BTP Thesis Part-1

Primolity Test :

A primolity test is a function that determine if a given integer greater than 1 is prime or composite.

Primolity Test

Probabilistic Primolity tests (also called non-deterministic Primolity test)

deterministic Primality tests

Deterministic Primality test:

If we put an integer n in the primality test, then the output will be "yes" if the integer is prime, "no" if integer is not prime (composite integer)

Non-deterministic Primality test:

This test takes an integer n and returns "no" if n is not prime, or returns "may be a prime".

A non-deterministic primality test also called probabilistic primality test returns either that the inputted integer is not a prime or that is "probably a prime" to some given degree of likelihood.

Primality test Algorithms:

(1) Sieve of Erastosthenes Primality test:

This is the simplest Algo. to test whether a given integer is prime or not. It also finds the prime numbers less than or equal to given integer n.

~~(1) Input: $n \in \mathbb{N}$~~

~~(2) $a[1 \ldots n]$ = integer array~~

~~(3)~~

To check Prime

Algorithm: (Input $n \in N$)

natural numbers

① $a[1 -- n]$ = Integer array

② for $j = 1$ to n do
{

③ $a[j] \leftarrow j$ }

④ $i \leftarrow 2$

⑤ while $i^2 < n$ do
if $a[i] \neq 0$ then
$t \leftarrow 2 \cdot i$
while ($t \leq n$) do
$a[t] \leftarrow 0$
$t \leftarrow t + i$

$i = i + 1$

⑥ for $j = 2$ to n do
if $a[j] \neq 0$ then return $a[j]$ = prime

Ex: $n = 10$

primes ≤ 10 (Sieve of erastosthenes)

Algo:

$\cancel{1}\quad 2\quad 3\quad \cancel{4}\quad 5\quad \cancel{6}\quad 7\quad \cancel{8}\quad 9\quad \cancel{10}$

$\cancel{1}\quad 2\quad 3\quad \cancel{4}\quad 5\quad \cancel{6}\quad 7\quad \cancel{8}\quad \cancel{9}\quad \cancel{10}$

$\cancel{1}\quad 2\quad 3\quad \cancel{4}\quad 5\quad \cancel{6}\quad 7\quad \cancel{8}\quad \cancel{9}\quad \cancel{10}$

$\Downarrow$

2, 3, 5, 7 are primes

$\Rightarrow$ 10 not a prime

Complexity Analysis:

Sieve method has $O(10^{\log_{10} n})$ operations to prove primality is n has $\log_{10} n$ digits

We can say this algorithm is of exponential time in terms of input length.

② Trial division Algorithm:

Algorithm:

For $k = 2, 3, 4 \,--\, \sqrt{n}$

Test if (for any k)

$n \equiv 0 \pmod{k}$ output composite

else

return n = prime

Complexity Analysis:

Trial division runs in time $O(\sqrt{n}\,\log^2(n))$, but this running time is exponential in the input size since the input represents n as binary number with $\lceil \log_2(n) \rceil$ digits.

---

③ Wilson's characterization of primes:

Wilson's theorem:

A natural number $n > 1$ is a prime number if and only if the product of all positive integers less than n is one less than a multiple of n

That is
~~That is~~ (using the notations of modular Arithmetic), the factorial $(n-1)! = 1 \times 2 \times 3 \cdots \times (n-1)$ satisfies:

$$\boxed{(n-1)! \equiv -1 \pmod{n}} \quad \text{OR} \quad \boxed{(n-1)! + 1 \bmod n = 0}$$

implies n is prime

Ex:

For $n = 2$

~~$(n-1)! = 2$~~
~~$-1 \bmod 2 = 2 - 1 = 1$~~

$(n-1)! = (2-1)! = 1$ LHS
$-1 \bmod 2 = 2 - 1 = 1$ RHS
LHS = RHS., Hence $n = 2$ prime

$n = 13$

$(n-1)! = (13-1)! = 12!$
$= 479001600$

$(n-1)! + 1 = 479001600 + 1 = 479001601$
since $479001601 \bmod 13 = 0$ ✓

$\Rightarrow$ 13 prime

Complexity Analysis:

From the Wilson's characterization of primes we can determine the primality of an integer n by calculating $(n-1)! \pmod{n}$.

But the computation requires $(n-1)$ multiplications, making it very time consuming.

---

(4) Euler test:

Euler test is based on simple lemma:

Lemma: n is prime iff $\phi(n) = n-1$

Proof: If n is prime number, then every integer less than n is relatively prime to it, hence by definition $\phi(n) = n-1$

Conversely, if $n > 1$ = composite, then n has a divisor d such that $1 < d < n$. It follows that there are atleast 2 integers among $1, 2 -- n$ which are not relatively prime to n, namely d & n itself. As a result, $\phi(n) \leq n-2$. This proves the lemma

Algorithm (Euler test):

Input: $n \in N$

check $\phi(n)$;

if $\phi(n) = n-1$

then output "prime"

else

output "composite"

Complexity Analysis:

From the euler test, we can determine the primality of integer n by calculating $\phi(n)$.

But for calculating $\phi(n)$, we require the factors of n and factorization is more difficult problem than primality test.