CS630 Homework 6

Qilin Ye

April 17, 2025

Solution to problem 1. We associate each vertex v with a polynomial P_v as suggest by the hint: define

$$P_v(x) = \begin{cases} x_0 & \text{if } v \text{ is a leaf} \\ \prod_{w \in \text{children}(v)} (x_h - P_w(x)) & \text{otherwise, and height}(v) = h. \end{cases}$$

If T_1 and T_2 are isomorphic, we claim that their corresponding polynomials defined at roots coincide. To see this, let $\varphi : T_1 \to T_2$ be an isomorphism; we prove that $P_v(x) = P_{\varphi(v)}(x)$ for all $v \in T_1$. The base case is trivial: a leaf is mapped to a leaf, so the corresponding polynomials are simply x_0. For the inductive step, suppose the claim holds for all nodes of height $< h$. If v has height h and has children $v_1, \cdots, v_k$, then

$$P_v(x) = \prod_{i=1}^{k} (x_h - P_{v_i}(x)) = \prod_{i=1}^{k} (x_h - P_{\varphi(v_i)}(x)) = P_{\varphi(v)}(x)$$

where the middle $=$ follows from IH. This completes the proof.

On the other hand, if two tress are not isomorphic, the difference betweeen the root polynomials is a nonzero polynomial whose total degree is at most n.

The rest is a standard fingerprinting proof. Fix a prime field $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$ with a sufficiently large prime p, say $p > n^2$. Choose uniformly $a_0, \cdots, a_H \in \mathbb{F}$ where H is the max height of both trees. We evaluate both root polynomials at $(a_0, \cdots, a_H)$, and output "isomprphic" iff the two evaluations agree. The inductive argument shows there will be no false ngatives. On the other hand, if the trees are non-isomorphic, the difference polynomial has degree $\leqslant n$, and so the probability of false positive is less than $n/p < 1/n$.

The polynomials can be evaluated in $\mathcal{O}(n)$ time, so we have derived a randomized algorithm that correctly tells if two trees are isomorphic with one-sided error $< 1/n$ in $\mathcal{O}(n)$ time.

Solution to problem 2. For a set S, define a polynomial $P_S(x) = \prod_{s_i \in S}(x - s_i)$. We will test whether P_{S_1} and P_{S_2} coincide. As usual we work over a prime field $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$ for a sufficiently large p, where we obtain a one-sided false positive rate w.p. $< 1/n$.

This PIT-based algorithm is preferable when the integers are not too large, so multiplication/evaluation will not take too much time. The major benefit is clearly the linear runtime. However, if (i) one-sided error is not acceptable or (ii) sets contain huge integers, one needs to still resort to the superlinear runtime of a standard sort-and-compare.

Solution to problem 3. Let $f(G)$ be an oracle that returns True iff G has a perfect matching, and suppose its runtime is $T(n, m)$ as described. We build the actual matching M on G as follows:

- Initialize $M \leftarrow \varnothing$ and $G_0 = G$.

- Repeat until G_i empty:
  - Pick any $u \in G_i$.
  - For each neighbor v of u in G_i: if the subgraph G_i' induced by $G_i \backslash \{u, v\}$ still admits a perfect matching, break loop and pair (u, v) in M, and set $G_{i+1} \leftarrow G_i'$.

The correctness of the oracle will guarantee that the for loop will always terminate by finding one such "good" neighbor to match. After $n/2$ steps (outside the loop), the algorithm will terminate. We will have at most $\sum_v \deg(v) = 2m$ calls to the oracle, where each subcall takes no more than $T(n, m)$ time. Hence the overall runtime is $\mathcal{O}(mT(n, m))$.

Solution to problem 4. For the three properties:

- The concatenation of bits is unambiguous, and matrix association is associate. Therefore $M(s)$ is well-defined.
- Suppose $M(x) = M(y)$. To see the last bit of x, we define

  $$N_0 = M(x)M_0^{-1} \qquad \text{and} \qquad N_1 = M(x)M_1^{-1}$$

  (observe M_0, M_1 are both invertible). If x ends in a 0, then N_0's entries are all non-negative (clearly, $M(s)$ has non-negative entries for all strings). On the other hand, since $M_0M_1^{-1} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}$, $N_1 = M(x')(M_0M_1^{-1})$ will in fact acquire a negative entry.

  Likewise, if the last bit of x is a 1, then N_1 will be element-wise non-negative but N_0 will not. Therefore, the uniqueness of this reconstruction shows that if $M(x) = M(y)$ then x, y must have "peeled off" the same last bit, and the recursion continues. Hence $x = y$.
- Suppose $M(t) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. Then

  $$M(t)M_0 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} a+b & b \\ c+d & d \end{bmatrix} \qquad \text{and} \qquad M(t)M_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} a & a+b \\ c & c+d \end{bmatrix}.$$

  To use induction, we state that the two elements in the same row for $M(s)$ where $|s| = n$ are bounded by the n^{th} and the $(n-1)^{\text{th}}$ Fibonacci number, respectively. The base case is trivial as $M(\epsilon)$ satisfies this property; the inductive step can be seen above: if a, b are bounded by the n^{th} and $(n-1)^{\text{th}}$ (order doesn't matter) Fibonacci numbers, then a and $a+b$ will be bounded by the $(n+1)^{\text{th}}$ and n^{th}.

Now, to the algorithm. Let the text be $T = (t_1, \cdots, t_n)$ and the pattern $P = (p_1, \cdots, p_m)$. We work with a prime field $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$ where p is sufficiently large (e.g. larger than the $(m+1)^{\text{th}}$ Fibonacci number, and also n). The pattern fingerprint will be given by $M(p_1)\cdots M(p_m)$. For the string, we will use a sliding window that computes the matrix products associated with m consecutive entries. We initialize

$$H[0] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{and} \qquad H[i] = H[i-1]M(t_i) \pmod p \qquad \text{for } 1 \leqslant i \leqslant n,$$

and for each $i = 1, 2, \cdots, n - m + 1$, we compare

$$W_i = H[i + m - 1](H[i - 1])^{-1} \pmod p$$

against F_p. If $W_i = F_p$, then the current sliding window matches pattern P. Observe that $H[\cdot] \in \mathrm{GL}_2$ so the determinant can be computed efficiently via closed-form formulas. Because $M(\cdot)$ is injective on length-m strings and all entries stay below p, this test will have no false positives.

Solution to problem 5. Let the degree sequence and sampled r_i be defined as in the problem statement. We show the claim via induction.

The base case $n = 1$ follows from the standard Schwartz-Zippel.

For the inductive step, assume the claim holds for $n - 1$ variables. Let $Q(x_1, \cdots, x_n)$ be a nonzero n-variate polynomial with degree sequence $(d_1, \cdots, d_n)$. Write

$$Q(x_1, \cdots, x_n) = \sum_{k=0}^{d_1} x_1^k Q_k(x_2, \cdots, x_n),$$

where d_1 is the max exponent of x_1, and each Q_k is an $(n-1)$-variate polynomial. By definition, Q_{d_1} is not identically zero, and its degree sequence is $(d_2, \cdots, d_n)$. We split the "bad" event $\{Q(r_1, \cdots, r_n) = 0\}$ into two cases, similar to the example shown in class:

- Case 1: $Q_{d_1}(r_2, \cdots, r_n) = 0$.
- Case 2: $Q_{d_1}(r_2, \cdots, r_n) \neq 0$ but $Q(r_1, \cdots, r_n) = 0$.

For case 1, since Q_{d_1} is nonzero in the last $n - 1$ variates with degree sequence $(d_2, \cdots, d_n)$, the IH implies this happens w.p. $\leqslant \sum_{i=2}^{n} d_i/|S_i|$.

For case 2, conditioned on any fixed choice $(r_2, \cdots, r_n)$ for which $Q_{d_1} \neq 0$, the polynomial $Q(x_1, r_2, \cdots, x_n)$ is a nonzero univariate polynomial in x_1 of degree eactly d_1, so the univariate bound over S_1 implies that this case happens w.p. $\leqslant d_1/|S_1|$. This bound holds for *every* choice of $(r_2, \cdots, r_n)$ with $Q_{d_1} \neq 0$, so the overall probability of case 2 is $\leqslant d_1/|S_1|$.

Summing over bad events,

$$\mathbb{P}(Q(r_1, \cdots, r_n) = 0) \leqslant \sum_{i=2}^{n} \frac{d_i}{|S_i|} + \frac{d_1}{|S_1|} = \sum_{i=1}^{n} \frac{d_i}{|S_i|}.$$