Homework #1

Due: 2025-3-20 23:59 | 8 Questions, 110 Pts

Name: 方嘉聪 ID: 2200017849

Note: The total points of this homework is 100 + 10, with 10 points being the *Challenge Problem.*

Question 1 (10') (Von Neumann Trick). Given a coin with probability p of getting "head" where $p \in (0,1)$. In this problem, we are going to use this coin to generate fair results in $\{0,1\}$. When $p = 1/2$, it's quite simple. However, it may not be so trivial for general p. John von Neumann gave the following procedure:

1. Toss the coin twice.
2. If two results are the same, start over, forgetting both results.
3. If two results are different, give 0 when the first result is "head", and 1 otherwise.

a. (3') Prove that, the procedure can generate uniform random results in $\{0,1\}$.

b. (7') Calculate the expectation for the number of throws. Express the answer as a function of p.

◀

Answer. a). In each throw round, there are 3 cases:

$$\Pr[\text{results are the same}] = p^2 + (1-p)^2$$
$$\Pr[\text{terminated, output } 0] = \Pr[\text{terminated, output } 1] = p(1-p).$$

And for the whole procedure, we have

$$\Pr[\text{output } 0] = \sum_{t=1}^{\infty} \left(p^2 + (1-p)^2\right)^{t-1} \cdot p(1-p) = p(1-p) \cdot \frac{1}{2p(1-p)} = \frac{1}{2}.$$

Similarly, we have $\Pr[\text{output } 1] = 1/2$. Thus, the procedure can generate uniform random results in $\{0,1\}$.

b). Denote the number of throws as X. Then we have

$$\Pr(X = k) = \left[p^2 + (1-p)^2\right]^{k-1} \cdot 2p(1-p).$$
$$\mathbb{E}[X] = 2\sum_{k=1}^{\infty} k \cdot 2p(1-p) \cdot \left[p^2 + (1-p)^2\right]^{k-1}$$
$$= 4p(1-p)\sum_{k=1}^{\infty} k \cdot \left[p^2 + (1-p)^2\right]^{k-1}.$$

Let $g(p) = \sum_{k=0}^{\infty}(2p^2 - 2p + 1)^k = \frac{1}{2p(1-p)}$, then

$$g'(p) = -\frac{(1-2p)}{2p^2(1-p)^2} = \sum_{k=1}^{\infty} k(4p-2)(2p^2-2p+1)^{k-1}$$

$$\Longrightarrow \sum_{k=1}^{\infty} k(2p^2-2p+1)^{k-1} = \frac{1}{4p^2(1-p)^2} \Longrightarrow \mathbb{E}[X] = \frac{4p(1-p)}{4p^2(1-p)^2} = \frac{1}{p(1-p)}.$$

Q.E.D. ◁

Question 2 (15') (Refined Error Bound for Schwartz–Zippel Algorithm). In this question, you will reflect on the error bound of Schwartz–Zippel Algorithm.

a. (3') Show the error bound in Schwartz–Zippel Algorithm is tight by giving an example. For convenience, assume $n = 3$ and $S = \{1, 2, 3, 4, 5\}$. Give two distinct polynomials $P(x_1, x_2, x_3)$ and $Q(x_1, x_2, x_3)$ where the degree of $P - Q$ is 3, such that Schwartz–Zippel Algorithm fails with probability *exactly* 3/5.

b. (12') Let $f_0(x_1, x_2, \cdots, x_n)$ be a multivariate polynomial over $\mathbb{R}$. For each $1 \le i \le n$, we inductively define d_i to be the maximum exponent of x_i in f_{i-1}, and $f_i(x_{i+1}, \cdots, x_n)$ to be the coefficient of $x_i^{d_i}$ in f_{i-1}. Assume $S_1, S_2, \cdots, S_n \subseteq \mathbb{R}$ be arbitrary finite subsets. For $r_i \in S_i$ chosen independently and uniformly at random, show that

$$\Pr\left[f_0(r_1, r_2, \cdots, r_n) = 0 | f_0 \not\equiv 0\right] \le \sum_{i=1}^{n} \frac{d_i}{|S_i|}.$$

◀

Answer. a). Let $P(x_1, x_2, x_3) = (x_1 - 1)(x_1 - 2)(x_1 - 3) + x_1x_2x_3$ and $Q(x_1, x_2, x_3) = x_1x_2x_3$. then

$$R := P - Q = (x_1 - 1)(x_1 - 2)(x_1 - 3), \text{ whose degree is 3.}$$

Therefore, the probability of failure is

$$\Pr[R(r_1, r_2, r_3) = 0 | R \not\equiv 0] = \frac{3 \cdot 5^2}{5^3} = \frac{3}{5}.$$

b). Prove the statement by induction on n.

Base case. When $n = 1$, f_0 is a single variable polynomial with at most d_1 roots in S_1. Thus,

$$\Pr[f_0(r_1) = 0 | f_0 \not\equiv 0] = \frac{d_1}{|S_1|}.$$

Induction hypothesis. Assume the statement holds for all multivariate polynomial with at most $k - 1$ variables.

Inductive step. Consider the polynomial $f_0(x_1, x_2, \cdots, x_k)$, then we can write

$$f_0(x_1, x_2, \cdots, x_k) = f_1(x_2, \cdots, x_k)x_1^{d_1} + N(x_1, x_2, \cdots, x_k).$$

Assume $r_2, r_3, \cdots, r_k$ are fixed, denote Ω as the event that $f_1(r_2, r_3, \cdots, r_k) = 0$. Then

- When Ω happenes, by the induction hypothesis,

$$\Pr[\Omega] \le \sum_{i=2}^{k} \frac{d_i}{|S_i|}$$

- When Ω does not happen, fix $r_2, r_3, \cdots, r_k$, then $f_0(r_1, r_2, \cdots, r_k)$ is a single variable polynomial with at most d_1 roots in S_1. Thus,

$$\Pr[f_0(r_1, r_2, \cdots, r_k) = 0 | \neg\Omega] \le \frac{d_1}{S_1}$$

Combine the two cases, we have

$$\begin{aligned}\Pr[f_0(r_1,\cdots,r_k)=0|f_0\not\equiv 0] &= \Pr[\Omega]\Pr[f_0(r_1,\cdots,r_k)=0|\Omega]+\Pr[\neg\Omega]\Pr[f_0(r_1,\cdots,r_k)|\neg\Omega]\\ &\le \sum_{i=2}^{k}\frac{d_i}{|S_i|}+\frac{d_1}{|S_1|}=\sum_{i=1}^{k}\frac{d_i}{|S_i|}.\end{aligned}$$

Therefore, the statement holds for all $n \in \mathbb{N}^{+}$.

Q.E.D. $\triangleleft$

Question 3 (20') (Schwartz–Zippel is More General). Our course begins with two examples of randomized algorithms: checking matrix multiplication and checking associativity. It turns out that both can be tackled using Schwartz–Zippel's algorithm. In this problem, you will give equal or better error bounds by constructing polynomials for the two problems.

a. (10') Given a binary operator $\circ$ on a set X of size n, we wish to decide if $\circ$ is associative. In our class, we defined operator $\circ$ on 2^X to check the associativity on $X = \{x_1, \cdots, x_n\}$. Similarly we can define operator $\circ$ on p^X where $p \geq 5$ is a given prime. In other words, every element in p^X are in the form of $\sum_{i=1}^n r_i x_i$ with $r_i \in \mathbb{F}_p$.
Base on this, please design a randomized algorithm similar to the one in class. However, this time, you may want to analyze its error bound by applying Schwartz–Zippel Lemma to a polynomial, rather than inclusive-exclusive argument. (You do not need to prove the equivalence to associativity. Schwartz–Zippel Lemma also works for finite fields, provided the size of field is larger than the degree of polynomial — you do not need to prove this, either.)
[Hint: Construct multivariate polynomials $F_1(\cdot), \cdots, F_n(\cdot)$ such that $\sum_{i=1}^n F_i(\cdot) x_i \equiv 0$ iff $\circ$ is associative over p^X.]

b. (10') Design a randomized algorithm that finds a counterexample when $\circ$ is not associative over X. Analyze the time complexity of your algorithm.

◀

Answer. a). The algorithm is as follows:

Algorithm 1 Check Associativity

Input: A binary operator $\circ$ on a set X of size n.

Output: Whether $\circ$ is associative over p^X

```
1: Randomly sample $\{r_i\}_{i=1}^n, \{s_j\}_{j=1}^n, \{t_k\}_{k=1}^n$ from $\mathbb{F}_p$ independently and uniformly.
2: $R \leftarrow \sum_{i=1}^n r_i x_i, S \leftarrow \sum_{j=1}^n s_j x_j, T \leftarrow \sum_{k=1}^n t_k x_k$.
3: if $(R \circ S) \circ T \neq R \circ (S \circ T)$ then
4:     return False
5: else
6:     return True
```

Let $F(x) = (R \circ S) \circ T - R \circ (S \circ T)$, and denote $F_i(\cdot)$ as the coefficient of x_i in $F(x)$. Then we have $F(x) = \sum_{i=1}^n F_i(\cdot) x_i$. And it's easy to verify that $F(x) \equiv 0$ iff $\circ$ is associative over p^X. Note that $\deg(F(x)) \leq 3$. By Schwartz-Zippel Lemma, the error bound of this algorithm is

$$\Pr[(R \circ S) \circ T = R \circ (S \circ T) | \circ \text{ is not associative}] \leq \frac{3}{|\mathbb{F}_p|} = \frac{3}{p}.$$

b). We first repeat **Algorithm 1** until finding R, S, T such that $(R \circ S) \circ T \neq R \circ (S \circ T)$. And then find the counterexample $r_i, s_j, t_k \in X$ by the following algorithm:

Algorithm 2 Find Counterexample

Input: A binary operator $\circ$ on a set X of size n.

Output: A counterexample (r_i, s_j, t_k) such that $(r_i \circ s_j) \circ t_k \neq r_i \circ (s_j \circ t_k)$.

```
1: Repeat **Algorithm 1** until finding $R, S, T$ such that $(R \circ S) \circ T \neq R \circ (S \circ T)$.
2: Denote $R = \sum_{i=1}^n r_i x_i, S = \sum_{j=1}^n s_j x_j, T = \sum_{k=1}^n t_k x_k$.
3: while $\max\{|R|, |S|, |T|\} > 1$ do                    ▷ $|R| := \#\{r_i | r_i \neq 0\}$, etc.
4:     Split $S = \sum_{i=1}^{\lfloor n/2 \rfloor} s_i x_i + \sum_{i=\lfloor n/2 \rfloor+1}^n s_i x_i := S_1 + S_2$.    ▷ assume $|S| = \max\{|R|, |S|, |T|\}$
5:     if $(R \circ S_1) \circ T = R \circ (S_1 \circ T)$ then
6:         $S \leftarrow S_2$.
7:     else
8:         $S \leftarrow S_1$.
9: $r_i \leftarrow R, s_j \leftarrow S, t_k \leftarrow T$.        ▷ Only one element exists in $R, S$ and $T$
10: return $(r_i, s_j, t_k)$.
```

Time Complexity: Denote $F(R, S, T) := (R \circ S) \circ T - R \circ (S \circ T)$. Similar with the argument presented in lecture note, if it exists r^*, s^*, t^* such that $F(r^*, s^*, t^*) \neq 0$, then we have

$$\begin{aligned} F(r^*, s^*, t^*) =& F(R_1, S_1, T_1) - F(R_1, S_1, T_0) - F(R_1, S_0, T_1) - F(R_0, S_1, T_1) \\ & F(R_1, S_0, T_0) + F(R_0, S_1, T_0) + F(R_0, S_0, T_1) - F(R_0, S_0, T_0) \end{aligned}$$

where $R_1 = R \cup \{r^*\}, R_0 = R$, etc. The first step of **Algorithm 2** terminates in $O(1)$ iterations, with a time complexity of $O(n^2)$.

The following steps halve the size of $R/S/T$ in each iteration, terminating in $O(\log n)$ iterations, resulting in a time complexity of $O(n^2 \log n)$.

The time complexity of the algorithm is $n^2 + n^2/2 + n^2/4 + \cdots = O(n^2)$.

Thus, the overall time complexity of the algorithm is $O(n^2 \log n)$.

Q.E.D. ◁

Question 4 (15') (More on Tutte Matrix). In this problem, we consider Tutte Matrix and matching in more general problems.

a. (7') Given a *bipartite graph* $G = (U, V, E)$, define its Tutte Matrix $A(G)$ as

$$A(G)_{ij} := \begin{cases} x_{ij} & \text{if } (u_i, v_j) \in E; \\ 0 & \text{otherwise.} \end{cases}$$

Show that the maximum size of a matching in G is exactly equal to the rank of $A(G)$.

b. (8') Given a *simple graph* $G = (V, E)$, define its Tutte matrix $B(G)$ as

$$B(G)_{ij} := \begin{cases} x_{ij} & \text{if } (v_i, v_j) \in E \text{ and } i < j; \\ -x_{ji} & \text{if } (v_i, v_j) \in E \text{ and } i > j; \\ 0 & \text{otherwise.} \end{cases}$$

Show that G has a perfect matching $\Leftrightarrow \det B(G) \not\equiv 0$.

[Hint: You can find some hints on the lecture notes.]

[**TA's Note:** This actually gives a $O(n^3)$ algorithm for solving general matching on a graph with n nodes. You're encouraged to find it out.]

◀

Answer. a). Notice that each possible matching $(I, J, E_{I,J})$ in G corresponds to a minor of $A(G)$, denoted as $A(G)_{I,J}$ where $I \subseteq U, J \subseteq V$ and $|I| = |J|$. Apply the similar argument in the notes to $A(G)_{I,J}$, we have

$$\det A(G)_{I,J} \not\equiv 0 \quad \text{iff} \quad (I, J, E_{I,J}) \text{ contains a perfect matching.}$$

Suppose the maximum size of a matching in G is k, then there exists a set of k rows and k columns in $A(G)$ such that the corresponding minor is non-zero, and any minor of G with size $k + 1$ is zero. Therefore, the rank of $A(G)$ is exactly equal to the maximum size of a matching in G.

b). Let $m = |V|$. Consider the matrix $B(G)$, we have

$$B(G) = \sum_{\sigma} \text{sgn}(\sigma) \prod_{i=1}^{n} B(G)_{i,\sigma(i)}. \tag{1}$$

Notice that each non-zero term in the summation (1) corresponds to a *directed cycle cover* in G, respectively. For example, suppose G has one directed cycle cover with k cycles, denote as $\{(v^i_{j_1} \to v^i_{j_2} \to \cdots \to v^i_{j_{r_i}} \to v^i_{j_1}) : i \in \{1, \cdots, k\}\}$ where the subscript j_t denotes the index of the vertex in V. Then the corresponding permutation is:

$$\text{for } i \in \{1, \cdots, k\}: \quad \sigma(j_t) = j_{t+1}, \forall t \in \{1, \cdots, r_i - 1\}, \sigma(j_{r_i}) = j_1.$$

$\underline{\text{We first prove that } G \text{ has a perfect matching } \implies \det B(G) \not\equiv 0}$. Assume the perfect matching of G is $\{(v_{2k-1}, v_{2k})\}_{k=1}^{m/2}$. Then the directed cycle cover is $\{(v_{2k-1} \to v_{2k} \to v_{2k-1})\}_{k=1}^{m/2}$, and the corresponding term is

$$\operatorname{sgn}(\sigma) \cdot \prod_{i=1}^{m/2} \left(-x_{2k-1,2k}^2\right) \not\equiv 0. \tag{2}$$

Furthermore, this term cannot be canceled by any other term beacuse any other non-zero term which corresponds to a cycle cover must contain at least one variable that is not in (2).
Therefore, $\det B(G) \not\equiv 0$.
$\underline{\text{Now we prove that } \det B(G) \not\equiv 0 \implies G \text{ has a perfect matching}}$. Consider any permutation σ_1 containing a directed cycle with *odd* length. Let σ_2 be the permutation obtained by reversing the direction of this cycle in σ_1 (keeping other cycles unchanged). Then we try to prove that **Claim: the corresponding terms will cancel each other in** $\det B(G)$.
Let the odd length cycle of σ_1 be $(v_{j_1} \to \cdots \to v_{j_g} \to v_{j_1})$ where g is odd. Notice that $\operatorname{sgn}(\sigma_1) = \operatorname{sgn}(\sigma_2)$, and beacuse $B(G)_{i,j} = -B(G)_{j,i}$, we have

$$\prod_{i=1}^{g} B(G)_{j_i,j_{i+1}} = x_{j_g,j_1} \prod_{i=1}^{g-1} x_{j_i,j_{i+1}} = -x_{j_1,j_g} \prod_{i=1}^{g-1} x_{j_{g-i+1},j_{g-i}}. = -\prod_{i=1}^{g} B(G)_{j_{g-i+1},j_{g-i}}.$$

Therefore, the **Claim** holds and any permutation contains odd length cycle will be canceled by its reverse permutation in $\det B(G)$.
Then we can conclude that the non-zero contributions in $\det B(G)$ are all from the permutations containing only *even* length cycle covers. Furthermore, we can easily construct a perfect matching from such even-length cycle cover by removing half of the edges in each cycle like the following figure:

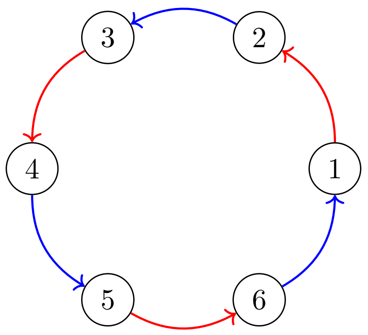

Figure 1: Construct a perfect matching by removing red edges.

Therefore, G has a perfect matching $\iff \det B(G) \not\equiv 0$.

Q.E.D. $\triangleleft$

Question 5 (20') (Checking Commutativity). Let G be a finite group. In this question we want to check if G is abelian, i.e., whether

$$x \circ y = y \circ x$$

holds for all $x, y \in G$. We assume G is presented as a set of *generators* $\{g_1, g_2, \cdots, g_k\}$ (i.e., every element of G can be written as a product of various g_i) together with a black box which returns any desired product $g_i \circ g_j$ in unit time.

There is a randomized algorithm making use of a *random product* $h = g_1^{b_1} \circ g_2^{b_2} \circ \cdots \circ g_k^{b_k}$, where $b_i \in \{0, 1\}$ are independent fair coin flips. It runs as follows

1. Let h, h' be two independent *random products*.
2. If $h \circ h' \neq h' \circ h$, then output "not abelian"; otherwise output "abelian".

Plainly this algorithm runs in $O(k)$ time, and is always correct when G is abelian. Here are the questions.

a. (10') Suppose H is any *proper* subgroup of G. For a random product h, show that

$$\Pr[h \notin H] \geq \frac{1}{2}.$$

[Hint: Construct $h' \notin H$ from any $h \in H$.]

b. (10') Show that when G is not abelian, the algorithm reports a correct answer with probability at least $1/4$.

[Hint: Consider the elements of G that commute with all elements of G.]

◀

Answer. a). $H < G$ implies that there exists $a \in G$ such that $a \notin H$. Therefore, for any random $h \in H$, $h \circ a \notin H$. And for any $h_1, h_2 \in H, h_1 \neq h_2$, it's easy to verify that $h_1 \circ a \neq h_2 \circ a$. Then for any random product h, we have

$$1 = \Pr[h \in H] + \Pr[h \notin H] \leq 2\Pr[h \notin H] \implies \Pr[h \notin H] \geq \frac{1}{2}.$$

b). Consider the center of G, denoted as $Z(G) = \{z \in G : z \circ g = g \circ z, \forall g \in G\}$. Note that $Z(G)$ is a subgroup of G. Thus, for a random product h, we have

$$\Pr[h \notin Z(G)] \geq \frac{1}{2}.$$

Given any $h_0 \notin Z(G)$, consider group(*the centralizer of a*) $U_{h_0} = \{z \in G : z \circ h_0 = h_0 \circ z\}$. It's easy to verify that U_{h_0} is a subgroup of G. Then for a random product h, we have

$$\Pr[h \notin U_{h_0}] \geq \frac{1}{2}.$$

Therefore, the probability that the algorithm reports a correct answer is

$$\Pr[h \circ h' \neq h' \circ h] \geq \Pr[h \notin Z(G)] \cdot \Pr[h' \notin U_h] \geq \frac{1}{4}.$$

Q.E.D. ◁

Question 6 (20') (New Pattern Matching). In this question we use a different fingerprinting technique for pattern matching. The idea is to map bit string s into a 2×2 matrix $M(s)$ as follows:

- For empty string ε, $M(\varepsilon) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.
- $M(0) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, M(1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$.
- For non-empty strings x, y, $M(xy) = M(x)M(y)$.

Here are the questions.

a. (10') Show that this fingerprint function has the following properties.
   1. (2') $M(x)$ is well-defined for all $x \in \{0,1\}^*$.
   2. (8') $M(x) = M(y) \implies x = y$.

b. (10') By considering the matrices modulo a suitable prime p, show an efficient randomized algorithm for pattern matching. Analyze the error bound of your algorithm.

◀

Answer. a). For **property 1**, let $x := x_1 x_2 \cdots x_n \in \{0,1\}^*$, then indection on n shows that $M(x)$ is well-defined, i.e.,

$$M(x) = M(\varepsilon) \prod_{i=1}^{n} M(x_i).$$

- **Base case:** When $n = 1$, it's easy to verify that

$$M(\varepsilon)M(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = M(1). \quad \text{Similarly for } M(0).$$

- **Inductive Hypothesis:** Assume that the statement holds for all $n \leq k$.
- **Inductive Step:** When $n = k+1$, we have

$$M(x_1 x_2 \cdots x_{k+1}) = M(x_1 x_2 \cdots x_k) M(x_{k+1}) = M(\varepsilon) \prod_{i=1}^{k} M(x_i) \cdot M(x_{k+1}).$$

  And for any $rs \in \{0,1\}^{k+1}$, if$r = \varepsilon$ or $s = \varepsilon$, then $M(rs) = M(r)M(s)$ holds by definition. Otherwise, by induction hypothesis, we have $M(rs) = M(\varepsilon) \prod_{i=1}^{k_r} M(r_i) \prod_{j=1}^{k_s} M(s_j) = M(r)M(s)$.

This completes the proof of **property 1**.

For **property 2**, we fisrt prove that for all $x \in \{0,1\}^*$, the elements of $M(x)$ are all non-negative. Note that $M(\varepsilon)$ and $M(0), M(1)$ are all non-negative. Then for all $x = x_1 x_2 \cdots x_n$, we can easily verify it by induction on n.

Then we prove that $M(x) = M(y) \implies x = y$ by induction on $|x| := n$ and $|y| := m$.

- **Base case:** When $n = 0$, for all $m \in \mathbb{N}^*$, denote $y = y_1 y_2 \cdots y_m$. If $y_1 = 0$, then

$$M(\varepsilon) = M(0)M(y_2 \cdots y_m) \implies M(y_2 \cdots y_m) = M(y) = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

which is impossible. Similarly for $y_1 = 1$. Thus, $M(\varepsilon) = M(y) \implies x = y, \forall m \in \mathbb{N}^*$ holds. Note that the case $m = 0$ is similar, i.e., $M(x) = M(\varepsilon) \implies x = y, \forall n \in \mathbb{N}^*$.

- **Inductive Hypothesis:** Assume that the statement holds for all $n \le k_1$ and $m \le k_2$.
- **Inductive Step:** When $n = k_1 + 1$ and $m = k_2 + 1$. Let $x = x_1 x_2 \cdots x_{k_1+1}$ and $y = y_1 y_2 \cdots y_{k_2+1}$. If $x_{k_1+1} = y_{k_2+1}$, then

$$M(x_1x_2\cdots x_{k_1})M(x_{k_1+1}) = M(y_1y_2\cdots y_{k_2})M(y_{k_2+1})$$
$$M(x_1x_2\cdots x_{k_1}) = M(y_1y_2\cdots y_{k_2}) \implies x_1x_2\cdots x_{k_1} = y_1y_2\cdots y_{k_2} \implies x = y.$$

Otherwise, without loss of generality, suppose $x_{k_1+1} = 0$ and $y_{k_2+1} = 1$. Let

$$M(x_1x_2\cdots x_{k_1}) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad M(y_1y_2\cdots y_{k_2}) = \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix}.$$

where $a, b, c, d, a', b', c', d'$ are all non-negative. Then we have

$$M(x) = \begin{bmatrix} a+b & b \\ c+d & d \end{bmatrix} = \begin{bmatrix} a' & a'+b' \\ c' & c'+d' \end{bmatrix} = M(y) \implies a = b' = c = d' = 0.$$

Then $\det(M(x_1x_2\cdots x_{k_1})) = 0$, which is impossible.

This completes the proof of **property 2**.

b). Like the fingerprinting technique in the lecture notes, the algorithm is as follows:
All operations are modulo p, ignore in the following algorithm for simplicity.

Algorithm 3 **Pattern Matching**

Input: A text Q of length n and a pattern P of length m.

Output: Whether P appears in Q.

```
1: Pick a random prime $p \in [2, \cdots, T]$.
2: Compute $M(P)$ and $M(Q[0:m])$.
3: for $i = 1$ to $n - m$ do
4:     $M(Q[i:i+m]) \leftarrow M^{-1}(Q[i-1]) \cdot M(Q[i-1:i+m-1]) \cdot M(Q[i+m-1])$.
5:     if $M(Q[i+1:i+m]) = M(P)$ then
6:         return True.
7:     else
8:         Continue.
9: return False.
```

Error Bound: the length of each element in the matrix is at most m. In order for a false match to occur, p must divide $(M(Q[i:i+m]) - M(P))$ for some i. Thus, p must divide $\prod_i M(Q[i:i+m]) - M(P)$. Therefore, the error bound is

$$\Pr[\text{Error}] \le \frac{\pi(mn)}{\pi(T)}.$$

Q.E.D. ◁

Question 7 (10') (Things Get Real). This question is a simplified version of a competitive programming problem, appeared in *CCPC 2022 Final*, where *CCPC* is short for *China Collegiate Programming Contest*. **You only need to give a solution, without proving it.**

Little Cyan Fish has devised a new game:

- The game is played with a string $s = s_1 s_2 \cdots s_n$ composed of the digits 0,1 and 2 .
- The game proceeds as follows:
  1. The player selects an index $i (1 \le i < n)$ such that $s_i = s_{i+1}$. Notably, if no such i exists, the game ends immediately.
  2. The characters s_i and s_{i+1} are then removed, and the player returns to step 1 .
- If the player can completely erase the string (leaving an empty string), he wins the game. If not, he loses.

Keen to challenge Little Cyan Fish, Big Flower Letter presents him with a string $s = s_1 s_2 \cdots s_n$ of length n. She then proposes q queries of the following type:

- l, r : Assuming $t = s[l \cdots r] = s_l s_{l+1} \cdots s_{r-1} s_r$, Little Cyan Fish needs to determine if the player can win the game if they were to play it using the string t.

Can you help Little Cyan Fish respond to all the queries?

Your algorithm should have time complexity of $O((n + q) \log n)$.

[**TA's Note:** You can assume that, if you can give the right answer to all queries with probability $\ge .9$, you will get *Accepted* in this question.]

Here is an example:

Let $n = 8, s = 01221220$.

The answer to query $1, 8$ is *Yes*, since $01221220 \to 011220 \to 0220 \to 00 \to \emptyset$.

The answer to query $2, 5$ is *Yes*, since $1221 \to 11 \to \emptyset$.

The answer to query $3, 7$ is *No*, since $22122 \to 122 \to 1$ is not an empty string.

The answer to query $5, 8$ is *No*, since $1220 \to 10$ is not an empty string. ◀

Answer. Notice that the matrix multiplication is associative but not commutative, then we can convert the problem to a matrix multiplication problem.

Get three matrices $A, B, C \in M^{2\times 2}(\mathbb{R})$ which satisfies (1) $A \ne B \ne C$, (2) $A^2 = B^2 = C^2 = I$ firstly. Then replace $0, 1, 2$ with A, B, C respectively, and the problem is equivalent to determine whether the matrix product is the identity matrix. The detailed algorithm is as follows:

Algorithm 4 Little Cyan Fish Game

Input: matrix $A, B, C \in M^{2\times 2}(\mathbb{R})$, queries $\{l, r\}_q$, string $s = s_1 s_2 \cdots s_n$.

Output: Whether the player can win the game

```
1: $M(0) \leftarrow A, M(1) \leftarrow B, M(2) \leftarrow C$.
```

```
 2: $P[0] \leftarrow I$.                 ▷ Initialize the prefix product array with the identity matrix
 3: for $i = 1$ to $n$ do
 4:     $P[i] \leftarrow P[i-1] \cdot M(s_i)$.
 5: for $i = 1$ to $q$ do
 6:     $l, r \leftarrow \{l, r\}_i$.
 7:     if $P[r] = P[l-1]$ then          ▷ If the matrix product between $l$ and $r$ is the identity matrix
 8:         return Yes.
 9:     else
10:         return No.
```

Q.E.D. ◁