INF 554: Using randomness in algorithms **Autumn 2013**

Lecture 2 — September 30th, 2013

Lecturer: Frédéric Magniez *Scribe: Clément Pit--Claudel*

2.1 Additional notes on primality testing

Note Although deterministic polynomial-time solutions to the PRIME problem are known (AKS), probabilistic algorithms remain significantly faster (Miller-Rabin's algorithm runs in $O\left(\lg(n)^2\right)$).

2.1.1 Application: Finding primes

Fast primality testing algorithms can be used to construct prime-finding algorithms (indeed, no easily computable formula to enumerate prime numbers is known).

FIND-PRIME

Input Integer N

Output Prime $p \in [\![N, 2N]\!]$

Algorithm

- Draw p uniformly from $[\![N, 2N]\!]$.
- Check if p is prime (e.g. using MILLER-RABIN):
  - If MILLER-RABIN accepts p, return p.
  - Otherwise, start over.

Theorem 2.1 (Chebyshev). *Let $\pi(x)$ be the number of primes $\leq x$. Then $\pi(x) \geq \frac{x}{2\ln(x)}$.*

Theorem 2.2. $\pi(x) \underset{x\to\infty}{\sim} \frac{x}{\ln(x)}$.

Corollary 2.3. *The number of primes in $[\![n, 2n]\!]$ is $\Omega\left(\frac{n}{\ln(n)}\right)$.*

Corollary 2.4. $\underset{p\in[\![n,2n]\!]}{\mathbb{P}}(p \text{ prime}) = \Omega\left(\frac{1}{\ln(n)}\right)$

Average time complexity $O(\ln(N))$ iterations; each iteration costs $O(\ln(N))$ modular additions/multiplications. Hence $O(\ln(N)^2)$.

Error Same as that of MILLER-RABIN.

Notes Errors do not accumulate. Also, the number of iterations can be bounded (thus turning this Las Vegas algorithm into a Monte-Carlo one) by failing after a set number of iterations (the probability of returning nothing after k iterations, or equivalently $\Theta(k \ln(n))$ operations, would then be $\frac{1}{2^k}$).

2.2 Polynomial identity testing

2.2.1 Problem definition

POLYNOMIAL-IDENTITY-TESTING (PIT)

Input Q and R, two n-variables polynomials of degree $\leq d$.

Output ACCEPT iff $Q = R$.

Notes Expanding P and Q and comparing individual coefficients takes exponential time in the size of their representation – in other words, compact representations exist that allow for fast evaluation of polynomials whose expanded form contains an exponential number of coefficients.
In the **black-box** model nothing is known about P and Q, and the only available operation is $x \mapsto P(x), Q(x)$. This single operation is assumed to be fast.

Example 1: Determinant Let $Q = \prod_{1 \leq i < j \leq n} (X_i - X_j)$ and $R = \det\left(X_i^j\right)$. Then $Q = R$, evaluating Q and R takes linear time in n, and expanding Q and R takes exponential time in n.

Example 2: Arithmetic circuits Arithmetic circuits are a tree-based representation of polynomial factorizations.

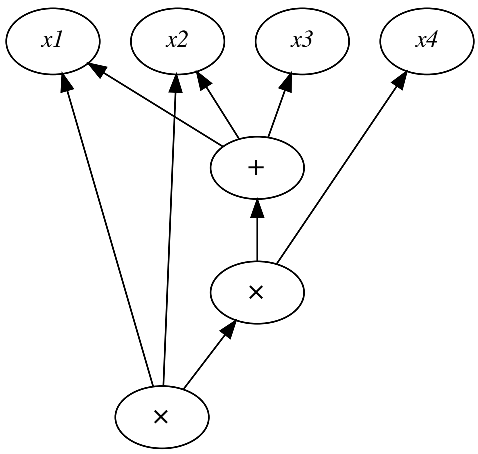

Figure 2.1. $x_1 x_2 x_4 (x_1 + x_2 + x_3)$

State of the art Deterministic solutions for the PIT problems are known for polynomials represented as arithmetic circuits of depth ≤ 2. Partial results were also obtained for multi-linear polynomials of depths 3, 4.
(Additional note: depth 4 is the most important one; deterministically solving PIT for arithmetic circuits of depth 4 would represent a significant leap forward for complexity theory.)

Lemma 2.5 (Schwartz-Zippel). *Let F denote an arbitrary field, and S denote a finite subset of F. Then for any non-zero polynomial $T(X_1, \dots, X_n)$ of degree d,*

$$\underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0) \leq \frac{d}{|S|}$$

Proof (by induction): If $n = 1$, then T has at most d roots, and $\underset{a \in S}{\mathbb{P}} (T(a) = 0) \leq \frac{d}{|S|}$.
If $n > 1$, expanding T by its first variable yields $T = \sum_i X_1^i T_i(X_2, \dots, X_n)$. Let j be the degree of T relative to X_1 – that is, the highest i such that $T_i \neq 0$. Then

$$\begin{aligned}\underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0) = &\underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0 \text{ and } T_j(a_2, \dots, a_n) = 0) \\ &+ \underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0 \text{ and } T_j(a_2, \dots, a_n) \neq 0)\end{aligned}$$

Noting that T_j is a $n-1$ variables polynomial of degree $d' = d - j$ and applying the induction hypothesis yields $\underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T_j(a_2, \dots, a_n) = 0) \leq \frac{d-j}{|S|}$, which implies that

$$\underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0 \text{ and } T_j(a_2, \dots, a_n) = 0) \leq \frac{d-j}{|S|}$$

To bound the second term, introduce $a_2, \dots, a_n$ such that $T_j(a_2, \dots, a_n) \neq 0$. The strong induction hypothesis applied to $T(X_1, a_2, \dots, a_n)$ (a single-variable polynomial of degree j) yields $\underset{a_1 \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0) \leq \frac{j}{|S|}$. In other words,

$$\underset{a_1,\dots,a_n \in S}{\mathbb{P}} \left(\underbrace{T(a_1, \dots, a_n) = 0}_{E} \middle| \underbrace{T_j(a_2, \dots, a_n) = 0}_{F} \right) \leq \frac{j}{|S|}$$

Finally, note that

$$\begin{aligned}\underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0 \text{ and } T_j(a_2, \dots, a_n) \neq 0) &= \mathbb{P}(E \cup F) \\ &= \mathbb{P}(E \mid F)\, \mathbb{P}(F) \\ &\leq \mathbb{P}(E \mid F) \\ &\leq \frac{j}{|S|}\end{aligned}$$

Combining both results yields the stated inequality:

$$\underset{a_1,\dots,a_n \in S}{\mathbb{P}} (T(a_1, \dots, a_n) = 0) \leq \frac{d}{|S|}$$

□

Algorithm

- Draw $a_1, \dots, a_n$ randomly from $[\![1, 2d]\!]$
- Accept iff $P(a_1, \dots, a_n) = Q(a_1, \dots, a_n)$

Time complexity Two polynomial evaluations.

Error

- One sided
- True-biased

If $P \neq Q$, then by Schwartz-Zippel's lemma $\mathbb{P}(ACCEPT) \leq \frac{d}{|S|} \leq \frac{1}{2}$.

Notes In practice, evaluating P and Q can yield extremely large values. To circumvent this problem, all calculations are generally made modulo a large prime value p. Carefully choosing this value is crucial to ensure that $P = Q \mod p$ is indeed equivalent to $P = Q$. Denoting the largest coefficient of P and Q as M, p can obtained by choosing a prime value larger than twice the maximum of d and M.

2.2.2 Fingerprints

FINGERPRINT Let A and B denote two players.

First player's input n-bits sequence u.

Second player's input n-bits sequence v.

Output ACCEPT iff $u = v$.

Complexity Number of bits exchanged.

Naive solution

- A sends u to B.
- B accepts iif $u = v$.

Complexity n bits.

Hash functions Vectors of $\mathbf{Z}_2^n$ are mapped to elements of $\mathbf{Z}_2[X_1, \dots, X_n]$ through the hash function $H : (a_i) \mapsto \sum_{0 \leq i < n} a_{i+1} X^i$ (or $\tilde{H} : (a_i) \mapsto \sum_{0 \leq i < n} a_{n-i} X^i$). These functions are such that $H(u) = H(v) \iff u = v$.

Algorithm

- A picks a prime number $p \in [\![n^2, 2n^2]\!]$.
- A picks a random number $a \in [\![1, n-1]\!]$.
- A sends $(p, a, H(u)(a) \mod p)$ to B.
- B accepts iff $H(v)(a) = H(u)(a) \mod p$.

Error

- One-sided
- True-biased

If $u \neq v$, then B accepts with probability $\leq \frac{1}{n}$.

Complexity $6\lg(n) + o(1)$ bits.

Time complexity n modular additions and multiplications for both A and B.

Note This algorithm is insecure: it is vulnerable to collision-based attacks.

2.2.3 Pattern-matching

PATTERN-MATCHING

Input Word $w \in \mathbf{Z}_2^n$, pattern $p \in \mathbf{Z}_2^k$. $k \leq n$.

Output Positions where p occurs in w: $\{i \mid p = w\,[i : i+k-1]\}$.

Note A naive deterministic algorithm (for each index $i \in [\![1, n-k+1]\!]$ in w, check whether $p = w\,[i : i+k-1]$) runs in $O(nk)$ time. Many efficient, deterministic, linear-time solutions are known (Rabin–Karp, Knuth–Morris–Pratt, Boyer-Moore, etc.), but all are tricky to implement. Probabilistic algorithms, on the other hand, achieve similar performance and are very easy to implement.

Note The nature of our hash functions allows for easy calculation of checksums of overlapping subwords. Recall that $\tilde{H} : (a_j) \mapsto \sum_{0 \leq j < n} a_{n-j} X^j$, and assume that $h_i(a) = \tilde{H}(w\,[i : i+k-1])(a) = \sum_{0 \leq j < k} w_{i-1+(k-j)} a^j$ is known. Then $h_{i+1}(a)$ can be derived in $O(1)$ from h_i. Indeed,

$$
\begin{aligned}
h_{i+1} &= \tilde{H}(w\,[i+1 : i+k]) \\
&= \sum_{0 \leq j < k} w_{i+(k-j)} X^j \\
&= \sum_{1 \leq j < k} w_{i+(k-j)} X^j + w_{i+k} \\
&= X \sum_{0 \leq j < k-1} w_{i-1+(k-j)} X^j + w_{i+k} \\
&= X(h_i - w_i X^{k-1}) + w_{i+k}
\end{aligned}
$$

Evaluating in a yields $h_{i+1}(a) = w_{i+k} + a\left(h_i(a) - w_i a^{k-1}\right)$.

Algorithm As usual, all calculations are run modulo a large enough prime value q. For each index i, we decide whether $w[i : i + k - 1]$ matches the pattern p by comparing $h_i(a)$ to $\tilde{H}(p)(a)$, for randomly sampled values of a.

- Pick a prime number $q \in [\![n^3, 2n^3]\!]$.
- Draw a randomly from $[\![0, q - 1]\!]$.
- Compute $h_p = \tilde{H}(p)(a)$.
- Compute $h = \tilde{H}(w[1 : k])$.
- For $i \in [\![1, n - k + 1]\!]$
  - If $h = h_p$, then append i to the list of accepted indices.
  - If $i \neq n - k + 1$, then update $h \leftarrow w_{i+k} + a\left(h - w_i a^{k-1}\right)$.

Time complexity $O(n)$ modular additions/multiplications.

Error

- One-sided
- True-biased

Errors consist in returning extraneous indices. For each non-matching index i,

$$\begin{aligned}\mathbb{P}\left(i \in \text{returned-values}\right) &= \mathbb{P}\left(h_i(a) = \tilde{H}(p)(a) \mid h_i \neq \tilde{H}(p)\right) \\ &\leq \frac{k}{p} \leq \frac{k}{n^3} \leq \frac{1}{n^2}\end{aligned}$$

Hence the union bound yields

$$\mathbb{P}\left(\text{incorrect output}\right) = \mathbb{P}\left(\exists i \in \text{returned-values} \mid p \neq w[i : i + k - 1]\right) \leq n \cdot \frac{1}{n^3} \leq \frac{1}{n^2}$$

Note Instead of choosing large prime numbers, one can reduce the probability of error by computing checksums for multiple different a.

2.2.4 Bipartite perfect matching

BIPARTITE-PERFECT-MATCHING (BPM)

Input Balanced bipartite graph $G = (E, U \sqcup V)$, with $|U| = |V| = n$.

Output ACCEPT iff a perfect matching exists in E, i.e. E contains n disjoint edges.

Note A deterministic $O(\sqrt{|U| + |V|} \cdot |E|) = O(n^{2,5})$ time solution yielding such a perfect matching if it exists is known (Hopcroft-Craft). Probabilistic algorithms by Lovasz (1979) achieve a time complexity for the decision problem equal to that of the calculation of a single $n \times n$ determinant modulo $p \in [\![n, 2n]\!]$. A 1987 extension by Mulmuley, U. Vazirani, and V. Vazirani gives a probabilistic estimate of the largest such matching in any general graph, in $O(1)$ matrix inversions time.

Note The calculation of a determinant can be reduced to a matrix multiplication problem.

Adjacency matrices Identify u and V with $[\![1, n]\!]$, and define the bi-adjacency matrix A as

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

Expanding $\det(A)$ yields $\det(A) = \sum_{\sigma \in \mathfrak{S}_n} (-1)^{\text{sgn}(\sigma)} \prod_i A_{i,\sigma(i)}$, and $\prod_i A_{i,\sigma(i)}$ is non-zero iff σ represents a perfect matching in G. Hence if $\det(A) \neq 0$ then there exists at least one perfect matching. The converse, unfortunately, does not hold due to the $(-1)^{\text{sgn}(\sigma)}$ term.

Note The permanent of A, defined as $\text{perm}\,(A) = \sum_{\sigma \in \mathfrak{S}_n} \prod_i A_{i,\sigma(i)}$, exactly equals the number of BPM in G, but computing it is a #P-complete problem ; the fastest known deterministic solution (Ryser's formula) has $O(2^n n)$ time complexity. The fastest known approximation (Jerrum, Sinclair and Vigoda) still requires $O(n^{10})$ time.

Tutte matrix Since computing the determinant of A is not sufficient, we introduce the Tutte matrix T of G as the $n \times n$ matrix

$$T_{i,j} = \begin{cases} X_{i,j} & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

Theorem 2.6. *$\det(T)$ is a $|E|$-variables polynomial of $\mathbf{Z}_2^n[X]$ whose degree d is $\leq n$, and $\det(T) \neq 0 \iff G$ has a BPM.*

Proof: If no BPM exists, then the determinant is null. Conversely, if a BPM exist, then the determinant is non-null. Indeed, each non-zero $\prod_i \delta_{(i,\sigma(i)) \in E} X_{i,\sigma(i)}$ monomial in the expansion of $\det(T)$ matches a single permutation, and is thus distinct of all other monomials in the expanded $\det(T)$ polynomial. □

Since the elements of T are polynomials, expanding $\det(T)$ is extremely costly. On the other hand, since $\forall x, \det(T)(x) = \det(T(x))$, evaluating $\det(T)$ in a single point is relatively cheap.

Algorithm

- Pick a prime number $p \in [\![n^2, 2n^2]\!]$.
- Draw $|E|$ random elements (a_i) from $[\![1, p-1]\!]$.
- Accept iff $\det(T(a)) \neq 0 \mod p$, where T is the Tutte matrix of G.

Error

- One sided
- False-biased (If the algorithm accepts, then the existence of a BPM is guaranteed)

The probability of incorrectly rejecting is exactly $\mathbb{P}(\det(T)(a) = 0 \mid \det(T) \neq 0)$, which by Schwartz-Zippel's lemma is $\leq \frac{d}{|S|} \leq \frac{n}{n^2} = \frac{1}{n}$.

Time complexity Equal to that of computing an $n \times n$ determinant ($O\left(n^{2.3727}\right)$ using Coppersmith-Winograd algorithms).

2.3 Exercises

2.3.1 Associativity testing

ASSOCIATIVE $S = [\![1, n]\!]$

Input $\circ : S \times S \to S$.

Output ACCEPT iff $\circ$ is associative.

Complexity Number of operations involving $\circ$.

Naive solution Checking all possible triples $(i, j, k) \in S^3$ requires $2n^3$ comparisons, and (assuming proper memoisation) n^2 evaluations of $\circ$.

Notes The number of witnesses of the non-associativity of an arbitrary law $\circ$ may be very small. As an example, consider defining $i \circ j = 3$ for all i, j except $1 \circ 2 = 1$. Then for all $\forall (a, b, c) \neq (1, 2, 2), a \circ (b \circ c) = 3 = (a \circ b) \circ c$, but $(1 \circ 2) \circ 2 = 1 \neq 3 = 1 \circ (2 \circ 2)$. In this case there exists a single witness $(1, 2, 2)$ of the non-associativity of $\circ$. The following sections are hence dedicated to expanding the search space to increase the relative frequency of witnesses.

Extension of the search space Let $S(p) = (\mathbf{Z}_p)^n$, and let $(e_1, \ldots, e_n)$ denote a basis of $S(p)$. Define the bilinear $\bullet$ operation over $S(p)$ by taking $e_i \bullet e_j = e_{i \circ j}$ and extending it to $S(p)$. Finally, note that if $(A_i)_i$ denotes the coefficients of A in the $(e_i)_i$ basis, then $A \bullet B = \sum_{i,j} A_i B_j e_{i \circ j}$.

Lemma 2.7. *$\bullet$ is associative iff. $\circ$ is.*

Proof: Assume $\circ$ is associative. Then $\forall (i, j, k), (e_i \bullet e_j) \bullet e_k = e_{(i \circ j) \circ k} = e_{i \circ (j \circ k)} = e_i \bullet (e_j \bullet e_k)$.
Conversely, assume $\bullet$ is associative. Then $\forall (i, j, k), e_{(i \circ j) \circ k} = (e_i \bullet e_j) \bullet e_k = e_i \bullet (e_j \bullet e_k) = e_{i \circ (j \circ k)}$, and hence $(i \circ j) \circ k = i \circ (j \circ k)$. $\square$

Lemma 2.8. *For all $(A, B, C) \in S(p)$, $(A \bullet B) \bullet C$ is a third-degree polynomial in the coefficients of A, B, C.*

Proof: Explicit expansion yields $(A \bullet B) \bullet C = \sum_{i,j,k} A_i B_j C_k e_{(i \circ j) \circ k}$. $\square$

Lemma 2.9. *Assume that $p = 7$ and that $\circ$ is not associative.*
Then $\underset{A,B,C \in S}{\mathbb{P}} \left((A \bullet B) \bullet C = A \bullet (B \bullet C)\right) \leq \frac{3}{7}$.

Proof: Given that $\circ$ is not associative, there exists a 3-tuple $(A, B, C) \in S(p)^3$ such that $(A \bullet B) \bullet C \neq A \bullet (B \bullet C)$. In other words, the third-degree polynomial $(A \bullet B) \bullet C - A \bullet (B \bullet C)$ in the A_i, B_j, C_k coefficients is not null. Hence (Schwartz-Zippel) $\underset{A,B,C \in S}{\mathbb{P}} \left((A \bullet B) \bullet C = A \bullet (B \bullet C)\right) \leq \frac{d}{\#S(p)} = \frac{3}{7}$. $\square$

Algorithm

- Draw A, B, C at random from $S(7)$.
- Compute $AB = A \circ B$,
  $BC = B \circ C$,
  $AB\_C = AB \circ C$,
  $A\_BC = A \circ BC$.
- Accept iff. $AB\_C = A\_BC$.

Complexity n^2 calls are required to build the full multiplication table of $\circ$.

Time complexity Each of the four subsequent calculations require $O(n^2)$ modular additions and multiplications, bringing the total time complexity to $O(n^2)$.

Error

- One-sided
- True-biased

If $\circ$ is not associative, then (by lemma 2.9) $\mathbb{P}\left(ACCEPT\right) \leq \frac{3}{7}$.

2.3.2 Notes on randomized algorithms as opposed to deterministic algorithms taking randomized input

Theorem 2.10. *Given a finite input set I and a set of random choices R, let $A(x, r), x \in I, r \in R^{(\mathbf{N})}$ denote a randomized algorithm such that $\forall x \in I$,*

- $\underset{r}{\mathbb{P}}\left(A(x, r) \text{ wrong}\right) \leq \epsilon$
- *$A(x, r)$ returns in time $\leq T$*

then there exists a deterministic algorithm B whose time complexity is $\leq T$ on all inputs, such that

$$\underset{x}{\mathbb{P}}\left(B(x) \text{ wrong}\right) \leq \epsilon$$

Proof: Let $\varepsilon(r) = \underset{x}{\mathbb{P}}\left(A(x, r) \text{ wrong}\right)$. Then

$$\begin{aligned}\underset{r}{\mathbb{E}}\left(\varepsilon(r)\right) &= \underset{x,r}{\mathbb{P}}\left(A(x, r) \text{ wrong}\right) \\ &= \underset{x}{\mathbb{E}}\left(\underbrace{\underset{r}{\mathbb{P}}\left(A(x, r) \text{ wrong}\right)}_{\leq \epsilon}\right) \\ &\leq \epsilon\end{aligned}$$

Hence there exists a sequence of random choices r such that $\varepsilon(r) \leq \epsilon$. □

Theorem 2.11 (Yao). *The converse holds in the following sense:*
Assume that for any probability distribution $\mathcal{D}$ over I there exists a deterministic algorithm $B_{\mathcal{D}}$ whose time complexity is $\leq T$ for all inputs x, such that

$$\mathbb{P}_{x \sim \mathcal{D}} (B_{\mathcal{D}}(x) \text{ wrong}) \leq \epsilon$$

Then there exists a probabilistic algorithm $A(x, r)$ whose time complexity is $\leq T$ for all inputs x, r, such that

$$\forall x, \mathbb{P}_{r} (A(x, r) \text{ wrong}) \leq \epsilon$$

Note The proof is based on linear programming.