Analysis of Willans' Formula Algorithms

Willans-Style Formula in Terms of the Detector Function

Let $D(j)$ denote any correct primality detector:

$$D(j) = \begin{cases} 1, & j \text{ is prime,} \\ 0, & \text{otherwise.} \end{cases}$$

Using this detector, the prime-counting function is given by

$$\pi(m) = \sum_{j=1}^{m} D(j).$$

However, for the purposes of the computational Willans formulation, we may eliminate explicit reference to $\pi(m)$ and express everything solely in terms of the detector function.

Let $B(n)$ be any upper bound satisfying $B(n) \geq p_n$. In the classical Willans formulation, one may choose $B(n) = 2^n$, while in the optimized variant we may take

$$B(n) = n(\log n + \log\log n) + 3.$$

The Willans-style expression for the n-th prime is then

$$p_n = 1 + \sum_{m=1}^{B(n)} \left\{ 1 + \sum_{j=1}^{m} D(j) \ \leq \ n \right\}.$$

Because the inner sum

$$\sum_{j=1}^{m} D(j)$$

counts exactly the number of primes up to m, the indicator evaluates to 1 precisely while fewer than n primes have been encountered. Once the sum reaches n, all subsequent terms of the outer summation contribute zero, ensuring that the total equals $p_n - 1$.

Thus the expression above yields the exact value of the n-th prime for any choice of primality detector $D(j)$. The correctness is detector-independent; only the computational complexity depends on whether one uses Wilson's congruence, Trial Division, Miller–Rabin, or any other suitable primality test.

1 Original Algorithm: Wilson's Theorem

The original formulation by C. P. Willans utilizes Wilson's Theorem to detect prime numbers. The prime detector $D(j)$ is defined as:

$$D_{wilson}(j) = \left\lfloor \cos^2 \pi \frac{(j-1)!+1}{j} \right\rfloor \tag{1}$$

Complexity Derivation

The computational cost is dominated entirely by the factorial term $(j-1)!$.

1. **Detector Cost:** Calculating $(j-1)!$ involves $j-2$ multiplications. However, the magnitude of the number grows super-exponentially. The bit-complexity of computing $j!$ is roughly $O((j \log j)^2)$. For simplicity, we denote the complexity of the detector as $O(j!)$.
2. **Total Complexity:** Willans' formula sums from $i = 1$ to 2^n. The nested structure implies that finding the n-th prime requires evaluating the detector for integers up to 2^n.

$$T_{orig}(n) \approx \sum_{i=1}^{2^n} (i!) \approx O((2^n)!)$$

Using Stirling's approximation, this is **Super-Exponential**.

2 Modified Algorithm: Trial Division

Mathematical Motivation for the Trial Division Detector

Trial Division relies on one of the most fundamental structural properties of integers: every composite number n can be expressed as a product of two positive integers,

$$n = ab,$$

where without loss of generality $a \leq b$. This immediately implies

$$a \leq \sqrt{n}.$$

Thus, if n is composite, it must have a divisor *no greater than* $\sqrt{n}$. Conversely, if no integer d in the range

$$2 \leq d \leq \sqrt{n}$$

divides n, then n cannot be composite and must therefore be prime.

This leads to the classical characterization:

$$n \text{ is prime} \iff n > 1 \quad \text{and} \quad \nexists d \in [2, \sqrt{n}] : d \mid n.$$

Formal Definition of the Trial Division Detector

The corresponding primality indicator function is:

$$D_{\text{trial}}(n) = \begin{cases} 0, & n \leq 1, \\ 1, & n = 2, \\ 1, & n > 2 \text{ and } \forall d \in \{2, 3, 4, \ldots, \lfloor \sqrt{n} \rfloor\},\ d \nmid n, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, $D_{\text{trial}}(n)$ returns 1 precisely when n has no divisors in the interval $[2, \sqrt{n}]$, and returns 0 otherwise. This detector is exact, deterministic, and significantly more efficient than the Wilson-based detector used in the original formulation.

We replace the factorial-based detector with a standard Trial Division algorithm. The detector $D(j)$ returns 1 if j is prime and 0 otherwise.

Complexity Derivation

1. **Detector Cost:** To determine if j is prime, we check divisibility by integers up to $\sqrt{j}$.

$$\text{Cost}(D_{trial}(j)) = O(\sqrt{j})$$

2. **Prime Counter Cost ($\pi(i)$):** To find the count of primes up to i, we sum the detector cost for all $k \leq i$:

$$\text{Cost}(\pi(i)) = \sum_{k=1}^{i} \sqrt{k} \approx \int_{1}^{i} x^{0.5} dx \approx \frac{2}{3} i^{1.5} = O(i^{1.5})$$

3. **Total Complexity:** The master formula iterates i from 1 to 2^n. We sum the cost of the prime counter for each step:

$$T_{mod}(n) \approx \sum_{i=1}^{2^n} i^{1.5} \approx \int_{1}^{2^n} x^{1.5} dx$$

$$T_{mod}(n) \approx \left[\frac{x^{2.5}}{2.5} \right]_{1}^{2^n} \approx (2^n)^{2.5} = 2^{2.5n}$$

This results in **Exponential Time** complexity $O(2^{2.5n})$, which is significantly more efficient than factorial time.

3 Side-by-Side Comparison

The table below compares the theoretical time complexity and practical feasibility of both algorithms.

Parameter	**Original (Wilson's)**	**Modified (Trial Division)**
Detector Logic	$(j-1)! \equiv -1 \pmod{j}$	$j \pmod{k} \neq 0, \forall k \leq \sqrt{j}$
Cost per Integer	$O(j!)$	$O(\sqrt{j})$
Total Complexity	$\boldsymbol{O((2^n)!)}$	$\boldsymbol{O(2^{2.5n})}$
Complexity Class	Super-Exponential	Exponential
Feasibility ($n = 10$)	**Impossible** (Requires 1023!)	**Instant** (< 1 ms)
Feasibility ($n = 50$)	Impossible	Impossible (Too slow)

Table 1: Algorithm Performance Comparison