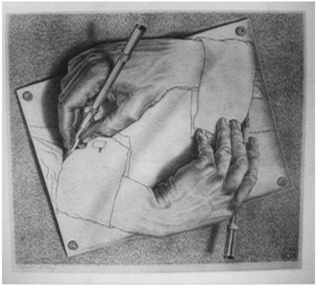
Recursion and Proofs by Induction

CS1200, CSE IIT Madras

Meghana Nasre

March 20, 2020

Recursion



Drawing Hands by M. C. Escher

- Familiar recursive functions
- Some important recursive functions
- Proving closed form solutions using induction

Some familiar examples

$\boxed{\text{Factorial Function}}$

$$
\begin{aligned}
\text{fact}(n) &= 1 && \text{if } n = 1 \\
&= n \cdot \text{fact}(n-1) && \textit{otherwise}
\end{aligned}
$$

Some familiar examples

Factorial Function

$$\begin{aligned} \text{fact}(n) &= 1 && \text{if } n = 1 \\ &= n \cdot \text{fact}(n-1) && \textit{otherwise} \end{aligned}$$

Fibonacci Sequence

$$0, 1, 1, 2, 3, 5, 8, \ldots$$

$$\begin{aligned} f(n) &= n && \text{if } n = 0 \text{ or } n = 1 \\ &= f(n-1) + f(n-2) && \textit{otherwise} \end{aligned}$$

Some more examples of recursive functions

$\boxed{\text{gcd(a, b)}}$: assume $a \geq b$

$$
\begin{aligned}
\text{gcd}(a, b) &= a && \text{if } b = 0 \\
&= \text{gcd}(b, a \bmod b) && \text{otherwise}
\end{aligned}
$$

Some more examples of recursive functions

$\boxed{\text{gcd(a, b)}}$: assume $a \geq b$

$$
\begin{aligned}
\text{gcd}(a, b) &= a && \text{if } b = 0 \\
&= \text{gcd}(b, a \bmod b) && \text{otherwise}
\end{aligned}
$$

$\boxed{\sum_{i=0}^{n} i}$

$$
\begin{aligned}
\sum_{i=0}^{n} i &= 0 && \text{if } n = 0 \\
&= n + \sum_{i=0}^{n-1} i && \text{otherwise}
\end{aligned}
$$

Proving bounds on recursive formulas using induction

An upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) < 2^n$.

An upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) < 2^n$.

$\boxed{\text{Base Case:}}$ $n = 0, n = 1$

An upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) < 2^n$.

$\boxed{\text{Base Case:}}$ $n = 0, n = 1$ verify

$\boxed{\text{Ind Hyp:}}$ Assume that the claim holds for $i = 0, \ldots, k$.

An upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) < 2^n$.

$\boxed{\text{Base Case:}}$ $n = 0, n = 1$ verify

$\boxed{\text{Ind Hyp:}}$ Assume that the claim holds for $i = 0, \ldots, k$.

$$f(n) = f(n-1) + f(n-2)$$

An upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) < 2^n$.

$\boxed{\text{Base Case:}}$ $n = 0, n = 1$ verify

$\boxed{\text{Ind Hyp:}}$ Assume that the claim holds for $i = 0, \ldots, k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &< 2^{n-1} + 2^{n-2} \end{aligned}$$

by strong induction

An upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) < 2^n$.

Base Case: $n = 0, n = 1$ verify

Ind Hyp: Assume that the claim holds for $i = 0, \ldots, k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &< 2^{n-1} + 2^{n-2} \qquad \text{by strong induction} \\ &< 2^{n-1} + 2^{n-1} = 2 \cdot 2^{n-1} = 2^n. \end{aligned}$$

An upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) < 2^n$.

$\boxed{\text{Base Case:}}$ $n = 0, n = 1$ verify

$\boxed{\text{Ind Hyp:}}$ Assume that the claim holds for $i = 0, \ldots, k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &< 2^{n-1} + 2^{n-2} && \text{by strong induction} \\ &< 2^{n-1} + 2^{n-1} = 2 \cdot 2^{n-1} = 2^n. \end{aligned}$$

$\boxed{\text{Tighter Bounds}}$

- $f(n) \leq 2^{n-1}$ for all $n \geq 1$

Claim: The n-th fibonacci number $f(n) < 2^n$.

Base Case: $n = 0, n = 1$ verify

Ind Hyp: Assume that the claim holds for $i = 0, \ldots, k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &< 2^{n-1} + 2^{n-2} && \text{by strong induction} \\ &< 2^{n-1} + 2^{n-1} = 2 \cdot 2^{n-1} = 2^n. \end{aligned}$$

Tighter Bounds

- $f(n) \leq 2^{n-1}$ for all $n \geq 1$
- $f(n) \leq \phi^{n-1}$ for all $n \geq 1$; $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$
  Does the same technique as above suffice to prove the second bound?

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

$\boxed{\text{Base Case:}}$ $n = 2, n = 3$

$f(2) = 1 \leq \phi^1 \approx 1.618$

$f(3) = 2 \leq \phi^2 \approx 2.618$

$\boxed{\text{Ind Hyp:}}$ Assume that the claim holds for $i = 2, \ldots, k$.

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

Base Case: $n = 2, n = 3$

$f(2) = 1 \leq \phi^1 \approx 1.618$

$f(3) = 2 \leq \phi^2 \approx 2.618$

Ind Hyp: Assume that the claim holds for $i = 2, \ldots, k$.

$$f(n) = f(n-1) + f(n-2)$$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

$\boxed{\text{Base Case:}}$ $n = 2, n = 3$

$f(2) = 1 \leq \phi^1 \approx 1.618$

$f(3) = 2 \leq \phi^2 \approx 2.618$

$\boxed{\text{Ind Hyp:}}$ Assume that the claim holds for $i = 2, \ldots, k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &\leq \phi^{n-1} + \phi^{n-2} \end{aligned}$$

by strong induction

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

Base Case: $n = 2, n = 3$

$f(2) = 1 \leq \phi^1 \approx 1.618$

$f(3) = 2 \leq \phi^2 \approx 2.618$

Ind Hyp: Assume that the claim holds for $i = 2, \ldots, k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &\leq \phi^{n-1} + \phi^{n-2} && \text{by strong induction} \\ &\leq 2 \cdot \phi^{n-1} && \text{similar to above proof} \end{aligned}$$

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

Base Case: $n = 2, n = 3$

$f(2) = 1 \leq \phi^1 \approx 1.618$

$f(3) = 2 \leq \phi^2 \approx 2.618$

Ind Hyp: Assume that the claim holds for $i = 2, \ldots, k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) & \\ &\leq \phi^{n-1} + \phi^{n-2} & \text{by strong induction} \\ &\leq 2 \cdot \phi^{n-1} & \text{similar to above proof} \end{aligned}$$

!! However the above does not help to prove the claim.
Hence we use some properties of ϕ.

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

$\boxed{\text{Ind Hyp:}}$ Assume that the claim holds for all values $i = 2, \ldots k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &\leq \phi^{n-2} + \phi^{n-3} \end{aligned}$$

by strong induction

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

Ind Hyp: Assume that the claim holds for all values $i = 2, \ldots k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &\leq \phi^{n-2} + \phi^{n-3} \end{aligned}$$

by strong induction

Note that ϕ (golden ratio) is a root of the equality

$$x^2 - x - 1 = 0$$

Thus we have $\phi + 1 = \phi^2$.

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

Ind Hyp: Assume that the claim holds for all values $i = 2, \ldots k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &\leq \phi^{n-2} + \phi^{n-3} && \text{by strong induction} \\ &\leq \phi^{n-3}(\phi + 1) = \phi^{n-3} \cdot \phi^2 = \phi^{n-1} \end{aligned}$$

Note that ϕ (golden ratio) is a root of the equality

$$x^2 - x - 1 = 0$$

Thus we have $\phi + 1 = \phi^2$.

Another upper bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \leq \phi^{n-1}$ for $n \geq 2$.

Ind Hyp: Assume that the claim holds for all values $i = 2, \ldots k$.

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ &\leq \phi^{n-2} + \phi^{n-3} && \text{by strong induction} \\ &\leq \phi^{n-3}(\phi + 1) = \phi^{n-3} \cdot \phi^2 = \phi^{n-1} \end{aligned}$$

Hence proved!

Note that ϕ (golden ratio) is a root of the equality

$$x^2 - x - 1 = 0$$

Thus we have $\phi + 1 = \phi^2$.

A lower bound on $f(n)$

Claim: The n-th fibonacci number $f(n) \geq \phi^{n-2}$ for $n \geq 2$.

Ex: complete the proof.

Ex: Read here about the Golden Ratio ϕ.

Recursively defined functions

A recursively defined function for non-negative integers as its domain:

- Basis step: Define the function for first k positive integers.
- Recursive step: Define the function for $i > k$ using function value at smaller integers.

Recursively defined functions

A recursively defined function for non-negative integers as its domain:

- Basis step: Define the function for first k positive integers.
- Recursive step: Define the function for $i > k$ using function value at smaller integers.

Recursive functions are **well-defined**.
That is, value of the function at any integer is determined unambiguously.

Recursively defined functions

A recursively defined function for non-negative integers as its domain:

- Basis step: Define the function for first k positive integers.
- Recursive step: Define the function for $i > k$ using function value at smaller integers.

Recursive functions are **well-defined**.
That is, value of the function at any integer is determined unambiguously.

Ex: For the functions below, determine if they are well-defined and if yes, find a (non-recursive) formula for them and prove your formula using induction.

- $h(0) = 0; h(n) = 2h(n-2) \quad \text{for } n \geq 1.$
- $g(0) = 0; g(n) = g(n-1) - 1 \quad \text{for } n \geq 1.$

Some important recursive functions

A fast growing function: Ackermann function

$$
A(m,n) = \begin{cases} 2n & \text{if } m = 0 \\ 0 & \text{if } m \geq 1 \text{ and } n = 0 \\ 2 & \text{if } m \geq 1 \text{ and } n = 1 \\ A(m-1, A(m, n-1)) & \text{if } m \geq 1 \text{ and } n \geq 2 \end{cases}
$$

A fast growing function: Ackermann function

$$
\begin{aligned}
A(m,n) &= 2n && \text{if } m = 0\\
&= 0 && \text{if } m \geq 1 \text{ and } n = 0\\
&= 2 && \text{if } m \geq 1 \text{ and } n = 1\\
&= A(m-1, A(m, n-1)) && \text{if } m \geq 1 \text{ and } n \geq 2
\end{aligned}
$$

Ex: Solve the following.

- Compute $A(1,1)$ and $A(2,2)$.
- Guess a value for $A(1,n)$ for $n \geq 1$ and prove your answer using induction on n.

A fast growing function: Ackermann function

$$
A(m,n) = \begin{cases} 2n & \text{if } m = 0 \\ 0 & \text{if } m \geq 1 \text{ and } n = 0 \\ 2 & \text{if } m \geq 1 \text{ and } n = 1 \\ A(m-1, A(m, n-1)) & \text{if } m \geq 1 \text{ and } n \geq 2 \end{cases}
$$

Ex: Solve the following.

- Compute $A(1,1)$ and $A(2,2)$.
- Guess a value for $A(1,n)$ for $n \geq 1$ and prove your answer using induction on n.
- Can you compute $A(2,3)$?

A slow growing function: iterated logarithm

$$
\begin{aligned}
\log^{(k)}(n) &= n && \text{if } k = 0\\
&= \log(\log^{(k-1)}(n)) && \text{if } \log^{(k-1)}(n) \text{ is defined and is positive}\\
&= \text{undefined} && \text{otherwise}
\end{aligned}
$$

A slow growing function: iterated logarithm

$$
\begin{aligned}
\log^{(k)}(n) &= n && \text{if } k = 0 \\
&= \log(\log^{(k-1)}(n)) && \text{if } \log^{(k-1)}(n) \text{ is defined and is positive} \\
&= \text{undefined} && \text{otherwise}
\end{aligned}
$$

Note: $\log^{(k)}(n)$ is NOT $\log(n) \cdot \log(n) \ldots \log(n)$, k times.
Assume base of logarithm is 2.

A slow growing function: iterated logarithm

$$
\begin{aligned}
\log^{(k)}(n) &= n && \text{if } k = 0 \\
&= \log(\log^{(k-1)}(n)) && \text{if } \log^{(k-1)}(n) \text{ is defined and is positive} \\
&= \text{undefined} && \text{otherwise}
\end{aligned}
$$

Note: $\log^{(k)}(n)$ is NOT $\log(n) \cdot \log(n) \ldots \log(n)$, k times.
Assume base of logarithm is 2.

Examples:

- $\log^{(2)}(16) = 2$ whereas $\log^2(16) = \log(16) \cdot \log(16) = 4 \cdot 4 = 16$.
- $\log^{(2)}(200) < \log^{(2)}(256) = 3$

A slow growing function: iterated logarithm

$$
\begin{aligned}
\log^{(k)}(n) &= n && \text{if } k = 0\\
&= \log(\log^{(k-1)}(n)) && \text{if } \log^{(k-1)}(n) \text{ is defined and is positive}\\
&= \text{undefined} && \text{otherwise}
\end{aligned}
$$

A slow growing function: iterated logarithm

$$
\begin{aligned}
\log^{(k)}(n) &= n && \text{if } k = 0 \\
&= \log(\log^{(k-1)}(n)) && \text{if } \log^{(k-1)}(n) \text{ is defined and is positive} \\
&= \text{undefined} && \text{otherwise}
\end{aligned}
$$

Note: $\log^{(k)}(n)$ is NOT $\log(n) \cdot \log(n) \ldots \log(n)$, k times.
Assume base of logarithm is 2.

A slow growing function: iterated logarithm

$$
\begin{aligned}
\log^{(k)}(n) &= n && \text{if } k = 0 \\
&= \log(\log^{(k-1)}(n)) && \text{if } \log^{(k-1)}(n) \text{ is defined and is positive} \\
&= \text{undefined} && \text{otherwise}
\end{aligned}
$$

Note: $\log^{(k)}(n)$ is NOT $\log(n) \cdot \log(n) \ldots \log(n)$, k times.
Assume base of logarithm is 2.

$\boxed{\text{Iterated Logarithm: } \log^*(n)}$: This is the smallest non-negative integer k such that $\log^{(k)}(n) \leq 1$.

A slow growing function: iterated logarithm

$$\begin{aligned}
\log^{(k)}(n) &= n && \text{if } k = 0 \\
&= \log(\log^{(k-1)}(n)) && \text{if } \log^{(k-1)}(n) \text{ is defined and is positive} \\
&= \text{undefined} && \text{otherwise}
\end{aligned}$$

Note: $\log^{(k)}(n)$ is NOT $\log(n) \cdot \log(n) \ldots \log(n)$, k times.
Assume base of logarithm is 2.

$\boxed{\text{Iterated Logarithm: } \log^*(n)}$: This is the smallest non-negative integer k such that $\log^{(k)}(n) \leq 1$.

Ex: What is $\log^*(4)$ and what is $\log^*(2^{2048})$?

A slow growing function: iterated logarithm

$$
\begin{array}{lcll}
\log^{(k)}(n) & = & n & \text{if } k = 0 \\
 & = & \log(\log^{(k-1)}(n)) & \text{if } \log^{(k-1)}(n) \text{ is defined and is positive} \\
 & = & \text{undefined} & \text{otherwise}
\end{array}
$$

Note: $\log^{(k)}(n)$ is NOT $\log(n) \cdot \log(n) \ldots \log(n)$, k times.
Assume base of logarithm is 2.

$\boxed{\text{Iterated Logarithm: } \log^*(n)}$: This is the smallest non-negative integer k such that $\log^{(k)}(n) \leq 1$.

Ex: What is $\log^*(4)$ and what is $\log^*(2^{2048})$?
Justify the title of the slide: slow growing function!

Summary

- Some well-known and not so well-known recursive functions.
- Use of induction to prove formulas.
- Reference: Section 5.3 [KT].

To iterate is human, to recurse is divine. L. Peter Deutsch