Lecture 22

Derandomization

Supplemental reading in CLRS: None

Here are three facts about randomness:

- Randomness can speed up algorithms (e.g., sublinear-time approximations)
- Under reasonable assumptions, we know that the speed-up can't be too large. Advances in complexity theory in recent decades have provided evidence for the following conjecture, which is generally believed to be true by complexity theorists:[1]

  Conecture. *Suppose there exists a randomized algorithm A which, on an input of size n, runs in time T and outputs the correct answer with probability at least* 2/3*. Then there exists a deterministic algorithm A' which runs in time* Poly(n,T) *and always outputs the correct answer.*

- In practice, randomization often doesn't buy any (or much) speed-up. Moreover, the randomized algorithm is often based on novel ideas and techniques which can equally well be used in a deterministic algorithm.

As we said in Lecture 9, a randomized algorithm can be viewed as a sequence of random choices along with a deterministic algorithm to handle the choices. *Derandomization* amounts to deterministically finding a possible sequence of choices which would cause the randomized algorithm to output the correct answer. In this lecture we will explore two basic methods of derandomization:

1. *Conditional expectations.* As we walk down the decision tree (see Figure 22.1), we may be able to use probabilistic calculations to guide our step. In this way, we can make choices that steer us toward a preponderance of correct answers at the bottom level.
2. *Pairwise independence.* Another way to find the correct answer is to simply run the randomized algorithm on *all* possible sequences of random choices and see which answer is reported most often (see Figure 22.2). In general this is not practical, as the number of sequences is exponential in the number of random choices, but sometimes it is sufficient to check only a relatively small collection of sequences (e.g., those given by a universal hash family).

In what follows, we will use the *big-cut problem* to explore each of these two methods of derandomization:

[1] See, for example, Impagliazzo and Wigderson, "P = BPP unless E has sub-exponential circuits: Derandomizing the XOR Lemma" ⟨http://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/IW97/proc.pdf⟩.

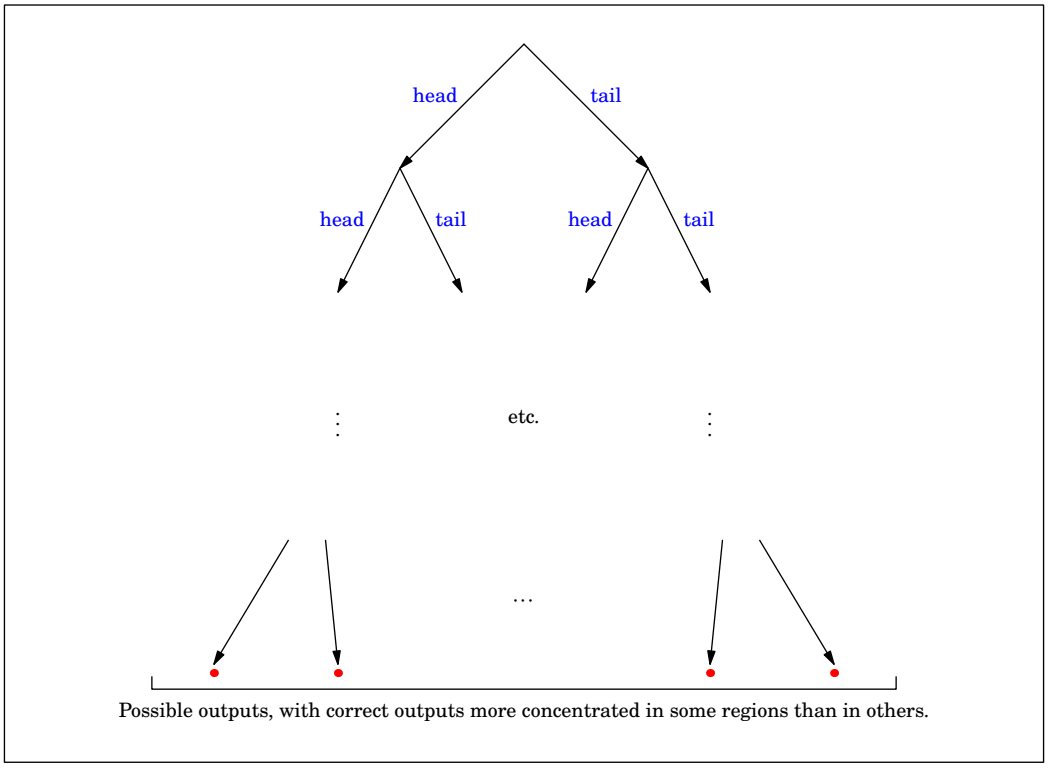

Figure 22.1. Probabilistic computations can often be used as a heuristic to find the regions in which correct answers are most concentrated.

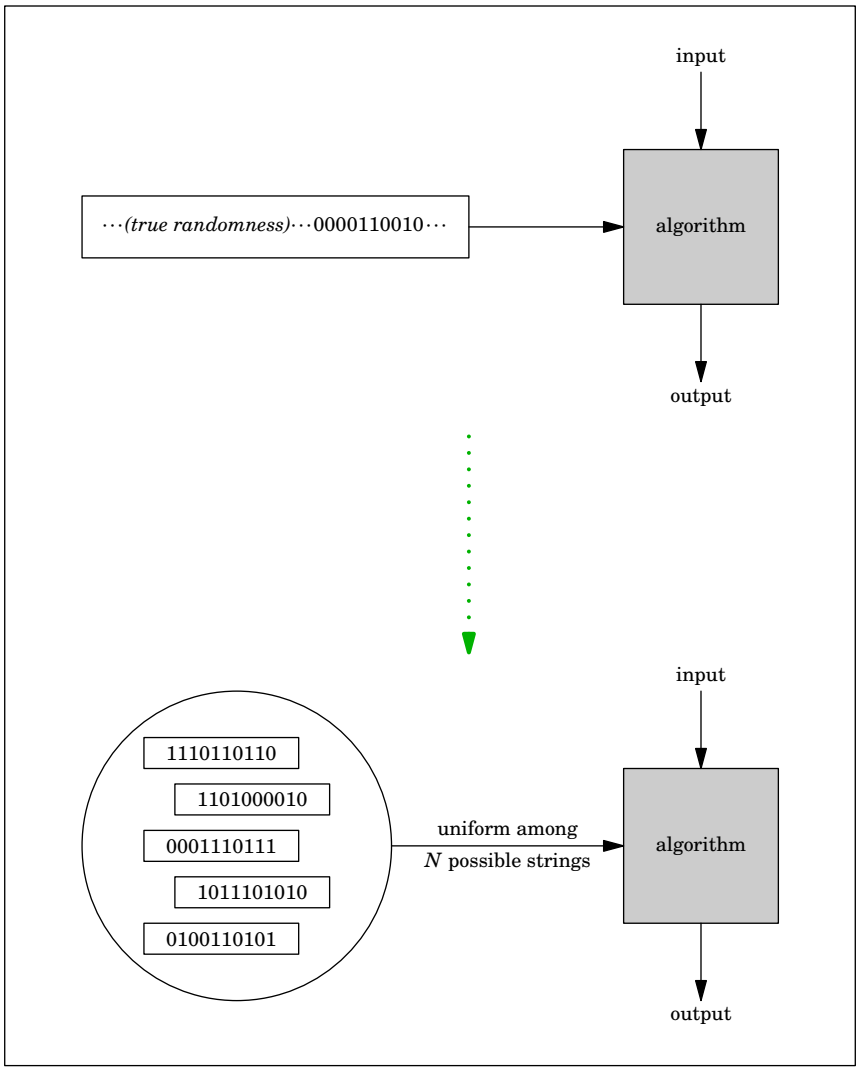

Figure 22.2. Sometimes the deterministic component of a randomized algorithm can't distinguish between a truly random sequence of choices and a well-chosen *pseudorandom* sequence of choices. Or, sometimes we can simulate a random choice from a large sample set (e.g., the set of all binary strings) using a random choice from a small sample set (e.g., a fixed set of N binary strings). If we can do this with N sufficiently small, it may be feasible to find the answer deterministically by running the randomized algorithm on all N of the possible random inputs.

Problem 22.1 (Big Cut)**.** Given an undirected graph $G = (V,E)$ with no loops, find a cut $(S, V \setminus S)$ which crosses at least half of the edges in E.

The solution to this problem is approximated by a simple randomized algorithm:

Algorithm:

```
S ← ∅
for each v ∈ V do
    Flip a coin
    If heads, then S ← S ∪ {v}
return (S, V \ S)
```

The running time is $\Theta(V)$.

Proof of approximation. For each edge $(u,v) \in E$, we have

$$\Pr\big[(u,v) \text{ crosses the cut}\big] = \Pr\left[\begin{matrix}(u \in S \text{ and } v \notin S) \text{ or} \\ (u \notin S \text{ and } v \in S)\end{matrix}\right] = \frac{1}{2}.$$

Thus, if $I_{(u,v)}$ denotes the indicator random variable which equals 1 if $(u,v) \in S$, then we have

$$\mathbb{E}\big[\#\text{ edges crossed by } S\big] = \mathbb{E}\left[\sum_{(u,v)\in E} I_{(u,v)}\right] = \sum_{(u,v)\in E} \mathbb{E}\big[I_{(u,v)}\big] = \tfrac{1}{2}\big|E\big|.$$

Chernoff's bound then tells us that, when E is large, it is extremely likely that the number of edges crossed is at least $0.499\big|E\big|$. □

22.1 Using Conditional Expectation

We can deterministically simulate the randomized approximation to Problem 22.1 by choosing to include a vertex v if and only if doing so (and making all future decisions randomly) would result in a higher expected number of crossed edges than choosing to exclude v. In more detail, say $V = \{v_1, \ldots v_n\}$, and suppose we have already decided whether S should contain each of the vertices $v_1, \ldots, v_k$. We pretend that all future decisions will be random (even though it's not true), so that we can use probabilistic reasoning to guide our choice regarding v_{k+1}. Ideally, our choice should maximize the value of

$$\mathop{\mathbb{E}}_{\substack{\text{future random}\\ \text{choices}}}\big[\#\text{ edges crossed} \,\big|\, \text{our existing decisions about } v_1, \ldots, v_{k+1}\big]. \tag{22.1}$$

Consider the partition

$$E = E_1 \sqcup E_2 \sqcup E_3,$$

where

$$\begin{aligned} E_1 &= \big\{(v_i, v_j) \in E : \ i,j \le k\big\} \\ E_2 &= \big\{(v_i, v_{k+1}) \in E : \ i \le k\big\} \\ E_3 &= \big\{(v_i, v_j) \in E : \ j > k+1\big\}. \end{aligned}$$

Whether we choose to include or exclude v_{k+1} from S, the number of crossed edges in E_1 is unaffected. Moreover, since each edge in E_3 has at least one of its vertices yet to be randomly assigned to S or

$V \setminus S$, the expected number of crossed edges in E_3 is $\frac{1}{2}|E_3|$ regardless of where we decide to put v_{k+1}. So in order to maximize (22.1), we should choose to put v_{k+1} in whichever set (either S or $V \setminus S$) produces more crossings with edges in E_2. (It will always be possible to achieve at least $\frac{1}{2}|E_2|$ crossings.) We can figure out what the right choice is by simply checking each edge in E_2.

Proceeding in this way, the value of $\mathbb{E}\big[\#\text{ edges crossed}\big]$ starts at $\frac{1}{2}|E|$ and is nondecreasing with each choice. Once we make the nth choice, there are no more (fake) random choices left, so we have

$$\big(\#\text{ edges crossed}\big) \;=\; \mathbb{E}\big[\#\text{ edges crossed}\big] \;\geq\; \tfrac{1}{2}\big|E\big|.$$

22.2 Using Pairwise Independence

The sequence of random choices in the randomized approximation to Problem 22.1 is equivalent to picking the set S uniformly at random from the collection $\mathscr{S} = \mathscr{P}(V)$ of all subsets of V. In hindsight, the only reason we needed randomness was to ensure that

$$\Pr_{S\in\mathscr{S}}\big[S \text{ doesn't cross } (u,v)\big] \leq \frac{1}{2} \qquad \text{for each } (u,v)\in E. \tag{22.2}$$

Recall from §10.1 that a subset of V can be represented as a function $V \to \{0,1\}$. In this way, $\mathscr{S}$ becomes a hash family $\mathscr{H} : V \to \{0,1\}$, and (22.2) becomes

$$\Pr_{h\in\mathscr{H}}\big[h(u) = h(v)\big] \leq \frac{1}{2} \qquad \text{for each } (u,v)\in E.$$

This will be satisfied as long as $\mathscr{H}$ is universal. So rather than taking $\mathscr{H}$ to be the collection $\{0,1\}^V$ of all functions $V \to \{0,1\}$, we can take $\mathscr{H}$ to be a much smaller universal hash family; there exist universal hash families of size $O(V)$.[2] In this way, we have

$$\mathop{\mathbb{E}}_{h\in\mathscr{H}}\begin{bmatrix}\#\text{ edges crossed by the}\\ \text{cut corresponding to } h\end{bmatrix} \;\geq\; \tfrac{1}{2}\big|E\big|.$$

In particular, this guarantees that there exists some $h \in \mathscr{H}$ such that

$$\begin{pmatrix}\#\text{ edges crossed by the}\\ \text{cut corresponding to } h\end{pmatrix} \;\geq\; \tfrac{1}{2}\big|E\big|.$$

We can simply check every $h \in \mathscr{H}$ until we find it.

Exercise 22.1. *What are the running times of the two deterministic algorithms in this lecture?*

[2] One such universal hash family is described as follows. To simplify notation, assume $V = \{1,\dots,n\}$. Choose some prime $p \geq n$ with $p = O(n)$ and let

$$\mathscr{H} = \{h_{a,b} : a,b \in \mathbb{Z}_p,\ a \neq 0\},$$

where

$$h_{a,b}(x) = (ax+b) \bmod p.$$

6.046J / 18.410J Design and Analysis of Algorithms
Spring 2012