Primitive Recursive Function and Ramsey Theory

Exposition by William Gasarch-U of MD
Definition of Primitive Recursive (PR)

Def $f(x_1, \ldots, x_n)$ is PR if either:
Definition of Primitive Recursive (PR)

Def $f(x_1, \ldots, x_n)$ is PR if either:

1. $f(x_1, \ldots, x_n) = 0$;
Definition of Primitive Recursive (PR)

Def \( f(x_1, \ldots, x_n) \) is PR if either:

1. \( f(x_1, \ldots, x_n) = 0; \)
2. \( f(x_1, \ldots, x_n) = x_i; \)
Definition of Primitive Recursive (PR)

Def $f(x_1, \ldots, x_n)$ is PR if either:

1. $f(x_1, \ldots, x_n) = 0$;
2. $f(x_1, \ldots, x_n) = x_i$;
3. $f(x_1, \ldots, x_n) = x_i + 1$;
Definition of Primitive Recursive (PR)

Def $f(x_1, \ldots, x_n)$ is PR if either:

1. $f(x_1, \ldots, x_n) = 0$;
2. $f(x_1, \ldots, x_n) = x_i$;
3. $f(x_1, \ldots, x_n) = x_i + 1$;
4. $g_1(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k), h(x_1, \ldots, x_n)$ PR $\implies$

$$f(x_1, \ldots, x_k) = h(g_1(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k))$$ is PR
Definition of Primitive Recursive (PR)

Def $f(x_1, \ldots, x_n)$ is PR if either:

1. $f(x_1, \ldots, x_n) = 0$;
2. $f(x_1, \ldots, x_n) = x_i$;
3. $f(x_1, \ldots, x_n) = x_i + 1$;
4. $g_1(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k), h(x_1, \ldots, x_n) \text{ PR } \implies f(x_1, \ldots, x_k) = h(g_1(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k)) \text{ is PR}$

5. $h(x_1, \ldots, x_{n+1})$ and $g(x_1, \ldots, x_{n-1}) \text{ PR } \implies$

\[
\begin{align*}
    f(x_1, \ldots, x_{n-1}, 0) &= g(x_1, \ldots, x_{n-1}) \\
    f(x_1, \ldots, x_{n-1}, m + 1) &= h(x_1, \ldots, x_{n-1}, m, f(x_1, \ldots, x_{n-1}, m))
\end{align*}
\]

is PR.
Examples of PR Functions

\[ f_0(x, y) = y + 1. \text{ Successor.} \]
Examples of PR Functions

\[ f_0(x, y) = y + 1. \text{ Successor.} \]
\[ f_1(x, y) = x + y \]
Examples of PR Functions

\[ f_0(x, y) = y + 1. \text{ Successor.} \]
\[ f_1(x, y) = x + y \]
\[ f_1(x, 0) = x \]
\[ f_1(x, y + 1) = f_1(x, y) + 1. \]
Used Rec Rule Once. Addition.

\[ f_2(x, y) = xy: \]
\[ f_2(x, 1) = x \] (Didn't start at 0. A detail.)
\[ f_2(x, y + 1) = f_2(x, y) + x. \]
Used Rec Rule Twice. Once to get x + y PR, and once here.

Multiplication

The PR functions can be put in a hierarchy depending on how many times the recursion rule is used to build up to the function.
Examples of PR Functions

\[ f_0(x, y) = y + 1. \text{ Successor.} \]
\[ f_1(x, y) = x + y \]
\[ f_1(x, 0) = x \]
\[ f_1(x, y + 1) = f_1(x, y) + 1. \]
Used Rec Rule Once. Addition.
\[ f_2(x, y) = xy: \]
Examples of PR Functions

\[ f_0(x, y) = y + 1. \text{ Successor.} \]
\[ f_1(x, y) = x + y \]
\[ f_1(x, 0) = x \]
\[ f_1(x, y + 1) = f_1(x, y) + 1. \]
Used Rec Rule Once. Addition.

\[ f_2(x, y) = xy: \]
\[ f_2(x, 1) = x \text{ (Didn’t start at 0. A detail.)} \]
\[ f_2(x, y + 1) = f_2(x, y) + x. \]
Used Rec Rule Twice. Once to get \( x + y \) PR, and once here. Multiplication
Examples of PR Functions

\[ f_0(x, y) = y + 1. \text{ Successor.} \]
\[ f_1(x, y) = x + y \]
\[ f_1(x, 0) = x \]
\[ f_1(x, y + 1) = f_1(x, y) + 1. \]
Used Rec Rule Once. Addition.

\[ f_2(x, y) = xy: \]
\[ f_2(x, 1) = x \text{ (Didn’t start at 0. A detail.)} \]
\[ f_2(x, y + 1) = f_2(x, y) + x. \]
Used Rec Rule Twice. Once to get \( x + y \) PR, and once here. Multiplication

The PR functions can be put in a hierarchy depending on how many times the recursion rule is used to build up to the function.
More PR Functions

More PR Functions

More PR Functions
More PR Functions

\[ f_3(x, y) = x^y : \]

Used Rec Rule three times. Exp.

\[ f_4(x, y) = \text{TOWER}(x, y). \]

Used Rec Rule four times. TOWER.

\[ f_5(x, y) = \text{WHAT SHOULD WE CALL THIS?} \]

Used Rec Rule five times. Its been called WOWER.
More PR Functions

\[ f_3(x, y) = x^y: \]
\[ f_3(x, 0) = 1 \]
\[ f_3(x, y + 1) = f_3(x, y)x. \]

Used Rec Rule three times. Exp.
More PR Functions

\[ f_3(x, y) = x^y: \]
\[ f_3(x, 0) = 1 \]
\[ f_3(x, y + 1) = f_3(x, y)x. \]

Used Rec Rule three times. Exp.

\[ f_4(x, y) = \text{TOW}(x, y). \]
More PR Functions

\[ f_3(x, y) = x^y: \]
\[ f_3(x, 0) = 1 \]
\[ f_3(x, y + 1) = f_3(x, y)x. \]

Used Rec Rule three times. Exp.

\[ f_4(x, y) = \text{TOW}(x, y). \]
\[ f_4(x, 0) = 1 \]
\[ f_4(x, y + 1) = f_4(x, y)x. \]

Used Rec Rule four times. TOWER.
More PR Functions

\[ f_3(x, y) = x^y: \]
\[ f_3(x, 0) = 1 \]
\[ f_3(x, y + 1) = f_3(x, y) x. \]
Used Rec Rule three times. Exp.

\[ f_4(x, y) = \text{TOW}(x, y). \]
\[ f_4(x, 0) = 1 \]
\[ f_4(x, y + 1) = f_4(x, y)^x. \]
Used Rec Rule four times. TOWER.

\[ f_5(x, y) = \text{WHAT SHOULD WE CALL THIS?} \]
More PR Functions

\[ f_3(x, y) = x^y: \]
\[ f_3(x, 0) = 1 \]
\[ f_3(x, y + 1) = f_3(x, y)x. \]

Used Rec Rule three times. Exp.

\[ f_4(x, y) = \text{TOW}(x, y). \]
\[ f_4(x, 0) = 1 \]
\[ f_4(x, y + 1) = f_4(x, y)^x. \]

Used Rec Rule four times. TOWER.

\[ f_5(x, y) = \text{WHAT SHOULD WE CALL THIS?} \]
\[ f_5(x, 0) = 1 \]
\[ f_5(x, y + 1) = \text{TOW}(f_5(x, y), x). \]

Used Rec Rule five times.

What should we call this? Discuss
More PR Functions

\[ f_3(x, y) = x^y : \]
\[ f_3(x, 0) = 1 \]
\[ f_3(x, y + 1) = f_3(x, y)x. \]
Used Rec Rule three times. Exp.

\[ f_4(x, y) = \text{TOW}(x, y). \]
\[ f_4(x, 0) = 1 \]
\[ f_4(x, y + 1) = f_4(x, y)^x. \]
Used Rec Rule four times. TOWER.

\[ f_5(x, y) = \text{WHAT SHOULD WE CALL THIS?} \]
\[ f_5(x, 0) = 1 \]
\[ f_5(x, y + 1) = \text{TOW}(f_5(x, y), x). \]
Used Rec Rule five times.
What should we call this? Discuss
Its been called WOWER.
The Functions That Have No Name

$f_a(x, y)$ is defined as
The Functions That Have No Name

\[ f_a(x, y) \text{ is defined as} \]
\[ f_a(x, 0) = 1 \]
\[ f_a(x, y + 1) = f_{a-1}(f_a(x, y), x, y) \]
The Functions That Have No Name

$f_a(x, y)$ is defined as

$f_a(x, 0) = 1$

$f_a(x, y + 1) = f_{a-1}(f_a(x, y), x, y)$

$f_0$ is Successor
The Functions That Have No Name

$f_a(x, y)$ is defined as

\[
f_a(x, 0) = 1 \\
f_a(x, y + 1) = f_{a-1}(f_a(x, y), x, y)
\]

$f_0$ is Successor

$f_1$ is Addition
The Functions That Have No Name

\( f_a(x, y) \) is defined as

\[
\begin{align*}
    f_a(x, 0) &= 1 \\
    f_a(x, y + 1) &= f_{a-1}(f_a(x, y), x, y)
\end{align*}
\]

\( f_0 \) is Successor

\( f_1 \) is Addition

\( f_2 \) is Multiplication
The Functions That Have No Name

\[ f_a(x, y) \text{ is defined as} \]
\[ f_a(x, 0) = 1 \]
\[ f_a(x, y + 1) = f_{a-1}(f_a(x, y), x, y) \]

\[ f_0 \text{ is Successor} \]
\[ f_1 \text{ is Addition} \]
\[ f_2 \text{ is Multiplication} \]
\[ f_3 \text{ is Exp} \]
The Functions That Have No Name

$f_a(x, y)$ is defined as

\[ f_a(x, 0) = 1 \]
\[ f_a(x, y + 1) = f_{a-1}(f_a(x, y), x, y) \]

$f_0$ is Successor

$f_1$ is Addition

$f_2$ is Multiplication

$f_3$ is Exp

$f_4$ is Tower (This name has become standard.)
The Functions That Have No Name

\[ f_a(x, y) \] is defined as

\[ f_a(x, 0) = 1 \]
\[ f_a(x, y + 1) = f_{a-1}(f_a(x, y), x, y) \]

\( f_0 \) is Successor
\( f_1 \) is Addition
\( f_2 \) is Multiplication
\( f_3 \) is Exp
\( f_4 \) is Tower (This name has become standard.)
\( f_5 \) is Wower (This name is not standard.)
The Functions That Have No Name

$f_a(x, y)$ is defined as

$f_a(x, 0) = 1$
$f_a(x, y + 1) = f_{a-1}(f_a(x, y), x, y)$

$f_0$ is Successor
$f_1$ is Addition
$f_2$ is Multiplication
$f_3$ is Exp
$f_4$ is Tower (This name has become standard.)
$f_5$ is Wower (This name is not standard.)
$f_6$ and beyond have no name.
Def $PR_a$ is the set of PR functions that can be defined with $\leq a$ uses of the Recursion rule.
**Def** $\text{PR}_a$ is the set of PR functions that can be defined with $\leq a$ uses of the Recursion rule.

**Note** One can show that any finite number of exponentials is in $\text{PR}_3$. 
More is PR than you Think

The following are PR:

1. \( f(x, y) = x - y \) if \( x \geq y \), 0 otherwise.
2. \( f(x, y) = \) the quotient when you divide \( x \) by \( y \).
3. \( f(x, y) = \) the remainder when you divide \( x \) by \( y \).
4. \( f(x, y) = x \) (mod \( y \)).
5. \( f(x, y) = \) GCD \( (x, y) \).
6. \( f(x) = 1 \) if \( x \) is prime, 0 if not.
7. \( f(x) = 1 \) if \( x \) is the sum of 2 primes, 0 otherwise.

Virtually any computable function from \( \mathbb{N}^k \) to \( \mathbb{N} \) that you encounter in mathematics is primitive recursive.
More is PR than you Think

The following are PR:

1. \( f(x, y) = x - y \) if \( x \geq y \), 0 otherwise.
More is PR than you Think

The following are PR:

1. $f(x, y) = x - y$ if $x \geq y$, 0 otherwise.
2. $f(x, y) =$ the quotient when you divide $x$ by $y$. 

Virtually any computable function from $\mathbb{N}^k$ to $\mathbb{N}$ that you encounter in mathematics is primitive recursive.
The following are PR:
1. \( f(x, y) = x - y \) if \( x \geq y \), 0 otherwise.
2. \( f(x, y) = \) the quotient when you divide \( x \) by \( y \).
3. \( f(x, y) = \) the remainder when you divide \( x \) by \( y \).
More is PR than you Think

The following are PR:

1. $f(x, y) = x - y$ if $x \geq y$, 0 otherwise.
2. $f(x, y) =$ the quotient when you divide $x$ by $y$.
3. $f(x, y) =$ the remainder when you divide $x$ by $y$.
4. $f(x, y) = x \pmod{y}$.

Virtually any computable function from $\mathbb{N}^k$ to $\mathbb{N}$ that you encounter in mathematics is primitive recursive.
More is PR than you Think

The following are PR:

1. \( f(x, y) = x - y \) if \( x \geq y \), 0 otherwise.
2. \( f(x, y) = \) the quotient when you divide \( x \) by \( y \).
3. \( f(x, y) = \) the remainder when you divide \( x \) by \( y \).
4. \( f(x, y) = x \pmod{y} \).
5. \( f(x, y) = \text{GCD}(x, y) \).
More is PR than you Think

The following are PR:

1. \( f(x, y) = x - y \) if \( x \geq y \), 0 otherwise.
2. \( f(x, y) = \) the quotient when you divide \( x \) by \( y \).
3. \( f(x, y) = \) the remainder when you divide \( x \) by \( y \).
4. \( f(x, y) = x \pmod{y} \).
5. \( f(x, y) = \text{GCD}(x, y) \).
6. \( f(x) = 1 \) if \( x \) is prime, 0 if not.

Virtually any computable function from \( \mathbb{N} \) to \( \mathbb{N} \) that you encounter in mathematics is primitive recursive.
More is PR than you Think

The following are PR:

1. \( f(x, y) = x - y \) if \( x \geq y \), 0 otherwise.
2. \( f(x, y) = \) the quotient when you divide \( x \) by \( y \).
3. \( f(x, y) = \) the remainder when you divide \( x \) by \( y \).
4. \( f(x, y) = x \pmod{y} \).
5. \( f(x, y) = \gcd(x, y) \).
6. \( f(x) = 1 \) if \( x \) is prime, 0 if not.
7. \( f(x) = 1 \) if \( x \) is the sum of 2 primes, 0 otherwise.

Virtually any computable function from \( \mathbb{N}^k \) to \( \mathbb{N} \) that you encounter in mathematics is primitive recursive.
Are There any Computable Functions that are Not PR?. Discuss.

This is really two questions:
1. Are there any, possibly contrived functions, that are computable but not PR?
2. Are there any natural functions that are computable but not PR?
Discuss both questions.
Are There any Computable Functions that are Not PR?

Are There any Computable Functions that are Not PR? Discuss.

This is really two questions
Are There any Computable Functions that are Not PR?

This is really two questions

1. Are there any, possibly contrived functions, that are computable but not PR?
Are There any Computable Functions that are Not PR?

Are There any Computable Functions that are Not PR?. Discuss.

This is really two questions

1. Are there any, possibly contrived functions, that are computable but not PR?
2. Are there any natural functions that are computable but not PR?

Discuss both questions.
There are Computable NON-PR functions

I won’t do this since the function is not natural.
A Natural non PR Function that is Computable

**Def** Ackerman’s function is the function defined by

\[
A(0, y) = y + 1 \\
A(x + 1, 0) = A(x, 1) \\
A(x + 1, y + 1) = A(x, A(x + 1, y))
\]
Def Ackerman’s function is the function defined by

\[
A(0, y) = y + 1 \\
A(x + 1, 0) = A(x, 1) \\
A(x + 1, y + 1) = A(x, A(x + 1, y))
\]

1. A is obviously computable.
Def. **Ackerman’s function** is the function defined by

\[
\begin{align*}
A(0, y) &= y + 1 \\
A(x + 1, 0) &= A(x, 1) \\
A(x + 1, y + 1) &= A(x, A(x + 1, y))
\end{align*}
\]

1. \(A\) is obviously computable.
2. \(A\) grows faster than any PR function.
Def Ackerman's function is the function defined by

\[
A(0, y) = y + 1 \\
A(x + 1, 0) = A(x, 1) \\
A(x + 1, y + 1) = A(x, A(x + 1, y))
\]

1. $A$ is obviously computable.
2. $A$ grows faster than any PR function.
3. Since $A$ is defined using a recursion which involves applying the function to itself there is no obvious way to take the definition and make it PR. Not a proof, an intuition.
Ackerman’s Function is Natural: Security

https://ackerman-security-systems.pissedconsumer.com/customer-service.html
https://www.ackermansecurity.com/
Ackerman’s Function is Natural: Security

https://ackerman-security-systems.pissedconsumer.com/customer-service.html
https://www.ackermansecurity.com/

They are called Ackerman Security since they claim that Burglar would have to be Ackerman(n)-good to break in.
DS is Data Structure.
UNION-FIND DS for sets that supports:

- If $a$ is a number then make $\{a\}$ a set.
- If $A, B$ are sets then make $A \cup B$ a set.
- Given $x$ find which, if any, set it is in.

There is a DS for this problem that can do $n$ operations in $n^2 - 1$ steps.

One can show that there is no better DS.

So $n^2 - 1$ is the exact upper and lower bound!
Ackerman’s Function is Natural: DS

DS is Data Structure.
UNION-FIND DS for sets that supports:
(1) If \( a \) is a number then make \( \{a\} \) a set.
DS is Data Structure.
UNION-FIND DS for sets that supports:
(1) If $a$ is a number then make $\{a\}$ a set.
(2) If $A, B$ are sets then make $A \cup B$ a set.
Ackerman’s Function is Natural: DS

DS is Data Structure.
UNION-FIND DS for sets that supports:
(1) If a is a number then make \{a\} a set.
(2) If A, B are sets then make \(A \cup B\) a set.
(3) Given \(x\) find which, if any, set it is in.
Ackerman’s Function is Natural: DS

DS is Data Structure.
UNION-FIND DS for sets that supports:
(1) If $a$ is a number then make $\{a\}$ a set.
(2) If $A, B$ are sets then make $A \cup B$ a set.
(3) Given $x$ find which, if any, set it is in.
Ackerman’s Function is Natural: DS

DS is Data Structure.
UNION-FIND DS for sets that supports:
(1) If $a$ is a number then make $\{a\}$ a set.
(2) If $A, B$ are sets then make $A \cup B$ a set.
(3) Given $x$ find which, if any, set it is in.

▶ There is a DS for this problem that can do $n$ operations in $nA^{-1}(n)$ steps.
DS is Data Structure.
UNION-FIND DS for sets that supports:
(1) If $a$ is a number then make $\{a\}$ a set.
(2) If $A, B$ are sets then make $A \cup B$ a set.
(3) Given $x$ find which, if any, set it is in.

▸ There is a DS for this problem that can do $n$ operations in $nA^{-1}(n)$ steps.
▸ One can show that there is no better DS.
DS is Data Structure.
UNION-FIND DS for sets that supports:
(1) If \(a\) is a number then make \(\{a\}\) a set.
(2) If \(A, B\) are sets then make \(A \cup B\) a set.
(3) Given \(x\) find which, if any, set it is in.

▶ There is a DS for this problem that can do \(n\) operations in \(nA^{-1}(n)\) steps.
▶ One can show that there is no better DS.
So \(nA^{-1}(n, n)\) is the exact upper and lower bound!
Writing a number as a sum of powers of 2.

$$1000 = 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3$$
Writing a number as a sum of powers of 2.

\[ 1000 = 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3 \]

But we can also write the exponents as sums of power of 2.

\[ 1000 = 2^{2^3 + 2^0} + 2^{2^3} + 2^{2^2 + 2^1 + 2^0} + 2^{2^1 + 2^0} \]
Writing a number as a sum of powers of 2.

\[ 1000 = 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3 \]

But we can also write the exponents as sums of power of 2

\[ 1000 = 2^{2^3+2^0} + 2^{2^3} + 2^{2^2+2^1+2^0} + 2^{2^1+2^0} \]

We can even write the exponents that are not already powers of 2 as sums of powers of 2.

\[ 1000 = 2^{2^{2^1+2^0}+2^0} + 2^{2^{2^1+2^0}} + 2^{2^2+2^1+2^0} + 2^{2^1+2^0} \]
Writing a number as a sum of powers of 2.

\[ 1000 = 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3 \]

But we can also write the exponents as sums of power of 2

\[ 1000 = 2^{2^3+2^0} + 2^{2^3} + 2^{2^{2^2+2^1+2^0}} + 2^{2^1+2^0} \]

We can even write the exponents that are not already powers of 2 as sums of powers of 2.

\[ 1000 = 2^{2^{2^1+2^0}+2^0} + 2^{2^{2^1+2^0}} + 2^{2^2+2^1+2^0} + 2^{2^1+2^0} \]

This is called \textbf{Hereditary Base } \textit{n} \textbf{Notation}
Ackerman’s Function and Goodstein Seq

\[ 1000 = 2^{2^{2^1+2^0}+2^0} + 2^{2^{2^1+2^0}} + 2^{2^2+2^1+2^0} + 2^{2^1+2^0} \]

Replace all of the 2’s with 3’s:

\[ 3^{3^{3^1+3^0}+3^0} + 3^{3^{3^1+3^0}} + 3^{3^3+3^1+3^0} + 3^{3^1+3^0} \]
Ackerman’s Function and Goodstein Seq

$1000 = 2^{2^{2^1}+2^0} + 2^{2^{2^1}+2^0} + 2^{2^2+2^1+2^0} + 2^{2^1+2^0}$

Replace all of the 2’s with 3’s:

$3^{3^{3^1}+3^0} + 3^{3^{3^1}+3^0} + 3^{3^3+3^1+3^0} + 3^{3^1+3^0}$

This number just went WAY up. Now subtract 1.

$3^{3^{3^1}+3^0} + 3^{3^{3^1}+3^0} + 3^{3^3+3^1+3^0} + 3^{3^1+3^0} − 1$
Ackerman’s Function and Goodstein Seq

\[ 1000 = 2^{2^{2^1+2^0}+2^0} + 2^{2^{2^1+2^0}} + 2^{2^2+2^1+2^0} + 2^{2^1+2^0} \]

Replace all of the 2’s with 3’s:

\[ 3^{3^{3^{3^1+3^0}+3^0}} + 3^{3^{3^{3^1+3^0}}} + 3^{3^3+3^1+3^0} + 3^{3^1+3^0} \]

This number just went WAY up. Now subtract 1.

\[ 3^{3^{3^{3^1+3^0}+3^0}} + 3^{3^{3^{3^1+3^0}}} + 3^{3^3+3^1+3^0} + 3^{3^1+3^0} - 1 \]

Repeat the process:
Replace 3 by 4, and subtract 1, Replace 4 by 5, and subtract 1, \ldots.
Ackerman’s Function and Goodstein Seq

\[ 1000 = 2^{2^{1+2^0} + 2^0} + 2^{2^{2^{1+2^0}} + 2^2 + 2^{1+2^0}} + 2^{2^{1+2^0}} \]

Replace all of the 2’s with 3’s:

\[ 3^{3^{3^{1+3^0} + 3^0} + 3^{3^{3^{1+3^0} + 3^0}} + 3^{3^3+3^{1+3^0}} + 3^{3^1+3^0}} \]

This number just went WAY up. Now subtract 1.

\[ 3^{3^{3^{1+3^0} + 3^0} + 3^{3^{3^{1+3^0} + 3^0}} + 3^{3^3+3^{1+3^0}} + 3^{3^1+3^0}} - 1 \]

Repeat the process:
Replace 3 by 4, and subtract 1, Replace 4 by 5, and subtract 1, · · · .

**Vote** Does the sequence:

- Goto infinity (and if so how fast- perhaps Ack-like?)
- Eventually stabilizes (e.g., goes to 18 and then stops there)
- Cycles- goes UP then DOWN then UP then DOWN . . . .
The sequence goes to 0. The number of steps for \( n \) to goto 0 is roughly \( \text{ACK}(n, n) \).
The sequence goes to 0.
The number of steps for $n$ to goto 0 is roughly $ACK(n, n)$. 