### Minimalization

**So either** \( p_n + 1 \) **is itself a prime or it is divisible by a prime** \( > p_n \).  

**In either case there is a prime** \( q \) **such that** \( p_n < q \leq p_n + 1 \), **which gives us the inequality that we wanted to verify:**  
\[ p_{n+1} \leq p_n + 1. \]

**But now look at the recursion again:**  
\[ p_0 = 0 \]
\[ p_{n+1} = \min_{t \geq p_n} \text{Prime}(t) \text{ & } t > p_n. \]

**This is not exactly how we defined recursion. We should reformulate this definition.**

**To do so, we define the (obviously) primitive recursive function**
\[ h(y, z) = \min_{t \geq y} \text{Prime}(t) \text{ & } t > y \]

**Then we set**  
\[ k(x) = h(x, x! + 1), \]
which is another primitive recursive function.  

**Then our recursion equations reduce to**  
\[ p_0 = 0 \]
\[ p_{n+1} = k(p_n), \]
**So that we can (finally!) conclude that** \( p_n \) **is a primitive recursive function.**

---

### Minimalization

**Finally, we want to discuss **minimalization without a bound.**

Let us write
\[ \min_y P(x_1, ..., x_n, y) \]
for the least value of \( y \) for which the predicate \( P \) is true if there is such a value.  

If there is no such value of \( y \), then \( \min_y P(x_1, ..., x_n, y) \) is **undefined.**

(Note the difference with bounded minimalization.)

**Obviously, unbounded minimalization of a predicate can produce a function that is not total.**

**Example:**

The function \( x - y = \min_z [y + z = x] \) is undefined for \( x < y \).

We will see later that there are primitive recursive predicates \( P(x, y) \) such that \( \min_y P(x, y) \) is a total function which is **not primitive recursive.**

---

### Theorem 7.2

**If** \( P(x_1, ..., x_n, y) \) **is a computable predicate and if**  
\[ g(x_1, ..., x_n, y) = \min_y P(x_1, ..., x_n, y), \]
then \( g \) **is a partially computable function.**

**Proof:**

The following program computes \( g \):

[A]  
\[ \text{IF } P(X_1, ..., X_n, Y) \text{ GOTO E} \]
\[ Y \leftarrow Y + 1 \]
\[ \text{GOTO A} \]

---

### Pairing Functions and Gödel Numbers

**How can we code pairs of numbers by single numbers?**

Let us define the following primitive recursive function:
\[ (x, y) = 2^y(2y + 1) \]

**Obviously,** \( 2^y(2y + 1) \) **can never be 0, so we have:**
\[ (x, y) + 1 = 2^y(2y + 1). \]
Pairing Functions and Gödel Numbers

\((x, y) + 1 = 2^x(2y + 1)\).

If \(z\) is any given number, there is a unique solution \(x, y\) to the equation

\((x, y) = z\).

\(x\) is the largest number such that \(2^x \mid (z + 1)\),
and \(y\) is the solution of the equation

\(2y + 1 = (z + 1)/2^x\).

This equation has a unique solution because

\((z + 1)/2^x\) must be odd
(if it were even, we could have chosen a greater \(x\)).