Lecture 31: Miller–Rabin Test
Recall

- In the previous lecture we considered an efficient randomized algorithm to generate prime numbers that need $n$-bits in their binary representation.
- This algorithm sampled a random element in the range $\{2^{n-1}, 2^{n-1} + 1, \ldots, 2^n - 1\}$ and test whether it is a prime number or not.
- By the Prime Number Theorem, we are extremely likely to hit a prime number.
- So, all that remains is an algorithm to test whether the random sample we have chosen is a prime number or not.
Primality Testing

- Given an \( n \)-bit number \( N \) as input, we have to ascertain whether \( N \) is a prime number or not in time polynomial in \( n \).
- Only in 2002, Agrawal–Kayal–Saxena constructed a deterministic polynomial time algorithm for primality testing. That is, the algorithm will always run in time polynomial in \( n \). For any input \( N \) (that has \( n \)-bits in its binary representation), if \( N \) is a prime number, the AKS primality testing algorithm will return 1; otherwise (if, the number \( N \) is a composite number), the AKS primality testing algorithm will return 0. In practice, this algorithm is not used for primality testing because this turns out to be too slow.
- In practice, we use a randomized algorithm, namely, the Miller–Rabin Test, that successfully distinguishes primes from composites with very high probability. In this lecture, we will study a basic version of this Miller–Rabin primality test.
Miller–Rabin outputs 1 to indicate that it has classified the input $N$ as a prime. If Miller–Rabin outputs 0, then it indicates that it has classified $N$ as a composite number.

<table>
<thead>
<tr>
<th>$N$ is</th>
<th>Miller–Rabin outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime</td>
<td>1 with probability 1</td>
</tr>
<tr>
<td></td>
<td>0 with probability 0</td>
</tr>
<tr>
<td>Composite</td>
<td>1 with probability $\leq 2^{-t}$</td>
</tr>
<tr>
<td></td>
<td>0 with probability $\geq 1 - 2^{-t}$</td>
</tr>
</tbody>
</table>

So, if $N$ is a prime, then Miller–Rabin algorithm is always correct. On the other hand, if $N$ is a composite number, then Miller–Rabin algorithm correctly classifies it as a composite number with probability $\geq (1 - 2^{-t})$, where $t$ is an input it takes. Intuitively, the Miller–Rabin only sometimes incorrectly classifies composite numbers as primes numbers.
In today’s lecture, we shall cover a basic form of Miller–Rabin primality testing algorithm.

This algorithm mimics the performance of the actual test on all inputs except a small set of bad composite numbers, namely, Carmichael Numbers. On all other inputs, it replicates the performance of the actual Miller–Rabin Test.

For example, our basic algorithm will correctly identify prime number with probability 1. Moreover, for any composite number that is not a Carmichael number, it will correctly classify it as a composite number with probability \( \geq (1 - 2^{-t}) \).

Our basic algorithm goes horribly wrong if the input \( N \) is a Carmichael number. It will incorrectly classify \( N \) as a prime number with probability 1.
Intuition Underlying the Basic Construction.

- Note that if \( N \) is a prime then we know that \( a^{p-1} = 1 \mod p \), for all \( a \in \{1, 2, \ldots, p - 1\} \)

- (We shall state this next statement without a proof) If \( N \) is a composite number that is not a Carmichael number, then at least half the elements \( a \in \{1, 2, \ldots, N - 1\} \) have the property that \( a^{N-1} \neq 1 \mod N \)

- So, if we pick a random \( a \in \{1, 2, \ldots, N - 1\} \) and compute \( a^{N-1} \mod N \), then
  1. If \( N \) is a prime, then it is always 1
  2. If \( N \) is a composite that is not a Carmichael number, then it is \( \neq 1 \) with probability at least 1/2
Basic Miller–Rabin Primality Testing

IsPrime($N, t$):

1. For $i = 1$ to $t$:
   1. Sample $a \leftarrow \{1, 2, \ldots, N - 1\}$
   2. If $a^{N-1} \mod N \neq 1$: Return 0

2. Return 1
Analysis.

- Suppose $N$ is a prime number. Then the test $a^{N-1} \mod N = 1$, for all $a \in \{1, 2, \ldots, N - 1\}$. Hence, the output of the algorithm is 1.

- Suppose $N$ is a composite number that is not a Carmichael number. Then, with probability $\geq 1/2$, the inner loop samples $a$ such that $a^{N-1} \mod N \neq 1$. So, the inner loop does not return 0, with probability $\leq 1/2$. Any one of the $t$-runs of the inner loop does not return 0, with probability $\leq 2^{-t}$. Hence, the probability that the basic test returns 1 (i.e., the algorithm incorrectly classifies a composite $N$ as a prime number) is $\leq 2^{-t}$.
Carmichael Numbers

- Carmichael numbers are composite numbers for which our basic algorithm fails
- There are infinitely many Carmichael numbers (otherwise, our basic algorithm could have simply checked whether $N$ lies in this finite list of Carmichael numbers)

**Definition (Carmichael Number)**

The composite number $N$ is a Carmichael number if $a^{N-1} \equiv 1 \mod N$, for all $a \in \{1, 2, \ldots, N - 1\}$

- $C(X)$ represents the number of Carmichael number $< X$. Erdös proved an upper-bound on $C(X)$

\[
C(X) \leq \frac{X}{\exp \left( x^{\frac{\lambda \log \log x}{\log x}} \right)},
\]

where $N = 2^x$ and $\lambda > 0$ is a constant.

- So, it is highly unlikely that a random number generated from the set $\{1, \ldots, 2^x - 1\}$ is a Carmichael number
Recall that our basic algorithm is incorrect only for Carmichael numbers.

We saw that Carmichael numbers are very rare.

So, when the input to our basic Rabin–Miller primality testing algorithm is chosen uniformly at random, then it works correctly with high probability.

The actual Rabin–Miller primality testing algorithm will not be covered in this course. Interested students are encouraged to read online resources on this algorithm.
A worked out example for $N = 15$

- For $a = 1, 2, 3, \ldots, 14$, we write down the values of $a^{N-1} \mod N$
  
  $\quad 1, 4, 9, 1, 10, 6, 4, 4, 6, 10, 1, 9, 4, 1$

- Only $a \in \{1, 4, 11, 14\}$ have $a^{N-1} = \mod N$

- 10-out-of-14 elements in $\{1, 2, \ldots, 14\}$ have $a^{N-1} \neq \mod N$