Karlsruher Institut für Technologie

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

Exercise Sheet 2 – The Power of Randomness

Probability and Computing

Exercise 1 – Checking Polynomial Equations

Let $\mathbb{F}$ be a field, for example $\mathbb{F} = \mathbb{Q}$. Given is a polynomial equation, for example:

$$(x^3 + 2x^2 - 5x - 6)(x^2 + x - 20)(x - 6) \stackrel{?}{=} x^6 - 7x^3 + 25$$

(a) Argue: In the case that the example equation does *not* hold, there are at most 6 values of x for which both sides yield the same result.

(b) Describe a randomized algorithm that decides whether a polynomial equation holds or not. This algorithm may accept false polynomial equations as correct with a small probability. What can be said about this probability?

Solution 1

(a) The equation has the form $f(x) = g(x)$ and can be rewritten as $f(x) - g(x) = 0$. If the equation does not hold, then $f(x) - g(x)$ is a polynomial of degree $0 \leq d \leq 6$. Such a polynomial has at most 6 roots. Therefore, there are at most 6 values of x for which $f(x) - g(x) = 0$, and hence $f(x) = g(x)$.

(b) First, consider the case $|\mathbb{F}| < \infty$. We then choose $X \sim \mathcal{U}(\mathbb{F})$ uniformly at random and substitute X into the equation. If the polynomial equation holds universally, it also holds for this X. If it does not hold, then as in (a) we can consider the maximum degree d of both sides. There are at most d elements of $\mathbb{F}$ for which both sides give the same result, and the probability that we picked one of them is at most $d/|\mathbb{F}|$.

If $|\mathbb{F}| = \infty$, there is no uniform distribution over $\mathbb{F}$. This is somewhat inconvenient, but we can choose X uniformly at random from a large finite subset $S \subseteq \mathbb{F}$. Then the upper bound on the error probability $d/|S|$ can be made arbitrarily small.

Remark: This is a false-biased Monte Carlo algorithm (see the lecture on Probability Amplification).

Exercise 2 – Checking Matrix Products[1]

Let $\mathbb{F}$ be a field, and $n \in \mathbb{N}$.

(a) Show: If $C, C' \in \mathbb{F}^{n \times n}$ are two different matrices and $v \in \{0,1\}^n$ is chosen uniformly at random, then $\Pr[C \cdot v \neq C' \cdot v] \geq \frac{1}{2}$.

(b) Describe an algorithm that, given $A, B, C \in \mathbb{F}^{n \times n}$, outputs a bit X with $X = 1$ if $A \cdot B = C$ and $\Pr[X = 1] \leq 1/2$ if $A \cdot B \neq C$. The algorithm should perform only $O(n^2)$ field operations.

Solution 2

(a) Let $C_1, \ldots, C_n \in \mathbb{F}^n$ and $C'_1, \ldots, C'_n \in \mathbb{F}^n$ be the columns of C and C' respectively, and let $v_1, \ldots, v_n \in \{0,1\}$ be the entries of v. Furthermore, let $i \in [n]$ be an index with $C_i \neq C'_i$, which must exist by assumption. We can then write:

$$D := C \cdot v - C' \cdot v = \underbrace{\sum_{\substack{j=1 \\ j \neq i}}^{n} (C_j - C'_j) \cdot v_j}_{w} + (C_i - C'_i) \cdot v_i.$$

Imagine that all entries of v except the i-th have already been chosen, and only v_i remains random. There are two cases:

Case 1. $w = 0$. In this case, $v_i = 1$ leads to $D = C_i - C'_i \neq 0^n$.

Case 2. $w \neq 0$. In this case, $v_i = 0$ leads to $D = w \neq 0^n$.

In both cases, at least one of the two possibilities for v_i results in $D \neq 0^n$. Hence overall, $\Pr[C \cdot v \neq C' \cdot v] = \Pr[D \neq 0^n] \geq \frac{1}{2}$.

(b) We use (a) with $C' = A \cdot B$.

sample $v \leftarrow \mathcal{U}(\{0,1\}^n)$
$w_1 \leftarrow C \cdot v$
$w_2 \leftarrow A \cdot B \cdot v$ // Computation order: $A \cdot (B \cdot v)$
return $X = \mathbb{1}_{w_1 = w_2}$

Clearly, $X = 1$ is guaranteed if $A \cdot B = C$. If $A \cdot B \neq C$, then by (a) $\Pr[X = 1] = \Pr[C \cdot v = C' \cdot v] \leq \frac{1}{2}$.

The three matrix–vector products can each be computed in $O(n^2)$ field operations. In particular, this is faster than a full matrix–matrix multiplication, which (with a naive algorithm) requires $\Omega(n^3)$ field operations.

Remark: This is a false-biased Monte Carlo algorithm (see the lecture on Probability Amplification).

[1] Known as Freivalds' Algorithm.

Exercise 3 – Deterministic Evaluation of $\bar{\wedge}$-Trees

Let A be a deterministic algorithm that takes as input a bit vector $I \in \{0,1\}^n$ (with $n = 2^d$) and computes the value of the complete balanced $\bar{\wedge}$-tree whose leaves are labeled according to I. Show: There exists an input $I_A \in \{0,1\}^n$ such that A must inspect every leaf label.

Solution 3

We act as an adversary to the algorithm and assign the value of a leaf only when it is queried. We ensure that it must query all leaves. The resulting leaf labeling is then the worst-case input I_A for A.

We prove the statement by induction. For $d = 1$ (leaf = root), there is nothing to show — the algorithm must obviously query the only leaf.

For $d > 1$, there are two subtrees of depth $d - 1$. In both, we use the strategy given by the induction hypothesis. Thus, the outcome of a subtree can only be determined once all 2^{d-1} leaves have been queried. Since the last leaf affects the result of the subtree, we can even choose its value at the end so that the entire subtree evaluates to 1. Hence, the total result at the root is $1 \bar{\wedge} b$, where b is the result of the other subtree. The algorithm therefore also needs to determine b, and thus must query all leaves in the second subtree as well.