6) &= \{(1, 5), (2, 4), (3, 3), (4, 2), (5, 1))\}\\
(X = 7) &= \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\}\\
(X = 8) &= \{(2, 6), (3, 5), (4, 4), (5, 3), (6, 2)\}\\
(X = 9) &= \{(3, 6), (4, 5), (5, 4), (6, 3)\}\\
(X = 10) &= \{(4, 6), (5, 5), (6, 4)\}\\
(X = 11) &= \{(5, 6), (6, 5)\}\\
(X = 12) &= \{(6, 6)\}
\end{aligned}$$

If the dice are fair, each outcome has equal probability. This gives us the following probability distribution for X:

$$\begin{aligned}
\Pr[X = 2] &= \frac{1}{36}\\
\Pr[X = 3] &= \frac{2}{36} = \frac{1}{18}\\
\Pr[X = 4] &= \frac{3}{36} = \frac{1}{12}\\
\Pr[X = 5] &= \frac{4}{36} = \frac{1}{9}\\
\Pr[X = 6] &= \frac{5}{36}\\
\Pr[X = 7] &= \frac{6}{36} = \frac{1}{6}\\
\Pr[X = 8] &= \frac{5}{36}\\
\Pr[X = 9] &= \frac{4}{36} = \frac{1}{9}\\
\Pr[X = 10] &= \frac{3}{36} = \frac{1}{12}\\
\Pr[X = 11] &= \frac{2}{36} = \frac{1}{18}\\
\Pr[X = 12] &= \frac{1}{36}
\end{aligned}$$

The probability distribution provides full detail about a random variable. However, we often are interested in more succinct information that in a sense summarizes the details. One such piece of information is the *expectation* or *expected value* of a random variable, which corresponds to the "average" value that the random variable takes on. We compute the expectation $\mathbb{E}[X]$ as follows:

$$\mathbb{E}[X] = \sum_i a_i \cdot \Pr[X = a_i]$$

Thus, the expectation is the average value of the variable weighted by the probability of obtaining each value.

Recall that

$$\Pr[X = a_i] = \sum_{\omega\in\Omega:X(\omega)=a_i} \Pr[\omega]$$

Plugging this into the formula for expectation, we get

$$\begin{aligned}
\mathbb{E}[X] &= \sum_i a_i \cdot \Pr[X = a_i]\\
&= \sum_i a_i \cdot (\sum_{\omega\in\Omega:X(\omega)=a_i} \Pr[\omega])\\
&= \sum_i \sum_{\omega\in\Omega:X(\omega)=a_i} X(\omega)\cdot \Pr[\omega]\\
&= \sum_{\omega\in\Omega} X(\omega)\cdot \Pr[\omega]
\end{aligned}$$

as an alternate expression of $\mathbb{E}[X]$.

Often, we can express a random variable X in terms of other random variables X_i. We can then apply *linearity of expectation* to compute the expectation of X from the expectations of each X_i.

Theorem 178 (Linearity of Expectation) *Let X be a weighted sum of random variables X_i, each multiplied by a constant c_i:*

$$X = \sum_i c_i \cdot X_i$$

Then $\mathbb{E}[X]$ is:

$$\mathbb{E}[X] = \sum_i c_i \cdot \mathbb{E}[X_i]$$

Proof 179 We prove the case where X is the weighted sum of two variables

$$X = c_1 X_1 + c_2 X_2$$

From the alternate formulation of expectation, we have

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{\omega\in\Omega} X(\omega) \cdot \Pr[\omega] \\
&= \sum_{\omega\in\Omega} (c_1 X_1(\omega) + c_2 X_2(\omega)) \cdot \Pr[\omega] \\
&= \sum_{\omega\in\Omega} c_1 X_1(\omega) \cdot \Pr[\omega] + c_2 X_2(\omega) \cdot \Pr[\omega] \\
&= (\sum_{\omega\in\Omega} c_1 X_1(\omega) \cdot \Pr[\omega]) + (\sum_{\omega\in\Omega} c_2 X_2(\omega) \cdot \Pr[\omega]) \\
&= c_1 \cdot (\sum_{\omega\in\Omega} X_1(\omega) \cdot \Pr[\omega]) + c_2 \cdot (\sum_{\omega\in\Omega} X_2(\omega) \cdot \Pr[\omega]) \\
&= c_1 \cdot \mathbb{E}[X_1] + c_2 \cdot \mathbb{E}[X_2]
\end{aligned}$$

Example 180 Suppose we roll a pair of distinguishable, six-sided dice. Let X_1 be the value of the first die and X_2 the value of the second. If the dice are fair, then each outcome is equally likely, so we have

$$\begin{aligned}
\mathbb{E}[X_1] = \mathbb{E}[X_2] &= \sum_{i=1}^{6} i \cdot \frac{1}{6} \\
&= 3.5
\end{aligned}$$

Let $X = X_1 + X_2$ be the sum of the values of the two dice. By linearity of expectation, we have

$$\mathbb{E}[X] = \mathbb{E}[X_1] + \mathbb{E}[X_2] = 7$$

Computing the expectation from the probability distribution of X gives us the same result, but it requires much more work to do so.

A sequence of random variables $X_1, X_2, \dots, X_n$ is *independent* if for every possible set of values $b_1, b_2, \dots, b_n$, we have:

$$\Pr[X_1 = b_1, X_2 = b_2, \dots, X_n = b_n] = \prod_i \Pr[X_i = b_i]$$

The notation $\Pr[X_1 = b_1, X_2 = b_2, \dots, X_n = b_n]$ represents the *joint probability* of the events $X_1 = b_1$, $X_2 = b_2$, …, $X_n = b_n$ occurring simultaneously. The random variables are independent if the joint probability of them taking on any sequence of values is the same as the product of each of them individually taking on the respective value.

A common type of random variable is an *indicator* random variable, also called a *0-1 random variable*. As the latter name suggests, it is a random variable that only takes on the values 0 or 1. If X is an indicator and $\Pr[X = 1] = p$, we have $\Pr[X = 0] = 1 - p$, and

$$\begin{aligned}\mathbb{E}[X] &= 0 \cdot \Pr[X = 0] + 1 \cdot Pr[X = 1]\\ &= \Pr[X = 1] = p\end{aligned}$$

If we can define a random variable Y as the sum of n indicator random variables X_i, each with the same probability $\Pr[X_i = 1] = p$, then:

$$\begin{aligned}\mathbb{E}[Y] &= \sum_i \mathbb{E}[X_i]\\ &= \sum_i \Pr[X_i = 1]\\ &= np\end{aligned}$$

Example 181 Suppose we repeatedly flip a biased coin with probability p of heads. What is the expected number of heads over n flips?

Define X_i to be 0 if the ith flip is tails and 1 if it is heads. Then X_i is an indicator with $\mathbb{E}[X_i] = \Pr[X_i] = p$. Let Y be the sum of the X_i. Since Y is the sum of n indicators X_i, each with probability $\Pr[X_i = 1] = p$, we have $\mathbb{E}[Y] = np$. Observe also that Y is the total number of heads, since X_i contributes 1 to the value of Y when the ith flip is heads and 0 otherwise. Thus, the expected number of heads is np.

Returning back to the rock-paper-scissors example, let R, P, and S be arbitrary numbers corresponding to rock, paper, and scissors. Let A be a random variable representing the action taken by our randomized program, B be a random variable representing the opponent's action. Let W be the event that the opponent wins – this is when the program chooses rock and the opponent paper, or the program paper and the opponent scissors, or the program scissors and the opponent rock. These are disjoint events, so the probability of any one of them happening is the sum of their individual probabilities. Then we have:

$$\begin{aligned}\Pr[W] &= \Pr[A = R, B = P] + \Pr[A = P, B = S] + \Pr[A = S, B = R]\\ &= \Pr[A = R] \cdot \Pr[B = P] + \Pr[A = P] \cdot \Pr[B = S] + \Pr[A = S] \cdot \Pr[B = R]\\ &= \frac{1}{3}(\Pr[B = P] + \Pr[B = S] + \Pr[B = R])\\ &= \frac{1}{3}\end{aligned}$$

In the second step, we make use of the fact that the program's action and the opponent's are independent. Then we apply the fact that the program chooses each action with equal probability $\frac{1}{3}$. Finally, the opponent must choose one of the three possible actions, so regardless of how the opponent makes their choice, the probabilities sum to one. The conclusion is that the probability the opponent wins is $\frac{1}{3}$ no matter what strategy they use. This is a significant improvement over the deterministic program, where the opponent could always win once they discover the program's strategy.

The rock-paper-scissors example is illustrative of a more general notion of considering algorithms as games. We allow one player to choose an algorithm to solve a problem. A second, more powerful player known as the *adversary* has the ability to choose an input. The goal of the adversary is to maximize the number of steps taken by the chosen algorithm on the given input. The game is unfair – the adversary knows the process by which the first player chooses an algorithm (e.g. from prior observations, or from reading the source code for the first player). If the first player chooses the algorithm deterministically, the adversary can always find an input that maximizes the number of steps. On the other hand, if the first player chooses an algorithm randomly according to some distribution, the adversary can no longer always find a worst input, even if the adversary knows the distribution the first player uses.

19.2 Randomized Approximations

We can exploit randomness to construct simple algorithms that produce an *α-approximation* (page 190) *in expectation*, meaning that the expected value of the output is within a factor α of the value of the optimal solution. As an example, we consider the problem of satisfying as many clauses as possible given an *exact 3CNF formula*. Such a formula is in *3CNF* (page 168), with the added stipulation that each clause has three distinct variables. For example, the clause

$$(x \vee \neg y \vee \neg z)$$

is in exact 3CNF, while the clause

$$(x \vee \neg y \vee \neg x)$$

is not, since it only has two distinct variables.

The problem of finding an assignment that maximizes the number of satisfied clauses in an exact 3CNF formula is known as Max-E3SAT. This is an NP-hard optimization problem. However, a simple, randomized algorithm that assigns the truth value of each variable randomly (with equal probability for true or false) achieves a 7/8-approximation in expectation. We demonstrate this as follows.

Suppose ϕ is an exact 3CNF formula with n variables and m clauses. Let a be a random assignment of the variables, and let $X(a)$ be the number of satisfied clauses for this assignment. Then X is a random variable, since it associates each output with a number. We define indicator random variables X_i, where $X_i = 1$ if clause i is satisfied and $X_i = 0$ if it is not. Then we have

$$X = X_1 + \cdots + X_m$$

An exact 3CNF clause has three literals, each with a distinct variable, and the probability that an individual literal is satisfied by a random assignment to its variable is $\frac{1}{2}$; a literal of the form x is satisfied when $x = 1$, which occurs with probability $\frac{1}{2}$, and a literal of the form $\neg x$ is satisfied when $x = 0$, which also occurs with probability $\frac{1}{2}$. A clause is unsatisfied when all three of its literals are false. Since the literals have distinct variables, the events that each is false are independent. Thus, we have

$$\begin{aligned}\Pr[\text{all three literals in clause } i \text{ are false}] &= \prod_{j=1}^{3} \Pr[\text{literal } j \text{ in clause } i \text{ is false}]\\ &= (\frac{1}{2})^3 = \frac{1}{8}\end{aligned}$$

If even a single literal is true, the clause is satisfied. Thus, the probability that clause i is satisfied with a random assignment is

$$\begin{aligned}\Pr[X_i = 1] &= 1 - \Pr[\text{all three literals in clause } i \text{ are false}]\\ &= 1 - \frac{1}{8} = \frac{7}{8}\end{aligned}$$

We then have $\mathbb{E}[X_i] = \frac{7}{8}$, so by linearity of expectation,

$$\begin{aligned}\mathbb{E}[X] &= \mathbb{E}[X_1] + \cdots + \mathbb{E}[X_m]\\ &= \frac{7}{8}m\end{aligned}$$

Thus, the expected number of satisfied clauses is $\frac{7}{8}m$. Since the optimal assignment can satisfy at most m clauses, the random assignment is a 7/8-approximation, in expectation, of the optimal.

We can further conclude from this result that any exact 3CNF formula has an assignment that satisfies at least $\frac{7}{8}$ of its clauses. We demonstrate this via contradiction. Suppose that a formula ϕ does not have an assignment that satisfies at least $\frac{7}{8}$ of its clauses. Let m be the number of clauses in ϕ, and let $X(a)$ be the number of clauses satisfied by an assignment a. We showed above that $\mathbb{E}[X] = \frac{7}{8}m$. By the alternate formulation of expectation, we have

$$\mathbb{E}[X] = \sum_{a\in\Omega} X(a) \cdot \Pr[a]$$

where Ω is the set of all possible assignments. By assumption, we have $X(a) < \frac{7}{8}m$ for all $a \in \Omega$. This gives us

$$\begin{aligned}\mathbb{E}[X] &< \sum_{a\in\Omega} \frac{7}{8}m \cdot \Pr[a]\\ &= \frac{7}{8}m \cdot \sum_{a\in\Omega} \Pr[a]\\ &= \frac{7}{8}m\end{aligned}$$

However, this contradicts our result that $\mathbb{E}[X] = \frac{7}{8}m$. Thus, our assumption that no assignment exists that satisfies $\frac{7}{8}$ of the clauses must be false.

More generally, the reasoning above gives us the following for an arbitrary random variable X:

Claim 182 (Averaging Argument) *Suppose X is a random variable over a sample space Ω. Then by the* averaging argument*:*

- *There is at least one outcome $\omega \in \Omega$ such that*

$$X(\omega) \geq \mathbb{E}[X]$$

- *There is at least one outcome $\omega \in \Omega$ such that*

$$X(\omega) \leq \mathbb{E}[X]$$

Applying the above, given m objects, if a property holds for each object with probability p, then there must be a case where it holds for at least mp objects simultaneously. This is because the expected number of objects for which the property holds is mp, and by Claim 182, there must be some case where it holds for at least mp objects.

Another question we may have is, what is the probability that a random assignment satisfies at least half the clauses of an exact 3CNF formula? In terms of the random variable X, what is $\Pr[X \geq \frac{1}{2}m]$ for a formula with m clauses? We can apply *Markov's inequality* to place a bound on this probability.

Theorem 183 (Markov's Inequality) *Let X be a nonnegative random variable (i.e. X never takes on a negative value), and let $v > 0$. Then*

$$\Pr[X \geq v] \leq \frac{\mathbb{E}[X]}{v}$$

Proof 184 We prove Markov's inequality for a *discrete* random variable X, one that can only take on a countable number of values. By definition of expectation, we have:

$$\begin{aligned}\mathbb{E}[X] &= \sum_i a_i \cdot \Pr[X = a_i] \\ &= (\sum_{a_i < v} a_i \cdot \Pr[X = a_i]) + (\sum_{a_i \geq v} a_i \cdot \Pr[X = a_i]) \\ &\geq \sum_{a_i \geq v} a_i \cdot \Pr[X = a_i] \\ &\geq \sum_{a_i \geq v} v \cdot \Pr[X = a_i] \\ &= v \cdot \Pr[X \geq v]\end{aligned}$$

The third step follows from the second because X is nonnegative, so $a_i \geq 0$ and therefore the left-hand summation is nonnegative.

Dividing both sides of the result by v, we get

$$\frac{\mathbb{E}[X]}{v} \geq \Pr[X \geq v]$$

In the case of Max-E3SAT, X is a count of the number of satisfied clauses, so it never takes on a negative value. Thus, we can apply Markov's inequality with $v = \frac{1}{2}$ to obtain:

$$\Pr\left[X \geq \frac{1}{2}m\right] \leq (\frac{7}{8}m)/(\frac{1}{2}m) = \frac{7}{4}$$

Unfortunately, this doesn't tell us anything we didn't know – we already know by definition of probability that $\Pr[X \geq \frac{1}{2}m] \leq 1$. Furthermore, we would like a lower bound on the probability that at least half the clauses are satisfied, not an upper bound. We need another tool to reason about this.

Theorem 185 (Reverse Markov's Inequality) *Let X be a random variable that is never larger than some value b, and let $v < b$. Then*

$$\Pr[X > v] \geq \frac{\mathbb{E}[X] - v}{b - v}$$

More loosely, we have

$$\Pr[X \geq v] \geq \frac{\mathbb{E}[X] - v}{b - v}$$

Proof 186 Define $Y = b - X$. Since X is never larger than b, Y is a nonnegative random variable. We have

$$\begin{aligned}\Pr[X \leq v] &= \Pr[b - Y \leq v] \\ &= \Pr[Y \geq b - v]\end{aligned}$$

Applying Markov's inequality, we get:

$$\begin{aligned}\Pr[Y \geq b - v] &\leq \frac{\mathbb{E}[Y]}{b - v} \\ &= \frac{b - \mathbb{E}[X]}{b - v}\end{aligned}$$

Then we have

$$\begin{aligned}\Pr[X > v] &= 1 - \Pr[X \le v]\\ &= 1 - \Pr[Y \ge b - v]\\ &\ge 1 - \frac{b - \mathbb{E}[X]}{b - v}\\ &= \frac{\mathbb{E}[X] - v}{b - v}\end{aligned}$$

To obtain the looser formulation, we note that

$$\begin{aligned}\Pr[X \ge v] &= \Pr[X > v] + \Pr[X = v]\\ &\ge \Pr[X > v]\end{aligned}$$

Combining this with the previous result, we obtain

$$\Pr[X \ge v] \ge \frac{\mathbb{E}[X] - v}{b - v}$$

Observe that for Max-E3SAT, we have an upper bound of m on the value of X, where m is the number of clauses. Setting $v = \frac{1}{2}m < m$, we apply the reverse Markov's inequality to get

$$\begin{aligned}\Pr\left[X \ge \frac{1}{2}m\right] &\ge (\frac{7}{8}m - \frac{1}{2}m)/(m - \frac{1}{2}m)\\ &= (\frac{3}{8})/(\frac{1}{2}) = \frac{3}{4}\end{aligned}$$

Thus, the randomized algorithm produces an assignment that satisfies at least half the clauses with probability $\ge \frac{3}{4}$.

Derandomized Algorithm for Max-E3SAT

The randomized algorithm for Max-E3SAT can be *derandomized* into a deterministic algorithm by computing conditional expectations for a choice and taking the best one. This is called the method of conditional probabilities[63] (or more specifically, the *method of conditional expectations*).

We start by defining the *conditional expectation* of a random variable given that event B has occurred as

$$\mathbb{E}[X|B] = \sum_k k \cdot \Pr[X = k|B]$$

The *conditional probability* $\Pr[A|B]$ for events A and B is the probability of event A having occurred knowing that event B has occurred. It can be computed as

$$\begin{aligned}\Pr[A|B] &= \frac{\Pr[A \cap B]}{\Pr[B]}\\ &= \frac{\sum_{\omega \in A \cap B} \Pr[\omega]}{\sum_{\omega \in B} \Pr[\omega]}\end{aligned}$$

The conditional expectation of a random variable X given an event B is the "average" value of X knowing that event B has occurred. For any random variable X and event B, we have

$$\mathbb{E}[X] = \Pr[B] \cdot \mathbb{E}[X|B] + \Pr[B] \cdot \mathbb{E}[X|\overline{B}]$$

where $B = \Omega \setminus B$ is the complement of B. Since $\mathbb{E}[X]$ is the weighted average of $\mathbb{E}[X|B]$ and $\mathbb{E}[X|\overline{B}]$, by the averaging argument, at least one of $\mathbb{E}[X|B] \ge \mathbb{E}[X]$ or $\mathbb{E}[X|\overline{B}] \ge \mathbb{E}[X]$ must hold.

We proceed to apply this to Max-E3SAT. Let ϕ be an exact 3CNF formula over n variables $x_1, \dots, x_n$ and with m clauses. Let X be the number of satisfied clauses for a random assignment. We have established that $\mathbb{E}[X] = 7m/8$. Now consider the events when $x_1 = 0$ and when $x_1 = 1$. These are complement events, since any assignment must set a variable to 0 or 1. By the reasoning above, at least one of $\mathbb{E}[X|x_1 = 0] \geq 7m/8$ or $\mathbb{E}[X|x_1 = 1] \geq 7m/8$ must hold. Thus, we can deterministically build an assignment $(a_1, a_2, \dots, a_n)$ by doing the following for each variable in order:

- Compute $\mathbb{E}[X|x_1 = a_1, \dots, x_{i-1} = a_{i-1}, x_i = 0]$ and $\mathbb{E}[X|x_1 = a_1, \dots, x_{i-1} = a_{i-1}, x_i = 1]$. In other words, given the partial assignment so far, try both setting $x_i = 0$ and setting $x_i = 1$ and compute the resulting conditional expectations.
- If setting $x_i = 0$ produces a better expectation, then fix $a_i = 0$, otherwise fix $a_i = 1$.

At each step, at least one of the two choices must satisfy at least $7m/8$ clauses by the averaging argument. Thus, the full assignment produced satisfies at least $7m/8$ clauses.

The full algorithm is as follows:

```
Input: an exact 3CNF formula
Output: an assignment that satisfies at least 7/8ths of the formula's clauses
  function ApproxExact3SAT($\phi$)
      $A$ = an empty assignment for $\phi$
      for all variables $x_i$ in $\phi$ do
          for all Boolean values $v = 0, 1$ do
              $\mu_v$ = ExpectedSatisfied($\phi, A + (x_i = v)$)
          if $\mu_0 > \mu_1$ then
              $A = A + (x_i = 0)$
          else
              $A = A + (x_i = 1)$
      return $A$
Input: an exact 3CNF formula and a partial assignment for it
Output: the expected number of satisfied clauses, over a uniformly random completion of the assignment
  function ExpectedSatisfied($\phi, A$)
      $\mu = 0$
      for all clauses $C_j$ in $\phi$ do
          if $C_j$ is satisfied by $A$ then
              $\mu = \mu + 1$
          else
              $k$ = the number of unassigned variables (by $A$) in $C_j$
              $\mu = \mu + (1 - 1/2^k)$
      return $\mu$
```

The ExpectedSatisfied subroutine uses linearity of expectation to compute the expected number of satisfied clauses. Let X be the total number, and let X_j be an indicator for whether clause C_j is satisfied. Then if C_j is already satisfied by the partial assignment A, $\mathbb{E}[X_j|A] = 1$. If C_j is not yet satisfied, the probability of it being satisfied by a random assignment of the remaining variables depends on how many unset variables there are in C_j. By the same reasoning as for the randomized algorithm, this is $1 - (1/2)^k$ for k unset variables. Thus, $\mathbb{E}[X_j|A] = 1 - (1/2)^k$.

Observe that this algorithm does not have any randomness, but it uses the tools of probability to deterministically produce a good assignment!

[63] https://en.wikipedia.org/wiki/Method_of_conditional_probabilities

Exercise 187 The following is a randomized algorithm for producing a cut in an undirected graph G:

```
function RandomCut($G = (V, E)$)
    $S = \emptyset$
    for all $v \in V$ do
        add $v$ to $S$ with probability $1/2$
    return $(S, T = V \setminus S)$
```

Recall that $C(S, T)$ is the set of crossing edges for the cut S, T, i.e., those that have one endpoint in S and the other endpoint in T. The size of the cut is $|C(S, T)|$.

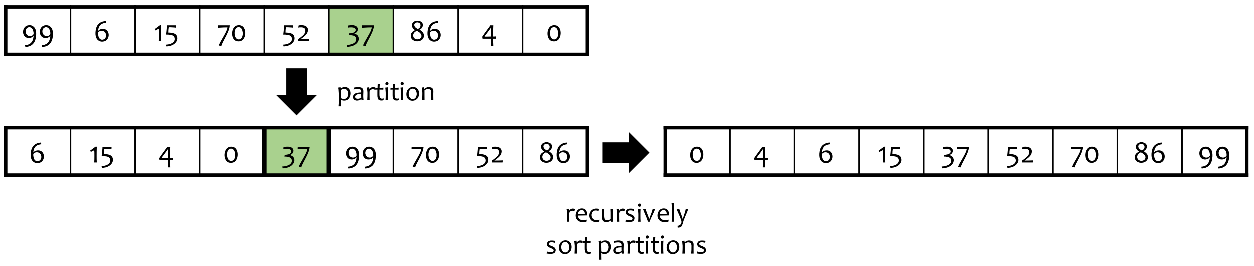
a) Prove that the randomized algorithm is a 1/2-approximation *in expectation* for finding a maximum cut of an undirected graph. That is, the expected size of the returned cut is at least half the size of a maximum cut.

   Hint: Use linearity of expectation to show that in expectation, half the edges of the graph are in $C(S, T)$.

b) Argue that, for any undirected graph $G = (V, E)$, there exists a cut of size at least $\frac{1}{2}|E|$.

19.3 Quick Sort

Another example of taking advantage of randomness in algorithm design is *quick sort*. The core algorithm requires selecting a *pivot* element from a sequence, and then partitioning the sequence into those elements that are less than the pivot and those that are greater than (or equal to) the pivot. The two partitions are recursively sorted. The following is an illustration of how this algorithm works, with the recursive steps elided:



It analyzing the runtime, we focus on the number of comparisons between elements – they dominate the runtime, and other aspects of the algorithm are dependent on choice of data structure. Partitioning an array of n elements requires comparing each element to the pivot, for a total of $n-1 = O(n)$ comparisons in each recursive step. If the two partitions corresponding to the elements less than and greater than the pivot are approximately equal in size, the recurrence for the number of comparisons is:

$$T(n) = 2T(\frac{n}{2}) + O(n)$$

This is the same recurrence as *merge sort* (page 13), and the algorithm achieves a runtime complexity of $O(n \log n)$ comparisons for a sequence of n elements. One way to accomplish this balanced partitioning is to find the median element – this can be done in linear time[64]. However, a simpler solution is to just pick a random element as the pivot. We proceed to analyze this version of the algorithm, which is as follows:

Algorithm 188 (Quick Sort)

```
function QuickSort($A[1, \dots, n]$)
    if $n = 1$ then return $A$
```

[64] https://en.wikipedia.org/wiki/Median_of_medians

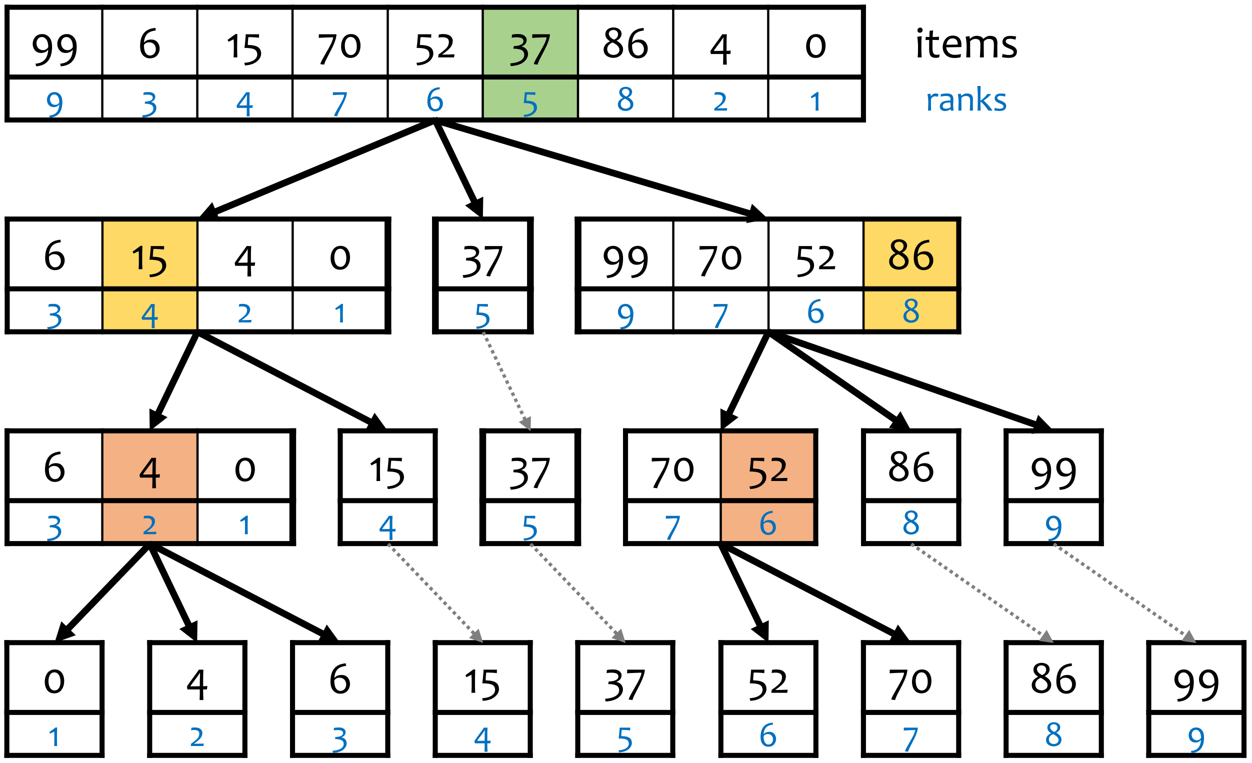
```
        $p =$ an index chosen uniformly at random from $\{1, \dots, n\}$
        $(L, R) =$ Partition$(A, p)$
        **return** QuickSort$(L)$ $+A[p]+$ QuickSort$(R)$
    **function** Partition$(A[1, \dots, n], p)$
        initialize empty arrays $L, R$
        **for** $i = 1$ to $n$ **do**
            **if** $i \neq p$ and $A[i] < A[p]$ **then**
                $L = L + A[i]$
            **else if** $i \neq p$ **then**
                $R = R + A[i]$
        **return** $(L, R)$
```

Let X be a random variable corresponding to the *rank* of the chosen pivot – the pivot has rank i if it is the ith largest element. The algorithm still works well when $n/6 \leq X \leq 5n/6$. By examining the probability distribution of X, we observe that:

$$\mathbb{E}[X] = \frac{n}{2}$$
$$\Pr\left[\frac{n}{6} \leq X \leq \frac{5n}{6}\right] \approx \frac{2}{3}$$

This follows from the fact that the element is chosen uniformly at random – half the outcomes have rank above $\frac{n}{2}$ and half below, and two-thirds have rank between $n/6$ and $5n/6$. Thus, we obtain a good pivot both with high probability and in expectation. Over the full course of the algorithm, most of the pivots chosen are good, and the expected runtime, in terms of number of comparisons, is still $O(n \log n)$.

For a more detailed analysis of the algorithm, we start by making the simplifying assumption that the input array does not have duplicates, so that each element has a unique rank. The following illustrates the execution of the algorithm over an example input, with the rank of each element denoted:



When are two elements compared? From Algorithm 188, we see that elements are compared in the Partition subroutine – in particular, the pivot element is compared to all other elements in the array that is passed to Partition. We also observe the following:

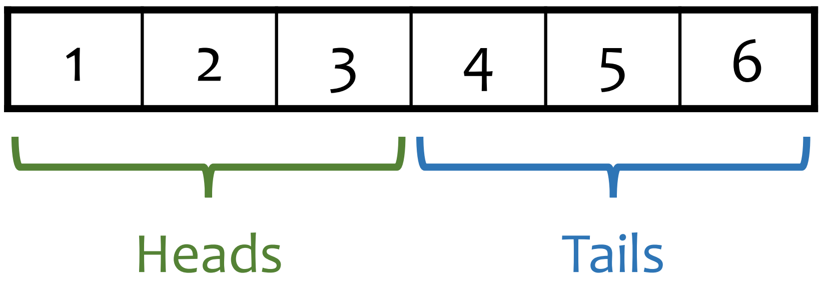
- The pivot element is not added to either partition (L or R), so it does not get compared to anything in subsequent recursive calls to QuickSort.
- For a pair of elements a and b, if one of them is less than the pivot (i.e. its rank is less than that of the pivot) and the other is greater than the pivot, the two elements get placed in separate partitions. Since the recursive calls operate on each partition separately, a and b do not get compared in any subsequent recursive calls to QuickSort.

In general, for two elements a and b, the elements get compared if and only if the first pivot chosen among the elements with value in the range $[a, b]$ is either a itself or b. As long as pivots are chosen outside of this range, a and b stay in the same partition (since the chosen pivot is either less than both a and b or greater than both of them), so they have a possibility of being compared. At the same time, since only pivots are compared to other elements, as long as neither a nor b are chosen as a pivot, they are not compared. When a pivot is selected in the range $[a, b]$, there are two possibilities:

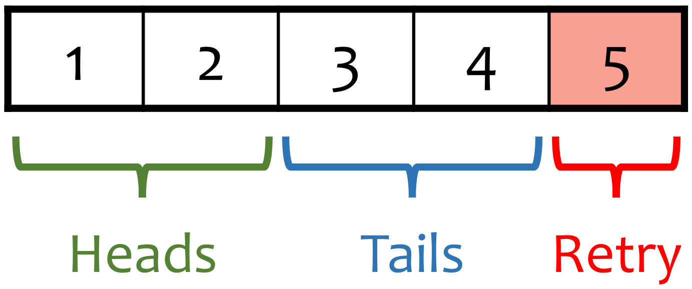
- a or b is selected as the pivot, in which case they are compared, since the pivot is compared to every element in the same subarray. They are not subsequently compared again, since as we observed above, the pivot element is not placed into either partition.
- An element c where $a < c < b$ is chosen as the pivot, in which case a and b are partitioned into separate subarrays and therefore never compared.

Thus, we need to determine the probability that when the first pivot is chosen from among the elements in the range $[a, b]$, the selected pivot is either a or b. Before we do so, let's take a look at a simpler problem, that of simulating a fair coin.

Suppose we have a fair six-sided die. How can we simulate the outcome of a flip of a fair coin by rolling the die? Since there are six possible outcomes of a single die roll, each with equal probability, we can assign three outcomes (say one, two, and three) to represent "heads" and three (e.g. four, five, and six) to represent "tails". Then heads and tails each have a probability of $1/2$, matching the outcomes of a fair coin flip.



What if we only have a fair five-sided die? We can't quite follow the same strategy, since there are an odd number of possible outcomes. Instead, we can use *rejection sampling* – assign two outcomes to heads (e.g. one and two), two outcomes to tails (e.g. three and four), and reject the fifth outcome, retrying with additional rolls until we get one of the first four outcomes.



We can see that the probability of heads is $1/2$ a couple of different ways:

- We can use *conditional probability*, which expresses the probability that an event A happened given that we know that another event B happened (denoted as $\Pr[A|B]$). Let R_i be the event that we need to roll i times (i.e. the outcomes of the first $i-1$ rolls were all five, and that of the ith roll was something other than five), and let H_i be the event that we get heads on the ith roll (i.e. the roll produces a one or two). Since there are four possible outcomes on the ith roll (since we know we don't get a five), and each are equally likely, we have $\Pr[H_i|R_i] = 2/4 = 1/2$. This is true for any i, so no matter when we stop rolling, the probability of getting heads is $1/2$.
- Alternatively, we can analyze the infinitely many possible outcomes directly. On the first roll we have $2/5$ probability of heads, and $1/5$ probability of needing to reroll. In the latter case, we again have $2/5$ probability of heads and $1/5$ probability of needing to roll a third time. Since the different rolls are independent, the probability of needing to roll a second time and getting heads in the second roll is $1/5 \cdot 2/5$. Similarly, the probability of needing to roll three times and getting heads in the third roll is $1/5 \cdot 1/5 \cdot 2/5$. In general, the probability of needing to roll i times and getting heads in the last roll is $(1/5)^i \cdot 2/5$, and the overall probability of heads is

  $$\Pr[\text{heads}] = \sum_{i=0}^{\infty} (\frac{1}{5})^i \cdot \frac{2}{5}$$

  This is a geometric series[65], and the sum of a geometric series $a + ar + ar^2 + \ldots$ is

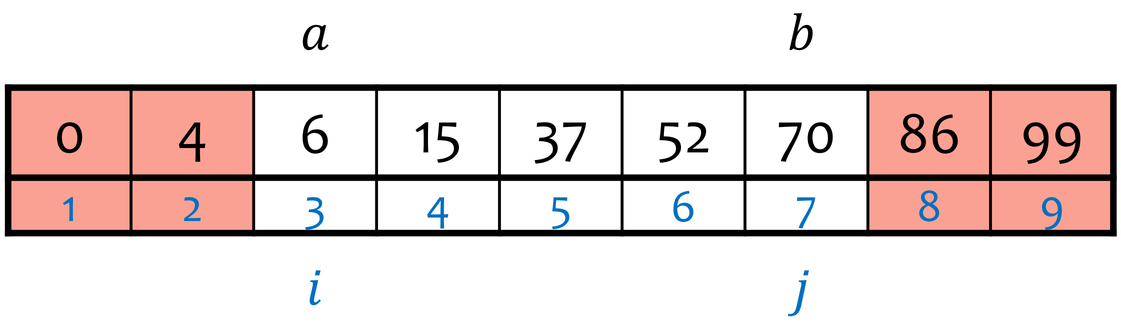
  $$\sum_{i=0}^{\infty} ar^i = \frac{a}{1-r}$$

  for $|r| < 1$. We have $a = 2/5$ and $r = 1/5$, so

  $$\begin{aligned}\Pr[\text{heads}] &= \sum_{i=0}^{\infty} (\frac{1}{5})^i \cdot \frac{2}{5} \\ &= \frac{2/5}{1-1/5} \\ &= \frac{1}{2}\end{aligned}$$

Returning to quick sort, we need to determine the probability of a or b being chosen as the pivot the first time a pivot is selected from the elements in the range $[a, b]$. If the element a has rank i and b has rank j, then there are $j - i + 1$ elements whose value lies in $[a, b]$. By the same reasoning as above, each one is equally likely to be chosen first, so the probability of a or b being chosen is $2/(j - i + 1)$.

[65] https://en.wikipedia.org/wiki/Geometric_series



Thus, the probability of a and b being compared at some point is $2/(j-i+1)$, where i is the rank of a and j is the rank of b.

We can now proceed to compute the expected number of comparisons. Let X_{ij} be an indicator random variable such that $X_{ij}=1$ if the elements of rank i and j are compared at some point in the algorithm, $X_{ij}=0$ otherwise. We have shown that

$$\Pr[X_{ij}=1]=\frac{2}{j-i+1}$$

Since each pair of elements is compared at most once, the total number of comparisons X is just the number of pairs that are compared, out of the $n(n-1)/2$ distinct pairs:

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

We have

$$\mathbb{E}[X_{ij}]=\Pr[X_{ij}=1]=\frac{2}{j-i+1}$$

and by linearity of expectation, we get

$$\begin{aligned}\mathbb{E}[X]&=\sum_{i=1}^{n}\sum_{j=i+1}^{n}\mathbb{E}[X_{ij}]\\&=\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{2}{j-i+1}\end{aligned}$$

Let $t=j-i+1$. We can then rewrite this sum as

$$\begin{aligned}\mathbb{E}[X]&=\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{2}{j-i+1}\\&=\sum_{i=1}^{n}\sum_{t=2}^{n-i+1}\frac{2}{t}\\&=2\cdot\sum_{i=1}^{n}\sum_{t=2}^{n-i+1}\frac{1}{t}\end{aligned}$$

The quantity $\sum_{t=2}^{n-i+1}\frac{1}{t}$ is a partial sum of the harmonic series[66], which has an upper bound

$$\sum_{m=1}^{k}\frac{1}{m}\leq\ln k+1$$

[66] https://en.wikipedia.org/wiki/Harmonic_series_(mathematics)

Thus, we have

$$\begin{aligned}\sum_{t=2}^{n-i+1} \frac{1}{t} &= -1 + \sum_{t=1}^{n-i+1} \frac{1}{t} \\ &\le -1 + \ln(n-i+1) + 1 \\ &= \ln(n-i+1) \\ &\le \ln n \quad (\text{for } i \ge 1)\end{aligned}$$

Plugging this into our expression for $\mathbb{E}[X]$, we get

$$\begin{aligned}\mathbb{E}[X] &= 2 \cdot \sum_{i=1}^{n} \sum_{t=2}^{n-i+1} \frac{1}{t} \\ &\le 2 \cdot \sum_{i=1}^{n} \ln n \\ &= 2n \ln n \\ &= O(n \log n)\end{aligned}$$

Thus, the expected number of comparisons is $O(n \log n)$, as previously claimed.

In practice, randomized quick sort is faster than the deterministic version as well as other deterministic $O(n \log n)$ sorts such as *merge sort* (page 13). It is also much simpler to implement *in place* (i.e. without using an auxiliary data structure). Thus, variants of randomized quick sort are widely used in practice.

Exercise 189 Randomized quick sort has worst-case running time $\Theta(n^2)$ but expected running time $O(n \log n)$ when run on any array of n elements.

Prove that for an array A of n elements and any constant $c > 0$,

$$\lim_{n\to\infty} \Pr[\text{randomized quick sort on input } A \text{ takes at least } cn^2 \text{ steps}] = 0$$

That is, randomized quick sort is very unlikely to run in $\Omega(n^2)$ time.

Hint: The number of steps a randomized algorithm takes is a random variable. You do not need to know anything about how randomized quick sort works; just use the facts above and Markov's inequality.

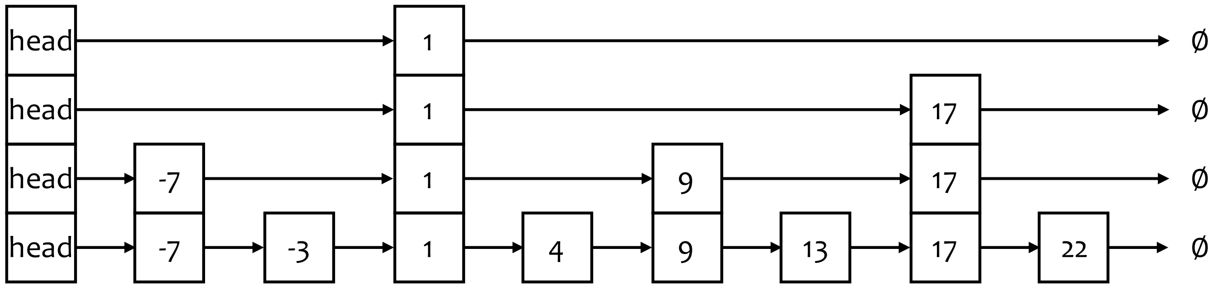
19.4 Skip Lists

Randomness can also be utilized to construct simple data structures that provide excellent expected performance. As an example, we consider a set abstract data type, which is a container that supports inserting, deleting, and searching for elements of some mutually comparable type. There are many ways to implement a set abstraction, using a variety of underlying data structures including linked lists, arrays, and binary search trees. However, simple, deterministic implementations suffer a linear runtime in the worst case for some operations. For example, if an adversary inserts the items $1, 2, \dots, n$ into a non-balancing binary search tree, the insertions take $O(n)$ time each, for a total time of $O(n^2)$. The usual way to avoid this is to perform complicated rebalancing operations, and there are a variety of schemes to do so (e.g. AVL trees[67], red-black trees[68], scapegoat trees[69], and so on). Can we use randomness instead to design a simple data structure that provides expected $O(\log n)$ runtime for all operations on a set of size n?

One solution is a *skip list*, which is essentially a linked list with multiple levels, where each level contains about half the elements of the level below (ratios other than $\frac{1}{2}$ may be used instead, trading off the space and time overheads). Whether

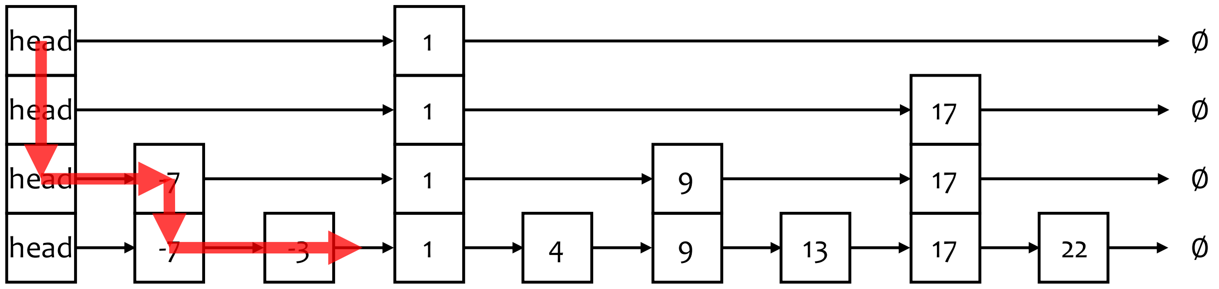
[67] https://en.wikipedia.org/wiki/AVL_tree
[68] https://en.wikipedia.org/wiki/Red%E2%80%93black_tree
[69] https://en.wikipedia.org/wiki/Scapegoat_tree

or not an element is duplicated to the next level is a random decision, which prevents an adversary from coming up with a sequence of operations that degrades performance. The following is an illustration of a skip list:

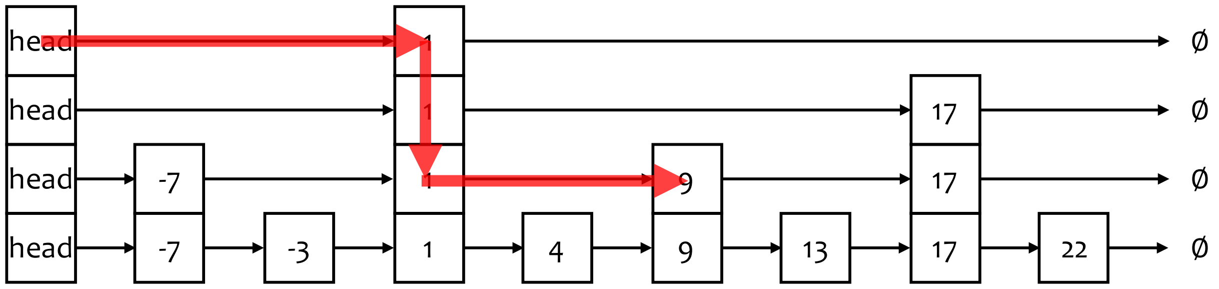


In the scheme illustrated above, each level has a sentinel head and is terminated by a null pointer. An element may appear at multiple levels, and the list may be traversed by following links rightward or by moving downward at a duplicated element.

To search for an item, we start at the sentinel node at the top left. We maintain the invariant that we are always located at a node whose value is less than the item we are looking for (or is a sentinel head node). Then in each step, we check the node to the right. If its value is greater than the item we are looking for, or if the right pointer is null, we move downward; however, if we are on the lowest level, we cannot move downward, so the item is not in the list. If the value of the node to the right is less than the item, we move to the right. If the value is equal to the item, we have found the item and terminate the search. The following illustrates a search for the item 0 in the list above:



The search terminates in failure, since we reach the value -3 in the bottom level with the value 1 to the right. The following illustrates searching for the item 9:



To insert an item, we first do a search for the item. If the item is in the list, then no further work is necessary. If it is not, the search terminates in the bottom level at the maximal element that is less than the item. Therefore, we insert the item immediately to the right. We then make a random decision of whether or not to duplicate the element to the next level. If the answer is yes, we make another random decision for the next level, and so on until the answer is no. If the probability of duplicating an item is $\frac{1}{2}$, the expected number of copies of an item is 2. Thus, insertion only takes an expected constant amount of time in addition to the search.

We proceed to determine the expected time of a search on a skip list. We start by determining the expected number of elements on each level i. Suppose the skip list contains n elements total. Let n_i be the number of elements on level i. We have $n_1 = n$, since the first level contains all elements. For $i > 1$, define X_{ij} to be an indicator random variable for whether or not the jth element is replicated to level i. We have

$$\Pr[X_{ij} = 1] = 1/2^{i-1}$$

since it takes $i - 1$ "yes" decisions for the element to make it to level i, and each decision is made independently with probability $1/2$. Thus, $\mathbb{E}[X_{ij}] = \Pr[X_{ij} = 1] = 1/2^{i-1}$.

The total number of elements n_i on level i is just the sum of the indicators X_{ij}. This gives us:

$$\begin{aligned}\mathbb{E}[n_i] &= \mathbb{E}\left[\sum_j X_{ij}\right]\\ &= \sum_j \mathbb{E}[X_{ij}]\\ &= \sum_j \frac{1}{2^{i-1}}\\ &= \frac{n}{2^{i-1}}\end{aligned}$$

Next, we consider how many levels we expect the list to contain. Suppose we only maintain levels that have at least one element. From our previous result, the expected number of elements is 1 for level i when

$$\begin{aligned}\frac{n}{2^{i-1}} &= 1\\ n &= 2^{i-1}\\ \log n &= i - 1\\ i &= \log n + 1\end{aligned}$$

Intuitively, this implies we should expect somewhere around $\log n$ levels. To reason about the expected number more formally, consider the number of elements on some level $i = \log n + k$. By Markov's inequality, we have

$$\begin{aligned}\Pr[n_{\log n+k} \geq 1] &\leq \mathbb{E}[n_{\log n+k}]/1\\ &= \frac{n}{2^{\log n+k-1}}\\ &= \frac{n}{n2^{k-1}}\\ &= \frac{1}{2^{k-1}}\end{aligned}$$

Then let Z_k be an indicator for whether $n_{\log n+k} \geq 1$. The number of levels L is at most

$$L \leq \log n + \sum_{k=1}^{\infty} Z_k$$

We have

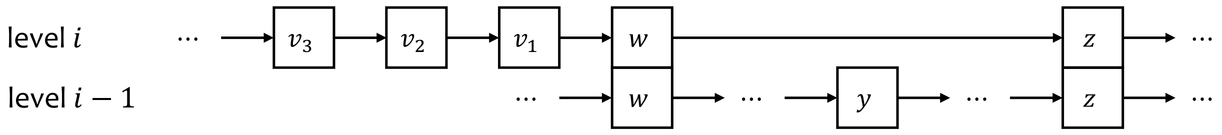
$$\begin{aligned}\mathbb{E}[Z_k] &= \Pr[Z_k = 1]\\ &= \Pr[n_{\log n+k} \geq 1]\\ &\leq \frac{1}{2^{k-1}}\end{aligned}$$

Then by linearity of expectation,

$$\begin{aligned}\mathbb{E}[L] &\le \mathbb{E}\left[\log n + \sum_{k=1}^{\infty} Z_k\right]\\ &= \log n + \sum_{k=1}^{\infty} \mathbb{E}[Z_k]\\ &\le \log n + \sum_{k=1}^{\infty} \frac{1}{2^{k-1}}\\ &= \log n + 2\end{aligned}$$

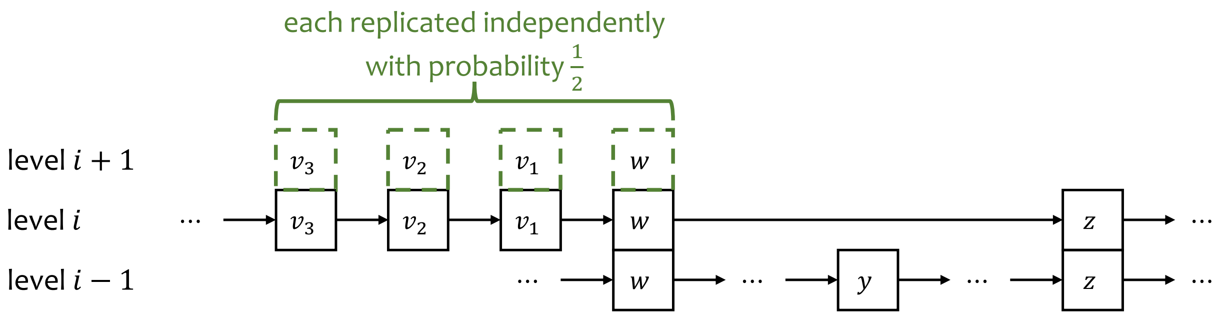
Thus, the expected number of levels is $O(\log n)$.

Finally, we reason about how many nodes a search encounters on each level i. Suppose we are looking for some element y. Let w be the maximal element on level i that is less than y (or the sentinel head if no such element exists), and let z be the minimal element on level i that is greater than or equal to y (or the null sentinel if no such element exists). Let $v_1, v_2, \dots$ be the elements on level i that are less than w, with v_1 the largest, v_2 the next largest, and so on.



The search encounters the nodes corresponding to w and z. It does not touch nodes on level i that come after z – either $y = z$, in which case the search terminates, or $z > y$ (or z is null), in which case the search moves down to level $i - 1$. Thus, we need only consider whether the search encounters the nodes corresponding to $v_1, v_2, \dots$.

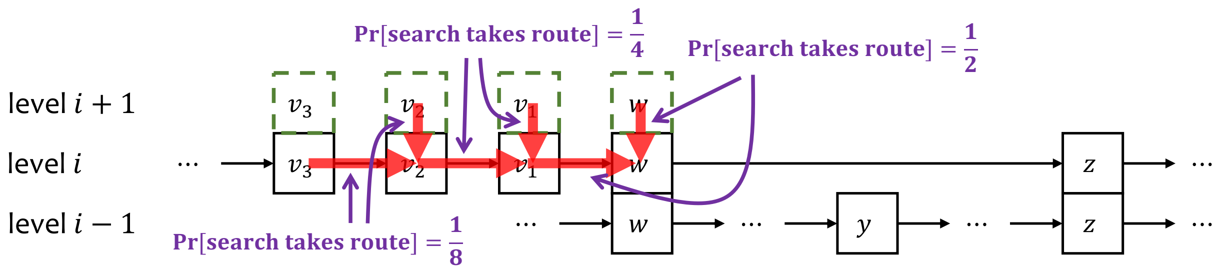
We observe that each element on level i is replicated on level $i + 1$ with probability $1/2$.


each replicated independently with probability $\frac{1}{2}$
level $i+1$
level i
level $i-1$
v_3
v_2
v_1
w
z
y
...

There are two possible routes the search can take to get to w on level i:

- The search can come down from level $i + 1$ to level i at w. This happens exactly when w is replicated on level $i+1$. The search process proceeds along the same level until it reaches a node that is greater than the target value y, and since $w < y$, it would not come down to level i before w if w appears on level $i + 1$. Thus, this possibility occurs with probability $1/2$.
- The search can come from the node corresponding to v_1 on level i. This occurs when w is not replicated on level $i + 1$. In that case, the search must come down to level i before w, in which case it goes through v_1. This possibility also occurs with probability $1/2$.

These two possibilities are illustrated below:


Pr[search takes route] = 1/4
Pr[search takes route] = 1/2
Pr[search takes route] = 1/8
level i + 1
level i
level i − 1
v3
v2
v1
w
z
y
…

Thus, v_1 is encountered on level i exactly when w is not replicated to level $i+1$, which occurs with probability $1/2$. We can inductively repeat the same reasoning for preceding elements. The search only encounters v_2 on level i if neither w nor v_1 are replicated on level $i+1$. Since they are replicated independently with probability $1/2$, the probability that neither is replicated is $1/4$. In general, the search encounters v_j on level i exactly when none of $w, v_1, v_2, \dots, v_{j-1}$ are replicated to level $i+1$, which happens with probability $1/2^j$.

We can now compute the expected number of nodes encountered on level i. Let Y_i be this number, and let V_{ij} be an indicator for whether or not element v_j is encountered. Then we have

$$Y_i = 2 + \sum_j V_{ij}$$

The first term 2 is due to the fact that the search encounters the nodes corresponding to w and z. From our reasoning above, we have

$$\mathbb{E}[V_{ij}] = \Pr[V_{ij} = 1] = \frac{1}{2^j}$$

Then by linearity of expectation,

$$\begin{aligned}\mathbb{E}[Y_i] &= \mathbb{E}\left[2 + \sum_j V_{ij}\right]\\ &= 2 + \sum_j \mathbb{E}[V_{ij}]\\ &= 2 + \sum_j \frac{1}{2^j}\\ &< 2 + \sum_{k=1}^{\infty} \frac{1}{2^k}\\ &= 3\end{aligned}$$

The second-to-last step follows from the fact that $\sum_j \frac{1}{2^j}$ has finitely many terms since there are only finitely many elements less than y. Thus, the expected number of nodes encountered on a level i is $O(1)$.

Combining this with our previous result that the expected number of levels is $O(\log n)$, the expected total number of nodes encountered across all levels is $O(\log n)$. Thus, a search on a skip list does an expected $O(\log n)$ comparisons. Since insertion does only a constant number of operations in addition to the search, it too does $O(\log n)$ operations, and the same reasoning holds for deletion as well. We achieve this expected runtime with a much simpler implementation than a self-balancing search tree.

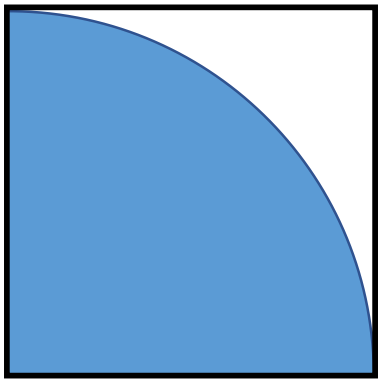
CHAPTER

TWENTY

MONTE CARLO METHODS AND CONCENTRATION BOUNDS

Some algorithms rely on repeated trials to compute an approximation of a result. Such algorithms are called *Monte Carlo methods*, which are distinct from *Monte Carlo algorithms* (page 278) – a Monte Carlo *method* does repeated sampling of a probability distribution, while a Monte Carlo *algorithm* is an algorithm that may produce the wrong result within some bounded probability. The former are commonly approximation algorithms for estimating a quantity, while the latter are exact algorithms that sometimes produce the wrong result. *As we will see* (page 281), there is a connection – repeated sampling (the strategy of a Monte Carlo method) can be used to amplify the probability of getting the correct result from a Monte Carlo algorithm.

As an example of a Monte Carlo method, we consider an algorithm for estimating the value of π, the area of a unit circle (a circle with a radius of one). If such a circle is located in the plane, centered at the origin, its top-right quadrant is as follows:



This quadrant falls within the square interval between $(0, 0)$ and $(1, 1)$. The area of the quadrant is $\pi/4$, while the area of the interval is 1. Thus, if we choose a random point in this interval, the probability that it lies within the quadrant of the circle is $\pi/4$[70]. This motivates the following algorithm for estimating the value of π:

```
function EstimatePi($n$)
    count $= 0$
    for $i = 1$ to $n$ do
        $x, y =$ values in $[0, 1]$ chosen uniformly and independently at random
        if $x^2 + y^2 \leq 1$ then
            count $=$ count $+ 1$
    return $4 \cdot$ count$/n$
```

The algorithm randomly chooses points between $(0, 0)$ and $(1, 1)$, counting how many of them fall within the unit circle, which is when the point's distance from the origin is at most one. We expect this number to be $\pi/4$ of the samples, so the algorithm returns four times the ratio of the points that fell within the circle as an estimate of π.

[70] This follows from applying the tools of continuous probability, which we will not discuss in any detail in this text.

We formally show that the algorithm returns the value of π in expectation. Let X be a random variable corresponding to the value returned, and let Y_i be an indicator random variable that is 1 if the ith point falls within the unit circle. As argued previously, we have:

$$\Pr[Y_i = 1] = \pi/4$$

We also have that

$$X = 4/n \cdot (Y_1 + \cdots + Y_n)$$

This leads to the following expected value for X:

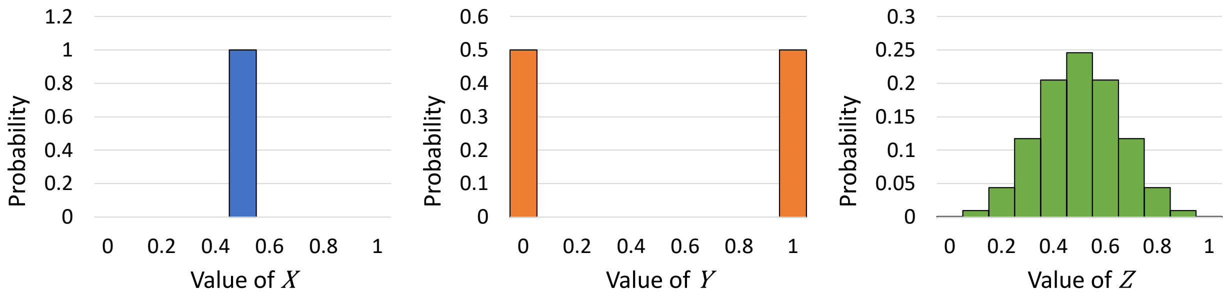
$$\begin{aligned}\mathbb{E}[X] &= 4/n \cdot (\mathbb{E}[Y_1] + \mathbb{E}[Y_2] + \cdots + \mathbb{E}[Y_n])\\ &= 4/n \cdot (\Pr[Y_1 = 1] + \Pr[Y_2 = 1] + \cdots + \Pr[Y_n = 1])\\ &= 4/n \cdot (\pi/4 + \pi/4 + \cdots + \pi/4)\\ &= \pi\end{aligned}$$

The expected value of the algorithm's output is indeed π.

When estimating π, how likely are we to actually get a result that is close to the expected value? While we can apply Markov's inequality, the bound we get is very loose, and it does not give us any information about how the number of samples affects the quality of the estimate. The *law of large numbers* states that the actual result converges to the expected value as the number of samples n increases. But how fast does it converge? There are many types of *concentration bounds* that allow us to reason about the deviation of a random variable from its expectation; Markov's inequality is just the simplest one. *Chebyshev's inequality* (page 221) is another simple bound that makes use of more information about a random variable, namely its *variance*. *Chernoff bounds* are yet another tool. There are multiple variants of Chernoff bounds, including the *multiplicative form* (page 268) and the additive *Hoeffding bounds* (page 226).

20.1 Variance and Chebyshev's Inequality

The expectation of a random variable gives us one piece of information about its probability distribution, but there are many aspects of the distribution that it does not capture. For instance, the following illustrates three random variables that have different distributions but the same expected value of 0.5:



Beyond the expectation, the next most important aspect of a probability distribution is its "spread"[71]. The *variance* of a random variable encapsulates this information.

[71] There are higher-order "moments"[Page 221, 72] of a distribution as well, such as skewness[73] and kurtosis[74]. However, expectation and variance (or standard deviation, its square root) are the most commonly used.

[72] https://en.wikipedia.org/wiki/Standardized_moment

[73] https://en.wikipedia.org/wiki/Skewness

[74] https://en.wikipedia.org/wiki/Kurtosis

Definition 190 (Variance) Suppose X is a random variable with expectation $\mathbb{E}[X]$. Then the *variance* of X is defined as

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

or, equivalently

$$\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

The second definition follows from the first due to linearity of expectation:

$$\begin{aligned}\text{Var}(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2]\\ &= \mathbb{E}[X^2 - 2\,\mathbb{E}[X]\cdot X + \mathbb{E}[X]^2]\\ &= \mathbb{E}[X^2] - 2\,\mathbb{E}[\mathbb{E}[X]\cdot X] + \mathbb{E}[\mathbb{E}[X]^2]\\ &= \mathbb{E}[X^2] - 2\,\mathbb{E}[X]\cdot\mathbb{E}[X] + \mathbb{E}[X]^2\\ &= \mathbb{E}[X^2] - 2\,\mathbb{E}[X]^2 + \mathbb{E}[X]^2\\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2\end{aligned}$$

In the fourth step, we used the fact that $\mathbb{E}[X]$ is a constant to pull it out of the outer expectation – by linearity of expectation, $\mathbb{E}[cY] = c\,\mathbb{E}[Y]$ for a constant c.

The variance tells us the average square of the distance of a random variable from its expectation. Taking the square root of the variance gives us an approximate measure of the distance itself; this is called the *standard deviation*, and it is often denoted by the symbol σ:

$$\sigma(X) = \sqrt{\text{Var}(X)}$$

Example 191 Suppose the random variable X has the distribution

$$X = \begin{cases} 0.5 & \text{with probability } 1 \end{cases}$$

and Y has the distribution

$$Y = \begin{cases} 0 & \text{with probability } 1/2\\ 1 & \text{with probability } 1/2 \end{cases}$$

Both X and Y have an expected value of 0.5. We compute their variances. The distributions of X^2 and Y^2 are as follows:

$$X^2 = \begin{cases} 0.25 & \text{with probability } 1 \end{cases}$$

$$Y^2 = \begin{cases} 0 & \text{with probability } 1/2\\ 1 & \text{with probability } 1/2 \end{cases}$$

Then we have:

$$\begin{aligned}\text{Var}(X) &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 = 0.25 - 0.5^2 = 0\\ \text{Var}(Y) &= \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 = 0.5 - 0.5^2 = 0.25\end{aligned}$$

We see that while X and Y have the same expectation, their variances do differ.

Example 192 Let X be an indicator random variable with probability p of being 1:

$$X = \begin{cases} 0 & \text{with probability } 1-p \\ 1 & \text{with probability } p \end{cases}$$

Then $\mathbb{E}[X] = \Pr[X = 1] = p$. Observe that X^2 is always equal to X, so $\mathbb{E}[X^2] = \mathbb{E}[X] = p$. Thus, the variance of X is

$$\begin{aligned} \operatorname{Var}(X) &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= p - p^2 \\ &= p(1-p) \end{aligned}$$

As a more complex example, define X to be the number of heads over n flips of a biased coin with probability p of heads. We computed in Example 181 that $\mathbb{E}[X] = np$. If we define X_i to be an indicator for whether or not flip i produces heads, we can infer from Example 192 that $\operatorname{Var}(X_i) = p(1-p)$. What about $\operatorname{Var}(X) = \operatorname{Var}(\sum_i X_i)$?

For expectation, we know from Theorem 178 that linearity of expectation always holds. However, for variance, this is not necessarily the case. As an example, for a random variable Y, define $Z = Y + Y = 2Y$. It is **not** the case that $\operatorname{Var}(Z) = 2\operatorname{Var}(Y)$; in actuality, $\operatorname{Var}(Z) = 4\operatorname{Var}(Y)$.

Theorem 193 *Suppose X is a random variable. Then for any constant c,* $\operatorname{Var}(cX) = c^2 \operatorname{Var}(X)$.

Proof 194 By definition of variance, we have

$$\begin{aligned} \operatorname{Var}(cX) &= \mathbb{E}[(cX)^2] - \mathbb{E}[cX]^2 \\ &= \mathbb{E}[c^2X^2] - (c\,\mathbb{E}[X])^2 \\ &= c^2\,\mathbb{E}[X^2] - c^2\,\mathbb{E}[X]^2 \\ &= c^2(\mathbb{E}[X^2] - \mathbb{E}[X]^2) \\ &= c^2 \operatorname{Var}(X) \end{aligned}$$

In the second and third steps, we applied linearity of expectation to pull the constants c and c^2 out of the expectations. □

For a more general sum of random variables, the variances only add if the random variables are independent. To establish this, we first demonstrate that the expectation of the product of independent random variables is the product of their expectations.

Lemma 195 *Let X and Y be independent random variables. Then*

$$\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$$

Proof 196 By definition of expectation, we have

$$\begin{aligned}\mathbb{E}[XY] &= \sum_{x,y} xy \Pr[X = x, Y = y] \\ &= \sum_x \sum_y xy \Pr[X = x] \cdot \Pr[Y = y] \\ &= \sum_x (x \Pr[X = x] \sum_y y \Pr[Y = y]) \\ &= \sum_x x \Pr[X = x] \cdot \mathbb{E}[Y] \\ &= \mathbb{E}[Y] \sum_x x \Pr[X = x] \\ &= \mathbb{E}[Y] \cdot \mathbb{E}[X]\end{aligned}$$

In the second step, we used the fact that X and Y are independent, so that $\Pr[X = x, Y = y] = \Pr[X = x] \cdot \Pr[Y = y]$. □

By induction, we can conclude that

$$\mathbb{E}\left[\prod_i X_i\right] = \prod_i \mathbb{E}[X_i]$$

when the X_i are independent.

We now consider the variance of the sum of independent random variables.

Theorem 197 *Suppose X and Y are independent random variables. Then* $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$.

Proof 198 By definition of variance, we have

$$\begin{aligned}\mathrm{Var}(X + Y) &= \mathbb{E}[(X + Y)^2] - \mathbb{E}[X + Y]^2 \\ &= \mathbb{E}[X^2 + 2XY + Y^2] - \mathbb{E}[X + Y]^2 \\ &= \mathbb{E}[X^2] + 2\,\mathbb{E}[XY] + \mathbb{E}[Y^2] - (\mathbb{E}[X] + \mathbb{E}[Y])^2 \\ &= \mathbb{E}[X^2] + 2\,\mathbb{E}[XY] + \mathbb{E}[Y^2] - (\mathbb{E}[X]^2 + 2\,\mathbb{E}[X]\,\mathbb{E}[Y] + \mathbb{E}[Y]^2) \\ &= \mathbb{E}[X^2] + 2\,\mathbb{E}[X]\,\mathbb{E}[Y] + \mathbb{E}[Y^2] - (\mathbb{E}[X]^2 + 2\,\mathbb{E}[X]\,\mathbb{E}[Y] + \mathbb{E}[Y]^2) \\ &= \mathbb{E}[X^2] + \mathbb{E}[Y^2] - \mathbb{E}[X]^2 - \mathbb{E}[Y]^2 \\ &= \mathrm{Var}(X) + \mathrm{Var}(Y)\end{aligned}$$

In the third step, we applied linearity of expectation, and we applied Lemma 195 in the fifth step. □

By induction, we can conclude that

$$\mathrm{Var}\left(\sum_i X_i\right) = \sum_i \mathrm{Var}(X_i)$$

when the X_i are independent. Thus, if X is the number of heads over n flips of a biased coin with probability p of heads, we have

$$\begin{aligned}\mathrm{Var}(X) &= \mathrm{Var}\left(\sum_i X_i\right)\\ &= \sum_i \mathrm{Var}(X_i)\\ &= \sum_i p(1-p)\\ &= np(1-p)\end{aligned}$$

While variance tells us the expected squared distance from the expectation, *Chebyshev's inequality* allows us to bound the probability of the absolute distance exceeding a given threshold.

Theorem 199 (Chebyshev's Inequality) *Let X be a random variable and let $a > 0$. Then*

$$\Pr[|X - \mathbb{E}[X]| \ge a] \le \frac{\mathrm{Var}(X)}{a^2}$$

Proof 200 We can derive Chebyshev's inequality by applying *Markov's inequality* (page 206) to the random variable $Y = (X - \mathbb{E}[X])^2$. Observe that since Y is a squared quantity, it is always nonnegative, so Markov is applicable. Furthermore, for a nonnegative a, $(X - \mathbb{E}[X])^2 \ge a^2$ exactly when $|X - \mathbb{E}[X]| \ge a$. Thus:

$$\begin{aligned}\Pr[|X - \mathbb{E}[X]| \ge a] &= \Pr[(X - \mathbb{E}[X])^2 \ge a^2]\\ &\le \frac{\mathbb{E}[(X - \mathbb{E}[X])^2]}{a^2}\\ &= \frac{\mathrm{Var}(X)}{a^2}\end{aligned}$$

Example 201 Suppose we flip a fair coin n times. What is the probability of of getting $\le 49\%$ or $\ge 51\%$ heads?

Let X be the number of heads. We previously computed that $\mathbb{E}[X] = \frac{n}{2}$, and $\mathrm{Var}(X) = n \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{n}{4}$. Then

$$\begin{aligned}\Pr\left[\left|X - \frac{n}{2}\right| \ge 0.01n\right] &\le \frac{\mathrm{Var}(X)}{(0.01n)^2}\\ &= 10000\frac{n \cdot 1/4}{n^2}\\ &= \frac{2500}{n}\end{aligned}$$

Thus, for 10,000 flips, the probability of deviating from the expectation by 1% is at most 1/4, while for 1,000,000 flips, it is at most 1/400.

In the example above, the random variable X can be defined as the sum of *independent, identically distributed (i.i.d.)* random variables $X = X_1 + \cdots + X_n$. In such a situation, we often prefer to reason about X/n rather than X itself, since the expectation of X/n is independent of n; in particular, $\mathbb{E}[X] = n\,\mathbb{E}[X_i]$, whereas $\mathbb{E}[X/n] = \mathbb{E}[X_i]$. The following is an alternative expression of Chebyshev's inequality for such a case:

Corollary 202 *Let $X = X_1 + \cdots + X_n$ be the sum of n independent, identically distributed random variables. Let $\varepsilon > 0$ be a deviation from the expectation $\mathbb{E}[X/n] = \mathbb{E}[X_i]$. Then*

$$\Pr\left[\left|\frac{1}{n}X - \mathbb{E}[X_i]\right| \ge \varepsilon\right] \le \frac{\mathrm{Var}(X_i)}{\varepsilon^2 n}$$

Proof 203 The event $|X/n - \mathbb{E}[X_i]| \geq \varepsilon$ is equivalent to the event $|X - \mathbb{E}[X]| \geq \varepsilon n$, since $\mathbb{E}[X] = n\,\mathbb{E}[X_i]$ (multiply both sides of the first inequality by n). By the standard form of Chebyshev's inequality, we have

$$\begin{aligned}\Pr\left[\left|\frac{1}{n}X - \mathbb{E}[X_i]\right| \geq \varepsilon\right] &= \Pr[|X - \mathbb{E}[X]| \geq \varepsilon n] \\ &\leq \frac{\mathrm{Var}(X)}{\varepsilon^2 n^2} \\ &= \frac{n\,\mathrm{Var}(X_i)}{\varepsilon^2 n^2} \\ &= \frac{\mathrm{Var}(X_i)}{\varepsilon^2 n}\end{aligned}$$

In the second-to-last step, we used Theorem 197 to deduce that $\mathrm{Var}(X) = n\,\mathrm{Var}(X_i)$ since the X_i are independent. □

Example 204 Suppose we flip a fair coin n times. What is the probability of of getting $\leq 49\%$ or $\geq 51\%$ heads?

Let X_i be an indicator for whether flip i produces heads, and let $X = X_1 + \cdots + X_n$. We have $\mathbb{E}[X/n] = \mathbb{E}[X_i] = 1/2$, and $\mathrm{Var}(X_i) = 1/4$. Then

$$\begin{aligned}\Pr\left[\left|\frac{1}{n}X - \frac{1}{2}\right| \geq 0.01\right] &\leq \frac{\mathrm{Var}(X_i)}{0.01^2 n} \\ &= 10000\frac{1/4}{n} \\ &= \frac{2500}{n}\end{aligned}$$

as before.

Using Chebyshev's inequality, we observe that the probability of deviating by 1% from the expectation over n flips of a coin decreases (at least) linearly in the number of flips n. In reality, the probability decreases much faster than this – using *multiplicative Chernoff bounds* (page 268) or *Hoeffding's inequality* (page 226), we can conclude that the probability decreases exponentially in the number of flips.

20.2 Hoeffding's Inequality

Hoeffding's inequality, also called *Chernoff-Hoeffding bounds*, is a set of concentration bounds that can give tighter results than Markov's inequality, Chebyshev's inequality, or other forms of Chernoff bounds. For simplicity, we restrict ourselves to the special case[75] of independent indicator random variables, with expectation

$$\Pr[X_i = 1] = p_i$$

for each indicator X_i.

Let p be the expected value of $\frac{1}{n}X$. We have

$$\begin{aligned}p &= \mathbb{E}\left[\frac{1}{n}X\right] \\ &= \frac{1}{n}\sum_i \mathbb{E}[X_i] \\ &= \frac{1}{n}\sum_i p_i\end{aligned}$$

[75] See the *appendix* (page 300) for the general case of Hoeffding's inequality, as well as proofs of the *special* (page 296) and *general* (page 300) cases.

Hoeffding's inequality places a limit on how much the actual value of $\frac{1}{n}X$ may deviate from the expectation p.

Theorem 205 (Hoeffding's Inequality – Upper Tail) *Let* $X = X_1 + \cdots + X_n$*, where the* X_i *are independent indicator random variables with* $\mathbb{E}[X_i] = p_i$*. Let*

$$p = \mathbb{E}\left[\frac{1}{n}X\right] = \frac{1}{n}\sum_i p_i$$

Let $\varepsilon > 0$ *be a deviation from the expectation. Then*

$$\Pr\left[\frac{1}{n}X \geq p + \varepsilon\right] \leq e^{-2\varepsilon^2 n}$$

Theorem 206 (Hoeffding's Inequality – Lower Tail) *Let* $X = X_1 + \cdots + X_n$*, where the* X_i *are independent indicator random variables with* $\mathbb{E}[X_i] = p_i$*. Let*

$$p = \mathbb{E}\left[\frac{1}{n}X\right] = \frac{1}{n}\sum_i p_i$$

Let $\varepsilon > 0$ *be a deviation from the expectation. Then*

$$\Pr\left[\frac{1}{n}X \leq p - \varepsilon\right] \leq e^{-2\varepsilon^2 n}$$

Theorem 207 (Hoeffding's Inequality – Combined) *Let* $X = X_1 + \cdots + X_n$*, where the* X_i *are independent indicator random variables with* $\mathbb{E}[X_i] = p_i$*. Let*

$$p = \mathbb{E}\left[\frac{1}{n}X\right] = \frac{1}{n}\sum_i p_i$$

Let $\varepsilon > 0$ *be a deviation from the expectation. Then*

$$\Pr\left[\left|\frac{1}{n}X - p\right| \geq \varepsilon\right] \leq 2e^{-2\varepsilon^2 n}$$

We consider again the example of flipping a fair coin with probability. Let H be the total number of heads, and let H_i be an indicator variable corresponding to whether the ith flip is heads. We have $\Pr[H_i = 1] = \frac{1}{2}$, and $\mathbb{E}[\frac{1}{n}H] = \frac{1}{2}$ for any number of flips n.

What is the probability of getting at least six heads out of ten flips? This is a deviation of $\varepsilon = 0.1$, and applying the upper-tail Hoeffding's inequality gives us:

$$\begin{aligned}\Pr\left[\frac{1}{n}H \geq p + \varepsilon\right] &= \Pr\left[\frac{1}{n}H \geq 0.5 + 0.1\right] \\ &\leq e^{-2\cdot 0.1^2\cdot 10} \\ &\approx 0.9802^{10} \\ &\approx 0.82\end{aligned}$$

What is the probability of getting at least 60 heads out of 100 flips, which is the same deviation $\varepsilon = 0.1$ from the expectation? Applying the upper tail again, we get

$$\Pr\left[\frac{1}{n}H \geq p + \varepsilon\right] \leq e^{-2\cdot 0.1^2\cdot 100} \approx 0.9802^{100} \approx 0.14$$

This is a significantly tighter bound than that produced by the *multiplicative Chernoff bound* (page 268).

Example 208 Suppose we have a coin that is biased by probability ε towards either heads or tails, but we don't know which one. In other words, either:

- $\Pr[\text{heads}] = \frac{1}{2} + \varepsilon$ and $\Pr[\text{tails}] = \frac{1}{2} - \varepsilon$, or
- $\Pr[\text{heads}] = \frac{1}{2} - \varepsilon$ and $\Pr[\text{tails}] = \frac{1}{2} + \varepsilon$

To determine in which direction the coin is biased, we flip the coin n times. If the results include at least $n/2$ heads, we assume the coin is biased towards heads, otherwise we assume it is biased towards tails. How many flips should we do to guarantee our answer is correct with probability at least $1 - \delta$?

Let X be the number of heads, and let X_i be an indicator that is 1 if the ith flip is heads, 0 if it is tails. Then $X = X_1 + \cdots + X_n$ is the sum of independent indicator random variables with $\mathbb{E}[X_i]$ equal to either $\frac{1}{2} + \varepsilon$ or to $\frac{1}{2} - \varepsilon$. We analyze the two cases individually.

- **Case 1:** The coin is biased towards heads. Then $\mathbb{E}[X_i] = p = \frac{1}{2} + \varepsilon$. Our guess is erroneous when:

$$\begin{aligned} X &< \frac{n}{2} \\ \frac{1}{n}X &< \frac{1}{2} \\ &= (\frac{1}{2} + \varepsilon) - \varepsilon \\ &= p - \varepsilon \end{aligned}$$

  By the lower-tail Hoeffding's inequality, we have

$$\begin{aligned} \Pr\left[\frac{1}{n}X < p - \varepsilon\right] &\le \Pr\left[\frac{1}{n}X \le p - \varepsilon\right] \\ &\le e^{-2\varepsilon^2 n} \end{aligned}$$

  In the first step, we used the fact that $\Pr[Y \le a] = \Pr[Y < a] + \Pr[Y = a] \ge \Pr[Y < a]$ for a random variable Y and any value a.

- **Case 2:** The coin is biased towards tails. Then $\mathbb{E}[X_i] = p = \frac{1}{2} - \varepsilon$. Our guess is erroneous when:

$$\begin{aligned} X &\ge \frac{n}{2} \\ \frac{1}{n}X &\ge \frac{1}{2} \\ &= (\frac{1}{2} - \varepsilon) + \varepsilon \\ &= p + \varepsilon \end{aligned}$$

  By the upper-tail Hoeffding's inequality, we have

$$\Pr\left[\frac{1}{n}X \ge p + \varepsilon\right] \le e^{-2\varepsilon^2 n}$$

In either case, the probability of error after n flips is upper-bounded by $e^{-2\varepsilon^2 n}$. We want this probability to be no more than δ:

$$\begin{aligned} e^{-2\varepsilon^2 n} &\le \delta \\ 1/\delta &\le e^{2\varepsilon^2 n} \\ \ln(1/\delta) &\le 2\varepsilon^2 n \\ \frac{\ln(1/\delta)}{2\varepsilon^2} &\le n \end{aligned}$$

If $\varepsilon = 0.01$ (i.e. the coin is biased towards heads or tails by 1%) and $\delta = 0.0001$ (we want to be correct at least 99.99% of the time), then

$$n \ge \frac{\ln(1/0.0001)}{2 \cdot 0.01^2} \approx 46502$$

flips suffice.

Exercise 209 Suppose we have a coin that is either fair, or is biased by probability ε towards heads, but we don't know which is the case. In other words, either:

- $\Pr[\text{heads}] = \frac{1}{2}$ and $\Pr[\text{tails}] = \frac{1}{2}$, or
- $\Pr[\text{heads}] = \frac{1}{2} + \varepsilon$ and $\Pr[\text{tails}] = \frac{1}{2} - \varepsilon$

We flip the coin n times and count the number of heads X.

a. For what values of X should we guess the coin is fair, and for what values that it is biased towards heads?

b. How many flips should we do to guarantee our answer is correct with probability at least $1 - \delta$?

c. How many flips should we do for $\varepsilon = 0.01$ and $\delta = 0.0001$? How does this compare to the situation in Example 208 with $\varepsilon = 0.005$ and $\delta = 0.0001$?

20.3 Polling

Rather than applying concentration bounds to compute the probability of a deviation for a specific sample size, we often wish to determine how many samples we need to be within a particular deviation with high confidence. One application of this is *big data*, where we have vast datasets that are too large to examine in their entirety, so we sample the data instead to estimate the quantities of interest. Similar to this is *polling* – outside of elections themselves, we typically do not have the resources to ask the entire population for their opinions, so we need to sample the populace to estimate the support for a particular candidate or political position. In both applications, we need to determine how many samples are needed to obtain a good estimate.

In general, a poll estimates the fraction of the population that supports a particular candidate by asking n randomly chosen people whether they support that candidate. Let X be a random variable corresponding to the number of people who answer this question in the affirmative. Then X/n is an estimate of the level of support in the full population.

A typical poll has both a *confidence level* and a *margin of error* – the latter corresponds to the deviation from the true fraction p of people who support the candidate, and the former corresponds to a bound on the probability that the estimate is within that deviation. For example, a 95% confidence level and a margin of error of $\pm 2\%$ requires that

$$\Pr\left[\left|\frac{X}{n} - p\right| \le 0.02\right] \ge 0.95$$

More generally, for a confidence level $1 - \gamma$ and margin of error ε, we require

$$\Pr\left[\left|\frac{X}{n} - p\right| \le \varepsilon\right] \ge 1 - \gamma$$

or equivalently

$$\Pr\left[\left|\frac{X}{n} - p\right| > \varepsilon\right] < \gamma$$

Formally, we define indicator variables X_i as

$$X_i = \begin{cases} 1 & \text{if person } i \text{ supports the candidate} \\ 0 & \text{otherwise} \end{cases}$$

for each person i in the set that we poll. Then $X = X_1 + \cdots + X_n$ is the sum of independent indicator variables, with

$$\mu = \mathbb{E}[X] = \sum_i \mathbb{E}[X_i] = np$$

20.3.1 Analysis with Hoeffding's Inequality

Applying Hoeffding's inequality to polling, the combined inequality gives us

$$\Pr\left[\left|\frac{1}{n}X - p\right| \geq \varepsilon\right] \leq 2e^{-2\varepsilon^2 n}$$

For a 95% confidence level and a margin of error of $\pm 2\%$, we require that

$$\Pr\left[\left|\frac{X}{n} - p\right| \leq 0.02\right] \geq 0.95$$

However, this isn't quite in the form where we can apply Hoeffding's inequality, so we need to do some manipulation first. We have:

$$\begin{aligned} \Pr\left[\left|\frac{X}{n} - p\right| \leq 0.02\right] &= 1 - \Pr\left[\left|\frac{X}{n} - p\right| > 0.02\right] \\ &\geq 1 - \Pr\left[\left|\frac{X}{n} - p\right| \geq 0.02\right] \end{aligned}$$

Hoeffding's inequality gives us:

$$\Pr\left[\left|\frac{X}{n} - p\right| \geq 0.02\right] \leq 2e^{-2\cdot 0.02^2\cdot n}$$

Substituting this into the above, we get:

$$\Pr\left[\left|\frac{X}{n} - p\right| \leq 0.02\right] \geq 1 - 2e^{-2\cdot 0.02^2\cdot n}$$

We want this to be at least 0.95:

$$\begin{aligned} 1 - 2e^{-2\cdot 0.02^2\cdot n} &\geq 0.95 \\ 2e^{-2\cdot 0.02^2\cdot n} &\leq 0.05 \\ e^{2\cdot 0.02^2\cdot n} &\geq 40 \\ 2\cdot 0.02^2\cdot n &\geq \ln 40 \\ n &\geq \frac{\ln 40}{2\cdot 0.02^2} \\ &\approx 4611.1 \end{aligned}$$

Thus, we obtain the given confidence level and margin of error by polling at least 4612 people. Observe that this does not depend on the total population size!

For an arbitrary margin of error $\pm\varepsilon$, we obtain:

$$\begin{aligned}\Pr\left[\left|\frac{X}{n} - p\right| \le \varepsilon\right] &= 1 - \Pr\left[\left|\frac{X}{n} - p\right| > \varepsilon\right] \\ &\ge 1 - \Pr\left[\left|\frac{X}{n} - p\right| \ge \varepsilon\right] \\ &\ge 1 - 2e^{-2\varepsilon^2 n}\end{aligned}$$

To achieve an arbitrary confidence level $1-\gamma$, we need:

$$\begin{aligned}1 - 2e^{-2\varepsilon^2 n} &\ge 1 - \gamma \\ 2e^{-2\varepsilon^2 n} &\le \gamma \\ e^{2\varepsilon^2 n} &\ge \frac{2}{\gamma} \\ 2\varepsilon^2 n &\ge \ln(\frac{2}{\gamma}) \\ n &\ge \frac{1}{2\varepsilon^2}\ln(\frac{2}{\gamma})\end{aligned}$$

More generally, if we wish to gauge the level of support for m different candidates, the *sampling theorem* tells us that the number of samples required is logarithmic in m.

Theorem 210 (Sampling Theorem) *Suppose n people are polled to ask which candidate they support, out of m possible candidates. Let $X^{(j)}$ be the number of people who state that they support candidate j, and let p_j be the true level of support for that candidate. We wish to obtain*

$$\Pr\left[\bigcap_j (\left|\frac{X^{(j)}}{n} - p_j\right| \le \varepsilon)\right] \ge 1 - \gamma$$

In other words, we desire a confidence level $1-\gamma$ that all estimates $X^{(j)}/n$ are within margin of error $\pm\varepsilon$ of the true values p_j. We obtain this when the number of samples n is

$$n \ge \frac{1}{2\varepsilon^2}\ln(\frac{2m}{\gamma})$$

The sampling theorem can be derived from applying the *union bound* (page 233), which we will discuss shortly.

In conclusion, when sampling from a large dataset, the number of samples required does depend on the desired accuracy of the estimation and the range size (i.e. number of possible answers). But it does not depend on the population size. This makes sampling a powerful technique for dealing with big data, as long as we are willing to tolerate a small possibility of obtaining an inaccurate estimate.

20.4 Load Balancing

Job scheduling is another application where we can exploit randomness to construct a simple, highly effective algorithm. In this problem, we have k servers, and there are n jobs that need to be distributed to these servers. The goal is to *balance* the load among the servers, so that no one server is overloaded. This problem is very relevant to content-delivery networks – there may be on the order of millions or even billions of concurrent users, and each user needs to be routed to one of only hundreds or thousands of servers. Thus, we have $n \gg k$ in such a case.

One possible algorithm is to always send a job to the most lightly loaded server. However, this requires significant coordination – the scheduler must keep track of the load on each server, which requires extra communication, space, and computational resources. Instead, we consider a simple, randomized algorithm that just sends each job to a random server. The expected number of jobs on each server is n/k. But how likely are we to be close to this ideal, balanced load?

We start our analysis by defining random variables $X^{(j)}$ corresponding to the number of jobs assigned to server j. We would like to demonstrate that the joint probability of $X^{(j)}$ being close to n/k for all j is high. We first reason about the load on an individual server. In particular, we wish to compute a bound on

$$\Pr\left[X^{(j)} \geq \frac{n}{k} + c\right]$$

for some value $c > 0$, i.e. the probability that server j is overloaded by at least c jobs. Let $X_i^{(j)}$ be an indicator random variable that is 1 if job i is sent to server j. Since the target server for job i is chosen uniformly at random out of k possible choices, we have

$$\mathbb{E}\left[X_i^{(j)}\right] = \Pr\left[X_i^{(j)} = 1\right] = \frac{1}{k}$$

We also have $X^{(j)} = X_1^{(j)} + \cdots + X_n^{(j)}$, giving us

$$\mathbb{E}\left[X^{(j)}\right] = \sum_i \mathbb{E}\left[X_i^{(j)}\right] = \frac{n}{k}$$

as we stated before. Then:

$$\begin{aligned}\Pr\left[X^{(j)} \geq \frac{n}{k} + c\right] &= \Pr\left[\frac{1}{n}X^{(j)} \geq \frac{1}{k} + \frac{c}{n}\right]\\ &\leq e^{-2(c/n)^2 n}\\ &= e^{-2c^2/n}\end{aligned}$$

by the upper-tail Hoeffding's inequality.

Now that we have a bound on an individual server being overloaded by c jobs, we wish to bound the probability that there is *some* server that is overloaded. The complement event is that no server is overloaded, so we can equivalently compute the joint probability that none of the servers is overloaded by c or more jobs:

$$\Pr\left[X^{(1)} < \frac{n}{k} + c, \dots, X^{(n)} < \frac{n}{k} + c\right]$$

However, we cannot do so by simply multiplying the individual probabilities together – that only works if the probabilities are independent, and in this case, they are not. In particular, if one server is overloaded, some other server must be underloaded, so the random variables $X^{(j)}$ are not independent.

Rather than trying to work with the complement event, we attempt to directly compute the probability that there is at least one overloaded server:

$$\Pr\left[(X^{(1)} \geq \frac{n}{k} + c) \cup \cdots \cup (X^{(n)} \geq \frac{n}{k} + c)\right]$$

More succinctly, we denote this as:

$$\Pr\left[\bigcup_j (X^{(j)} \geq \frac{n}{k} + c)\right]$$

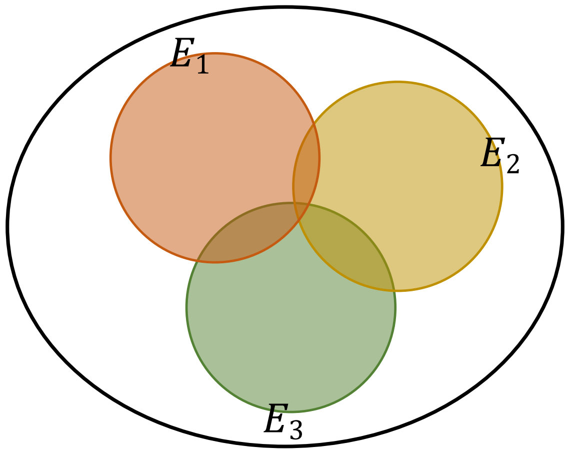
We have a union of events, and we want to compute the probability of that union. The *union bound* allows us to compute a bound on that probability.

Theorem 211 (Union Bound) *Let $E_1, E_2, \ldots, E_m$ be a sequence of events over the same sample space. Then*

$$\Pr\left[\bigcup_{i=1}^{m} E_i\right] \leq \sum_{i=1}^{m} \Pr[E_i]$$

In other words, the probability of the union is no more than the sum of the probabilities of the individual events.

To see why the union bound holds, consider the outcomes that are in the union event $\cup_i E_i$. The probability of this event is the sum of the probabilities of the individual outcomes in $\cup_i E_i$. In the maximal case, the events E_i are disjoint, so that an individual outcome is not a member of more than one E_i. Then the sum of the probabilities of the events E_i is also the sum of the probabilities of the individual outcomes in $\cup_i E_i$. In the general case, the E_i's may overlap:



In this case, the sum $\sum_i \Pr[E_i]$ overcounts the probability $\Pr[\cup_i E_i]$, since we "double count" outcomes that occur in more than one E_i. Thus, $\sum_i \Pr[E_i]$ is an upper bound on the probability $\Pr[\cup_i E_i]$.

Returning to the job-scheduling problem, we have:

$$\begin{aligned}\Pr\left[\bigcup_j (X^{(j)} \geq \frac{n}{k} + c)\right] &\leq \sum_j \Pr\left[X^{(j)} \geq \frac{n}{k} + c\right]\\ &\leq \sum_j e^{-2c^2/n}\\ &= k \cdot e^{-2c^2/n}\\ &= e^{\ln k} \cdot e^{-2c^2/n}\\ &= e^{\ln k - 2c^2/n}\end{aligned}$$

For $c = \sqrt{n \ln k}$, we get:

$$\begin{aligned} \Pr\left[\bigcup_j (X^{(j)} \geq \frac{n}{k} + \sqrt{n \ln k})\right] &\leq e^{\ln k - 2(\sqrt{n \ln k})^2/n} \\ &= e^{\ln k - 2(n \ln k)/n} \\ &= e^{-\ln k} \\ &= 1/e^{\ln k} \\ &= \frac{1}{k} \end{aligned}$$

With concrete values $n = 10^{10}$ and $k = 1000$, we compute the overload relative to the expected value as:

$$\begin{aligned} \frac{\sqrt{n \ln k}}{n/k} &= \frac{\sqrt{10^{10} \ln 1000}}{10^7} \\ &\approx 0.026 \end{aligned}$$

This is an overhead of about 2.6% above the expected load. The probability that there is a server overloaded by at least 2.6% is bounded from above by $1/k = 0.001$, or $\leq$0.1%. Thus, when there are $n = 10^{10}$ jobs and $k = 1000$ servers, the randomized algorithm has a high probability ($\geq$99.9%) of producing a schedule where the servers all have a low overhead ($\leq$2.6%).

Exercise 212 Previously, we *demonstrated* (page 230) that to use polling to achieve a confidence level $1 - \gamma$ and a margin of error $\pm\varepsilon$,

$$n \geq \frac{1}{2\varepsilon^2} \ln(\frac{2}{\gamma})$$

samples are sufficient when there is a single candidate. We also saw the *sampling theorem* (page 231), which states that for m candidates,

$$n \geq \frac{1}{2\varepsilon^2} \ln(\frac{2m}{\gamma})$$

samples suffice to achieve a confidence level $1 - \gamma$ that all estimates $X^{(j)}/n$ are within margin of error $\pm\varepsilon$.

Use the union bound to demonstrate that this latter result follows from the result for a single candidate.