Strong Induction

\[ P(0) \]
\[ \forall k \ ( (P(0) \land P(1) \land P(2) \land \cdots \land P(k)) \rightarrow P(k + 1)) \]

\[ \therefore \forall n \ P(n) \]

1. By induction we will show that \( P(n) \) is true for every \( n \geq 0 \)
2. **Base Case:** Prove \( P(0) \)
3. **Inductive Hypothesis:**
   Assume that for some arbitrary integer \( k \geq 0 \), \( P(j) \) is true for every \( j \) from 0 to \( k \)
4. **Inductive Step:**
   Prove that \( P(k+1) \) is true using the Inductive Hypothesis (that \( P(j) \) is true for all values \( \mathbb{W} k \))
5. **Conclusion:** Result follows by induction
Fibonacci Numbers

\[ f_0 = 0 \]
\[ f_1 = 1 \]
\[ f_n = f_{n-1} + f_{n-2} \quad \text{for all } n \geq 2 \]
Bounding the Fibonacci Numbers

Define $f_n$ as:

\[
\begin{align*}
  f_0 &= 0 \\
  f_1 &= 1 \\
  f_n &= f_{n-1} + f_{n-2} \text{ for all } n \geq 2
\end{align*}
\]

Theorem:

\[2^{n/2} - 1 \leq f_n \text{ and } f_n < 2^n\]

Proof:

Let $P(n)$ be “$2^{n/2} - 1 \leq f_n \text{ and } f_n < 2^n$” for all $n \geq 2$.

We go by strong induction on $n$.

Base Case: $2^{2/2} - 1 = 2^0 = 1 \leq 0 + 1 = f_2$, and $f_2 = 0 + 1 = 1 < 4 = 2^2$. So, $P(2)$ is true.

Induction Hypothesis:

Suppose $P(j)$ for all integers $j$ s.t. $2 \leq j \leq k$ for some $k \geq 2$.

Induction Step: We want to show $2^{(k+1)/2} - 1 \leq f_{k+1}$ and $f_{k+1} < 2^n$
Bounding the Fibonacci Numbers

Define \( f_n \) as:
\[
\begin{align*}
  f_0 &= 0 \\
  f_1 &= 1 \\
  f_n &= f_{n-1} + f_{n-2} \quad \text{for all } n \geq 2
\end{align*}
\]

Theorem:
\[
2^{n/2} - 1 \leq f_n \quad \text{and} \quad f_n < 2^n
\]

Induction Step: We want to show \( 2^{(k+1)/2} - 1 \leq f_{k+1} \) and \( f_{k+1} < 2^n \)

If \( k+1=3 \), \( 2^{3/2} - 1 = 2^{1/2} \leq 2 = 1 + 1 = f_3 \), and
\[
f_3 = 1 + 1 = 2 < 8 = 2^3.
\]
So, \( P(3) \) is true.

Otherwise, note that \( f_{k+1} = f_k + f_{k-1} \) by definition.

Taking each inequality separately:
\[
f_{k+1} = f_k + f_{k-1} < 2^k + 2^{k-1} \quad \text{(by IH)}
\]
\[
< 2^k + 2^k \quad \text{(}2^{k-1} < 2^k\text{)}
\]
\[
= 2^{k+1}
\]

\[
f_{k+1} = f_k + f_{k-1} \geq 2^{k/2-1} + 2^{(k-1)/2-1} \quad \text{(by IH)}
\]
\[
\geq 2^{(k-1)/2-1} + 2^{(k-1)/2-1} \quad \text{(Because } 2^{k/2-1} > 2^{(k-1)/2-1}\text{)}
\]
\[
= 2(2^{(k-1)/2-1}) \quad \text{(Combining terms)}
\]
\[
= 2^{2/2+(k-1)/2-1} \quad \text{(Multiplying)}
\]
\[
= 2^{(k+1)/2-1}
\]

So, the claim is true by strong induction.
Running time of Euclid’s algorithm

**Theorem:** Suppose that Euclid’s Algorithm takes \( n \) steps for \( \gcd(a,b) \) with \( a > b \). Then, \( a \geq f_{n+1} \).

We go by strong induction on \( n \).
Let \( P(n) \) be “\( \gcd(a,b) \) with \( a > b \) takes \( n \) steps \( \rightarrow a \geq f_{n+1} \)” for all \( n \geq 1 \).

**Base Case:**
If Euclid’s Algorithm on \( a, b, \) with \( a > b \), takes 1 step, then it must be the case that \( b \mid a \).
Note that \( f_2 = 1 \).
Note that if \( a \) were 0, then \( \gcd(0, b) \), which takes zero steps. So, the smallest possible value for \( a \) is 1, which is \( f_2 \).

**Induction Hypothesis:** Suppose \( P(j) \) for all integers \( j \) s.t. \( 1 \leq j \leq k \) for some \( k \geq 1 \).

**Induction Step:** We want to show if \( \gcd(a,b) \) takes \( k+1 \) steps, then \( a \geq f_{k+2} \).
If \( k = 2 \), note that \( a > 1 \), because \( \gcd(1, b) \) takes one step. Also, \( f_3 = 2 \).
Running time of Euclid’s algorithm

Theorem: Suppose that Euclid’s Algorithm takes \( n \) steps for \( \text{gcd}(a,b) \) with \( a > b \). Then, \( a \geq f_{n+1} \).

Since the algorithm took \( k+1 \) steps, let’s give them names:

Say \( r_{k+1} = a \) and \( r_k = b \), and \( r_i = r_{i-1} \mod r_{i-2} \).

So, \( \text{gcd}(a, b) = \text{gcd}(r_{k+1}, r_k) \)

\[
= \text{gcd}(r_k, r_k \mod r_{k+1}) = \text{gcd}(r_k, r_{k-1}) \\
= \text{gcd}(r_{k-1}, r_{k-1} \mod r_k) = \text{gcd}(r_{k-1}, r_{k-2}) \\
= ...
\]

Writing these as equations, we have:

\[
\begin{align*}
  r_{k+1} &= q_k r_k + r_{k-1} \\
  r_k &= q_{k-1} r_{k-1} + r_{k-2} \\
  &\vdots \\
  r_3 &= q_2 r_2 + r_1 \\
  r_2 &= q_1 r_1 \\
\end{align*}
\]

Note that \( q_i \geq 1, r_i \geq 1 \).

Note that after one iteration of the algorithm, we’re left with \( \text{gcd}(r_k, r_{k-1}) \) which takes \( k \) steps.

By the IH, \( r_k \geq f_{k+1} \). So,

\[
\begin{align*}
  r_{k+1} &= q_k r_k + r_{k-1} \quad \text{(by gcd algorithm)} \\
  &\geq q_k f_{k+1} + f_k \quad \text{(by IH)} \\
  &\geq f_{k+1} + f_k \quad \text{(since } q_k \geq 1) \\
  &\geq f_{k+2} \quad \text{(definition of } f) \\
\end{align*}
\]
Recursive Definition of Sets

Recursive Definition

• Basis Step: $0 \in S$
• Recursive Step: If $x \in S$, then $x + 2 \in S$
• Exclusion Rule: Every element in $S$ follows from basis steps and a finite number of recursive steps.
Recursive Definitions of Sets

Basis: \(6 \in S, 15 \in S\)
Recursive: If \(x,y \in S\), then \(x+y \in S\)

Basis: \([1, 1, 0] \in S, [0, 1, 1] \in S\)
Recursive: If \([x, y, z] \in S\), then \([\alpha x, \alpha y, \alpha z] \in S\)
If \([x_1, y_1, z_1] \in S\) and \([x_2, y_2, z_2] \in S\), then
\([x_1 + x_2, y_1 + y_2, z_1 + z_2] \in S\).

Powers of 3:
Basis: \(1 \in S\)
Recursive: If \(x \in S\), then \(3x \in S\).
Recursive Definitions of Sets: General Form

Recursive definition

– *Basis step:* Some specific elements are in $S$
– *Recursive step:* Given some existing named elements in $S$, some new objects constructed from these named elements are also in $S$.
– *Exclusion rule:* Every element in $S$ follows from basis steps and a finite number of recursive steps
Strings

• An \textit{alphabet} $\Sigma$ is any finite set of characters.

• The set $\Sigma^*$ of \textit{strings} over the alphabet $\Sigma$ is defined by
  – \textbf{Basis:} $\varepsilon \in \Sigma^*$ (the empty string)
  – \textbf{Recursive:} if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$
Palindromes

Palindromes are strings that are the same backwards and forwards

**Basis:**

\( \epsilon \) is a palindrome and any \( a \in \Sigma \) is a palindrome

**Recursive step:**

If \( p \) is a palindrome then \( apa \) is a palindrome for every \( a \in \Sigma \)
All Binary Strings with no 1’s before 0’s

Basis:
\[ \varepsilon \in S \]

Recursive:
\[ \text{If } x \in S, \text{ then } 0x \in S \]
\[ \text{If } x \in S, \text{ then } x1 \in S \]
Function Definitions on Recursively Defined Sets

Length:
\[ \text{len}(\varepsilon) = 0 \]
\[ \text{len}(wa) = 1 + \text{len}(w) \text{ for } w \in \Sigma^*, a \in \Sigma \]

Reversal:
\[ \varepsilon^R = \varepsilon \]
\[ (wa)^R = aw^R \text{ for } w \in \Sigma^*, a \in \Sigma \]

Concatenation:
\[ x \cdot \varepsilon = x \text{ for } x \in \Sigma^* \]
\[ x \cdot wa = (x \cdot w)a \text{ for } x \in \Sigma^*, a \in \Sigma \]