Project Euler Problem 248: Numbers with Euler Totient 13!

Aditya Bhimalingam

Problem 248. (Numbers with Euler Totient 13!)
Find the 150,000th positive integer n such that $\phi(n) = 13!$, where ϕ is Euler's totient function.
Source: `https://projecteuler.net/problem=248`

Restatement

We are asked to compute the 150,000th integer n with $\phi(n) = 13!$. Direct enumeration is infeasible due to the magnitude of $13! = 6{,}227{,}020{,}800$. We therefore exploit **number-theoretic structure** and recursive generation based on divisors and prime powers.

Background

Euler's totient function $\phi(n)$ is defined as the number of integers $1 \leq k \leq n$ that are coprime to n.

Theorem 0.1 (Euler's Product Formula)**.** *If $n = \prod_{i=1}^{r} p_i^{\alpha_i}$ is the prime factorization of n, then*

$$\phi(n) = \prod_{i=1}^{r} p_i^{\alpha_i - 1}(p_i - 1).$$

Proof. For each prime power $p_i^{\alpha_i}$, the numbers not coprime to $p_i^{\alpha_i}$ are multiples of p_i: exactly $p_i^{\alpha_i - 1}$ of them. By multiplicativity of ϕ over coprime integers, the formula follows. □

Lemma 0.2 (Inverse Totient Problem)**.** *Given m, find all positive integers n such that $\phi(n) = m$. This is called the inverse totient problem.*

Key Insight: Prime-Power Decomposition

Lemma 0.3 (Prime-Power Construction)**.** *Let m be a positive integer and $d \mid m$. If $p = d + 1$ is prime, then there exists $k \geq 1$ such that*

$$\phi(p^k) = d \cdot p^{k-1}.$$

Proof. By Euler's formula, $\phi(p^k) = p^{k-1}(p - 1)$. Substitute $p - 1 = d$, then $\phi(p^k) = p^{k-1}d$, as required. □

Corollary 0.4. *If f is an integer with $\phi(f) = m/(d \cdot p^{k-1})$, then $\phi(f \cdot p^k) = m$.*

Proof. Because $\gcd(f, p) = 1$, $\phi(f \cdot p^k) = \phi(f)\phi(p^k) = (m/(d \cdot p^{k-1}))(d \cdot p^{k-1}) = m$. □

Concrete Numerical Example

Definition 0.5 (Recursive Construction Example)**.** *Let* $m = 12$*. Divisors of* 12 *are* $1, 2, 3, 4, 6, 12$*. Take* $d = 2$*:* $d + 1 = 3$ *is prime. Valuation* $v_3(12) = 1$*, so* $k = 1, 2$ *are valid.*

- $k = 1$*:* $n = 2 \cdot 3^0 \cdot f = 2f$*,* $\phi(f) = 6$*.*
- $k = 2$*:* $n = 2 \cdot 3^1 \cdot f = 6f$*,* $\phi(f) = 2$*.*

Recursively applying to each divisor f *generates all integers* n *with* $\phi(n) = 12$*.*

Recursive Algorithm

Construction

Let $r[n]$ denote the list of integers x such that $\phi(x) = n$. Initialize $r[1] = [1]$.

For each divisor d of m:

1. If $d + 1$ is prime p, iterate $k = 1, \ldots, v$, where $v = v_p(m) + 1$.
2. Set $u = d \cdot p^{k-1}$ and $v = p^k$.
3. For each $f \mid m/u$, if $f \in r$, append $v \cdot e$ for all $e \in r[f]$ to $r[u \cdot f]$.

Finally, sort $r[m]$ and pick the 150,000th element.

Correctness

Theorem 0.6 (Completeness)**.** *The recursive construction generates all* n *such that* $\phi(n) = m$*.*

Proof. Euler's totient function is multiplicative: if $\gcd(a, b) = 1$, then $\phi(ab) = \phi(a)\phi(b)$. All integers with $\phi(n) = m$ can be factorized into prime powers.

For each prime power p^k, Lemma 1 ensures that every candidate prime $p = d + 1$ divides m correctly. Recursively combining with smaller solutions f ensures all multiplicative combinations are generated. Since we consider all divisors d and all possible k, no solution is missed. Sorting yields the numbers in ascending order. □

Algorithm in Pseudocode

```
Input: m = 13!
1. r[1] = [1]
2. For each divisor d of m:
      If d+1 is prime:
          For k = 1 to v_p(m, d+1)+1:
              u = d * (d+1)^(k-1)
              v = (d+1)^k
              For each f | m/u:
                  If f in r:
                      Add v*e for e in r[f] to r[f*u]
3. Sort r[m]
4. Return r[m][150_000-1]
```

Complexity Analysis

- Let $d(m)$ denote the number of divisors of m.
- Each recursive step generates all multiplicative combinations. Total number of generated integers is manageable (a few million for 13!).
- Time complexity: $O(d(m) \cdot \text{number of solutions})$.
- Space complexity: $O(\text{number of solutions})$.

Results

Running the algorithm yields

$$n_{150,000} = 23507044290$$

matching published references.

Discussion

- This problem elegantly combines combinatorics, number theory, and algorithmic design.
- Recursive generation based on multiplicative structure avoids infeasible brute-force search.
- Proofs show correctness: every integer with totient 13! is generated exactly once.
- The approach generalizes to any factorial or highly composite m.

References

- Project Euler, Problem 248. `https://projecteuler.net/problem=248`.
- D. Jacobsen, *Math::Prime::Util Inverse Totient Examples*. `https://github.com/danaj/Math-Prime-Util/blob/master/examples/inverse_totient.pl`.
- Max Alekseyev, *invphi.gp v1.3*, 2013.
- Python Documentation, `https://docs.python.org/3/library/collections.html`.