Discrete Structures: Recursion

Amotz Bar-Noy

Department of Computer and Information Science
Brooklyn College
Recursion

Definition
- Recursion occurs when “something” is defined in terms of itself or of its type.

Focus
- Recursive formulas in mathematics
- Recursive programs in computer science

Illustrations
- https://storage.googleapis.com/algodailyrandomassets/curriculum/recursion/cover.jpg
Recursive Formulas

Definition
- A recursive formula is defined on the set of integers greater than or equal to some number $m$ (usually 0 or 1)
- The formula computes the $n^{th}$ value based on some or all of the previous $n - 1$ values

Goal
- Given initial values and a recursive formula, find its closed-form expression that does not depend on previous values

Recursion and induction
- Usually proving the correctness of a solution (a closed-form expression) to a recursive formula is done by induction
The Non-Negative Integers

The recursive formula

\[ N(n) = \begin{cases} 
0 & \text{for } n = 0 \\
N(n - 1) + 1 & \text{for } n > 0 
\end{cases} \]

The recursive pseudocode

```
function N(n) (* integer n \geq 0 *)
    if n = 0
        then return (0)
    else return (N(n - 1) + 1)
```
The Non-Negative Integers

Bottom-Up evaluation

\[
\begin{align*}
N(0) &= 0 \\
N(1) &= N(0) + 1 = 1 \\
N(2) &= N(1) + 1 = 2 \\
N(3) &= N(2) + 1 = 3 \\
N(4) &= N(3) + 1 = 4 \\
&\vdots \\
N(n) &= n
\end{align*}
\]
The Non-Negative Integers

Top-Down evaluation

\[ N(n) = (N(n - 1) + 1) \]
\[ = (N(n - 2) + 1) + 1 \]
\[ = (N(n - 3) + 1) + 1 + 1 \]
\[ = (N(n - 4) + 1) + 1 + 1 + 1 \]
\[ \vdots \]
\[ = N(0) + \underbrace{1 + \cdots + 1}_n \]
\[ = 0 + n \]
\[ = n \]
The Non-Negative Integers

The closed-form expression

\[ N(n) = n \]

Proof by induction

- Induction base: \( N(0) = 0 \)
- Induction hypothesis: \( N(n - 1) = n - 1 \) for \( n > 0 \)
- Inductive step for \( n > 0 \):

\[
N(n) = N(n - 1) + 1 \\
= (n - 1) + 1 \\
= n
\]
The non-recursive pseudocode

function $N(n)$ (* integer $n \geq 0$ *)

$k = 0$

for $i = 1$ to $n$

$k = k + 1$

return $(k)$

Why it is the same as the recursive pseudocode?

- The **recursive pseudocode** implements the **Top-Down** evaluation
- The **non-recursive pseudocode** implements the **Bottom-Up** evaluation
The Non-Positive Integers

The recursive formula

\[ N(n) = \begin{cases} 
0 & \text{for } n = 0 \\
N(n - 1) - 1 & \text{for } n > 0 
\end{cases} \]

The recursive pseudocode

```plaintext
function N(n) (* integer n ≥ 0 *)
    if n = 0
        then return (0)
    else return (N(n - 1) - 1)
```
The Non-Positive Integers

Bottom-Up evaluation

\[ N(0) = -0 \]
\[ N(1) = N(0) - 1 = -1 \]
\[ N(2) = N(1) - 1 = -2 \]
\[ N(3) = N(2) - 1 = -3 \]
\[ N(4) = N(3) - 1 = -4 \]
\[ \vdots \]
\[ N(n) = -n \]
The Non-Positive Integers

Top-Down evaluation

\[ N(n) = (N(n - 1) - 1) \]
\[ = (N(n - 2) - 1) - 1 \]
\[ = (N(n - 3) - 1) - 1 - 1 \]
\[ = (N(n - 4) - 1) - 1 - 1 - 1 \]
\[ \vdots \]
\[ = N(0) - 1 - 1 - \cdots - 1 \]
\[ = 0 - n \]
\[ = -n \]
The Non-Positive Integers

The closed-form expression

\[ N(n) = -n \]

Proof by induction

- Induction base: \( N(0) = 0 = -0 \)
- Induction hypothesis: \( N(n - 1) = -(n - 1) \) for \( n > 0 \)
- Inductive step for \( n > 0 \):

\[
\begin{align*}
N(n) & = N(n - 1) - 1 \\
& = -(n - 1) - 1 \\
& = -n
\end{align*}
\]
The Non-Positive Integers

The non-recursive pseudocode

function $N(n)$ (* integer $n \geq 0$ *)
    $k = 0$
    for $i = 1$ to $n$
        $k = k - 1$
    return $k$

Why it is the same as the recursive pseudocode?

- The recursive pseudocode implements the Top-Down evaluation
- The non-recursive pseudocode implements the Bottom-Up evaluation
Another Recursive Formula

The recursive formula

\[ T(n) = \begin{cases} 
2 & \text{for } n = 0 \\
T(n - 1) + 10 & \text{for } n > 0 
\end{cases} \]

The recursive pseudocode

\begin{verbatim}
function T(n) (* integer n \geq 0 *)
    if n = 0
        then return (2)
    else return (T(n - 1) + 10)
\end{verbatim}
Another Recursive Formula

Bottom-Up evaluation

\[
T(0) = 2 \\
T(1) = T(0) + 10 = 12 \\
T(2) = T(1) + 10 = 22 \\
T(3) = T(2) + 10 = 32 \\
T(4) = T(3) + 10 = 42 \\
\vdots \\
T(n) = 2 + 10n
\]
Another Recursive Formula

Top-Down evaluation

\[
T(n) = (T(n - 1) + 10) \\
= (T(n - 2) + 10) + 10 \\
= (T(n - 3) + 10) + 10 + 10 \\
= (T(n - 4) + 10) + 10 + 10 + 10 \\
\vdots \\
= T(0) + \underbrace{10 + 10 + \cdots + 10}_{n} \\
= 2 + 10n
\]
Another Recursive Formula

The closed-form expression

\[ T(n) = 2 + 10n \]

Proof by induction

- Induction base: \( T(0) = 2 + 10 \cdot 0 = 2 \)
- Induction hypothesis: \( T(n - 1) = 2 + 10(n - 1) \) for \( n > 0 \)
- Inductive step for \( n > 0 \):

\[
T(n) = T(n - 1) + 10 \\
= (2 + 10(n - 1)) + 10 \\
= 2 + 10n
\]
Another Recursive Formula

The non-recursive pseudocode

function T(n) (* integer n ≥ 0 *)
  t = 2
  for i = 1 to n
    t = t + 10
  return(t)

Why it is the same as the recursive pseudocode?

- The recursive pseudocode implements the Top-Down evaluation
- The non-recursive pseudocode implements the Bottom-Up evaluation
A Generalization

The recursive formula

\[
T(n) = \begin{cases} 
  b & \text{for } n = 0 \text{ and a real number } b \\
  T(n-1) + a & \text{for } n > 0 \text{ and a real number } a
\end{cases}
\]

The recursive pseudocode

```plaintext
function T(n) (* integer n ≥ 0 *)
    if n = 0
        then return (b)
    else return (T(n - 1) + a)
```
A Generalization

Bottom-Up evaluation

\[
\begin{align*}
T(0) &= b \\
T(1) &= T(0) + a = b + a \\
T(2) &= T(1) + a = b + 2a \\
T(3) &= T(2) + a = b + 3a \\
T(4) &= T(3) + a = b + 4a \\
& \vdots \\
T(n) &= b + an
\end{align*}
\]
A Generalization

Top-Down evaluation

\[ T(n) = (T(n - 1) + a) \]
\[ = (T(n - 2) + a) + a \]
\[ = (T(n - 3) + a) + a + a \]
\[ = (T(n - 4) + a) + a + a + a \]
\[ \vdots \]
\[ = T(0) + \overbrace{a + a + \cdots + a}^{n} \]
\[ = b + an \]
A Generalization

The closed-form expression

\[ T(n) = b + an \]

Proof by induction

- Induction base: \( T(0) = b = b + a \cdot 0 \)
- Induction hypothesis: \( T(n - 1) = b + a(n - 1) \) for \( n > 0 \)
- Inductive step for \( n > 0 \):

\[
T(n) = T(n - 1) + a \\
= b + a(n - 1) + a \\
= b + an
\]
A Generalization

The non-recursive pseudocode

function $T(n)$ (* integer $n \geq 0$ and reals $a$ and $b$ *)

$t = b$

for $i = 1$ to $n$

$t = t + a$

return $(t)$

Why it is the same as the recursive pseudocode?

- The recursive pseudocode implements the Top-Down evaluation
- The non-recursive pseudocode implements the Bottom-Up evaluation
A Generalization

The recursive formula

\[ T(n) = \begin{cases} 
  b & \text{for } n = 0 \text{ and a real number } b \\
  T(n - 1) + a & \text{for } n > 0 \text{ and a real number } a 
\end{cases} \]

The closed-form expression

\[ T(n) = b + an \]

Special cases

- **Non-negative integers:** \( b = 0 \land a = 1 \implies T(n) = n \)
- **Non-positive integers:** \( b = 0 \land a = -1 \implies T(n) = -n \)
- **Non-negative Even integers:** \( b = 0 \land a = 2 \implies T(n) = 2n \)
- **Positive Odd integers:** \( b = 1 \land a = 2 \implies T(n) = 2n + 1 \)
Powers of Two

The recursive formula

\[ P(n) = \begin{cases} 
  1 & \text{for } n = 0 \\
  2P(n - 1) & \text{for } n > 0 
\end{cases} \]

The recursive pseudocode

```plaintext
function P(n) (* integer \( n \geq 0 \) *)
    if \( n = 0 \)
      then return (1)
    else return (2P(n - 1))
```

The non-recursive pseudocode

```plaintext
function P(n) (* integer \( n \geq 0 \) *)
    p = 1
    for i = 1 to n
        p = 2 \cdot p
    return(p)
```
Powers of Two

The closed-form expression

\[ P(n) = 2^n \]

Proof by induction

- Induction base: \( P(0) = 1 = 2^0 \)
- Induction hypothesis: \( P(n - 1) = 2^{n-1} \) for \( n > 0 \)
- Inductive step for \( n > 0 \):

\[
\begin{align*}
P(n) &= 2P(n - 1) \\
    &= 2 \cdot 2^{n-1} \\
    &= 2^n
\end{align*}
\]
Factorials

The recursive formula

\[ F(n) = \begin{cases} 
1 & \text{for } n = 0 \\
nF(n - 1) & \text{for } n > 0 
\end{cases} \]

The recursive pseudocode

function \( F(n) \) (* integer \( n \geq 0 \) *)
    if \( n = 0 \)
      then return (1)
    else return (\( n \cdot F(n - 1) \))

The non-recursive pseudocode

function \( F(n) \) (* integer \( n \geq 0 \) *)
    \( f = 1 \)
    \( f = 1 \)
    for \( i = 1 \) to \( n \)
      \( f = i \cdot f \)
    return (\( f \))
The closed-form expression

\[ F(n) = n! \]

Proof by induction

- Induction base: \( F(0) = 1 = 0! \)
- Induction hypothesis: \( F(n - 1) = (n - 1)! \) for \( n > 0 \)
- Inductive step for \( n > 0 \):

\[
F(n) = nF(n - 1) \\
= n(n - 1)! \\
= n!
\]
The $\log_2$ Function

The recursive formula

$$L(n) = \begin{cases} 
0 & \text{for } n = 1 \\
L(n/2) + 1 & \text{for } n = 2^h \text{ and } h > 0
\end{cases}$$

The recursive pseudocode

function $L(n)$ (* power of two integer $n = 2^h$ for $h \geq 0$ *)

if $n = 1$

then return (0)

else return $(L(n/2) + 1)$

The non-recursive pseudocode

function $L(n)$ (* power of two integer $n = 2^h$ for $h \geq 0$ *)

$h = 0$

while $n > 1$

$n = n/2$ and $h = h + 1$

return($h$)
The \( \log_2 \) Function

The closed-form expression

\[
L(n) = \log_2(n)
\]

Proof by induction

- Induction base: \( L(1) = 0 = \log_2(1) \)
- Induction hypothesis: \( L(2^{h-1}) = \log_2(2^{h-1}) = h - 1 \) for \( h > 0 \)
- Inductive step: Assume \( n = 2^h \) for \( h > 0 \)

\[
\begin{align*}
L(n) &= L(2^h) \\
    &= L(2^{h-1}) + 1 \\
    &= \log_2(2^{h-1}) + 1 \\
    &= (h - 1) + 1 \\
    &= h \\
    &= \log_2(n)
\end{align*}
\]
The Sum $1 + 2 + \cdots + n$

The recursive formula

$$S(n) = \begin{cases} 
0 & \text{for } n = 0 \\
S(n - 1) + n & \text{for } n > 0
\end{cases}$$

The recursive pseudocode

function $S(n)$ (* integer $n \geq 0$ *)

if $n = 0$
then return $(0)$
else return $(S(n - 1) + n)$

The non-recursive pseudocode

function $S(n)$ (* integer $n \geq 0$ *)

$s = 0$

for $i = 1$ to $n$

$s = s + i$

return$(s)$
The Sum $1 + 2 + \cdots + n$

The closed-form expression

$$S(n) = \frac{n(n + 1)}{2}$$

Proof by induction

- Induction base: $S(0) = 0 = (0 \cdot 1)/2$
- Induction hypothesis: $S(n - 1) = ((n - 1)n)/2$ for $n > 0$
- Inductive step for $n > 0$:

$$S(n) = S(n - 1) + n$$
$$= \frac{(n - 1)n}{2} + n$$
$$= \frac{n^2 - n}{2} + \frac{2n}{2}$$
$$= \frac{n^2 + n}{2}$$
$$= \frac{n(n + 1)}{2}$$
The Sum $1 + 3 + \cdots + n$ For Odd $n$

**The recursive formula**

$$S(n) = \begin{cases} 
1 & \text{for } n = 1 \\
S(n - 2) + n & \text{for odd } n > 1 
\end{cases}$$

**The recursive pseudocode**

```plaintext
function S(n) (* odd integer $n \geq 1$ *)
    if $n = 1$
        then return (1)
    else return ($S(n - 2) + n$)
```

**The non-recursive pseudocode**

```plaintext
function S(n) (* odd integer $n \geq 1$ *)
    $s = 0$
    for $i = 1$ to $n$ step 2
        $s = s + i$
    return($s$)
```
The Sum $1 + 3 + \cdots + n$ For Odd $n$

Small values of $n$

\[
\begin{align*}
S(1) &= 1 = 1^2 = \left(\frac{1 + 1}{2}\right)^2 \\
S(3) &= S(1) + 3 = 1 + 3 = 4 = 2^2 = \left(\frac{3 + 1}{2}\right)^2 \\
S(5) &= S(3) + 5 = 4 + 5 = 9 = 3^2 = \left(\frac{5 + 1}{2}\right)^2 \\
S(7) &= S(5) + 7 = 9 + 7 = 16 = 4^2 = \left(\frac{7 + 1}{2}\right)^2 \\
S(9) &= S(7) + 9 = 16 + 9 = 25 = 5^2 = \left(\frac{9 + 1}{2}\right)^2 \\
S(11) &= S(9) + 11 = 25 + 11 = 36 = 6^2 = \left(\frac{11 + 1}{2}\right)^2
\end{align*}
\]
The Sum $1 + 3 + \cdots + n$ For Odd $n$

The closed-form expression

$$S(n) = \left( \frac{n + 1}{2} \right)^2$$

Proof by induction

- Induction base: $S(1) = 1 = \left( \frac{1+1}{2} \right)^2$
- Induction hypothesis: $S(n - 2) = \left( \frac{n-1}{2} \right)^2$ for odd $n > 1$
The Sum $1 + 3 + \cdots + n$ For Odd $n$

Proof by induction: inductive step for odd $n > 1$

$$S(n) = S(n - 2) + n$$

$$= \left( \frac{(n - 2) + 1}{2} \right)^2 + n$$

$$= \left( \frac{n - 1}{2} \right)^2 + \frac{4n}{4}$$

$$= \frac{n^2 - 2n + 1 + 4n}{4}$$

$$= \frac{n^2 + 2n + 1}{4}$$

$$= \frac{(n + 1)^2}{2^2}$$

$$= \left( \frac{n + 1}{2} \right)^2$$
A General Recursive Formula

The recursive formula
For real numbers \(a\) and \(b\) and a positive real number \(r \neq 1\)

\[
T(n) = \begin{cases} 
  b & \text{for } n = 0 \\
  rT(n-1) + a & \text{for } n > 0
\end{cases}
\]

The closed-form expression

\[
T(n) = br^n + a\frac{r^n - 1}{r - 1}
\]

Proof
- By induction on \(n \geq 0\)
A General Recursive Formula

**Bottom-Up evaluation**

\[ T(0) = b \]
\[ T(1) = rT(0) + a = r(b) + a \]
\[ T(2) = rT(1) + a = r(br + a) + a \]
\[ T(3) = rT(2) + a = r(br^2 + a(r + 1)) + a \]
\[ T(4) = rT(3) + a = r(br^3 + a(r^2 + r + 1)) + a = br^4 + a(r^3 + r^2 + r^1 + r^0) \]

\[ \vdots \]

\[ T(n) = br^n + a \sum_{i=0}^{n-1} r^i \]

\[ T(n) = br^n + a \frac{r^n - 1}{r - 1} \]
A General Recursive Formula

Top-Down evaluation

\[ T(n) = r^1 T(n - 1) + a \]

\[ = r^1 (rT(n - 2) + a) + a \]

\[ = r^2 (rT(n - 3) + a) + a(r + 1) \]

\[ = r^3 (rT(n - 4) + a) + a(r^2 + r + 1) \]

\[ = r^n T(0) + a(r^{n-1} + r^{n-2} + \cdots + r + 1) \]

\[ = br^n + a \sum_{i=0}^{n-1} r^i \]

\[ = br^n + a \frac{r^n - 1}{r - 1} \]
First Special Case: \( b = 1, a = 0, \text{ and } r \neq 1 \)

The recursive formula

\[
T(n) = \begin{cases} 
1 & \text{for } n = 0 \\
rT(n - 1) & \text{for } n \geq 1
\end{cases}
\]

The closed-form expression

\[
T(n) = br^n + a \frac{r^n - 1}{r - 1} \\
= 1 \cdot r^n + 0 \cdot \frac{r^n - 1}{r - 1} \\
= r^n
\]
Second Special Case: \( b = 0, \ a = 1, \) and \( r = 2 \)

The recursive formula

\[
T(n) = \begin{cases} 
0 & \text{for } n = 0 \\
2T(n - 1) + 1 & \text{for } n \geq 1 
\end{cases}
\]

The closed-form expression

\[
T(n) = br^n + a \frac{r^n - 1}{r - 1} = 0 \cdot r^n + 1 \cdot \frac{2^n - 1}{2 - 1} = 2^n - 1
\]
Third Special Case: $b = 0$, $a = 1/2$, and $r = 1/2$:

The recursive formula

$$T(n) = \begin{cases} 0 & \text{for } n = 0 \\ \frac{1}{2} T(n - 1) + \frac{1}{2} & \text{for } n \geq 1 \end{cases}$$

The closed-form expression

$$T(n) = br^n + a \frac{r^n - 1}{r - 1}$$

$$= 0 \cdot r^n + \frac{1}{2} \cdot \frac{(\frac{1}{2})^n - 1}{\frac{1}{2} - 1}$$

$$= \frac{1}{2} \cdot \frac{1 - (\frac{1}{2})^n}{\frac{1}{2}}$$

$$= 1 - \left(\frac{1}{2}\right)^n$$
Tower of Hanoi

Definition by example
https://www.youtube.com/watch?v=5Wn4EboLrMM

General definition

- There are three pegs (rods) called A, B, and C and \( n \geq 1 \) disks of different sizes
- Initially all the disks are located on peg A ordered from the largest at the bottom to the smallest at the top
- A **legal move** takes any top disk and moves it to another peg as long as it is not placed on top of a smaller disk
- **Goal:** Move the \( n \) disks from A to B using only legal moves
- **Efficiency:** Move the disks with as few as possible legal moves

History, Background, and beyond
Tower of Hanoi

**Solution: demo**

- [https://www.mathsisfun.com/games/towerofhanoi.html](https://www.mathsisfun.com/games/towerofhanoi.html)

**General Recursive solution**

- **Initial call:** Move \( n \geq 1 \) disks from \( A \) to \( B \)

- **Recursive base:** For \( n = 1 \) move the single disk from \( A \) to \( B \)

- **Recursive step:** Assume \( k > 1 \) disks are to be moved from peg \( X \) to peg \( Y \) for \( X \neq Y \) and \( \{X, Y, Z\} = \{A, B, C\} \):
  - Move the top \( k - 1 \) disks from \( X \) to \( Z \notin \{X, Y\} \)
  - Move the top disk from \( X \) to \( Y \)
  - Move the top \( k - 1 \) disks from \( Z \) to \( Y \)

**Recursive solution for four disks**

- [https://www.youtube.com/watch?v=YstLjLCGmgg](https://www.youtube.com/watch?v=YstLjLCGmgg)
Correctness: proof by induction (sketch)

**Induction base:** When $n = 1$, a largest top disk can be legally moved from a peg to an empty peg.

**Induction hypothesis:** The smallest $1 \leq k < n$ disks can be legally moved from a peg to any of the other two pegs.

**Inductive step:**
- The $n - 1$ smallest disks are legally moved from peg $A$ to peg $C$ by the induction hypothesis.
- The largest disk is legally moved from peg $A$ to the empty peg $B$.
- The $n - 1$ smallest disks are legally moved from peg $C$ to peg $B$, which has the largest disk by the induction hypothesis.
Number of moves

Let $M(n)$ be the number of legal moves made by the recursive solution for $n \geq 1$ disks.

Trivially, $M(1) = 1$ and by definition $M(0) = 0$.

Recursively, $M(n) = 2M(n - 1) + 1$.

By the generalized formula (second special case)

$M(n) = 2^n - 1$
Binomial Coefficients: The Recursive Formula

**Definition**

for \( n \geq 0 \) \( \binom{n}{0} = \binom{n}{n} = 1 \)

for \( n \geq 1 \) and \( 1 \leq k \leq n - 1 \) \( \binom{n}{k} = \frac{n-1}{k-1} + \binom{n-1}{k} \)

**Closed-form expression**

\[
\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1) \cdots (n-k+1)}{k(k-1) \cdots 2}
\]
Proof By Induction

Induction base for any \( n \geq 1 \) and \( k = 0 \) or \( k = n \):

\[
\binom{n}{0} = 1 = \frac{n!}{1 \cdot n!} = \frac{n!}{0! \cdot n!} = \frac{n!}{0! \cdot (n - 0)!}
\]

\[
\binom{n}{n} = 1 = \frac{n!}{n! \cdot 1} = \frac{n!}{n! \cdot 0!} = \frac{n!}{n! \cdot (n - n)!}
\]

Induction hypothesis for \( 1 \leq m < n \) and \( 1 \leq h \leq m \):

\[
\binom{m}{h} = \frac{m!}{h!(m-h)!}
\]
Proof By Induction

**Inductive step for** \( n > 1 \) **and** \( 0 < k < n \):**

\[
\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} = \frac{(n-1)!}{(k-1)!(n-k)!} + \frac{(n-1)!}{k!(n-k-1)!} = \frac{k(n-1)!}{k!(n-k)!} + \frac{(n-k)(n-1)!}{(n-k)k!(n-k-1)!} = \frac{k(n-1)!}{k!(n-k)!} + \frac{(n-k)(n-1)!}{k!(n-k)!} = \frac{k(n-1)! + (n-k)(n-1)!}{k!(n-k)!} = \frac{(k + (n-k))(n-1)!}{k!(n-k)!} = \frac{n(n-1)!}{k!(n-k)!} = \frac{n!}{k!(n-k)!}
\]
The Fibonacci Numbers

The sequence:

\[ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 \ldots \]

The recursive definition

\[ F_n = \begin{cases} 
0 & \text{for } n = 0 \\
1 & \text{for } n = 1 \\
F_{n-1} + F_{n-2} & \text{for } n \geq 2 
\end{cases} \]
The Fibonacci Numbers

The recursive pseudocode

function $F(n)$ (* integer $n \geq 2$ *)
  if $n = 0$ then return (0)
  if $n = 1$ then return (1)
  otherwise return ($F(n - 1) + F(n - 2)$)

The non-recursive pseudocode

function $F(n)$ (* integer $n \geq 2$ *)
  $F_0 = 0$
  $F_1 = 1$
  for $i = 2$ to $n$
    $F_i = F_{i-1} + F_{i-2}$
  return ($F_n$)
Fibonacci Numbers - The Original Problem

Story

- A young pair of rabbits (one of each sex) is placed on an island
- A pair of rabbits does not breed until they are 2 months old
- After they are two months old, each pair of rabbits produces another pair each month
- No rabbits ever die

Problem

- How many pairs of rabbits are there on the island after $n$ months?

Solution

- There are $F_n$ pairs of rabbits on the island after $n$ months
Online Resources

The original story
https://www.youtube.com/watch?v=sjQlW6cH3Ko

The Fibonacci’s soup
https://me.me/i/
todays-special-fibonaccis-soup-ingredients-yesterdays-soup-the-day-before-20394428

Some basics
https://www.youtube.com/watch?v=ZC-d4dKTyKw
Additional Online Resources

Texts

- **Math is Fun:**
  https://www.mathsisfun.com/numbers/fibonacci-sequence.html

- **The life and numbers of Fibonacci:**
  https://plus.maths.org/content/life-and-numbers-fibonacci

- **Wikipedia:**
  https://en.wikipedia.org/wiki/Fibonacci_number

- **Fibonacci Numbers and the Golden Section:**
  http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fib.html

Videos

- **The magic of Fibonacci numbers:**
  https://www.youtube.com/watch?v=SjSHVDfXHQ4&vl=ja

- **The Fibonacci Sequence and Experiences with Learning:**
  https://www.youtube.com/watch?v=uk6CLffEuZM
### Three Consecutive Fibonacci Numbers

#### Identity for $n \geq 1$

\[
F_{n-1}F_{n+1} = F_n^2 + (-1)^n
\]

#### Correctness for small $n$

\[
\begin{align*}
F_0F_2 &= 0 \cdot 1 = 1^2 - 1 = F_1^2 + (-1)^1 \\
F_1F_3 &= 1 \cdot 2 = 1^2 + 1 = F_2^2 + (-1)^2 \\
F_2F_4 &= 1 \cdot 3 = 2^2 - 1 = F_3^2 + (-1)^3 \\
F_3F_5 &= 2 \cdot 5 = 3^2 + 1 = F_4^2 + (-1)^4 \\
F_4F_6 &= 3 \cdot 8 = 5^2 - 1 = F_5^2 + (-1)^5 \\
F_5F_7 &= 5 \cdot 13 = 8^2 + 1 = F_6^2 + (-1)^6 \\
F_6F_8 &= 8 \cdot 21 = 13^2 - 1 = F_7^2 + (-1)^7
\end{align*}
\]

#### “Almost” like integers and powers of 2

\[
(n - 1)(n + 1) = n^2 - 1
\]

\[
2^{n-1} \cdot 2^{n+1} = (2^n)^2
\]
Proof By Induction

Notations

\[ L(n) = F_{n-1}F_{n+1} \]
\[ R(n) = F_n^2 + (-1)^n \]

The induction base: \( n = 1 \)

\[ L(1) = F_0F_2 = 0 \cdot 1 = 0 = 1 - 1 = F_1^2 + (-1)^1 = R(1) \]

The induction hypothesis: \( L(n-1) = R(n-1) \) for \( n > 1 \)

\[ F_{n-2}F_n = F_{n-1}^2 + (-1)^{n-1} \]
\[ F_{n-1}^2 = F_{n-2}F_n - (-1)^{n-1} = F_{n-2}F_n + (-1)^n \]
Proof By Induction

The inductive step: \( L(n) = R(n) \) for \( n > 1 \)

\[
L(n) = F_{n-1}F_{n+1} \\
= F_{n-1}(F_{n-1} + F_n) \\
= F_{n-1}^2 + F_{n-1}F_n \\
= F_{n-2}F_n + (-1)^n + F_{n-1}F_n \quad \text{(* the induction hypothesis *)} \\
= (F_{n-2} + F_{n-1})F_n + (-1)^n \\
= F_n^2 + (-1)^n \\
= R(n)
\]
The Missing or the Extra Part Paradox

- **64 = 65?**
  - [https://www.youtube.com/watch?v=QTxQDjGQh-0](https://www.youtube.com/watch?v=QTxQDjGQh-0)
  - [https://www.youtube.com/watch?v=ee0dksxkog0](https://www.youtube.com/watch?v=ee0dksxkog0)
  - **Explanation:** [http://yozh.org/2011/05/15/an-area-paradox/](http://yozh.org/2011/05/15/an-area-paradox/)

- **More Missing Squares:**
  - [https://www.youtube.com/watch?v=ExUV3GOTDqE](https://www.youtube.com/watch?v=ExUV3GOTDqE)
  - [https://www.youtube.com/watch?v=h_aRDdpulMU](https://www.youtube.com/watch?v=h_aRDdpulMU)
  - [https://www.youtube.com/watch?v=eFw0878Ig-A](https://www.youtube.com/watch?v=eFw0878Ig-A)

- **60=58=59?**
  - [https://www.youtube.com/watch?v=iMgFDhpa000](https://www.youtube.com/watch?v=iMgFDhpa000)

- **The Vanishing Square:**
  - [https://www.youtube.com/watch?v=BqhuX52QEIs](https://www.youtube.com/watch?v=BqhuX52QEIs)

- **The Infinite Chocolate:**
  - [https://www.youtube.com/watch?v=dmBsPgPu0Wc](https://www.youtube.com/watch?v=dmBsPgPu0Wc)
  - [https://www.youtube.com/watch?v=7Yo6mwSBlAs](https://www.youtube.com/watch?v=7Yo6mwSBlAs)
  - [https://www.youtube.com/watch?v=tf1YbDV0508](https://www.youtube.com/watch?v=tf1YbDV0508)
Sum of First $n$ Fibonacci numbers

Identity for $n \geq 1$

$$\sum_{i=1}^{n} F_i = F_1 + F_2 + \cdots + F_n = F_{n+2} - 1$$

Correctness for small $n$

- $F_1 = 1$
- $F_1 + F_2 = 1 + 1 = 2 = 3 - 1 = F_4 - 1$
- $F_1 + F_2 + F_3 = 1 + 1 + 2 = 4 = 5 - 1 = F_5 - 1$
- $F_1 + F_2 + F_3 + F_4 = 1 + 1 + 2 + 3 = 7 = 8 - 1 = F_6 - 1$
- $F_1 + F_2 + F_3 + F_4 + F_5 = 1 + 1 + 2 + 3 + 5 = 12 = 13 - 1 = F_7 - 1$

“Almost” like powers of 2

$$\sum_{i=0}^{n} 2^i = 1 + 2 + 4 + \cdots + 2^n = 2^{n+1} - 1$$
Proof By Induction

**Notations**

\[ L(n) = F_1 + F_2 + \cdots + F_n \]
\[ R(n) = F_{n+2} - 1 \]

**The induction base:** \( n = 1 \)

\[ L(1) = F_1 = 1 = 2 - 1 = F_3 - 1 = R(1) \]

**The induction hypothesis:** \( L(n - 1) = R(n - 1) \) for \( n > 1 \)

\[ F_1 + F_2 + \cdots + F_{n-1} = F_{n+1} - 1 \]
Proof By Induction

The inductive step: \( L(n) = R(n) \) for \( n > 1 \)

\[
L(n) = 1 + 1 + 2 + \cdots + F_{n-1} + F_n \\
= L(n-1) + F_n \\
= R(n-1) + F_n \\
= (F_{n+1} - 1) + F_n \\
= (F_n + F_{n+1}) - 1 \\
= F_{n+2} - 1 \\
= R(n)
\]
A Generalized Fibonacci Sequence

Definition

\[ G_n = \begin{cases} 
  a & \text{for } n = 0 \\
  b & \text{for } n = 1 \\
  G_{n-1} + G_{n-2} & \text{for } n \geq 2 
\end{cases} \]

Identity

\[
\sum_{i=0}^{n} G_i = G_0 + G_1 + \cdots + G_n = G_{n+2} - G_1 = G_{n+2} - b
\]

The Fibonacci sequence special case

- \[ \sum_{i=0}^{n} G_i = G_{n+2} - 1 \] for \( a = 0 \) and \( b = 1 \)

Example: \( a = 3 \) and \( b = 2 \)

- The sequence: 3, 2, 5, 7, 12, 19, 31, 50, 81, \ldots
- The identity: \( 3 + 2 + 5 + 7 + 12 + 19 + 31 = 79 = 81 - 2 \)
Proof by Induction (Sketch)

The inductive step: \( R(n) = L(n) \) for \( n > 1 \)

\[
G_{n+2} - G_1 = G_n + G_{n+1} - G_1
\]
\[
= G_n + G_{n-1} + G_n - G_1
\]
\[
= G_n + G_{n-1} + G_{n-2} + G_{n-1} - G_1
\]
\[
= G_n + G_{n-1} + G_{n-2} + G_{n-3} + G_{n-2} - G_1
\]
\[\vdots\]
\[
= G_n + G_{n-1} + G_{n-2} + \cdots + G_i + G_{i-1} + G_i - G_1
\]
\[\vdots\]
\[
= G_n + G_{n-1} + G_{n-2} + \cdots + G_1 + G_0 + G_1 - G_1
\]
\[
= G_n + G_{n-1} + G_{n-2} + \cdots + G_1 + G_0
\]
Sum of First $n$ Odd-Indexed Fibonacci numbers

Identity for $n \geq 1$

$$\sum_{i=1}^{n} F_{2i-1} = F_1 + F_3 + \cdots + F_{2n-1} = F_{2n}$$

Correctness for small $n$

\begin{align*}
F_1 &= 1 \\
F_1 + F_3 &= 1 + 2 = 3 = F_4 \\
F_1 + F_3 + F_5 &= 1 + 2 + 5 = 8 = F_6 \\
F_1 + F_3 + F_5 + F_7 &= 1 + 2 + 5 + 13 = 21 = F_8 \\
F_1 + F_3 + F_5 + F_7 + F_9 &= 1 + 2 + 5 + 13 + 34 = 55 = F_{10}
\end{align*}

“Almost” like powers of 2

$$\sum_{i=0}^{n} 2^{2i-1} = 2 + 8 + 32 + \cdots + 2^{2n-1} = \frac{2}{3}(2^{2n} - 1)$$
Sketches of Two Proofs

Expanding $F_{2n}$

\[
F_{2n} = F_{2n-1} + F_{2n-2} \\
= F_{2n-1} + F_{2n-3} + F_{2n-4} \\
= F_{2n-1} + F_{2n-3} + F_{2n-5} + F_{2n-6} \\
\vdots \\
= F_{2n-1} + F_{2n-3} + F_{2n-5} + \cdots + F_3 + F_1 + F_0 \\
= F_{2n-1} + F_{2n-3} + F_{2n-5} + F_3 + F_1
\]

Evaluating the sum

\[
\sum_{i=1}^{n} F_{2i-1} = F_1 + F_3 + F_5 + \cdots + F_{2n-3} + F_{2n-1} \\
= (F_2 - F_0) + (F_4 - F_2) + (F_6 - F_4) + \cdots + (F_{2n-2} - F_{2n-4}) + (F_{2n} - F_{2n-2}) \\
= F_{2n} - F_0 \\
= F_{2n}
\]
Sum of First $n$ Squares of Fibonacci numbers

Identity for $n \geq 1$

$$\sum_{i=1}^{n} F_i^2 = F_1^2 + F_2^2 + \cdots + F_n^2 = F_n F_{n+1}$$

Correctness for small $n$

- $F_1^2 = 1^2 = 1 = 1 \cdot 1 = F_1 F_2$
- $F_1^2 + F_2^2 = 1^2 + 1^2 = 2 = 1 \cdot 2 = F_2 F_3$
- $F_1^2 + F_2^2 + F_3^2 = 1^2 + 1^2 + 2^2 = 6 = 2 \cdot 3 = F_3 F_4$
- $F_1^2 + F_2^2 + F_3^2 + F_4^2 = 1^2 + 1^2 + 2^2 + 3^2 = 15 = 3 \cdot 5 = F_4 F_5$
- $F_1^2 + F_2^2 + F_3^2 + F_4^2 + F_5^2 = 1^2 + 1^2 + 2^2 + 3^2 + 5^2 = 40 = 5 \cdot 8 = F_5 F_6$

"Almost" like powers of 2

$$\sum_{i=0}^{n} (2^i)^2 = \sum_{i=0}^{n} 4^i = \frac{4^{n+1} - 1}{3} = \frac{2}{3} 2^n 2^{n+1} - \frac{1}{3}$$
Proof By Induction

Notations

\[ L(n) = F_1^2 + F_2^2 + \cdots + F_n^2 \]
\[ R(n) = F_n F_{n+1} \]

The induction base: \( n = 1 \)

\[ L(1) = F_1^2 = 1^2 = 1 = 1 \cdot 1 = F_1 F_2 = R(1) \]

The induction hypothesis: \( L(n - 1) = R(n - 1) \) for \( n > 1 \)

\[ F_1^2 + F_2^2 + \cdots + F_{n-1}^2 = F_{n-1} F_n \]
Proof By Induction

The inductive step: \( L(n) = R(n) \) for \( n > 1 \)

\[
L(n) = F_1^2 + F_2^2 + \cdots + F_{n-1}^2 + F_n^2 \\
= L(n-1) + F_n^2 \\
= R(n-1) + F_n^2 \\
= F_{n-1}F_n + F_n^2 \\
= F_n(F_{n-1} + F_n) \\
= F_nF_{n+1} \\
= R(n)
\]
Proof Without Words: \[ \sum_{i=1}^{n} F_i^2 = F_n F_{n+1} \]
The Golden Ratio

Definition

- The golden ratio is the positive root of the equation $x^2 = 1 + x$

Formula

$$\phi = \frac{1 + \sqrt{5}}{2} = 1.618 \ldots$$

The Fractional Part of the Golden Ratio

$$\phi^2 = 1 + \phi \implies \phi = \frac{1}{\phi} + 1 \implies \phi - 1 = \frac{1}{\phi} = 0.618 \ldots$$

Online Resources

- https://www.youtube.com/watch?v=fmaVqkR0ZXg
- https://www.youtube.com/watch?v=6nSfJEDZ_WM
The Kepler Triangle

\[ \phi^2 = 1^2 + (\sqrt{\phi})^2 \]
Two infinite expressions for $\phi$

\[
\phi = \sqrt{1 + \sqrt{1 + \sqrt{1 + \cdots}}} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \cdots}}}
\]

**Proof I**

- Define $S = \sqrt{1 + \sqrt{1 + \sqrt{1 + \cdots}}}

\[
S^2 = 1 + S \implies S = \phi
\]

**Proof II**

- Define $F = 1 + \frac{1}{1 + \frac{1}{1 + \cdots}}

\[
F = 1 + \frac{1}{F} \implies F = \phi
\]
Solving the Equation $x = 1 + \frac{1}{x}$

The iteration method: $\phi = 1.618033988749894848204586834 \ldots$

- Initially: $x_1 = 1$
- Iteratively: $x_i = 1 + \frac{1}{x_{i-1}}$ for $i > 1$
  * $x_2 = 1 + \frac{1}{1} = 2$
  * $x_3 = 1 + \frac{1}{2} = \frac{3}{2} = 1.5$
  * $x_4 = 1 + \frac{1}{3/2} = \frac{5}{3} \approx 1.667$
  * $x_5 = 1 + \frac{1}{5/3} = \frac{8}{5} = 1.6$
  * $x_6 = 1 + \frac{1}{8/5} = \frac{13}{8} = 1.625$
  * $x_7 = 1 + \frac{1}{13/8} = \frac{21}{13} \approx 1.61538461538$
  * $x_8 = 1 + \frac{1}{21/13} = \frac{34}{21} \approx 1.61904761905$
  * $x_9 = 1 + \frac{1}{34/21} = \frac{55}{34} \approx 1.61764705882$
  * $x_{20} = 1 + \frac{1}{6765/4181} = \frac{10946}{6765} \approx 1.61803399852$
Solving the Equation $x = 1 + \frac{1}{x}$

$x_1 = 1$

$x_2 = 1 + \frac{1}{1}$

$x_3 = 1 + \frac{1}{1 + \frac{1}{1}}$

$x_4 = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1}}}$

$x_5 = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1}}}}$

$x = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \ldots}}}}}}$
Solving the Equation $x = 1 + \frac{1}{x}$

**Theorem**

$$x_n = \frac{F_{n+1}}{F_n}$$

**Proof By Induction**

- **Induction Base:** $x_1 = 1 = \frac{1}{1} = \frac{F_2}{F_1}$
- **Assume correctness for $x_{n-1}$, prove correctness for $x_n$**

  $$x_n = 1 + \frac{1}{x_{n-1}} = 1 + \frac{1}{\frac{F_{n-1}}{F_{n-2}}} = 1 + \frac{F_{n-1}}{F_n} = \frac{F_n + F_{n-1}}{F_n} = \frac{F_{n+1}}{F_n}$$
Approximating $\phi$ with $\sqrt{1 + \sqrt{1 + \sqrt{1 + \cdots}}}$

The first 3 digits of the fractional part of $\phi = 1.6180339887\ldots$

- Initially: $x_1 = \sqrt{1} = 1$
- Iteratively: $x_i = \sqrt{1 + x_{i-1}}$ for $i > 1$
  * $x_2 = \sqrt{1 + 1.000} = \sqrt{2.000} \approx 1.414$
  * $x_3 = \sqrt{1 + 1.414} = \sqrt{2.414} \approx 1.554$
  * $x_4 = \sqrt{1 + 1.554} = \sqrt{2.554} \approx 1.598$
  * $x_5 = \sqrt{1 + 1.598} = \sqrt{2.598} \approx 1.612$
  * $x_6 = \sqrt{1 + 1.612} = \sqrt{2.612} \approx 1.616$
  * $x_7 = \sqrt{1 + 1.616} = \sqrt{2.616} \approx 1.617$
  * $x_8 = \sqrt{1 + 1.617} = \sqrt{2.617} \approx 1.618$
  * $x_8 = \sqrt{1 + 1.618} = \sqrt{2.618} \approx 1.618$
The Fibonacci Numbers and the Golden Ratio

The two roots of the equation $x^2 - x - 1 = 0$

- The positive root: $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$
- The negative root: $\hat{\phi} = \frac{1-\sqrt{5}}{2} = 1 - \phi \approx -0.618$

Fibonacci numbers as a function of the Golden Ratio

- $F_k = \frac{\phi^k - \hat{\phi}^k}{\sqrt{5}}$
- $F_{k+1} = \phi F_k + \hat{\phi}^k$
- $|\hat{\phi}| < 1 \implies F_k = \frac{\phi^k}{\sqrt{5}}$ rounded to the nearest integer