Randomized Algorithms

Last Updated: May 4th, 2025

Introduction

A **randomized algorithm** is an algorithm that, as part of its procedure, may observe zero or more independent outcomes of one or more random variables. The purpose of such algorithms is to obtain a reduction in either time or space complexity (when compared to a non-randomized algorithm) at perhaps the cost of possibly returning an incorrect result. Similar to approximataion algorithms, a randomized algorithm is usually combined with one or more of the fundamental algorithm strategies described in previous lectures, and it serves a similar purpose as an approximation algorithm in that it provides a reduction in computational resources, but at the cost of degrading the correctness of the output.

However, not every randomized algorithm returns a degraded output. In fact, the first two algorithms that we study are randomized versions of `Find Statistic` and `Quicksort` and which always return a correct output while still performing at least as well if not better than their deterministic counterparts.

1 Randomized Versions of `Find Statistic` and `Quicksort`

Recall that the `Median-of-Five Find Statistic` algorithm takes as input an integer array a and natural number k, and returns the kth least member of a. Although the `Median-of-Five Find Statistic` seems both clever in its design and valuable for establishing a worst-case linear bound on computing an array statistic (including the median), on average it can be significantly outperformed by a randomized algorithm that randomly selects the pivot in each round rather than use the "median-of-five" heuristic.

We may also randomize Hoare's `Quicksort` algorithm by also randomly selecting the pivot, although in this case the improvement is not as significant given that the "median-of-three" heuristic only requires O(1) steps and tends to induce good array splits on average.

Definition 1.1. A **finite random variable** X is one whose domain is a finite set of real numbers $\text{dom}(X) = \{x_1, \ldots, x_n\}$, and associated with each domain member x_i is a probability value p_i, with the property that $p_1 + \cdots + p_n = 1$. The collection of probabilities, denoted as $\vec{p}$, is called the **probability distribution** of X. Finally, we say that X is **uniformly distributed** in case

$$p_1 = \cdots = p_n = \frac{1}{n}.$$

A similar definition may be given for a **discrete random variable**, with the only difference being that its domain is now also capable of being countably infinite: $\{x_1, x_2, \ldots\}$.

Definition 1.2. Let X be a finite random variable whose domain is $\{x_1, \ldots, x_n\}$ and whose distribution is $\vec{p} = (p_1, \ldots, p_n)$. Then the **expectation** (also referred to as **average** or **mean**) of X, denoted $E[X]$, is defined as

$$E[X] = \sum_{i=1}^{n} x_i \cdot p_i.$$

A similar definition may be given for a discrete random variable.

Example 1.3. Let X be a uniform random variable having domain $\{1, \ldots, n\}$. Then

$$E[X] = \sum_{i=1}^{n} i \cdot \frac{1}{n} = \frac{1}{n} \sum_{i=1}^{n} i = \frac{1}{n} \left(\frac{n(n+1)}{2} \right) = \frac{(n+1)}{2}. \quad \square$$

The above example shows that, in the case n is even, the average of X is a value that cannot be observed when obtaining a random sample of X.

The following result, often referred to as "linearity of expectation", will prove quite useful.

Proposition 1.4. Let X, Y, and Z be random variables, with $Z = X + Y$. Then

$$E[X + Y] = E[X] + E[Y].$$

Proof. Without loss of generality, we assume X has domain $\{x_1, \ldots, x_m\}$ and distribution $\vec{p}$, Y has domain $\{y_1, \ldots, y_n\}$ and distribution $\vec{q}$, while Z has domain $\{x_i + y_j | 1 \leq i \leq m \text{ and } 1 \leq j \leq n\}$ and distribution r_{ij}. Then we have

$$E[Z] = \sum_{i=1}^{m}\sum_{j=1}^{n}(x_i+y_j)r_{ij} = [(x_1+y_1)r_{11}+\cdots+(x_1+y_n)r_{1n}]+\cdots+[(x_m+y_1)r_{m1}+\cdots+(x_m+y_n)r_{mn}] =$$

$$[x_1(r_{11} + \cdots + r_{1n}) + \cdots + x_m(r_{m1} + \cdots + r_{mn})] + y_1(r_{11} + \cdots + r_{m1}) + \cdots + y_n(r_{1n} + \cdots + r_{mn}) =$$

$$(x_1 \cdot p_1 + \cdots + x_m \cdot p_m) + (y_1 \cdot q_1 + \cdots + y_n \cdot q_n) = E[X] + E[Y],$$

where we have used the fact that, for all $i = 1, \ldots, m$

$$p_i = r_{i1} + \cdots + r_{in},$$

and, for all $j = 1, \ldots, n$,

$$q_j = r_{1j} + \cdots + r_{mj}.$$

□

Example 1.5. Let R and B both be uniform random variables, and each having domain $\{1, \ldots, 6\}$. R (respectively, B) represents the outcome of rolling a six-sided red (respectively, blue) die. Let S be the random variable defined by $S = R + B$, i.e. S assumes the sum of the values showing for R and B. First notice that

$$E[R] = E[B] = \frac{1}{6}(1 + 2 + 3 + 4 + 5 + 6) = 21/6 = 3.5.$$

Therefore, by linearity of expectation we have

$$E[S] = E[R] + E[B] = 3.5 + 3.5 = 7. \quad \square$$

Definition 1.6. Two discrete random variables X and Y are said to be **independent** iff, for all $x_i \in \mathrm{dom}(X)$ and $y_j \in \mathrm{dom}(Y)$,

$$P(X = x_i \wedge Y = y_j) = P(X = x_i) \cdot P(Y = y_j).$$

Example 1.7. Let R and B be the random variables defined in Example 1.5. Rolled together, there are $6 \times 6 = 36$ different possible outcomes in terms of the pair of values (r, b) that appear face up after the dice are rolled. Moreover, for a pair of "fair" dice it is customary to assume that each pair is equally likely to appear, i.e.

$$P(R = r \wedge B = b) = \frac{1}{36} = \frac{1}{6} \times \frac{1}{6} = P(R = r) \cdot P(B = b).$$

Therefore, as a consequence of the fairness assumption, we are also assuming that the outcome of one die is independent of the outcome of the other. □

Definition 1.8. A **Bernouli random variable** X, written $X \sim \text{Be}(p)$, $0 < p < 1$, is a finite random variable having domain $\{0, 1\}$ and for which $P(X = 1) = p$.

A Bernoulli random variable is sometimes referred to as a **binary random variable** or as an **indicator random variable** since it is often used to indicate whether or not some event has occurred.

Definition 1.9. A **Geometric random variable**, denoted $G(p)$, $0 < p < 1$, is a discrete random variable having domain $\{1, 2, \ldots\}$ and for which $p_i = (1 - p)^{i-1}p$.

A useful way to view a geometric random variable is that it represents an infinite sequence of independent Bernoulli random variables $X_1, X_2, \ldots$, and assumes the value i, where i is the least index for which $X_i = 1$. Thus, given a sequence of independent experiments in which each experiment has a probability p of success, a geometric random variable assumes the number of trials that have elapsed when the first successful experiment is observed.

Proposition 1.10. If $X \sim G(p)$ be a geometric random variable, then $E[X] = \frac{1}{p}$

Proof. For all $0 < x < 1$, the geometric-series summation formula gives

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}.$$

Moreover, differentiating both sides of the equation yields, for all $0 < x < 1$,

$$\sum_{i=1}^{\infty} i \cdot x^{i-1} = \frac{1}{(1-x)^2}.$$

Finally, substituting $1 - p$ for x and multiplying both sides by p yields

$$E[X] = \sum_{i=1}^{\infty} i \cdot (1-p)^{i-1} p = \frac{p}{(1-(1-p))^2} = \frac{p}{p^2} = \frac{1}{p}. \quad \square$$

In the case of `Randomized Find Statistic` and `Randomized Quicksort`, the number of steps $t(x)$ that is required of the algorithm on some input x is now itself a random variable. For example, in the case of `Quicksort`, x takes the form of an array and the sorting of x will depend on the quality of the pivots that are randomly generated during the algorithm's execution. Moreover, the number of steps can vary anywhere from $\Omega(n \log n)$ to $\mathrm{O}(n^2)$, and so the average lies somewhere in between those extremes.

Definition 1.11. Given algorithm $\mathcal{A}$, its **expected number of steps** for an input of size n, denoted $ET_{\mathcal{A}}(n)$ (or $ET(n)$ when $\mathcal{A}$ is understood) is defined as

$$ET(n) = \frac{1}{2^n} \sum_{|x|=n} t(x).$$

Now, for the `Randomized Quicksort` algorithm and any two length-n arrays a and b, notice that $t(a) = \Theta(t(b))$. This is because `Quicksort`'s running time depends only on the pivots that are selected throughout the algorithm, and these pivots are independent of the input array assuming a fixed array length n.

The same can be said for `Randomized Find Statistic`, although the k input may change the proportionality constant C that is hidden in the big-O average number of steps. This is not an issue since all of the results in this lecture are expressed in big-O notation.

1.1 Analysis of `Randomized Find-Statistic` Algorithm

We first analyze the expected runnning time of the `Randomized Find-Statistic` algorithm and make use of the linearity of expectation.

1. Let $E[Y]$ denote the expected time it takes to narrow the search to an array of size $\leq \frac{3n}{4}$.
2. Let $E[Z]$ denote the expected time it takes to locate the desired statistic in an array of size at most $\frac{3n}{4}$. In other words,
$$E[Z] = ET(\frac{3n}{4}).$$
3. Then by the linearity of expectation we have
$$ET(n) = E[Z] + E[Y] = ET(\frac{3n}{4}) + E[Y].$$
4. To bound $E[Y]$, notice that the worst case occurs when the desired statistic lies in the middle at $k = \lfloor n/2 \rfloor$. Then to reduce the array length by at least 25%, the pivot's order (from 1 to n in terms of its relative size in the array) must lie in the interval $[n/4, 3n/4]$ which will happen with probability $1/2$.
5. Moreover, assuming the random pivot selections are independent, then the number of required selections follows a geometric distribution with $p \geq \frac{1}{2}$, and so
$$E[Y] \leq 2 \cdot \mathrm{O}(n) = \mathrm{O}(n),$$
since each pivot selection requires use of the `Partitioning` algorithm which requires $\mathrm{O}(n)$ steps.
6. Therefore, $ET(n)$ satisfies
$$ET(n) \leq ET(3n/4) + \mathrm{O}(n)$$
which by Case 1 of the Master Theorem implies that $ET(n) = \mathrm{O}(n)$. □

Example 1.12. Suppose a is an array of integers with size equal to 124 and suppose $k = 88$. Provide a good upper bound on the expected number of rounds (i.e. pivot selections) that will be required in order for the scope of the search to be reduced to a subarray of size $\frac{3}{4} \cdot (124) = 93$.

Solution. Let i denote the order ($0 \leq i \leq 123$) of the randomly and uniformly selected pivot.

Case 1. $i = 88$. The algorithm terminates and the 88th least element of a is returned.

Case 2. $i > k = 88$. Then k would be be in the left array whose size is at most 93 iff $i \in \{89, \ldots, 92\}$.

Case 3. $i < 88$. In this case, k would be be in the right array whose size is at most 93 iff

$$123 - M + 1 \leq 93 \iff i \geq 31 \iff i \in \{31, \ldots, 87\}.$$

Therefore, there are $1 + 4 + 57 = 62$ different pivots that will reduce a to a length of 93 or less, and so the probability of selecting one of them equals $62/124 = 0.5$, and so the expected number of pivot selections before the array is reduced by 25% is at most 2. □

1.2 Analysis of `Randomized Quicksort` Algorithm

To analyze `Randomized Quicksort`, we make use of conditional probability and conditional expectation.

Definition 1.13. Given discrete random variables X and Y the **conditional probability** that $X = x_i$ on condition that $Y = y_j$, denoted $P(X = x_i|Y = y_j)$ is defined as

$$P(X = x_i \mid Y = y_j) = \frac{P(X = x_i \wedge Y = y_j)}{P(Y = y_j)}.$$

In words $P(X = x_i \mid Y = y_j)$ equals the probability that X will assume the value x_i on condition that Y assumes the value y_j. Thus, we are assuming $P(Y = y_j) > 0$.

Notice that, in case the events $X = x_i$ and $Y = y_j$ are independent events, then

$$P(X = x_i \mid Y = y_j) = P(X = x_i)$$

as one would hope, since knowing the outcome of Y should have no effect on the outcome of X.

Example 1.14. Suppose Let $X_i \sim \text{Be}(0.5)$, $i = 1, \ldots, 5$, are five independent Bernoulli random variables, each with $p = 0.5$. Let Bernoulli random variables `Odd` and `Three` be indicators of whether or not i) the binary string $S = X_1 \cdots X_5$ has an odd number of 1's and ii) S is the binary representation of a number that is divisible by 3, respectively. Compute

$$P(\texttt{Odd} \mid \texttt{Three}).$$

Solution. We have

$$P(\texttt{Odd} \mid \texttt{Three}) = \frac{P(\texttt{Odd} \wedge \texttt{Three})}{P(\texttt{Three})}.$$

It turns out that only one binary string of length 5 has an odd number of 1's *and* forms a number divisible by 3, namely

$$(21)_2 = 10101.$$

Thus, $P(\texttt{Odd} \wedge \texttt{Three}) = 1/32$. Therefore, since $P(\texttt{Three}) = 11/32$, we have

$$P(\texttt{Odd} \mid \texttt{Three}) = (1/32)/(11/32) = 1/11.$$

We may also condition the expectation of some random variable X with respect to the outcome of some other random variable Y.

Definition 1.15. Let X be a finite random variable whose domain is $\{x_1, \ldots, x_m\}$ and let Y be a random variable whose domain is $\{y_1, \ldots, y_n\}$. Let $p(i|j)$ denote the probability $P(X = x_i \mid Y = y_j)$. Then the **conditional expectation** of X with respect to event that $Y = y_j$ is defined as

$$E[X|Y = y_j] = \sum_{i=1}^{n} x_i \cdot p(i|j).$$

Example 1.16. Let R, B, and S be the random variables defined in Example 1.5. Suppose that the red die is rolled and it shows the value 2. Determine the average value of S conditioned on this event.

Solution.

$$E[S|R = 2] = 2 \cdot P(S = 2|R = 2) + 3 \cdot P(S = 3|R = 2) + \cdots + 8 \cdot P(S = 8|R = 2) + 9 \cdot P(S = 9|R = 2) + \cdots$$

$$+12 \cdot P(S = 12|R = 2) =$$

$$0 + 3 \cdot \frac{1}{6} + \cdots + 8 \cdot \frac{1}{6} + 0 + \cdots + 0 = 33/6 = 5.5. \quad \square$$

Notice in the above example that

$$33/6 = (21 + 2 \cdot 6)/6 = 3.5 + 2 = 5.5.$$

In words, the average value of the sum of the two dice is just the average of average of B added to 2. In general, one can show that $E[S|R = r] = 3.5 + r$.

A fruitful way of viewing $E[X|Y = y_j]$ is to think of it as the outcome of some random variable that assumes the value $E[X|Y = y_j]$ once Y assumes the value y_j. We call this random variable the **conditional expectation random variable** of X with respect to Y, and denote it as $E[X|Y]$. Moreover we have

$$\mathrm{dom}(E[X|Y]) = \{E[X|Y = y_1], \ldots, E[X|Y = y_n]\},$$

and its probability distribution is the same as for Y, say $(p_1, \ldots, p_n)$. In other words, with probability p_j,

1. Y assumes the value y_j, and
2. in turn $E[X|Y]$ assumes the value $E[X|Y = y_j]$.

We leave the proof of the following proposition as an exercise. It makes use of the following formula, sometimes called the **chain rule**, which is just a re-statment of conditional probability:

$$P(X = x_i \wedge Y = y_j) = P(X = x_i \mid Y = y_j) \cdot P(Y = y_j).$$

We may write the above more succinctly as

$$p(i, j) = p(i|j)p(j),$$

where, e.g., $p(j) = p_j = P(Y = y_j)$.

Proposition 1.17. Given random variables X and Y,

$$E[E[X|Y]] = E[X].$$

Example 1.18. Let R, B, and S be the random variables defined in Example 1.5. Then based on the discussion below Example 1.16, we have

$$E[S|R] = E[B] + R = 3.5 + R.$$

Therefore, by linearity of expectation and Proposition 1.17,

$$E[S] = E[E[S|R]] = E[3.5] + E[R] = 3.5 + 3.5 = 7. \quad \square$$

We now apply Proposition 1.17 to proving an upper bound on the expected running time of `Randomized Quicksort`. Again we assume an array input of size n that is an array a of distinct numbers.

1. Random variable X denotes the running time of `Randomized Quicksort` on some input a of size n.

2. Random variable Y has domain $\{1, \ldots, n\}$, where a sample $i \in \{1, \ldots, n\}$ represents the order of the pivot that is randomly selected at the top level of the recursion tree.

3. Then, for all $i = 1, \ldots, n$, by linearity of expectation we have

$$E[T(n)|Y = i] = E[X|Y = i] = ET(i-1) + ET(n-i) + \mathrm{O}(n).$$

   In words, after the i th least element in a is selected as pivot, the Partitioning algorithm requires $\mathrm{O}(n)$ steps, followed by two applications of `Randomized Quicksort`: once on a_{left} whose size equals $i-1$, and once on a_{right} whose size equals $n-i$, each having respective expected running times $ET(i-1)$ and $ET(n-i)$.

4. Thus, by Proposition 1.17, and the fact that each i has probability $1/n$ of being chosen, we have

$$ET(n) = E[E[T(n)|Y]] = \frac{1}{n}\sum_{i=1}^{n}(ET(i-1) + ET(n-i) + \mathrm{O}(n)) = \frac{2}{n}\sum_{i=1}^{n} ET(i-1) + \mathrm{O}(n) =$$

$$\frac{2}{n}\sum_{i=0}^{n-1} ET(i) + \mathrm{O}(n).$$

5. We leave it as an exercise to show that this recurrence induces $ET(n)$ to grow as $\mathrm{O}(n \log n)$.

2 Probability Spaces

A more general way of developing probability is to define what is called a **probability space**

Definition 2.1. A **probability space** is a triple $(\mathcal{S}, \mathcal{E}, P)$, where $\mathcal{S}$, $\mathcal{E}$, and P, are defined as follows.

1. $\mathcal{S}$ is a set of elements, called the **sample space**.
2. $\mathcal{E}$ is called the **event space** and is a collection of subsets of $\mathcal{S}$ that includes S, $\emptyset$, and is closed under countable unions, countable intersections, and complement. Each subset $E \in \mathcal{E}$ is called an **event**.
3. $P : \mathcal{E} \to [0, 1]$ is called a **probability measure** that assigns a number between 0 and 1 to each event, and satisfies the following axioms.
   (a) $P(\mathcal{S}) = 1$ and
   (b) If $A_1, A_2, \ldots$ is an infinite sequence of disjoint events, then
   $$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i).$$

Proposition 2.2. Let $(S, \mathcal{E}, P)$ be a probability measure space. Then the following statements hold for arbitrary $A, B \in \mathcal{E}$.

1. $P(\emptyset) = 0$
2. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
3. $P(\overline{A}) = 1 - P(A)$

Proof. We prove the first two statements and leave the third as an exercise. By Axiom 2, $P(A) = P(A \cup \emptyset) = P(A) + P(\emptyset)$ which implies $P(\emptyset) = 0$.

For proving the second statement, first it is an exercise to show that a σ-algebra is closed under set difference. In other words, if $A, B \in \mathcal{E}$, then $A - B \in \mathcal{E}$. Then by Axiom 2 we have $P(A) = P(A - B) + P(A \cap B)$, which implies $P(A - B) = P(A) - P(A \cap B)$. Similarly, $P(B - A) = P(B) - P(A \cap B)$. Also by Axiom 2,

$$P(A \cup B) = P(A - B) + P(B - A) + P(A \cap B) = (P(A) - P(A \cap B)) + (P(B) - P(A \cap B)) + P(A \cap B) =$$
$$P(A) + P(B) - P(A \cap B).$$

Example 2.3. Given a random variable X for which $\text{dom}(X) = \{x_1, \ldots, x_n\}$, and whose probability distribution is $(p_1, \ldots, p_n)$, we may define the probability space $(\mathcal{S}, \mathcal{E}, P)$ with respect to D, where

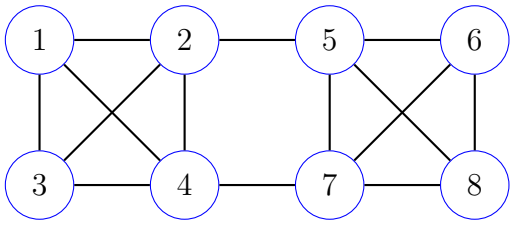
1. $\mathcal{S} = \{x_1, \ldots, x_n\}$
2. $\mathcal{E} = \mathcal{P}(\mathcal{S})$, the power set of $\mathcal{S}$, i.e. the set of all subsets of $\mathcal{S}$
3. $P : \mathcal{E} \to [0, 1]$ is defined so that
$$P(E) = \sum_{x_i \in E} p_i$$
is the sum of the probabilities that are associated with the members of E.

   As an example, suppose D is a random variable that represents the outcome of rolling a fair six-sided die. Then an event of the probability space associated with D is a subset of $\{1, 2, \ldots, 6\}$. For example, the event E of rolling an even number corresponds with the subset $E = \{2, 4, 6\}$ and $P(E) = 3(1/6) = 1/2$. □

3 Karger's Randomized Min-Cut Algorithm

Definition 3.1. Given a connected simple graph $G = (V, E)$, a **cut set** $C \subseteq E$ for G is a subset of edges of G whose removal from G disconnects G in the sense that there are at least two vertices $u, v \in V$ for which there is no longer a path beetween u and v. Moreover, C is said to be a **minimum cut set** iff it is a cut set of G and has the least cardinality of any cut set for G.

Example 3.2. The following graph has a unique minimum cut set equal to $\{(2, 5), (4, 7)\}$.



Idea Behind Karger's Algorithm for Finding a Minimum Cut

1. Think of each vertex $v \in V$ as a communication station and each edge $e = (u, v) \in E$ as a communication channel between two stations $u, v \in V$.

2. Allow for multigraphs, i.e. the possibility of multiple communication channels between two stations.

3. A **minimum cut communication failure (mccf) event** occurs when a minimum number of communication channels fail, resulting in the loss of communication between two or more stations, meaning there is no working communication path between some stations.

4. Greedy heuristic for estimating a minimum cut set of channels that would cause an mccf event: repeat the following until only two stations remain.

   (a) Choose two stations u and v that have the most communication channels between them.

   (b) Because of having the most channels between them, guess that an mccf event will *not* cut off communication between u and v.

   (c) Because of the fail-safe assumption about u and v, we may view them as a single communication entity and merge them into the single **compound station** uv, and eliminate any channels between them.

   (d) Replace any channel of the form (w, u) or (w, v), with the new channel (w, uv). In general if there are $a \geq 0$ (w, u)-channels and $b \geq 0$ (w, v)-channels, then replace them with $a + b$ (w, uv)-channels.

5. What remains are two stations. Estimate that the channels between them form a minimum cut set.

6. It can be shown that the above algorithm does *not* always return a minimum cut set.

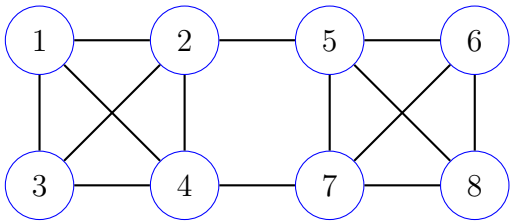
7. Randomized Version (Karger): instead of deterministically choosing a pair of stations to merge in each iteration, we instead randomly select a pair by first assigning each pair (u, v) a probability that is proportional to the number of channels between u and v. Note: equivalently, we may randomly choose a pair of stations by randomly choosing one of the edges via a uniform distribution.

Example 3.3. Demonstrate Karger's algorithm on the following graph and assuming the sequence of i) random station-pair selections

$$(2,3),(6,7),(5,8),(58,67),(23,5678),(1,235678),$$

and ii) random station-pair selections

$$(2,3),(6,7),(5,8),(58,67),(1,23),(123,4).$$



Solution.

The following theorem can be proved via induction.

Theorem 3.4. Chain Rule of Probability.

$$P(E_1, \ldots, E_n) = P(E_1)P(E_2|E_1)P(E_3|E_2, E_1) \cdots P(E_n|E_1, E_2, \ldots, E_{n-1}).$$

Lemma 3.5. When performing Karger's algorithm, suppose two vertices u and v are to be merged, and $\deg(u), \deg(v) \geq k$, then $\deg(uv) \geq k$.

Proof. Suppose u and v share $j \leq k$ edges. Then, upon merging u and v, we have

$$\deg(uv) = \deg(u) + \deg(v) - j = \deg(u) + (\deg(v) - j) \geq \deg(u) \geq k$$

since $j \leq k \leq \deg(v)$. □

Theorem 3.6. Given a multigraph G with n vertices, With probability at least $\frac{2}{n(n-1)}$, Karger's randomized algorithm results in a minimum cut set for G.

Proof. Let C be some minimum cut set of edges in G and having size k. Then once all edges of C are removed from G, the vertices of G are then partitioned into two sets: S and $V - S$ in such a way that there is no edge that is incident with both a vertex in S and in $V - S$. The goal is to prove that, after $n-2$ rounds of selecting pairs of stations to merge, all such pairs are either selected from S, or from $V - S$ with probability at least $2/(n(n-1))$. This would imply that the final two compound vertices consist of S and $V - S$, with the edges between them being equal to C, and hence a minimum cut has been successfully identified.

1. E_i, $i = 1, \ldots, n-2$, occurs provided the vertices merged in round i do not have a cut edge between them.
2. F_i, $i = 1, \ldots, n-2$ occurs provided $\bigcap_{j=1}^{i} E_i$ occurs, meaning that $E_1, E_2, \ldots, E_i$ all occur, i.e. none of the vertices merged in rounds 1 through i had a cut edge between them.
3. **Goal:** compute $P(F_{n-2})$.
4. **Observation:** $\deg(v) \geq k$ for all $v \in V$ (why?). Hence, G has at least $nk/2$ edges.
5. $P(E_1) = P(F_1) =$
$$1 - k/(nk/2) = 1 - \frac{2k}{nk} = 1 - \frac{2}{n} = \frac{n-2}{n}.$$
6. **General Observation:** If F_{i-1} occurs, then, going into round i, by Lemma 3.5, each vertex has degree at least k, none of the k min-cut edges have been selected, the graph has $n - i + 1$ vertices, and hence there are at least $(n - i + 1)k/2$ edges.
7. $P(E_i|F_{i-1}) =$
$$1 - k/((n-i+1)k/2) = 1 - \frac{2k}{(n-i+1)k} = 1 - \frac{2}{n-i+1} = \frac{n-i-1}{n-i+1}.$$
8. Finally, applying the chain rule of probability yields
$$P(F_{n-2}) = P(E_1, E_2, E_3, E_4, \ldots, E_{n-2}) =$$
$$P(E_1)P(E_2|E_1)P(E_3|E_1, E_2)P(E_4|E_1, E_2, E_3)\cdots$$
$$P(E_{n-3}|E_1, E_2, E_3, E_4, \ldots, E_{n-4})P(E_{n-2}|E_1, E_2, E_3, E_4, \ldots, E_{n-3}) =$$
$$P(E_1)P(E_2|F_1)P(E_3|F_2)P(E_4|F_3)\cdots P(E_{n-2}|F_{n-3}) =$$
$$\left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\left(\frac{n-5}{n-3}\right)\cdots\left(\frac{2}{4}\right)\left(\frac{1}{3}\right) =$$
$$\frac{2}{n(n-1)},$$
since all factors cancel except for the first two denominators and the last two numerators. □

3.1 If at First You Don't Succeed, Try Try Again!

To improve the chances of Karger's algorithm returning an actual minimum cut set for some graph G, we may make a number of independent runs of the algorithm and keep track of the smallest cut set that is returned by any of the runs. The following proposition tells us how many runs are needed for the probability of finding the smallest cut set is approximately $1/e$. It can be proved using elementary mathematical analysis which makes use of L'Hospital's rule and limit theorems involving continuous functions.

Proposition 3.7. Let p_n denote the probability that some experiment is deemed a success, where $p_n \to 0$ as $n \to \infty$. Then if $\lceil 1/p_n \rceil$ independent trials of the experiment are performed, the probability that none of them will succeed converges to $1/e$ as $n \to \infty$.

Note that the convergence to $1/e$ happens rather rapidly. For example, if $p = 1/100$, then

$$(1 - 1/100)^{100} = (0.99)^{100} \approx 0.3660,$$

whereas $1/e \approx 0.3679$.

Thus, by performing

$$\frac{n(n-1)}{2}(\ln n^2) = n(n-1)\ln n$$

indpendepent runs, the probability of not finding a min cut set is less than $1/n^2$, giving an increasing degree of confidence that a minimum cut set has been found.

Note that the Stoer-Wagner algorithm has the best known running time for a deterministic algorithm that solves the min-cut optimization problem. It's running time is $O(mn + n^2 \log n)$ and works on graphs with non-negatively weighted edges.

3.2 A Monte Carlo Bootstrapping Method for Limiting Trials of Karger's Algorithm

As was demonstrated in Example 3.3, if the minimum cut size k is known, then Karger's algorithm not only produces an approximate minimum cut C', but it also yields an estimate of the probability that C' is in fact a minimum cut. Moreover, it can be shown that if we perform t independent trials of Karger's algorithm, and obtain probability estimates $\hat{p}_1, \ldots, \hat{p}_t$, then

$$\hat{p} = \frac{\hat{p}_1 + \cdots + \hat{p}_t}{t} \to p$$

as t increases indefinitely, where p is the true probability that Karger's algorithm returns a minimum cut. In other words, the average of the estimates converges to the true probability p and we may use statistics (i.e. sample mean, standard deviation, and relative error) to estimate how close $\hat{p}$ is to p. Finally, once we are satisfied with $\hat{p}$'s degree of accuracy, we may then perform $c/\hat{p}$ additional trials in order to guarantee that a minimum cut will be optained with a probability that is approximately $1 - e^{-c}$, where c is some constant that depends on our desired degree of cut-size accuracy. This is an example of what is called a **Monte Carlo simulation**, i.e. repeating the same random experiment a number of times in order to estimate it's true likelihood of success.

Now suppose, as is usually the case, we do not know the value of k. Then we may use a **bootstrapping** procedure that begins by running Karger's algorithm a handful of times and choosing k_0 to be the size of the smallest cut found in any of the trials. Then we may repeat the Monte Carlo procedure described above using k_0 in place of k. Notice that, since $k_0 \geq k$, the probability estimates $\hat{p}_1, \ldots, \hat{p}_t$ will average to a probability $\hat{p}' \leq \hat{p}$. This is because in each round of Karger's algorithm the probability of avoiding $k_0 \geq k$ edges is less than or equal to that of avoiding only k edges. Of course, if during the Monte-Carlo simulation a better cut size $k_1 < k_0$ is found, then the MC simulation can be restarted using k_1 in place of k_0. Therefore, we obtain a nested iterative approach in which the outer loop keeps track of the least cut size found, while the inner loop performs the MC simulation for the current k value.

We call it "bootstrapping" because we start with almost zero knowledge about the min-cut size, and for each new cut-size upper bound k' that is obtain via Karger's algorithm, the improved bound leads to an improved Monte Carlo simulation because the computed $\hat{p}_i$ probabilities become increasingly more accurate.

4 Verifying Matrix Multiplication

Definition 4.1. An instance of the `Verifty Matrix Multiplication (VMM)` decision problem consists of three $n \times n$ 0-1 matrices A, B, and C, and the problem is to decide if $AB = C$.

Of course, Strassen's algorithm can decide this problem in $O(n^{\log 7})$. We now provide a randomized algorithm that requires $O(n^2)$ steps and, in case $AB \neq C$, identifies the inequality with probability at least 0.5. Thus the algorithm does not produce any false negatives, but it may produce a false positive with probability at most 0.5.

Randomized Algorithm $\mathcal{A}$

1. Let $D = AB - C$.
2. Randomly and uniformly select a 0-1 vector $\vec{x}$ from $\{0,1\}^n$.
3. Evaluate $B\vec{x}$, $A(B\vec{x})$, and $C\vec{x}$. //Note: all additons/subtractions are mod 2.
4. Evaluate $\vec{y} = A(B\vec{x}) - C\vec{x} = D\vec{x}$.
5. Return $(\vec{y} = \vec{0})$. //Return 1 iff $D\vec{x} = \vec{0}$.

Proof that $\mathcal{A}$ returns a false positive with probability at most 0.5

1. Assume $D \neq 0$.
2. Without loss of generality (WLOG), assume $d_{11} \neq 0$, where d_{11} is entry $(1,1)$ of D.
3. Let $\vec{x} = (x_1, x_2, \ldots, x_n)$ be the randomly selected vector.
4. Then the first component of $D\vec{x}$ equals
$$x_1 \oplus \sum_{i=2}^{n} d_{1i}x_i.$$
5. Now imagine that we are obtaining $\vec{x}$ by randomly sampling from $\{0,1\}^n$ using n independent Bernoulli variables in the order $X_n, \ldots, X_2, X_1 \sim \text{Be}(0.5)$ to obtain the components of $\vec{x}$, namely $x_n, \ldots, x_2, x_1$, in that order.
6. Then before we sample X_1, we will already know the value of $\sum_{i=2}^{n} d_{1i}x_i$, and at that point we see that, by the independence of the Bernoulli variables, the first component of $D\vec{x}$ will be nonzero with probability at least 0.5 which is the probability of returning a true negative.
7. Furthermore, by running this algorithm, say, 1000 times, the probability of a false positive is gets reduced to
$$\leq \left(\frac{1}{2}\right)^{1000}. \quad \square$$

5 The WalkSAT Algorithm

Recall the two logic decision problems `2SAT` and `3SAT`. The generalization of these problems is the k-`SAT` decision problem, $k \geq 2$, where each clause of a k-`SAT` instance $\mathcal{C}$ has k literals. `WalkSAT` is a randomized algorithm for solving k-`SAT`. Let n denote the number of variables of k-`SAT` instance $\mathcal{C}$. Let $r(n)$ be some efficiently computable integer function.

`WalkSAT`.

1. Create an initial assignment α over the variables of $\mathcal{C}$ by sampling from n independent binary random variables with $p = 0.5$.
2. If α satisfies $\mathcal{C}$, then return 1. //Found a satisfying assignment.
3. count $\leftarrow 1$.
4. While count $< r(n)$,
   (a) Randomly and uniformly select a clause $c \in \mathcal{C}$ unsatisfied by α.
   (b) Randomly and uniformly select a literal l of c.
   (c) $\alpha \leftarrow \overline{\alpha(l)}$. //Flip the value of α at literal l.
   (d) If α satisfies $\mathcal{C}$, then return 1. //Found a satisfying assignment.
   (e) count = count + 1.
5. Return 0. //Failed to find a satisfying assignment.

Theorem 5.1. If $k \geq 3$ and $\mathcal{C}$ is satisfiable, then when $r(n) = 3n$, `WalkSAT` returns 1 with probability approximately $(\frac{k}{2(k-1)})^n$.

5.1 Preliminary results

Proposition 5.2. (**Law of Total Probability**) Let A be an event of some probability space $(\mathcal{S}, \mathcal{E}, P)$ and suppose $E_1, E_2, \ldots, E_n$ are pairwise disjoint events for which

$$\mathcal{S} = E_1 \cup \cdots \cup E_n.$$

Then

$$P(A) = P(A|E_i)P(E_1) + \cdots + P(A|E_n)P(E_n).$$

Example 5.3. Suppose a bag has three coins: two fair coins and a two-headed coin. Suppose one of the coins is randomly and uniformly selected, tossed in the air, and lands "heads". Determine the probability that the two-headed coin was selected.

Solution.

Definition 5.4. A **binomial random variable** $X \sim B(n,p)$ is one for which $\text{dom}(X) = \{1,\ldots,n\}$ and for which

$$P(X=k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k}.$$

A binomial random variable $X \sim B(n,p)$ is essentially the sum of n independent Bernoulli random variables $X_i \sim Be(p)$, $i = 1,\ldots,n$:

$$X = X_1 + \cdots + X_n.$$

To see this, notice that, for given $0 \leq k \leq n$, there are $\left(\begin{array}{c} n \\ k \end{array} \right)$ different ways to choose exactly k of the Bernoulli variables to assume the value 1, and the remaining $n-k$ assume the value 0. Moreover, by independence, the probability of each such way is $p^k(1-p)^{n-k}$. Therefore,

$$P(X=k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k}.$$

Example 5.5. If a fair coin is tossed independently five times, what is the probability that it lands heads exactly three times?

Solution.

Theorem 5.6. (**Binomial Theorem**) For any two real numbers x and y,

$$(x+y)^n = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) x^k y^{n-k}.$$

Definition 5.7. The **Hamming distance** between any two binary words $u, v \in \{0,1\}^n$, denoted $d(u,v)$ is equal to the number of bit places i for which $u_i \neq v_i$.

Example 5.8. Let β be a fixed Boolean assignment of 10 variables and α be a random assignment of those same 10 variables. What is the probability that $d(\alpha, \beta) = 5$.

Solution.

Proof of Theorem 5.1.

1. Let α denote the current `WalkSAT` assignment, and β denote a fixed assignment that satisfies k-`SAT` instance $\mathcal{C}$.

2. Assuming $r(n) = \infty$, let E denote the event that $d(\alpha, \beta) = 0$ at some point during the `WalkSAT` algorithm. Note: $P(E)$ is less than or equal to the probability that `WalkSAT` terminates, since it's possible that α may assume a different satisfying assignment.

3. **Goal.** Calculate $P(E)$.

4. **Observation.** $d(\alpha, \beta)$ either increases by 1 or decreases by 1 after each `WalkSAT` iteration.

5. Let $p(x)$ denote the probability that E occurs when initially $d(\alpha, \beta) = x$, where $x \geq 0$. Note: by assuming that x can exceed n we can solve for $p(x)$ and the true value for $p(x)$ will be not worse than the obtained solution.

6. In a given `WalkSAT` iteration, let c denote the selected unsatisfied clause. Then in the worst case there is only one literal of c whose current assigned value can be flipped and result in a *decrease* of $d(\alpha, \beta)$ by 1, and flipping the current assigned value of any of the remaining $k - 1$ literals will result in an *increase* of by 1. Therefore, By the Law of Total Probability and the definition of $p(x)$, we have the recurrence

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

with boundary conditions $p(0) = 1$ and $p(\infty) = 0$.

7. **Exercise.** Show that $p(x) = (k-1)^{-x}$ satisfies the above recurrence.

8. Let X be the random variable that assumes the initial Hamming distance

9. We now calculate $P(E)$ by conditioning on X:

$$P(E) = \sum_{i=0}^{n} P(E|X=i)P(X=i) = \sum_{i=0}^{n} \left(\frac{1}{k-1}\right)^i \binom{n}{i} \frac{1}{2^n} = \sum_{i=0}^{n} \binom{n}{i} \left(\frac{1}{2(k-1)}\right)^i \left(\frac{1}{2}\right)^{n-i} =$$

$$\left(\frac{1}{2} + \frac{1}{2(k-1)}\right)^n = \left(\frac{k}{2(k-1)}\right)^n.$$

10. Finally, with more work it can be shown that if `WalkSAT` is able to converge to β, then it should so within $3n$ steps with high probability. □

Corrolary 5.9. For $k = 3$ `WalkSAT` can be used to find a satisfying assignment for satisfiable $\mathcal{C}$ in $\Theta((4/3)^n n^2)$ number of steps with a probability that converges to 1.

6 Mathematical Optimization and a Randomized Approximation Algorithm for `Max Cut`

The field of **mathematical optimization** represent a branch of the theory of algorithms that is focused on solving optimization problems that generally occur within the domain of continuous spaces, including both finite and infinite dimensional vector spaces over the field of real numbers. As we'll see in this section mathematical-optimization algorithms can sometimes be applied to combinatorial optimization problems in order to provide either an exact or approximate solution.

Two of the most famous and useful mathematical optimization algorithms are for solving the following problems, respectively.

Linear Programming (LP) An instance of this problem consists of a linear real-valued function $f(x_1, \ldots, x_n)$ along with m linear constraints $c_1, \ldots, c_m$, where each constraint c_i has the form

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i,$$

where $a_{i1}, \ldots a_{in}, b_i \in \mathbb{R}$. The problem is to find a vector $(x_1^*, \ldots, x_n^*) \in \mathbb{R}^n$ which minimizes f subject to satisfying each of the m constraints.

Semi Definite Programming (SDP) An instance of this problem consists of a quadratic real-valued function of the form

$$f(\vec{x}_1, \ldots, \vec{x}_n) = \sum_{i,j=1}^{n} c_{ij}(\vec{x}_i \cdot \vec{x}_j),$$

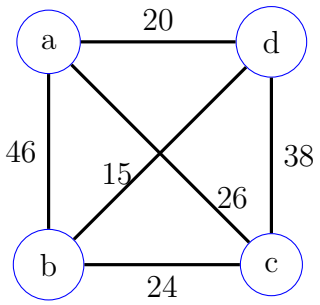
subject to the m constraints

$$\sum_{i,j=1}^{n} a_{ijk}(\vec{x}_i \cdot \vec{x}_j) \leq b_k,$$

for $k = 1, \ldots, m$.

For the purposes of this lecture, it is useful to know that there are polynomial-time algorithms for solving both problems, and that these algorithms have been optimized over the past several years to become indispensible tools for general problem solving. It's interesting to note that both `LP` and `SDP` are **P-complete** problems, meaning that any problem in `P` can be map reduced to either of the two problems, and where the map requires a $O(\log n)$ amount of memory to perform (Why is every problem in `P` trivially polynomial-time reducible to `LP` and `SDP`?).

Example 6.1. Recall that Christofides' approximation algorithm for `Triangle Inequality TSP` requires finding a minimum-cost matching for the odd-degree vertices of a minimum spanning tree that is a subgraph of some complete graph. Suppose the subgraph with those odd-degree vertices is shown below.



Then we may reduce the problem of finding a minimum-cost matching for these vertices to the following instance of `LP`. In this case we have six variables $x_{ab}, x_{ac}, x_{ad}, x_{bc}, x_{bd}$ and x_{cd}, one for each edge of the graph. We (linearly) constrain each variable:

$$0 \leq x_{ij} \leq 1,$$

for all $i, j \in \{a, b, c, d\}$ and $i \neq j$. Here, setting, e.g., $x_{ab} = 1$ implies that edge (a, b) will be part of the matching. Next, for each vertex $v \in \{a, b, c, d\}$, we need a constraint that implies that there is exactly one edge that is incident with v and is part of the matching. For example, in the case of $v = a$, we have the constraint

$$x_{ab} + x_{ac} + x_{ad} = 1.$$

Finally, the linear function to be minimized is

$$46x_{ab} + 26x_{ac} + 20x_{ad} + 24x_{bc} + 15x_{bd} + 38x_{cd}.$$

It can be shown that the optimal solution for this instance of `LP` will be such that every variable is assigned the value of either 0 or 1. In this case, $x_{ac} = x_{bd} = 1$, while the other four are assigned 0. □

Recall that an instance of the `Max Cut` optimization problem is a non-negatively weighted undirected graph $G = (V, E, w)$, and the problem is the find a subset $U \subseteq V$ for which

$$w(e_1) + w(e_2) + \cdots + w(e_r)$$

is a maximum, where $\{e_1, e_2, \ldots, e_r\}$ is the set of all edges e for which one vertex of e is a member of U, and the other vertex is a member of $E - U$. We leave it as an exercise to show that there is a greedy algorithm for finding a U that achieves an approximation ratio of at least 0.5. The algorithm begins by setting $U = \emptyset$ and in each step chooses a vertex to move to/from U based on the largest positive difference between the sum of its intra-connection weights and the sum of its inter-connection weights. For example, if $u, v \in U$, and $(u, v) \in E$, then $w(u, v)$ contributes to the intra-connection weight sum of both u and v. On the other hand, if $u \in U$ and $v \in V - U$, then $w(u, v)$ contributes to the inter-connection weight sum of both. The algorithm terminates when there is no vertex whose intra-connection weight sum exceeds its inter-connection weight sum. It can be shown that this algorithm does always work.

We now present the Goemans-Williamson randomized approximation algorithm that attains a 0.878 approximation ratio for `Max Cut`. The idea behind the algorithm is to map an instance of $G = (V, E, w)$ of `Max Cut` to an instance of `Semidefinite Programming` to where the optimal solution to the `SDP` problem instance implies an approximate solution to `Max Cut` having the desire 0.878 ratio.

1. Finding a maximum cut for G is equivalent to finding an assignment $s : V \to \{-1, 1\}$ that maximizes
$$W = \sum_{i<j} w_{ij} \frac{1 - s_i s_j}{2},$$
where, e.g., s_i is used to denote $s(i)$, and $w_{ij} = 0$ if $(i, j) \notin E$.

2. The above can be rewritten as
$$W = \frac{1}{2} \left(\sum_{i<j} w_{ij} - E \right),$$
where
$$E = \sum_{i<j} w_{ij} s_i s_j.$$

3. Notice that maximizing W is equivalent to minimizing E.

4. **Problem Relaxation.** The search space is $\{-1, 1\}^n$, where $n = |V|$. Thinking of -1 and 1 as one-dimensional unit vectors, we expand the search space to all n-dimensional unit vectors to obtain the new objective function
$$E' = \sum_{i<j} w_{ij} (\vec{s}_i \cdot \vec{s}_j),$$
subject to the constraint that $|\vec{s}_i| = 1$ for all $i = 1, 2, \ldots n$. We leave it as an exercise to prove that objective E' along with the unit-length constraints represent an instance of `SDP`.

5. **Mapping Back to 1-D Space.** Notice that, since $\mathbb{R}^1 \subset \mathbb{R}^n$, we have $E_{\text{opt}} \leq E'_{\text{opt}}$. Thus, ideally, we would define assignment s in such a way that the sign of $s_i s_j$ would equal the sign of $\vec{s}_i \cdot \vec{s}_j$. However, finding such an assignment may become very complex and/or may not be possible. So, instead we randomly generate a unit vector $\vec{r} \in \mathbb{R}^n$, and let $\vec{r}$ represent the normal vector of a plane $P_{\vec{r}}$ in $\mathbb{R}^n$. Then
   (a) set $s_i s_j = 1$ in case both $\vec{s}_i$ and $\vec{s}_j$ are on the same side of P, i.e. $(\vec{s}_i \cdot \vec{r})(\vec{s}_j \cdot \vec{r}) > 0$
   (b) set $s_i s_j = -1$ in case both $\vec{s}_i$ and $\vec{s}_j$ are on opposite sides of P, i.e. $(\vec{s}_i \cdot \vec{r})(\vec{s}_j \cdot \vec{r}) < 0$

6. Given an arbitrary pair $i < j$, consider the 2-dimensional plane P_{ij} spanned by $\vec{s}_i$ and $\vec{s}_j$. We may represent both $\vec{r}$ and $P_{\vec{r}}$ using an orthonormal basis for which two of the basis vectors $\vec{e}_1$ and $\vec{e}_2$ span P_{ij}. Then, when projecting both $\vec{r}$ and $P_{\vec{r}}$ on to P_{ij} the dot product of $\vec{r}$'s projection with both $\vec{s}_i$ and $\vec{s}_j$ will yield the same value that is obtained with $\vec{r}$. Moreover, $P_{\vec{r}}$ will project as a line in P_{ij} that is orthogonal to $\vec{r}$.

7. From the previous statement, we thus have Given an arbitrary pair $i < j$, we have
$$P(s_i \neq s_j) = P(\text{projection of } P_{\vec{r}} \text{ onto } P_{ij} \text{ lies between } \vec{s}_i \text{ and } \vec{s}_j) =$$
$$(\text{radian measure between } \vec{s}_i \text{ and } \vec{s}_j)/\pi = \theta_{ij}/\pi.$$

8. Note that $\vec{s}_i \cdot \vec{s}_j = \cos \theta_{ij}$. Thus, returning to the definition of W from Statement 1, we have
$$E[W_{\text{approx}}] = \sum_{i<j} w_{ij} \frac{\theta_{ij}}{\pi}.$$

while

$$W_{\text{opt}} \leq \sum_{i<j} w_{ij} \frac{1 - \cos\theta_{ij}}{2}$$

9. Therefore,

$$\frac{E[W_{\text{approx}}]}{W_{\text{opt}}} \geq \min_{i<j} \frac{\theta_{ij}/\pi}{(1 - \cos\theta_{ij})/2} = 0.878\ldots.$$

10. Why do we use $E[W_{\text{approx}}]$ instead of W_{approx}? Remember that θ_{ij}/π is a *probability*. Therefore, the quantity

$$\frac{1 - s_i s_j}{2}$$

equals 1 with probability θ_{ij}/π and equals 0 with probability $1 - \theta_{ij}/\pi$. Therefore, its expected value is θ_{ij}/π. Finally, by using the linearity of expecation, the expectation provided in statement 8.

11. Finally, by the Law of Large Numbers, the average of repeated trials converges to the average and so the average approximation ratio should be attained after a relatively small number of trials.

Exercises

1. Use the substitution method to show that $T(n) = \mathrm{O}(n \log n)$, where $T(n)$ satisfies the recurrence.

$$T(n) = \frac{2}{n} \sum_{i=0}^{n-1} T(i) + C'n$$

where $C' > 0$ is a constant. Hint: use the following result from mathematical analysis. If $f(x)$ is an increasing real-valued function on the interval $[a, b]$, then there exists a real number $\gamma \in [0, 1]$ for which

$$\sum_{i=a}^{b} f(i) = \int_a^b f(x)\ \mathrm{dx} + \gamma(f(b) - f(a)).$$

Conclude that randomized Quicksort has a long-linear expected running time.

2. An urn contains five red, three blue, and two orange balls. Two balls are randomly selected from the urn. Let X denote the number of blue balls selected. Provide the domain of X along with its probability distribution.

3. A fair coin is tossed independently $n \geq 1$ times. Letting H and T denote the number of heads and tails observed, provide the domain of the random variable $D = H - T$. Determine D's probability distribution in case that $n = 3$.

4. Team A is to play Team B in a sequence of games that together comprise a **match**. Since Team A proved to be the best team (i.e. most wins) in the league during the season, it has been rewarded so that it wins the match if it is the first team to win either two or three games. On the other hand, Team B only wins the match if it is the first team to win three games. Finally, before the match begins, the experts believe that both teams have a 50% chance of winning any of the upcoming games. Determine the probability that Team A will win the match as well as the expected number of games that will have been played when the match ends (regardless of who wins the match).

5. Suppose X and Y are geometric random variables, both with parameter p. For $n \geq 2$, $i = 1, \ldots, n-1$, compute

$$P(X = i | X + Y = n).$$

6. Given random variables X and Y, where $\mathrm{dom}(X) = \{x_1, \ldots, x_m\}$ and $\mathrm{dom}(Y) = \{y_1, \ldots, y_n\}$, the **joint probability mass function** for X and Y is the function

$$p : \mathrm{dom}(X) \times \mathrm{dom}(Y) \to [0, 1]$$

for which $p(x_i, y_j)$ denotes the probability of simultaneously observing $X = x_i$ and $Y = y_j$. Moreover,

$$\sum_{i=1}^{m} \sum_{j=1}^{n} p(x_i, y_j) = 1.$$

Notice that the probability distribution for either X or Y can be computed using p. For example,

$$p_i = \sum_{j=1}^{n} p(x_i, y_j).$$

Suppose that the joint distrubiton is given by the values

$$\begin{array}{lll} p(1,1) = \frac{1}{9} & p(1,2) = \frac{1}{9} & p(1,3) = 0 \\ p(2,1) = \frac{1}{3} & p(2,2) = 0 & p(2,3) = \frac{1}{6} \\ p(3,1) = \frac{1}{9} & p(3,2) = \frac{1}{18} & p(3,3) = \frac{1}{9} \end{array}$$

Use it to compute $E[X|Y = i]$ for $i = 1, 2, 3$.

7. For the joint distribution given in the previous exercise, are X and Y independent random variables. Prove or disprove.

8. An urn contains three white, six red, and five black balls. Six of the balls are randomly selected from the urn. Let W and B denote the number of white and black balls that were selected, respectively. Compute the probability distribution for W conditioned on the fact that $B = 3$ black balls were selected. Compute $E[W|B = 1]$.

9. Repeat the previous exercise, but now assume that, when a ball is selected, its color is noted and it is placed back in the urn (i.e. we assume that the balls are being selected one at a time with replacement).

Exercise Hints and Solutions

1. Inductive assumption: $T(k) \leq Ck \log k$ for all $k < n$ and some constant $C > 0$. Prove that $T(n) \leq Cn \log n$. From the provided hint and performing integration by parts on the integral $\int_0^{n-1} x \log x \, \mathrm{dx}$, we see that

$$T(n) = Cn(1 - \frac{1}{n}) \log(n-1) - \frac{C}{2 \ln 2} n(1 - \frac{1}{n}) + C'n + \gamma(1 - \frac{1}{n}) \log(n-1) \leq Cn \log n$$

so long as $C \geq 2 \ln 2(C' + \gamma)$, and n is sufficiently large.

2. $\mathrm{dom}(X) = \{0, 1, 2\}$. Also,

$$P(X = 0) = \frac{7}{10} \times \frac{6}{9} = \frac{7}{15},$$

$$P(X = 1) = \frac{3}{10} \times \frac{7}{9} + \frac{7}{10} \times \frac{3}{9} = \frac{7}{15},$$

and

$$P(X = 2) = \frac{3}{10} \times \frac{2}{9} = \frac{1}{15}.$$

Notice that the probabilities sum to 1.

3. $\mathrm{dom}(D) = \{0, \pm 2, \pm 4, \ldots, \pm n\}$ if n is even, and $\mathrm{dom}(D) = \{\pm 1, \pm 3, \ldots, \pm n\}$ if n is odd. For $n = 3$ we have $P(D = -3) = 1/8$, $P(D = -1) = 3/8$, $P(D = 1) = 3/8, P(D = 3) = 1/8$.