1. It suffices to show that the complement problem \textbf{PRIMES} is in \textbf{NP}. But this is quite obvious:

\textbf{PRIMES} (n)

1. \( n = 1 \) return \( \top \)
2. \( n = 2 \) return \( \top \)

Guess \( m, m' \) \( 1 \leq m, m' < n \)
3. \( m, m' = n \) return \( \top \)
4. Return \( \bot \)

This algorithm runs in time \( O(\log n^2) \) and has a successful run iff \( n \) is not prime.

2. pour \((a, b, p)\)

\% raised \( a \) to power \( b \) mod \( p \)

\% assume \( 1 \leq a < p \)

\( d := a \)
\( e := b \)
\( s := 1 \)

\textbf{until} \( e = 0 \)

\( \text{if } e \text{ is odd} \)

\( s := s \cdot d \mod p \)
\( e := e/2 \mod p \)
\( d := d^2 \mod p \)

\textbf{return} \( s \)

Number of iterations \( \log_h b + 1 \). At each iteration, we have at most 2 arithmetic operations on numbers no larger than \( p \), which is taken to be polynomial in \( \ln p \), plus one division by 2, which takes constant time.
3. Long division \( x^N \) can be performed in time bounded by a polynomial function of \( N \) using the standard method. To test whether \( p-1 = q_1 \cdots q_m \) for some \( d_1, \ldots, d_m \), do:

\[
\begin{align*}
\text{let } z & = p - 1 \\
\text{for } i & = 1 \ldots m \\
\text{if } q_i \text{ does not divide } z & \text{, return } N \\
& \text{ until } q_i \text{ does not divide } z \\
& \quad z = z / q_i \\
\text{if } z = 1 & \text{, return } \gamma \\
\text{return } N
\end{align*}
\]

The inner loop can be executed no more than \( \log(p-1) \) times because (assuming 1's have been filtered out of the \( q_i \)'s), each division reduces \( z \) by a factor of at least 2.

4. Let \( p = q_1 \cdots q_m \geq 2 \cdots 2 = 2^m \). So \( \log_2 p \geq m \).

5. \text{CPP consists of the following table:}

<table>
<thead>
<tr>
<th>Opening and closing parents</th>
<th>Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 m separators (where ( m \leq \log_2 p ))</td>
<td>( 2 \log_2 p )</td>
</tr>
<tr>
<td>( q_1, \ldots, q_m ) (the prime factors)</td>
<td>( 2 \log p )</td>
</tr>
<tr>
<td>( \log_2 p + 1 )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( z_{k+2}^{m}</td>
<td>\text{CPP}(q_k) )</td>
</tr>
</tbody>
</table>
The only non-obvious sum is \( x_0, 1 \ldots, x_0, m \).

Since \( \log_2 q_1 + \ldots + \log_2 q_m \leq \log_2 p \) and
\( \sum ||q|| \leq \log_2 q_1 + 1 \),
\( \sum ||q|| \leq m + \log_2 p \leq 2 \log_2 p \).

Adding everything up, we have
\[ 5 \log_2 p + 4 + \sum_{i=1}^{m} |\text{CPP}(q_i)| \]
(Actually, you can do a bit better than this, but no matter.)

6. To show \( \text{PRIMES} \in \text{NP} \), run the following algorithm.

(i) Guess a structure \( \text{CPP}(p) \) with the size bounds
given \( \leq q_1 \ldots q_m \).
\( \text{CPP}(p) = (r_1, q_1, \text{CPP}(q_1), \ldots, q_m, \text{CPP}(q_m)) \).

(ii) Compute \( r_1, r_2^{p-1/q_1} \) mod \( p \) in polynomial time
by \( Q2 \). For each \( q_1 = q_1, \ldots, q_m \), check
\( r_2^{p-1} = 1 \) (mod \( p \)) and \( r_2^{p-1/q_i} \) \( \equiv 1 \) mod \( p \) for all
\( q_1 = q_1, \ldots, q_m \).

(iii) Check \( p = q_1 \ldots q_m \) for some \( d_1, \ldots, d_m \)
in polynomial time by \( Q3 \).

(iv) Recursively check \( \text{CPP}(q_1) \ldots \text{CPP}(q_m) \).

Suppose steps (ii) and (iii) take time bounded by
\( a x_0^k \), where \( x_0 = \lceil \text{CPP}(p) \rceil = \sum_{k=2}^{m} |\text{CPP}(q_k)| \).

Clearly, such an \( a \) and \( k \) exist. We claim that
the whole algorithm runs in time bounded by
\( a (x_0 + \sum_{k=2}^{m} |\text{CPP}(q_k)|)^k \).

This is trivial since we just say \( Y \). For \( p > 2 \), time taken is, by
inductive hypothesis,
\( a x_0^k + \sum_{k=2}^{m} a x_0^k |\text{CPP}(q_k)|^k \leq \)
\( a (x_0 + \sum_{k=2}^{m} |\text{CPP}(q_k)|)^k \) = \( a (x_0 + \sum_{k=2}^{m} |\text{CPP}(q_k)|)^k \) as required.