Probability & Computing

Lecture One

1 Introduction

An Example: Verifying polynomial multiplication

Is $(x^2 - x + 3)(x^3 + 4x^2 - x - 1) = x^5 + 3x^4 - x^3 + 12x^2 - 3x - 3$?

Given three polynomials $f(x), g(x), h(x) \in \mathbb{Z}[x]$, how can we efficiently verify whether $f(x)g(x) = h(x)$? The direct method is to just multiply $f(x)$ and$g(x)$ in time $O(n \log n)$ (using FFT) and compare it with $h(x)$; this can be done in time $O(n \log n)$, where $n = deg(h)$.

Using randomness, we can verify this in $O(n)$ time, provided we allow for a small probability that our algorithm will output an incorrect answer.

Here's an algorithm: pick uniformly at random, an element $a \in \{1, 2, \ldots, 100n\}$; evaluate $f(a), g(a), h(a)$ in $O(n)$ time, and compare the product of the first two values with the third. If they match, we output TRUE, otherwise we output FALSE.

Notice that if it were indeed true that $f(x)g(x) = h(x)$, our values would match, and we shall output the correct answer (TRUE). On the other hand, it can happen that $f(x)g(x) \neq h(x)$, but $f(a)g(a) = h(a)$ for the value a that we picked. In this case, the correct answer is FALSE, but our output is TRUE.

What is the probability that we give this incorrect answer? To answer this, suppose that $f(x)g(x) \neq h(x)$, and let $S = \{a | f(a)g(a) = h(a)\}$. We note that the non-zero polynomial $f(x)g(x) - h(x)$ can have at most n roots, and therefore $|S| \leq n$. Now, the probability of our giving the incorrect answer is exactly equal to $\dfrac{|S \cap \{1, 2, \ldots, 100n\}|}{100n}$, which is at most $\dfrac{1}{100}$.

Thus, we have a (simple) randomized algorithm that verifies the polynomial-multiplication identity in $O(n)$ time, and which returns the correct answer with probability at least 0.99.

Exercise 1 *Substitute random values for the polynomials at the beginning of this section and compare. Are they equal?*

2 Monte Carlo and Las Vegas algorithms

Randomized algorithms often offer an advantage in speed or simplicity (or both). The random choices that the algorithm makes can result in one or both of the following:

(A) the algorithm does not always return the correct value: instead, it returns the correct value with some probability;

(B) the running time is a random variable.

Algorithms with property (A) are called Monte Carlo algorithms, and algorithms with property (B) are called Las Vegas algorithms. The first algorithm we saw in Section 1 is an example of a Monte Carlo algorithm. The most well-known Las Vegas algorithm is quick-sort which always returns the sorted sequence correctly, but its running time depends on the choice of random pivots, and is therefore a random variable.

3 Search-vs-decision

A decision problem is one whose answer is True/False, such as verifying a polynomial identity, or whether a CNF formula has a satisfying assignment.

A search problem, on the other hand, is about finding an object with some property, and in optimization problems, the property is to minimize or maximize some objective function.

For a Monte-Carlo algorithm for these two kinds of problems, what is a good probablity of correctness/success respectively? As the exercises below show, the answer isn't the same for both!

Exercise 2 *For a Monte-Carlo algorithm that solves a decision problem, what is a good value for the probability of correctness?*

Exercise 3 *For a Monte-Carlo algorithm that solves a search problem, what is a good value for the probability of correctness?*

The two exercises leave open-ended the meaning of "good", but let's say that 0.99 is certainly a good value. Let's also say that an algorithm is efficient if its worse-case running time is polynomial in the size of the input, and in both exercises, the algorithms are efficient.

4 Basic probability

Some exercises from class:

1. A number X is chosen uniformly at random from $\{1, 2, 3, \ldots, 100\}$. What is the probability that X is divisible by 2 or by 3?

Solution: Let A be the event that X is divisible by 2, B the event that X is divisible by 3. We are asking for $Pr[A \cup B] = Pr[A] + Pr[B] - Pr[A \cap B] = \frac{50}{100} + \frac{33}{100} - \frac{16}{100} = \frac{67}{100}$.

2. A fair coin is tossed 100 times. What is the probability that the tosses result in at least 70 heads?

   Solution: The direct expression is $\frac{1}{2^{100}} \left(\sum_{i=70}^{100} \binom{100}{i} \right)$. Note, however that from this expression we are unable to gauge how small or large the actual value is. As some-one in class pointed out, an upper bound is $\frac{30\binom{100}{70}}{2^{100}}$.

3. A pair of 6-sided dice is thrown. Given that the sum of the observed values is at least 10, what is the probability that the first value is even?

   Solution: The reduced sample space is $\{(4,6),(6,4),(5,5),(5,6),(6,5),(6,6)\}$ and the desired probability is $\frac{4}{6}$.

Abstract definition of probability:

Given any set Ω, the universe (or referred to as the sample space) or universse, a probability distribution on Ω is a function P from subsets of Ω to $[0,1]$, that satisfies the following properties:

(a) $P(\Omega) = 1$; (b) If $E_1, E_2, \ldots$ are mutually disjoint, then $P(E_1 \cup E_2 \ldots) = P(E_1) + P(E_2) + \ldots$.

An *event* is any subset of Ω.

Three further useful properties are:

1. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.
2. For an event E, we have $P(\bar{E}) = P(\Omega \setminus E) = 1 - P(E)$.
3. If A, B are two independent events, then $P(A \cap B) = P(A)P(B)$.

Conditional probability: If $A, B \subseteq \Omega$ are two events, the conditional probability $P(A|B)$ is obtained by restricting the sample space to B. Thus, we have:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

which is the definition of $P(A|B)$.

5 Sampling with a coin

Suppose that we have a fair coin as a source of randomness. We can use it to sample from the uniform distribution on $\{1, 2, \ldots, 8\}$ by tossing it three times, and assigning the values 1 through 8 to the eight possible outcomes. More generally, we can also sample from a set such as $\{1, 2, \ldots, n\}$ by tossing it k times for k such that $2^k \geq n$, and assigning n distinct outcomes the values 1 through n. If the outcome is not one of these n, we repeat the experiment.

In particular, we can also make choices with probability p when p is a rational number. We will also think of this as having a biased coin with probabilities p and $1 - p$ for the two sides.

Exercise 4 *Simulate a coin of any real bias p using a fair coin.*

Exercise 5 *Simulate a fair coin using a biased coin of unknown bias.*

6 Minimum cut

For an undirected graph $G = (V, E)$, a cut is a partition $(A, V \setminus A)$ of the vertices, and the set of edges $\{\{x, y\} : x \in A, y \in V \setminus A\}$ is called a cut-set, and sometimes the edge-set is also referred to as the cut, as we shall do. An example is shown in the slides.

A Greedy Algorithm that doesn't work:

We maintain sets of vertices, with each set initially consisting of singleton vertices, and repeatedly merge them, until we have exactly two sets remaining; we then output the set of edges across these two sets as the solution.

Which pair of sets should we merge at any time-step? When there are three sets remaining, we can see that we should merge the pair that has the maximum number of edges across it. For more sets, however, the greedy algorithm can give an incorrect answer.

Karger's Algorithm:

The algorithm of David Karger is similar to the greedy algorithm, in the sense that we start with singleton sets, and then repeatedly merge them. However, instead of doing this greedily, Karger's algorithm merges each pair of sets with probability proportional to the number of (multi)-edges between them.

An example of the merging process is shown in the slides, where we think of each set as a single (block) vertex, and have multi-edges between them. Instead of merge, we use the word contract.

Exercise 6 *For the example shown in the attached slides, calculate the probability of that particular sequence being taken by the algorithm.*

Exercise 7 *If the minimum cut size is k, show that the minimum degree of G is at least k, and that hence the number of edges is at least $nk/2$.*

We shall show in the second class that Karger's algorithm outputs a given minimum cut with probability at least $\frac{2}{n(n-1)}$. The probability of finding a minimum cut can be increased by repeating the algorithm several times, and taking the minimum cut among all the cuts that were output.

7 Probability Amplification

Let E be an experiment that we perform, and which has a probablity p of success. If we repeat the experiment k times independently, then the probability of failing every time is equal to $(1-k)^p$. In particular, if we repeat the experiment $\frac{c}{p}$ times, then the probability of failure is at most e^{-c} (where we used the fact that $e^x \geq 1 + x$). Taking a value of $c = 10$ for example, gives a probability of success more than 99.9%.