Advanced Algorithms Course. Lecture Notes. Part 7

Randomized Algorithms

Basics of Probability Theory

This section is not a full-fledged introduction to probability theory, but only a recap of the absolute minimum knowledge needed for a real understanding of randomized algorithms and their analysis.

The mathematical essence of the notion of probability can be described by **Kolmogorov's axioms**, without recurring to any interpretation of proabilities. We deal with a **probability space** which is a set Ω with a probability function. For simplicity we focus on discrete (finite or countably infinite) sets Ω, which is the most relevant case in algorithmic contexts. Subsets of Ω are called **events**. The probability $Pr(A)$ of an event A is a real number from the interval $[0, 1]$, and probabilities have to satisfy the following simple properties (these are Kolmogorov's axioms): $Pr(\emptyset) = 0$; $Pr(\Omega) = 1$; if $A \cap B = \emptyset$ then $Pr(A \cup B) = Pr(A) + Pr(B)$. The last property called additivity must also hold for countably infinite sets of disjoint events.

Single-element events $A = \{\omega\}$ are also called elementary events. We may simply write $Pr(\omega)$ instead of $Pr(\{\omega\})$.

From the axioms it follows immediately that $Pr(\Omega \setminus A) = 1 - Pr(A)$, and $Pr(A \cup B) \leq Pr(A) + Pr(B)$ for any events A and B. The latter inequality is so useful that it deserves an own name: we call it the **union bound**. One can use it to bound the probablity of a complicated event which is, however, the disjunction of simpler events with easily computable probabilities.

Sometimes we know already that some event B occurs, and we are interested in the probability of A, given this additional knowledge. This **conditional probability** is $Pr(A|B) := Pr(A \cap B)/Pr(B)$. Pronounce $Pr(A|B)$ as "probability of A given B" or "probability of A conditional on B".

We call an event A **independent** of an event B if $Pr(A|B) = Pr(A)$. In that case we obviously get $Pr(A \cap B) = Pr(A) \cdot Pr(B)$, hence the independence relation is symmetric, and we can simply say "A and B are independent". It is not always intuitively clear whether two events are independent. In such cases we have to calculate the relevant probabilities. Also, do not confuse independent and disjoint events ($A \cap B = \emptyset$) – these are totally different things!

Mutual independence of k events can also be characterized by the product formula: the probability of the intersection equals the product of the probabilities of the events; moreover, this equality must hold for each of the 2^k subsets of these k events.

Random Variables

A **random variable** is a function X from a probability space into, e.g., the set R of real numbers. (We only consider the case of real-valued X and discrete Ω.) Formally, we can write a random variable as $X : \Omega \longrightarrow R$.

Every possible value x of X gets a probability in an obvious way, by defining $Pr(X = x) = Pr(X(\omega) = x)$. The **distribution** of X is $Pr(X = x)$ viewed as a function of x. Note carefully that a random variable and its distribution are two different objects. Two random variables with equal distributions do not necessarily come from the same function on Ω. This distinction is important when we combine several random variables by algebraic operations (see below).

The **expected value** or **expectation** of a random variable X is defined as $E[X] := \sum_{\omega \in \Omega} Pr(\omega)X(\omega)$. Note that $E[X] = \sum_x Pr(X = x) \cdot x$, that is, the expectation depends only on the distribution of X. Intuitively, $E[X]$ is the long-term average of X when we observe the random variable many times independently.

A frequent misunderstanding is that $Pr(X > E[X]) = 1/2$. This is far from being true in general. For instance, let X be the random variable that describes a win in a lottery (where the stake is not considered in X). The expected win is some small positive amount, but the probability of winning anything is extremely small, certainly not $1/2$. A "probability-free" formulation of this insight is: The average of a set of values is in general distinct from its median!

Random variables X and Y on the same discrete probability space are **independent** if $Pr(X = x, Y = y) = Pr(X = x) \cdot Pr(Y = y)$ holds for all values x and y. In the same way as for random events we could instead

define independence by the property that knowing the value of X has no impact on the distribution of Y, and then this product rule comes out. Also, mutual independence of a set of random variables is similarly defined as for events.

Random variables, without loss of generality defined on the same probability space, can be combined by arbitrary algebraic operations: We simply apply the operation to their random values. For instance, the sum $X + Y$ of random variables X and Y is defined by $(X + Y)(\omega) := X(\omega) + Y(\omega)$. Similarly we can define the product of random variables, and so on.

A useful and powerful property is the **linearity of expectation**. It says that E is a linear operator, that means, $E[X + Y] = E[X] + E[Y]$. This holds for arbitrary random variables, not only for independent ones. The proof is a straighforward calculation in a few lines:

$$E[X+Y] = \sum_{\omega\in\Omega} Pr(\omega)(X+Y)(\omega) = \sum_{\omega\in\Omega} Pr(\omega)(X(\omega)+Y(\omega))$$

$$= \sum_{\omega\in\Omega} Pr(\omega)X(\omega) + \sum_{\omega\in\Omega} Pr(\omega)Y(\omega) = E[X] + E[Y].$$

A similar property for the product does not hold in general. We have $E[XY] = E[X]E[Y]$ in special cases only. The most important sufficient condition is that X and Y are independent. Again, the proof is a straighforward calculation, but this time it is easier to work on the range of values rather than on Ω. Observe carefully in which step independence is used:

$$E[XY] = \sum_{z} Pr(XY = z)z = \sum_{z} \sum_{x,y:\, xy=z} Pr(X = x, Y = y)xy$$

$$= \sum_{z} \sum_{x,y:\, xy=z} Pr(X = x)xPr(Y = y)y = \sum_{x,y} Pr(X = x)xPr(Y = y)y$$

$$= \sum_{x} Pr(X = x)x + \sum_{y} Pr(Y = y)y = E[X]E[Y].$$

Repeat Until Success

An important basic "algorithm" is to repeat a random experiment until it succeeds: Suppose that we have a $0, 1$-valued random variable that attains value 1 with probability p. We observe this variable many times independently and successively, until the result 1 appears for the first time. What is the expected number of iterations needed?

Intuitively one would expect $1/p$, but intuition is often misleading, therefore we'd better derive this result more rigorously. Although this is still a basic exercise, a strict formal treatment would already be a bit laborious, as our probability space is the Cartesian product of infinitely many copies of a probability space with two events. However, we will skip some technicalities and think in a semi-formal way: Let E_i be the event that the ith iteration is successful. Then $Pr(E_i) = (1-p)^{i-1}p$. Note that the first $i-1$ iterations have failed, and the probabilities can be multiplied because the trials are independent. Hence our expected value is

$$\sum_{i=1}^{\infty} Pr(E_i) \cdot i = \sum_{i=1}^{\infty} (1-p)^{i-1} pi.$$

Now some standard algebra (that we omit here) confirms the result $1/p$.

The World's Simplest(?) Randomized Algorithms

Suppose that some treasure is hidden behind either door A or door B. We wish to find the treasure, opening a minimum number of doors.

There exist only two deterministic "algorithms" for this toy problem:
(1) First open A, and if the treasure is not there, open B.
(2) First open B, and if the treasure is not there, open A.

Regardless which algorithm we choose, the treasure may be behind the door chosen last, hence any deterministic algorithm needs 2 steps in the worst case.

Now let us apply the following randomized algorithm instead:
First open A or B, both chosen with probability 1/2, and if the treasure is not there, open the other door.

Regardless of the location of the treasure, we find it with probability 1/2 in 1 step, and with probability 1/2 in 2 steps. This yields an expected number of only $0.5 \cdot 1 + 0.5 \cdot 2 = 1.5$ steps.

It might appear paradoxical that randomized rules can be more efficient than purposeful deterministic rules, but this simple problem shows some of the "magic" behind that: Imagine that some nasty adversary selects the input. Given any deterministic algorithm, the adversary may choose an input that makes our life as hard as possible (here: hide the treasure). The adversary loses power when faced with a randomized algorithm: Still the adversary knows the algorithm in advance, but not the random decisions that will be made. This makes it impossible to present an inconvenient input in the same way.

Next, note that we computed the expected time *for every fixed instance* and then took the *maximum over all instances*. (In this simple problem, the expected values were equal, but in general they may be different, and we may be interested in the worst case.) Thus, the worst-case expected time of randomized algorithms should not be confused with average-case analysis. All randomness is on the algorithm's side, while *nothing* is assumed here about the probability distribution of inputs. If not said otherwise, we will consider algorithms that are randomized in this sense, while inputs are not random.

This also illustrates that "randomness" and "arbitrariness" is not the same: In the scenario above we have never assumed that the treasure is behind door A or B with probability 1/2. Let us assume this now, for a moment. Then, even a deterministic algorithm succeeds in 1.5 expected steps, as opposed to the 2 steps in the worst case. That is, knowing a probability distribution on the inputs, even though it is the uniform distribution, would have been valuable information.

Rendezvous problems are similar to the scenario above: Imagine that two persons P and Q can move around, they want to meet as early as possible, but unfortunately, P and Q cannot communicate to each other before they meet. Perhaps they have not even agreed on a search strategy beforehand. Now, if P and Q apply deterministic strategies, it may happen that they are always at different locations and will never meet. If at least one of P or Q follows a certain randomized strategy, they will meet after some expected time. The details depend on the space and the possible moves. In the simplest case, imagine again two rooms A and B, and assume that time is divided in discrete slots. In every time slot, each person can freely choose a room. How should P and Q behave, in order to meet in the same room as soon as possible? (We leave it to you to think about it.) Unlike this minimalistic case, problems of this type can be mathematically very deep.

Global Minimum Cut Revisited

Here and later, when we say that an element from a set is chosen **at random**, we mean that every element is selected with the same probability.

In a graph $G = (V, E)$ with n nodes and m edges we wish to find a global min-cut (A, B), that is, a partitioning $V = A \cup B$ such that the number of cut edges (edges with one node in A and one node in B) is minimized. Motivations include the assessment of reliability of networks, finding clusters in graphs, and efficient hierarchical computation of distances in graphs.

We can easily reduce the problem to Minimum Cut, by trying all possible pairs of sources and sinks $s, t \in V$. But since flow and cut algorithms are somewhat sophisticated, it may be pleasant to see an extremely simple randomized algorithm that solves the Global Min-Cut problem as well. However this comes with a price: Success is no longer guaranteed. We will get a correct solution only with high probability.

For simplicity we discuss only the basic randomized algorithm for Global Min-Cut, although faster algorithms are known.

In the following we have to allow graphs with parallel edges (also called multiple edges). That is, there can exist several edges between the same two nodes.

The algorithm will use an operation called edge contraction: Let $e = (u, v)$ be any edge. Contracting e means: shrink e to one point, merge its end nodes u, v into one new node, and delete e and all edges that were parallel to e. Note that all nodes that were adjacent to u or v (or both) are now adjacent to the new node, and no further edges are deleted. (It is advisable to draw some small examples.) In particular, no loops are produced, but some more edges may become parallel.

The overall algorithm is now easily described. Choose an edge e at random(!) and contract e. Iterate this step $n - 2$ times, such that exactly two nodes a and b remain. This two-node graph represents a cut, in the obvious sense: A is the set of the original nodes merged into a, and B is the set of the original nodes merged into b. The number of parallel edges (a, b) is the size of the cut.

It may seem that this algorithm has nothing to do with the problem. It would just blindly and repeatedly contract random edges. Why should this lead to a minimum cut, or even to any small cut?

The intuition is that a small cut has a reasonable chance to be avoided by all these random edge contractions, thus it may be preserved in the end. But how large is actually the probability of this lucky event?

The analysis which has to confirm this intuition is not so obvious. It uses a clever combination of several elementary tools from probability theory.

Consider any global min-cut (A, B). Let F denote the set of its cut edges, and $k := |F|$. (If several minimum cuts exist, we fix any one of them.)

We study the contracted graph after j steps of the algorithm. Clearly, it has exactly $n - j$ nodes.

Assume that some node in the contracted graph has a degree smaller than k. Then this node and its complement set would form a cut smaller than k, a contradiction. Hence every node has a degree at least k.

Since the number of edges equals half the sum of degrees, the contracted graph has at least $k(n-j)/2$ edges. Therefore, the probability that, unfortunately, some of the k edges in F is contracted in the next step is at most $2/(n-j)$.

That means, our specific cut (A, B) is returned with probability at least

$$\prod_{j=0}^{n-3}(1-2/(n-j)) = \prod_{j=0}^{n-3}((n-j-2)/(n-j)) = 2/n(n-1) > 2/n^2$$

after the contraction procedure. This small success probability seems to be disappointing. However, the denominator is only $O(n^2)$. Now we simply repeat the contraction algorithm tn^2 times from scratch, where t is some free parameter, and we output the smallest cut found in all these trials. Hence we still need only polynomial time. How large is now the probability of finding a minimum cut?

As seen above, each trial fails with probability at most $1-2/n^2$. The probabilty that all tn^2 trials miss a minimum cut is therefore $(1-2/n^2)^{tn^2} \approx e^{-2t}$. Note that the failure probability decreases exponentially with the time (number of trials), and the user can choose t and trade time for reliability of the result. You may also recognize and appreciate the superficial similarity to approximation schemes.

Stop!! Was this calculation correct? Why can we simply multiply probabilities as we did? Repeated trials of the algorithm from scratch were independent, hence it was correct to multiply their failure probabilities. But inside the contraction algorithm, the events "step j avoids selecting an edge from F" are barely independent for different j, as every choice changes the edge set and thus influences the probabilities of further choices.

Nevertheless, the product formula leading to $2/n^2$ was correct, but for a different reason. What we actually multiplied were conditional probabilities of the following events: step j avoids selecting an edge from F, under the condition that all previous steps have already avoided F. Using the definition of conditional probabilities and some simple algebra one can verify that the product equals the probability that all steps avoid F, as claimed.

We omit the somewhat lengthy derivation here, but we want to draw your attention to the possible traps of probbaility calculations. A danger in the field is that we usually do not write down the probability spaces and events formally. Thus we must always critically reflect what we are doing.

Gambling in Monte Carlo and in Las Vegas

This Global Minimum Cut algorithm is an example of a **Monte Carlo algorithm**. These are algorithms that run in polynomial time in the worst case, but whose results can be wrong, with some small but acceptable probability. This failure probability can be reduced exponentially by repeated runs. The latter technique is called **probability amplification**. By way of contrast, **Las Vegas algorithms** give always correct results, but only expected time bounds can be shown, and running times can be much higher in the worst case. A famous example of a Las Vegas algorithm is Quicksort.

Appendix

A standard example for probability spaces is rolling dice. There we have $\Omega = \{1, 2, 3, 4, 5, 6\}$, and every elementary event has the same probability $1/6$. We emphasize that this is a model assumption about rolling dice, not a mathematical statement that follows from something else.

In this probability space, let us consider the following pairs of events:

A_1: the result is odd.
B_1: the result is at most 3.

A_2: the result is odd.
B_2: the result is at most 4.

What do you think: Are A_1 and B_1 independent? Are A_2 and B_2 independent? If you do not see it, simply do the calculations. You should get the result that one of these pairs is independent, and the other one is not. But what is the difference? (We only made a slight change in B after all.) Can you intuitively explain the result?