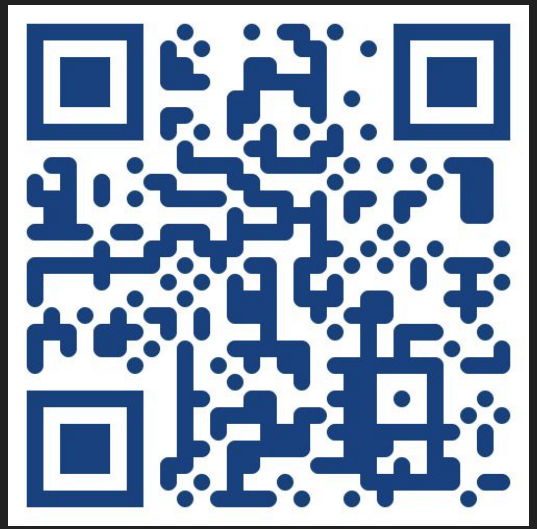
Algorithms and Probability

Exercise Session 8



https://n.ethz.ch/~ahmala/anw

MINITEST

Time limit: 11 minutes

Number of questions: 11

Threshold to get 1 point: 7 correct answers

Threshold to get 2 points: 9 correct answers

Do Not Rush!

PASSWORD: *******

PASSWORD: capital

Monte Carlo Algorithms: One Sided Errors

We say that a Monte Carlo algorithm A for a decision problem has a one-sided error if and only if, for some $\varepsilon > 0$:

$\Pr[A(I) = YES] = 1$ if the correct answer for input I is YES, and

$\Pr[A(I) = NO] \geq \varepsilon$ if the correct answer for input I is NO

for any input I.

Primality Tests I

```
1  Choose 1 ≤ a < n uniformly at random
2  if gcd(a, n) ≠ 1
3      return "not prime"
4  else
5      return "prime"
```

Is this one sided error?

Pr[A(I) = “not prime”] = ?

Pr[A(I) = “prime”] = ?

Primality Tests I

```
1  Choose 1 ≤ a < n uniformly at random
2  if gcd(a, n) ≠ 1
3      return "not prime"
4  else
5      return "prime"
```

Is this one sided error? Yes

Pr[A(I) = “not prime”] = 1

Pr[A(I) = “prime”] >= ε

Primality Tests I

```
1  Choose 1 ≤ a < n uniformly at random
2  if gcd(a, n) ≠ 1
3      return "not prime"
4  else
5      return "prime"
```

Is this one sided error? Yes

Pr[A(I) = “not prime”] = 1

Pr[A(I) = “prime”] >= ε

Can you increase the chance of success to a higher probability using only this algorithm?

Primality Tests I

```
1  Choose 1 ≤ a < n uniformly at random
2  if gcd(a, n) ≠ 1
3      return "not prime"
4  else
5      return "prime"
```

Is this one sided error? Yes

Pr[A(I) = “not prime”] = 1

Pr[A(I) = “prime”] >= ε

Can you increase the chance of success to a higher probability using only this algorithm?

Just run it multiple times!

Primality Tests I

```
1  Choose 1 ≤ a < n uniformly at random
2  if gcd(a, n) ≠ 1
3      return "not prime"
4  else
5      return "prime"
```

Is this one sided error? Yes

Pr[A(I) = “not prime”] = 1

Pr[A(I) = “prime”] >= ε

Can you increase the chance of success to a higher probability using only this algorithm?

How many times to run?

Primality Tests I

```java
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;

public class prime {
    public static void main(String[] args) {
        Random rng = new Random(System.currentTimeMillis());

        int N = 100000; // 10^5
        int number_of_trials = 1;
        int x = 99899; // 283*353


        for (int i = 0; i < number_of_trials; ++i) {
            int random_number = chooseRandom(rng, N);
            if (gcd(random_number, x) != 1) {
                System.out.println("Not Prime");
                return;
            }
        }

        System.out.println("Prime");
    }

    // choose a random integer in [1, N];
    public static int chooseRandom(Random rng, int N) {
        return rng.nextInt(N - 1) + 1;
    }

    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }
}
```

We want to check if x=99899 is a prime number.

99899 = 283*353

How many times should we run the algorithm

```
1  Choose 1 ≤ a < n uniformly at random
2  if gcd(a, n) ≠ 1
3      return "not prime"
4  else
5      return "prime"
```

Primality Tests I

Pr[gcd(random_number, x) != 1)
= Pr[random_number being a multiple of 283)
+ Pr[random_number being a multiple of 353)
 - Pr[random_number being a multiple of 283*353)

```java
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;

public class prime {
    public static void main(String[] args) {
        Random rng = new Random(System.currentTimeMillis());

        int N = 100000; // 10^5
        int number_of_trials = 1;
        int x = 99899; // 283*353


        for (int i = 0; i < number_of_trials; ++i) {
            int random_number = chooseRandom(rng, N);
            if (gcd(random_number, x) != 1) {
                System.out.println("Not Prime");
                return;
            }
        }

        System.out.println("Prime");
    }

    // choose a random integer in [1, N];
    public static int chooseRandom(Random rng, int N) {
        return rng.nextInt(N - 1) + 1;
    }

    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }
}

```

Primality Tests I

Pr[gcd(random_number, x) != 1)
= Pr[random_number being a multiple of 283)
+ Pr[random_number being a multiple of 353)
 - Pr[random_number being a multiple of 283*353)

Pr[gcd(random_number, x) != 1)
= (353+283-1) / 100000 = 0.00635

```java
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;

public class prime {
    public static void main(String[] args) {
        Random rng = new Random(System.currentTimeMillis());

        int N = 100000; // 10^5
        int number_of_trials = 1;
        int x = 99899; // 283*353


        for (int i = 0; i < number_of_trials; ++i) {
            int random_number = chooseRandom(rng, N);
            if (gcd(random_number, x) != 1) {
                System.out.println("Not Prime");
                return;
            }
        }

        System.out.println("Prime");
    }

    // choose a random integer in [1, N];
    public static int chooseRandom(Random rng, int N) {
        return rng.nextInt(N - 1) + 1;
    }

    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }
}

```

Primality Tests I

Pr[gcd(random_number, x) != 1)
= Pr[random_number being a multiple of 283)
+ Pr[random_number being a multiple of 353)
 - Pr[random_number being a multiple of 283*353)

Pr[gcd(random_number, x) != 1)
= (353+283-1) / 100000 = 0.00635

With probability 1 - 0.00635 = 0.99365 the algorithm fails.

```java
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;

public class prime {
    public static void main(String[] args) {
        Random rng = new Random(System.currentTimeMillis());

        int N = 100000; // 10^5
        int number_of_trials = 1;
        int x = 99899; // 283*353


        for (int i = 0; i < number_of_trials; ++i) {
            int random_number = chooseRandom(rng, N);
            if (gcd(random_number, x) != 1) {
                System.out.println("Not Prime");
                return;
            }
        }

        System.out.println("Prime");
    }

    // choose a random integer in [1, N];
    public static int chooseRandom(Random rng, int N) {
        return rng.nextInt(N - 1) + 1;
    }

    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }
}

```

Primality Tests I

Pr[gcd(random_number, x) != 1)
= Pr[random_number being a multiple of 283)
+ Pr[random_number being a multiple of 353)
 - Pr[random_number being a multiple of 283*353)

Pr[gcd(random_number, x) != 1)
= (353+283-1) / 100000 = 0.00635

With probability 1 - 0.00635 = 0.99365 the algorithm fails.

If run 100 times, you get 0.99365^100 = 0.53 failure rate.

If run 1000 times, you get 0.99365^1000 = 0.002 failure rate.

```java
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;

public class prime {
    public static void main(String[] args) {
        Random rng = new Random(System.currentTimeMillis());

        int N = 100000; // 10^5
        int number_of_trials = 1;
        int x = 99899; // 283*353


        for (int i = 0; i < number_of_trials; ++i) {
            int random_number = chooseRandom(rng, N);
            if (gcd(random_number, x) != 1) {
                System.out.println("Not Prime");
                return;
            }
        }

        System.out.println("Prime");
    }

    // choose a random integer in [1, N];
    public static int chooseRandom(Random rng, int N) {
        return rng.nextInt(N - 1) + 1;
    }

    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }
}

```

In Class Exercise

https://n.ethz.ch/~ahmala/anw/material/Broken_Servers.pdf

You are in a network of n servers, numbered from 1 to n.

Every server can be contacted in O(1).

If the server is damaged, sends an i.i.d. bit

If the server is undamaged, sends always the same bit

a) Let $\delta > 0$. Describe a Monte Carlo Algorithm finding out if server i is damaged. Make sure Error Probability is $<= \delta/N$

b)

- If a server is damaged, does the algorithm from a) find it?
- If a server is undamaged, does the algorithm from b) find it?
- When the algorithm returns “Server is damaged”, is it always correct?
- When the algorithm returns “Server is undamaged”, is it always correct?

c)

- Given δ, describe a Monte Carlo Algorithm returning the list of damaged servers with error probability <= δ.

Fermat

$$n \text{ is prime} \implies \forall a \in \{1, \ldots, n-1\} : a^{n-1} \equiv_n 1$$

Example:

6 ^ 12 = 1 mod 13

4 ^ 4 = 1 mod 5

Fermat

$$n \text{ is prime} \implies \forall a \in \{1, \ldots, n-1\} : a^{n-1} \equiv_n 1$$

Example:

6 ^ 12 = 1 mod 13

4 ^ 4 = 1 mod 5

Primality Tests II - Using Fermat

```
1  Choose 1 < a < n uniformly at random
2  if  a^{n-1} ≢_n 1
3      return "not prime"
4  else
5      return "prime"
```

Primality Tests II - Using Fermat

```
1 Choose 1 < a < n uniformly at random
2 if a^{n-1} ≢_n 1
3     return "not prime"
4 else
5     return "prime"
```

Does it have one-sided error?

Primality Tests II - Using Fermat (success prob of min 0.5)

```
1 Choose 1 < a < n uniformly at random
2 if a^{n-1} ≢_n 1
3     return "not prime"
4 else
5     return "prime"
```

Let’s consider n = 561 = 3*11*17
4^560 = 1 mod 561
7^560 = 1 mod 561
For all b relatively prime to n, we get b ^ {n-1} = 1 mod n.

Primality Tests III - Miller Rabin (success prob of min 0.75)

n is prime $\Rightarrow$ the equation (x^2 = 1 mod n) has exactly the solutions 1 and $n - 1$

```
1  Write n − 1 as 2^k · d with d odd
2  Choose 1 < a < n uniformly at random
3  if a^d ≢_n 1
4      for i = 1, \dots, k do
5          if a^(2^i · d) ≡_n n − 1
6              return "prime"
7      return "not prime"
8  else
9      return "prime"
```

Hard DP+Probability Problem

https://n.ethz.ch/~ahmala/anw/tasks/Taking%20Balls.pdf