Fermat's Primality Test

The Algorithm

Sample Algorithms

```
function Fermat(N):
    // N is an n-bit number
    a = random n-bit number < N
    z = mod-exponentiation(a, N-1, N)

    if z == 1:
        return True
    return False
```

Correct?

Run time?
Primalty 1

Is $N$ Fermat 1 prime?

$N = 24$
$a = 7$

$N = 29$
$a = 14$
\[ z = a^{N-1} \mod N \]

From algorithm,

Assume \( N \) is prime.

Define \( S = \{ 1, 2, 3, \ldots, N-1 \} \)

Define \( S' = \{ a \cdot 1, a \cdot 2, a \cdot 3, \ldots, a \cdot (N-1) \} \mod N \)

Claim \( S = S' \)

Each number in \( S \) is distinct and \( \not= 0 \).

Two numbers from \( S \) \( (i, j) \) map to \( (i' = a \cdot i, j' = a \cdot j) \) in \( S' \).

If \( i' = j' \), then \( a \cdot i = a \cdot j \mod N \)

Since \( N \) is prime, \( a \) and \( N \) are relatively prime, \( a^i \) exists.

\[ a \cdot i = a \cdot j \Rightarrow i \equiv j \mod N \]

So, if \( i' = j' \), then \( i = j \), but \( i \not= j \), so \( i' \not= j' \).

\( \Rightarrow \) All elements of \( S' \) are distinct.

Because no element of \( S \) is 0, and \( a \not= 0 \), no element of \( S' \) is 0.

\[ \Rightarrow S = S' \]
Fermat's Primality Test

\[ S = \sum_{i=1}^{N-1} a^i \]

\[ S' = \sum_{i=1}^{N-1} a^i \equiv a^{N-1} \pmod{N} \]

\[ S = S' \]

\[ P = \prod_{i \in S} i = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot N-1 = (N-1)! \]

\[ P' = \prod_{i \in S'} i' = a^1 \cdot a^2 \cdot a^3 \cdot \ldots \cdot a^{N-1} = a^{N-1} (N-1)! \pmod{N} \]

Since \( S = S' \), \( P = P' \)

\[ (N-1)! \equiv a^{N-1} (N-1)! \pmod{N} \]

Divide by \((N-1)!\)

\[ 1 \equiv a^{N-1} \pmod{N} \]

This is the test in the algorithm!

\[ \Rightarrow \text{Fermat's Little Theorem} \]
Fermat's Primality Test  correctness

Correctness assumes $N$ is prime.

What if $N$ is not prime?

Sometimes non-prime numbers still report True.
Fermat's Primality Test  Runtime

Generate n-bit number : $O(n)$
Modular Exponentiation : $O(n^3)$
Test for 1 : $O(n)$
Subtract 1 from N : $O(n)$

$O(n^3)$
function Fermat2(N, k):  // N is an n-bit number
    // k is a number.
    for i in 1..k:
        a = random n-bit number 0 < a < N
        z = mod-exponentiation(a, N-1, N)
        if z ≠ 1:
            return False
    return True

Correct?

Runtime?
Fermat's Primality Test #2

- The code inside the loop is just Fermat's theorem. A False means not prime. A True means maybe prime. Let's quantify "maybe".

- Hand waving: Except for some rare composite (non-prime) numbers, called Carmichael numbers, all composite numbers have at least 1 value of a \( \alpha \) s.t. \( \alpha^{N-1} \not\equiv 1 \) (mod N).

- Assume \( N \) is not prime, and is not Carmichael.

- \( \alpha^{N-1} \not\equiv 1 \) (mod N), \( \alpha \) exists.

- If no \( b \) exists s.t. \( b^{N-1} \equiv 1 \) (mod N), the Fermat primality test will return False.

- If \( b \) does exist s.t. \( b^{N-1} \equiv 1 \) (mod N), then

  \[
  (a \cdot b)^{N-1} = a^{N-1} b^{N-1} = a^{N-1} \not\equiv 1 \pmod{N}
  \]

  \Rightarrow \text{every possible } b \text{ produces an } a \cdot b, \text{ where } (a \cdot b)^{N-1} \not\equiv 1 \pmod{N}

  \Rightarrow \text{there are at least as many numbers that } \not\equiv 1 \text{ as do}

  \Rightarrow \text{Prob(Fermat's test says composite number is prime) } \leq \frac{1}{2}
Fermat's Primality Test #2  

- How likely is Fermat2 to give "prime" label to a composite?

$$\Pr(\text{incorrect}) \leq \left(\frac{1}{2}\right)^k = \frac{1}{2^k}$$
Fermat's Primality Test #2

- for loop repeats $O(k)$
  - random number $O(n)$
  - modular exponentiation $O(n^3)$
  - $z \neq 1$ $O(n)$

$\Rightarrow O(k) \cdot [O(n) + O(n^3) + O(n)] = O(kn^3)$

We choose desired confidence vs. desired runtime with $k$. 
function generate_prime(n, k):  // create a prime n-bit number, 
    while prime not found:
        N = random n-bit number
        if Fermat2(N, k):
            return N

Correct?

Runtime?
Generate Random Prime  Correctness

- Fermat\(^2\) is correct with error \(\leq \frac{1}{2^k}\)
- Generate \( N \) : \( O(n) \) \( \Rightarrow \) \( O(kn^3) \) inside loop.
- \( \text{FermatZ}(N,k) \) : \( O(kn^3) \)
- While loop repetition?

According to Lagrange's prime number theorem! (sorry, no proof)

\[
\pi(x) \approx \frac{x}{\ln(x)} = \text{number of primes } \leq x.
\]

Probability of number being prime = \[
\frac{\pi(x)}{x} \approx \frac{x/\ln(x)}{x} = \frac{1}{\ln(x)} \approx \frac{1}{\ln(2)}
\]

= \[
\frac{1}{n \cdot \ln(2)} \approx \frac{1.44}{n}
\]

or, about 1 in \( n \) chance of being prime for \( n \)-bit number.

\( \Rightarrow \) on average \( O(n) \) repetitions.

\( \Rightarrow \) total = \( O(kn^4) \).