CSE 6512 Lecture 6 Notes

Morad Behandish

September 15, 2011

1 Finger Printing (Review)

In the previous lecture, three examples were presented as applications of randomized algorithms, in which the idea of "finger printing" was utilized:

Example 1. Given three $n \times n$ matrices A, B and C, check if $AB = C$.

Note: This is equivalent to checking if a given $n \times n$ matrix $D := AB - C \equiv 0$.

Solution. A Monte Carlo algorithm was proposed in which one checks if the equality $A(Br_i) = Cr_i$ (equivalently $Dr_i = 0$) holds for a number of randomly selected $n \times 1$ binary vectors $r_i \in \{0,1\}^n$, where i runs from 1 to s. It was demonstrated that if the equality holds for $s \geq \alpha \log n$ random binary vectors, the probability of the wrong answer is very low:

$$Prob.\left[Dr_1 = Dr_2 = \cdots = Dr_s = 0 \;\middle|\; D \neq 0\right] \leq \left(\frac{1}{2}\right)^s \leq \left(\frac{1}{2}\right)^{\alpha \log n} = n^{-\alpha} \quad (1)$$

The error is one-sided. The run-time is $\mathcal{O}(n^2 \log n)$.

Example 2. Given two degree-(n) polynomials $P_1(x), P_2(x)$ and a degree-$(2n)$ polynomial $P_3(x)$ check if $P_1(x)P_2(x) \equiv P_3(x)$ for all values of $x \in \mathbb{F} \subseteq \mathbb{R}$.

Note: This is equivalent to checking if a given degree-$(2n)$ polynomial $Q(x) := P_1(x)P_2(x) - P_3(x) \equiv 0$ for all values of $x \in \mathbb{F} \subseteq \mathbb{R}$.

Solution. A Monte Carlo algorithm was proposed in which one checks if the equality $P_1(r)P_2(r) = P_3(r)$ (equivalently $Q(r) = 0$) holds for a *single* random number $r \in \mathbb{S}$ where $\mathbb{S} \subseteq \mathbb{F}$. It was demonstrated that if r is selected from a subset as large as $|\mathbb{S}| \geq 2n^{\alpha+1}$ the probability of the wrong answer is very low:

$$Prob.\left[Q(r) = 0 \;\middle|\; Q(x) \neq 0\right] \leq \frac{2n}{|\mathbb{S}|} \leq \frac{2n}{2n^{\alpha+1}} = n^{-\alpha} \quad (2)$$

The error is one-sided. The run-time is linear $\mathcal{O}(n)$.

Example 3. Check if a given n-variable polynomial $Q(x_1, x_2, \cdots, x_n) \equiv 0$ for all values of $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{F} \subseteq \mathbb{R}^n$.

Note: The degree of a term in Q is defined as the sum of the exponents of the variables in that term. The total degree of Q (denoted d) is the maximum of the degrees of individual terms.

Solution. A Monte Carlo algorithm was proposed in which one checks if the equality $Q(r_1, r_2, \cdots, r_n) = 0$ holds for a *single* random point (consisting of n random real numbers) $\mathbf{r} = (r_1, r_2, \cdots, r_n) \in \mathbb{S}$ where $\mathbb{S} \subseteq \mathbb{F}$. It was demonstrated that if $\mathbf{r}$ is selected from a subset as large as $|\mathbb{S}| \geq dn^{\alpha}$ the probability of the wrong answer is very low:

$$Prob.\left[Q(\mathbf{r}) = 0 \;\middle|\; Q(\mathbf{x}) \neq 0\right] \leq \frac{d}{|\mathbb{S}|} \leq \frac{d}{dn^{\alpha}} = n^{-\alpha} \tag{3}$$

where Schwartz-Zippel lemma was used - proved in this lecture in more detail:

Theorem. (Schwartz and Zippel's) Given a multi-variable polynomial $Q(\mathbf{x})$ of degree-d, where $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{F} \subseteq \mathbb{R}^n$, given a point $\mathbf{r} = (r_1, r_2, \cdots, r_n)$ randomly selected from $\mathbb{S} \subseteq \mathbb{F}$,

$$Prob.\left[Q(r_1, r_2, \cdots, r_n) = 0 \;\middle|\; Q(x_1, x_2, \cdots, x_n) \neq 0\right] \leq \frac{d}{|\mathbb{S}|} \tag{4}$$

Proof. (By induction on n) The theorem is true for $n = 1$ (See previous lecture's notes - Example 2 - for the proof). [1]

$$Prob.\left[Q(r) = 0 \;\middle|\; Q(x) \neq 0\right] \leq \frac{d}{|\mathbb{S}|} \tag{5}$$

Now assuming the lemma is correct for any $(n-1)$-variable polynomial, we shall prove it for n-variable polynomial $Q(x_1, x_2, \cdots, x_n)$. Let us write the latter as a summation of its individual terms; factor-out the powers of the first variable x_1^i from each term and collect the terms with the same i. The collective coefficient of x_1^i is denoted as $Q_i(x_2, x_3, \cdots, x_n)$:

$$Q_i(x_2, x_3, \cdots, x_n) = \sum_{i=0}^{i=k} x_1^i Q_i(x_2, x_3, \cdots, x_n) \tag{6}$$

where k is the degree of Q in terms of x_1, which means the largest exponent i

[1] Comparing eqs. 2 and 5 one should keep in mind that in eq. 2, the degree of single-variable polynomial $Q(x)$ is $(2n)$; this should not be confused with notation in equation 4 where n refers to the number of variables and d is used to denote the degree.

of x_1 among the above terms with $Q_i(x_2, x_3, \cdots, x_n) \neq 0$. [2]

Note that the total degree of $Q_k(x_2, x_3, \cdots, x_n)$ is $\leq (d-k)$. Note also that all Q_i's are $(n-1)$-variable polynomials. Applying the induction hypothesis:

$$Prob.\Bigg[Q_k(r_2, r_3, \cdots, r_n) = 0 \;\Bigg|\; Q_k(x_2, x_3, \cdots, x_n) \neq 0\Bigg] \leq \frac{d-k}{|\mathbb{S}|} \tag{7}$$

where the condition $Q_k(x_2, x_3, \cdots, x_n) \neq 0$ is always true as a result of the definition of k. So eq. 7 can be re-written as:

$$Prob.\big(\mathbf{B}\big) := Prob.\Bigg[Q_k(r_2, r_3, \cdots, r_n) = 0\Bigg] \leq \frac{d-k}{|\mathbb{S}|} \tag{8}$$

Where the "event $\mathbf{B}$" is defined as $Q_k(r_2, r_3, \cdots, r_n) = 0$. Now consider the "event $\bar{\mathbf{B}}$" i.e. $Q_k(r_2, r_3, \cdots, r_n) \neq 0$. Let:

$$q(x_1) := Q(x_1, r_2, \cdots, r_n) = \sum_{i=0}^{i=k} x_1^i Q_i(r_2, \cdots, r_n) \tag{9}$$

Note that $q(x_1)$ is a degree-k single-variable polynomial and cannot be identically zero, in the case of "event $\bar{\mathbf{B}}$". Applying the induction hypothesis:

$$Prob.\big(\mathbf{A}|\bar{\mathbf{B}}\big) := Prob.\Bigg[q(r_1) = 0 \;\Bigg|\; q(x_1) \neq 0\Bigg] \leq \frac{k}{|\mathbb{S}|} \tag{10}$$

Where the "event $\mathbf{A}$" is defined as $q(r_1) = Q(r_1, r_2, \cdots, r_n) = 0$ with the condition $Q(x_1, x_2, \cdots, x_n) \neq 0$ as an implicit assumption. On the other hand, we know from probability theory that: [3]

$$Prob.\big(\mathbf{A}\big) \leq Prob.\big(\mathbf{A}|\bar{\mathbf{B}}\big) + Prob.\big(\mathbf{B}\big) \tag{11}$$

---

[2] For example:

$$\begin{aligned} Q(x_1, x_2, x_3) :&= 5x_1x_2^2x_3^2 + 7x_1x_2x_3 + 10x_1^5x_2^4 - 14x_1^3x_2^2x_3^5 + 3x_3^3 + 1 \\ &= x_1^0\big(3x_3^3 + 1\big) + x_1^1\big(5x_2^2x_3^2 + 7x_2x_3\big) + x_1^3\big(-14x_2^2x_3^5\big) + x_1^5\big(10x_2^4\big) \end{aligned}$$

where $Q_0(x_1, x_2, x_3) = 3x_3^3 + 1$, $Q_1(x_1, x_2, x_3) = 5x_2^2x_3^2 + 7x_2x_3$, $Q_2(x_1, x_2, x_3) = 0$, $Q_3(x_1, x_2, x_3) = -14x_2^2x_3^5$, $Q_4(x_1, x_2, x_3) = 0$ and $Q_5(x_1, x_2, x_3) = 10x_2^4$. Here $k = 5$.

[3] Here is the proof:

$$Prob.\big(\mathbf{A}\big) = Prob.\big(\mathbf{A}|\bar{\mathbf{B}}\big)Prob.\big(\bar{\mathbf{B}}\big) + Prob.\big(\mathbf{A}|\mathbf{B}\big)Prob.\big(\mathbf{B}\big)$$

$$\begin{aligned} Prob.\big(\bar{\mathbf{B}}\big) \quad &\leq 1 \Longrightarrow Prob.\big(\mathbf{A}|\bar{\mathbf{B}}\big)Prob.\big(\bar{\mathbf{B}}\big) \leq Prob.\big(\mathbf{A}|\bar{\mathbf{B}}\big) \\ Prob.\big(\mathbf{A}|\mathbf{B}\big) &\leq 1 \Longrightarrow Prob.\big(\mathbf{A}|\mathbf{B}\big)Prob.\big(\mathbf{B}\big) \leq Prob.\big(\mathbf{B}\big) \\ \Longrightarrow Prob.\big(\mathbf{A}\big) &\leq Prob.\big(\mathbf{A}|\bar{\mathbf{B}}\big) + Prob.\big(\mathbf{B}\big) \;\blacksquare \end{aligned}$$

Substituting for the RHS terms in eq. 11 from eqs. 8 and 10 we obtain:

$$Prob.\big(\mathbf{A}\big) \leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \tag{12}$$

Substituting for the LHS term from its definition:

$$Prob.\Big[Q(r_1, r_2, \cdots, r_n) = 0 \;\Big|\; Q(x_1, x_2, \cdots, x_n) \neq 0\Big] \leq \frac{d}{|\mathbb{S}|} \tag{13}$$

And if $|\mathbb{S}| \geq dn^{\alpha}$ then the above probability is $\leq n^{-\alpha}$. ■

2 Applications

2.1 Vandermonde's Identity

Vandermonde's matrix is defined as:

$$\mathbf{A}_{n\times n} = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & x_3^3 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \cdots & x_n^{n-1} \end{pmatrix}$$

Vandermonde's identity states that:

$$\det\big(\mathbf{A}\big) = \prod_{1\leq j<i\leq n} \big(x_i - x_j\big). \tag{14}$$

The above identity can be verified using the Schwatz and Zippel's lemma. The run time of such a verification is $O(M(n))$ where $M(n)$ is the time needed to multiply two $n \times n$ matrices.

2.2 Perfect Matching

Input. A bi-partite graph $\mathbb{G}\big(\mathbf{U}, \mathbf{V}, \mathbf{E}\big)$ in which the vertices are organized into two sets $\mathbf{U} = \{u_1, u_2, \cdots, u_n\}$ and $\mathbf{V} = \{v_1, v_2, \cdots, v_n\}$ with equal cardinality $n = |\mathbf{U}| = |\mathbf{V}|$, interconnected with $|\mathbf{E}| = m$ edges. [4]

[4] A bi-partite graph (aka a bi-graph) is one whose vertices can be divided into two disjoint sets $\mathbf{U}$ and $\mathbf{V}$ such that every edge (u_i, v_j) connects a vertex $u_i \in \mathbf{U}$ to a vertex $v_j \in \mathbf{V}$:

$$\mathbf{E} = \{(u_i, v_j) : u_i \in \mathbf{U}, v_j \in \mathbf{V}\}$$

In other words, there is no edge between two vertices of $\mathbf{U}$ such as (u_i, u_j) nor between two vertices of $\mathbf{V}$ such as (v_i, v_j). (Equivalently, a bi-partite graph is a graph that does not contain any odd-length cycles...)

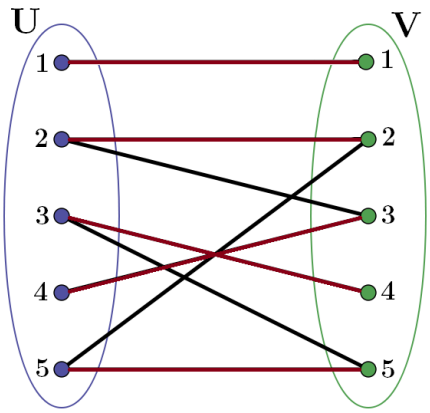

Figure 1: Example of a bi-partite graph with 5+5 vertices, 9 edges and a perfect matching (colored red)

Definition. A "Matching" is a subset of edges $\mathbf{E}$ such that each vertex occurs at most once. A "Perfect Matching" aka "Maximum Matching" is a matching of size $n = |\mathbf{U}| = |\mathbf{V}|$.

Example. Fig. 1 illustrates an example of a bi-partite graph with $2n = 10$ vertices and 8 edges, 5 of which are grouped into a perfect matching named $\mathbf{M}$:

$$\mathbf{U} = \{u_1, u_2, u_3, u_4, u_5\}; \quad \mathbf{V} = \{v_1, v_2, v_3, v_4, v_5\}$$
$$\mathbf{E} = \{(u_1, v_1), (u_2, v_2), (u_2, v_3), (u_3, v_4), (u_3, v_5), (u_4, v_3), (u_5, v_2), (u_5, v_5)\}$$
$$\mathbf{M} = \{(u_1, v_1), (u_2, v_2), (u_3, v_4), (u_4, v_3), (u_5, v_5)\} \subset \mathbf{E}$$

Output. "YES" if the given bi-partite graph $\mathbb{G}\big(\mathbf{U}, \mathbf{V}, \mathbf{E}\big)$ with $n = |\mathbf{U}| = |\mathbf{V}|$ has a perfect matching; "NO" otherwise.

Theorem. (Edmond's) Let $\mathbf{A}$ be an $n \times n$ matrix whose elements are:

$$A_{i,j} = \begin{cases} x_{i,j} & \text{if } (u_i, v_j) \in \mathbf{E} \\ 0 & \text{otherwise} \end{cases}$$

where $x_{i,j}$ denotes a different independent variable for every value of $1 \leq i \leq n$ and $1 \leq j \leq n$. $\mathbb{G}$ has a perfect matching iff $\det\big(\mathbf{A}\big) \neq 0$ (not *identically* zero).

Proof. From definition, the determinant is evaluated as:

$$\det\big(\mathbf{A}\big) = \sum_{\pi \in \mathcal{S}_n} sign(\pi) \cdot A_{1,\pi(1)} \cdot A_{2,\pi(2)} \cdots A_{n,\pi(n)} \tag{15}$$

where $j = \pi(i)$ stands for any possible permutation (i, j) and $\mathcal{S}_n$ is the set of all such permutations. If the graph has a perfect matching, then for every given vertex $u_i \in \mathbf{U}$, there is a $v_j \in \mathbf{V}$ such that $(u_i, v_j) \in \mathbf{E}$, this corresponds to a permutation $(i, j = \hat{j}(i))$ [5] for which $A_{i,j} = x_{i,j} \neq 0$. Therefore, there exists a non-zero term in the above equation, namely $\pm x_{1,\hat{j}(1)} \cdot x_{2,\hat{j}(2)} \cdots x_{n,\hat{j}(n)}$. This corresponds to a perfect matching $\{(u_1, v_{\hat{j}(1)}), (u_2, v_{\hat{j}(2)}), \cdots, (u_n, v_{\hat{j}(n)})\}$. Since the variables $x_{i,j}$ are independent, no two terms in the sum can cancel each other, hence $\det(\mathbf{A}) \neq 0$.

One the other hand, if there is no perfect matching, there will not exist such a term in eq. 15 - otherwise this contradicts the assumption of no perfect matching - thus all the terms in it will have at least one $A_{i,j} = 0$ consequently $\det(\mathbf{A}) = 0$. ■

Fact. The determinant of any $n \times n$ matrix can be computed in $\mathcal{O}(M(n))$ time, [6] where $M(n)$ is the time needed to multiply two $n \times n$ matrices. [7]

Fact. There exists a deterministic algorithm for constructing maximum matching, that takes $\mathcal{O}(m\sqrt{n})$ time where $n = |\mathbf{U}| = |\mathbf{V}|$ and $m = |\mathbf{E}|$.

However, the proposed randomized algorithm, which is based on calculating the determinant of the Edmond's matrix works for arbitrary graphs. It is also very suitable for parallel implementation.

3 Pattern Matching

3.1 String Comparison

Input. Two *ordered* sets $\mathbf{A} = \{a_1, a_2, \cdots, a_n\}$ and $\mathbf{B} = \{b_1, b_2, \cdots, b_n\}$. Without loss of generality, we can assume that a_i's and b_j's are all binary values (either 0 or 1). Thus these sets represent two long binary numbers A and B respectively.

Output. "YES" if $A = B$; and "NO" otherwise.

[5] Note that $\hat{j} : i \in [1, n] \longrightarrow j \in [1, n]$ gives the index of a vertex $v_j \in \mathbf{V}$ that is connected to the vertex $u_i \in \mathbf{U}$ with given index i. For a bi-partite graph that has a perfect matching, there always exists such a value of j for any given i.

[6] Using the trivial "paper-and-pencil" algorithm it takes $\mathcal{O}(n!)$ time which is impractical.

[7] $M(n) = \mathcal{O}(n^3)$ using the naive matrix multiplication algorithm (i.e. dot product of rows and columns) while $M(n) \approx \mathcal{O}(n^{2.807})$ using Strassen's algorithm and $M(n) \approx \mathcal{O}(n^{2.376})$ using Coppersmith-Winograd's algorithm. (The best asymptotic time-complexity known to date, though with a very large constant that makes it slower than Strassen's algorithm unless the matrices are very large.)

Note. Let $Q := \sum_{i=0}^{i=n-1} a_{i+1}2^i$ and $R := \sum_{j=0}^{j=n-1} b_{j+1}2^j$. The above problem is equivalent to checking whether two integers Q and R are equal or not.

A trivial comparison algorithm takes n bitwise comparisons, thus the run-time is $\mathcal{O}(n)$. We will see that it can be improved using randomization and the idea of "finger printing":

Solution. Pick a random prime number $p \leq t$ where t is to be determined during the analysis. Then check if $\big(A \bmod p\big) = \big(B \bmod p\big)$. We will show that this implies $A = B$ with very high probability if t is selected appropriately.

Let $K := (A - B)$. The question is whether $K = 0$ or not. The algorithm checks if $\big(K \bmod p\big) = \big(A \bmod p\big) - \big(B \bmod p\big) = 0$ i.e. if "p divides K".

Note. $K \leq 2^n$ since this is the maximum possible difference between Q and R. So there exist less than (or equal to) $\log K = n$ prime divisors for K.

Fact. The number of prime numbers p such that $p \leq t$ is $= \Theta(\frac{t}{\log t})$. [8]

$$\begin{aligned} \implies & \mathit{Prob.}\Big[\big(A \bmod p\big) = \big(B \bmod p\big) \;\Big|\; A \neq B\Big] \\ = & \mathit{Prob.}\Big[\big(K \bmod p\big) = 0 \;\Big|\; K \neq 0\Big] \leq \frac{n}{t/\log t} \end{aligned} \tag{16}$$

We want this probability to be $\leq n^{-\alpha}$. So we can pick $t \geq n^\alpha \log n^\alpha$ (an even safer choice would be $t \geq n^{\alpha+1}$).

$$\mathit{Prob.}\Big[\big(K \bmod p\big) = 0 \;\Big|\; K \neq 0\Big] \leq n^{-\alpha} \tag{17}$$

Note. The number of binary bits required to represent p is $\log p \leq \log t \leq \log n^{\alpha+1} = (\alpha + 1)\log n$. Therefore, the bit-complexity (i.e. time-complexity with bitwise operations as basic operations) is $\mathcal{O}(\log n)$. Note also that a single random prime number p is sufficient.

3.2 String Search

Input. A "text" $\mathbf{T} = t_1 t_2 \cdots t_n$ of size $n = |\mathbf{T}|$ and a "pattern" $\mathbf{P} = p_1 p_2 \cdots p_m$ of size $m = |\mathbf{P}|$, both of which using some alphabet $t_i, p_j \in \sum^*$. ($m \ll n$.)

Output. The starting indices in $\mathbf{T}$ wherever an instance of $\mathbf{P}$ occurs.

[8] This is a theorem from number theory, the proof of which was not discussed in the class.

Fact. This problem can be trivially solved in $\mathcal{O}(mn)$ time using the following algorithm: Scan through $\mathbf{T}$, stepping forward one letter at a time, from index 1 up to $(n - m + 1)$. Then compare the next m letters of $\mathbf{T}$ with all letters of $\mathbf{P}$.

3.3 Knuth-Morris-Pratt's Algorithm

The KMP algorithm solves the above problem *deterministically* by employing the observation that when a mismatch occurs, the pattern itself embodies sufficient information to determine where the next match could begin, thus bypassing re-examination of previously matched characters. [9] The KMP algorithm takes $\mathcal{O}(m + n)$ time, where $n = |\mathbf{T}|$ and $m = |\mathbf{P}|$.

3.4 Rabin-Karp's Algorithm

The following string search algorithm was proposed by Michael O. Rabin and Richard M. Karp in 1987. The idea is the same as previous algorithm using "finger printing": Without loss of generality, assume that $\sum = \{0, 1\}$. Then both $\mathbf{T}$ and $\mathbf{P}$ correspond to binary numbers T and P respectively.

Let $\mathbf{T}_i$ be the m-digit string (same length as $\mathbf{P}$) that starts at i'th position in string $\mathbf{T}$: $\mathbf{T}_i = t_i t_{i+1} \cdots t_{i+m-1}$. Let T_i be the corresponding binary number.

Algorithm 1 Rabin-Karp's Algorithm

```
for i := 1 → (n − m + 1) do
  pick a random prime p ≤ t
  if (P mod p) = (T_i mod p) then
    output i as a match;
  end if
end for
```

Run-time and probability analysis will be carried out in the next lecture.

[9] Details about the KMP algorithm were not discussed in the class.