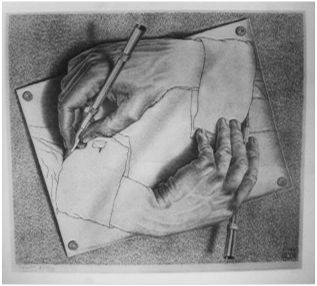
Recursion and Proofs by Induction – Part II

CS1200, CSE IIT Madras

Meghana Nasre

March 20, 2020

Recursion Continued



Drawing Hands by M. C. Escher

- Familiar recursive functions ✓
- Some important recursive functions ✓
- Proving closed form solutions using induction ✓
- Defining objects and sequences using recursion

All images are courtsey Google Images

Recursive Sets

We have seen different ways of defining sets. Lets see some sets which can be recursively defined.

Recursive Sets

We have seen different ways of defining sets. Lets see some sets which can be recursively defined.

$$S = \{1, 3, 9, 81, ...\}$$

Attempt to give a recursive definition for the set above.

Recursive Sets

We have seen different ways of defining sets. Lets see some sets which can be recursively defined.

$$S = \{1, 3, 9, 81, \ldots\}$$

Attempt to give a recursive definition for the set above.

Basis Step: $1 \in S$.
Recursive Step: if $a \in S$, then $3a \in S$.

Recursive Sets

We have seen different ways of defining sets. Lets see some sets which can be recursively defined.

$$S = \{1, 3, 9, 81, ...\}$$

Attempt to give a recursive definition for the set above.

Basis Step: $1 \in S$.
Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

We have seen different ways of defining sets. Lets see some sets which can be recursively defined.

$$S = \{1, 3, 9, 81, ...\}$$

Attempt to give a recursive definition for the set above.

Basis Step: $1 \in S$.
Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof: Let A be the set of all non-negative powers of 3. Show that $S = A$.
Note that $A = \{\ 3^n \ \mid \ n \in \mathbb{Z}_{\geq 0}\}$

- Show that $A \subseteq S$
- Show that $S \subseteq A$

Recursive Sets

$$S = \{1, 3, 9, 81, \ldots\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 1: $A \subseteq S$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 1: $A \subseteq S$.

Use induction on n. Let $P(n)$: 3^n belongs to S.

- Base case: $n = 0$. This is true since $3^0 = 1 \in S$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 1: $A \subseteq S$.

Use induction on n. Let $P(n)$: 3^n belongs to S.

- Base case: $n = 0$. This is true since $3^0 = 1 \in S$.
- Inductive step: Assume $P(k)$ is true, that is, $3^k \in S$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 1: $A \subseteq S$.

Use induction on n. Let $P(n)$: 3^n belongs to S.

- Base case: $n = 0$. This is true since $3^0 = 1 \in S$.
- Inductive step: Assume $P(k)$ is true, that is, $3^k \in S$.
- $3^k \in S$ and by Recursive Step, we know that $3 \cdot 3^k \in S \implies 3^{k+1} \in S$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 1: $A \subseteq S$.

Use induction on n. Let $P(n)$: 3^n belongs to S.

- Base case: $n = 0$. This is true since $3^0 = 1 \in S$.
- Inductive step: Assume $P(k)$ is true, that is, $3^k \in S$.
- $3^k \in S$ and by Recursive Step, we know that $3 \cdot 3^k \in S \implies 3^{k+1} \in S$.

Thus, we know that all non-negative powers of 3 belong to S. That is, $A \subseteq S$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

Recursive Sets

$$S = \{1, 3, 9, 81, \ldots\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

We use the recursive definition of the set S.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

We use the recursive definition of the set S.

- Observe that basis step $1 = 3^0$ is a power of 3. Thus, $3 \in A$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

We use the recursive definition of the set S.

- Observe that basis step $1 = 3^0$ is a power of 3. Thus, $3 \in A$.
- We need to show that all integers generated by recursive step are in A.

Recursive Sets

$$S = \{1, 3, 9, 81, \ldots\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

We use the recursive definition of the set S.

- Observe that basis step $1 = 3^0$ is a power of 3. Thus, $3 \in A$.
- We need to show that all integers generated by recursive step are in A.
- We need to show that if $x \in S$ and $x \in A$, then $3x \in A$.

Recursive Sets

$$S = \{1, 3, 9, 81, \ldots\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

We use the recursive definition of the set S.

- Observe that basis step $1 = 3^0$ is a power of 3. Thus, $3 \in A$.

- We need to show that all integers generated by recursive step are in A.
- We need to show that if $x \in S$ and $x \in A$, then $3x \in A$.
- Since $x \in A$, we know $x = 3^i$ for some $i \geq 0$.

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

We use the recursive definition of the set S.

- Observe that basis step $1 = 3^0$ is a power of 3. Thus, $3 \in A$.
- We need to show that all integers generated by recursive step are in A.
- We need to show that if $x \in S$ and $x \in A$, then $3x \in A$.
- Since $x \in A$, we know $x = 3^i$ for some $i \geq 0$.
- Thus, $3x = 3 \cdot 3^i = 3^{i+1}$. And $3^{i+1} \in A$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Basis Step: $1 \in S$.

Recursive Step: if $a \in S$, then $3a \in S$.

Claim: The set S is the set of all non-negative powers of 3.

Proof Part 2: $S \subseteq A$.

Note: This implies S contains only those integers which are non-negative powers of 3.

We use the recursive definition of the set S.

- Observe that basis step $1 = 3^0$ is a power of 3. Thus, $3 \in A$.

- We need to show that all integers generated by recursive step are in A.
- We need to show that if $x \in S$ and $x \in A$, then $3x \in A$.
- Since $x \in A$, we know $x = 3^i$ for some $i \geq 0$.
- Thus, $3x = 3 \cdot 3^i = 3^{i+1}$. And $3^{i+1} \in A$.

Thus, S contains only those integers that are non-negative powers of 3, i.e., $S \subseteq A$.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

$\boxed{\text{Recap and ponder}}$

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

$\boxed{\text{Recap and ponder}}$

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

Revisit the proof.

For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

  Revisit the proof.

  For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Ex:

- Give a recursive definition for the set of even integers.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

Revisit the proof.

For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Ex:

- Give a recursive definition for the set of even integers.

Write down your definition on a sheet of paper.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

  Revisit the proof.

  For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Ex:

- Give a recursive definition for the set of even integers.

  Write down your definition on a sheet of paper. Does it cover only positive even integers? If yes, correct it.

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

  Revisit the proof.

  For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Ex:

- Give a recursive definition for the set of even integers.

  Write down your definition on a sheet of paper. Does it cover only positive even integers? If yes, correct it.

- Consider the following definition of a set S.

  Basis Step: $(0, 0) \in S$.

  Recursive Step: if $(a, b) \in S$, then $(a + 2, b + 3) \in S$ and $(a + 3, b + 2) \in S$.

Recursive Sets

$$S = \{1, 3, 9, 81, \ldots\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

  Revisit the proof.

  For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Ex:

- Give a recursive definition for the set of even integers.

  Write down your definition on a sheet of paper. Does it cover only positive even integers? If yes, correct it.

- Consider the following definition of a set S.

  Basis Step: $(0, 0) \in S$.

  Recursive Step: if $(a, b) \in S$, then $(a + 2, b + 3) \in S$ and $(a + 3, b + 2) \in S$.

Write down at least 5 elements in the set S.

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

Revisit the proof.

For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Ex:

- Give a recursive definition for the set of even integers.

Write down your definition on a sheet of paper. Does it cover only positive even integers? If yes, correct it.

- Consider the following definition of a set S.

Basis Step: $(0, 0) \in S$.

Recursive Step: if $(a, b) \in S$, then $(a + 2, b + 3) \in S$ and $(a + 3, b + 2) \in S$.

Write down at least 5 elements in the set S. Show that if $(a, b) \in S$, then 5 divides $a + b$?

Recursive Sets

$$S = \{1, 3, 9, 81, ...\}$$

Recap and ponder

- A simple recursively defined set.
- Part 1 of the proof uses induction on n.
- Part 2 of the proof uses structural definition of the set.

Revisit the proof.

For instance, does the claim go through if Basis Step was $2 \in S$ instead of $1 \in S$?

Ex:

- Give a recursive definition for the set of even integers.

Write down your definition on a sheet of paper. Does it cover only positive even integers? If yes, correct it.

- Consider the following definition of a set S.

Basis Step: $(0, 0) \in S$.
Recursive Step: if $(a, b) \in S$, then $(a + 2, b + 3) \in S$ and $(a + 3, b + 2) \in S$.

Write down at least 5 elements in the set S. Show that if $(a, b) \in S$, then 5 divides $a + b$? Is the converse true?

Recursive Structures

In CS you will encounter many recursively defined (data) structures. In fact you have seen some of them already.

Recursive Structures

In CS you will encounter many recursively defined (data) structures. In fact you have seen some of them already.

Lets define a linked list recursively.

Recursive Structures

In CS you will encounter many recursively defined (data) structures. In fact you have seen some of them already.

Lets define a linked list recursively.

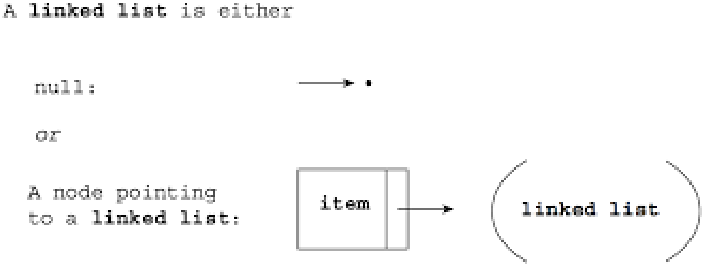
Basis Step: A null node is a linked list.
Recursive Step: A linked list is a node (containing data and) pointing to a linked list.

Recursive Structures

In CS you will encounter many recursively defined (data) structures. In fact you have seen some of them already.

Lets define a linked list recursively.

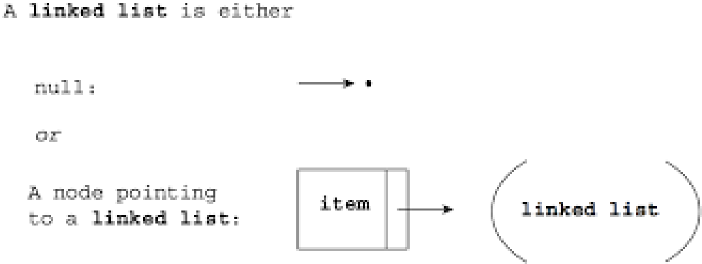
Basis Step: A null node is a linked list.
Recursive Step: A linked list is a node (containing data and) pointing to a linked list.



In CS you will encounter many recursively defined (data) structures. In fact you have seen some of them already.

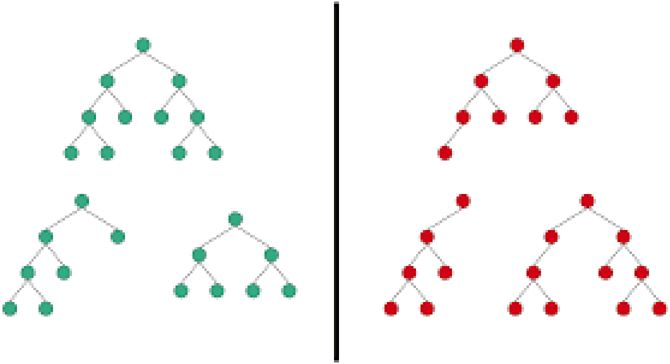
Lets define a linked list recursively.

Basis Step: A null node is a linked list.
Recursive Step: A linked list is a node (containing data and) pointing to a linked list.

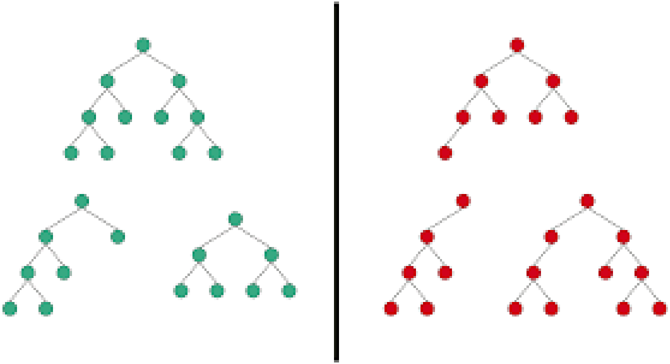


Ex: Define length of a linked list recursively.

Trees defined recursively



Trees defined recursively



Trees are drawn upside down in CS!

Trees defined recursively

Lets define trees (in mathematics and CS) using recursion. We start with binary trees.

Lets define trees (in mathematics and CS) using recursion. We start with binary trees.

$\boxed{\text{Def.1: Binary trees}}$

Basis Step: A null node represents an empty binary tree.

Trees defined recursively

Lets define trees (in mathematics and CS) using recursion. We start with binary trees.

Def.1: Binary trees

Basis Step: A null node represents an empty binary tree.

Recursive Step: If T_1 and T_2 are disjoint binary trees then we can get a binary tree (denoted as $T_1 \cdot T_2$) with a root r together with edges connecting r to the roots of the left subtree T_1 and right subtree T_2.

Trees defined recursively

Lets define trees (in mathematics and CS) using recursion. We start with binary trees.

Def.1: Binary trees

Basis Step: A null node represents an empty binary tree.

Recursive Step: If T_1 and T_2 are disjoint binary trees then we can get a binary tree (denoted as $T_1 \cdot T_2$) with a root r together with edges connecting r to the roots of the left subtree T_1 and right subtree T_2.

Consider another definition of Trees.

Def.2: Full binary trees

Trees defined recursively

Lets define trees (in mathematics and CS) using recursion. We start with binary trees.

Def.1: Binary trees

Basis Step: A null node represents an empty binary tree.

Recursive Step: If T_1 and T_2 are disjoint binary trees then we can get a binary tree (denoted as $T_1 \cdot T_2$) with a root r together with edges connecting r to the roots of the left subtree T_1 and right subtree T_2.

Consider another definition of Trees.

Def.2: Full binary trees

Basis Step: A single node is a full binary tree

Trees defined recursively

Lets define trees (in mathematics and CS) using recursion. We start with binary trees.

Def.1: Binary trees

Basis Step: A null node represents an empty binary tree.

Recursive Step: If T_1 and T_2 are disjoint binary trees then we can get a binary tree (denoted as $T_1 \cdot T_2$) with a root r together with edges connecting r to the roots of the left subtree T_1 and right subtree T_2.

Consider another definition of Trees.

Def.2: Full binary trees

Basis Step: A single node is a full binary tree

Recursive Step: If T_1 and T_2 are disjoint full binary trees then we can get a full binary tree (denoted as $T_1 \cdot T_2$) with a root r together with edges connecting r to the roots of the left subtree T_1 and right subtree T_2.

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

  Write down your answer. If yes, construct an example tree, if no, attempt a proof.

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

  Write down your answer. If yes, construct an example tree, if no, attempt a proof.

Height of a full binary tree

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

  Write down your answer. If yes, construct an example tree, if no, attempt a proof.

Height of a full binary tree

Basis Step: If T consists of a single root node, then $h(T) = 0$.

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

  Write down your answer. If yes, construct an example tree, if no, attempt a proof.

Height of a full binary tree

Basis Step: If T consists of a single root node, then $h(T) = 0$.

Recursive Step: If $T = T_1 \cdot T_2$ is a full binary tree where T_1 and T_2 are themselves full binary trees then, $h(T) = 1 + \max\{h(T_1), h(T_2)\}$.

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

  Write down your answer. If yes, construct an example tree, if no, attempt a proof.

Height of a full binary tree

Basis Step: If T consists of a single root node, then $h(T) = 0$.

Recursive Step: If $T = T_1 \cdot T_2$ is a full binary tree where T_1 and T_2 are themselves full binary trees then, $h(T) = 1 + \max\{h(T_1), h(T_2)\}$.

Ex:

- Write a similar definition for number of nodes $n(T)$ for a full binary tree.

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

  Write down your answer. If yes, construct an example tree, if no, attempt a proof.

Height of a full binary tree

Basis Step: If T consists of a single root node, then $h(T) = 0$.

Recursive Step: If $T = T_1 \cdot T_2$ is a full binary tree where T_1 and T_2 are themselves full binary trees then, $h(T) = 1 + \max\{h(T_1), h(T_2)\}$.

Ex:

- Write a similar definition for number of nodes $n(T)$ for a full binary tree.

  Write down your answer. Do these functions $h(T)$ and $n(T)$ work for binary trees?

Binary Trees continued

Revisit the definitions on the previous slide and the picture on the even earlier slide carefully.

Clearly an empty tree is not a full binary tree.

Ex:

- Is there any other tree that is a binary tree but not a full binary tree?

  Write down your answer. If yes, construct an example tree, if no, attempt a proof.

Height of a full binary tree

Basis Step: If T consists of a single root node, then $h(T) = 0$.

Recursive Step: If $T = T_1 \cdot T_2$ is a full binary tree where T_1 and T_2 are themselves full binary trees then, $h(T) = 1 + \max\{h(T_1), h(T_2)\}$.

Ex:

- Write a similar definition for number of nodes $n(T)$ for a full binary tree.

  Write down your answer. Do these functions $h(T)$ and $n(T)$ work for binary trees? What is the change needed?

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step:

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step:

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$n(T) \quad = \quad 1 + n(T_1) + n(T_2)$$

this answers last Ex: on prev. slide partly

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$\begin{aligned} n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\ &\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \end{aligned}$$

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$
\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\}
\end{aligned}
$$

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$
\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\} \\
&= 1 + 2 \cdot 2^{\max\{h(T_1)+h(T_2)\}} = 1 + 2^{1+\max\{h(T_1)+h(T_2)\}}
\end{aligned}
$$

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$
\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\} \\
&= 1 + 2 \cdot 2^{\max\{h(T_1)+h(T_2)\}} = 1 + 2^{1+\max\{h(T_1)+h(T_2)\}} \\
&= 1 + 2^{h(T)}
\end{aligned}
$$

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\} \\
&= 1 + 2 \cdot 2^{\max\{h(T_1)+h(T_2)\}} = 1 + 2^{1+\max\{h(T_1)+h(T_2)\}} \\
&= 1 + 2^{h(T)}
\end{aligned}$$

Hence proved!

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\} \\
&= 1 + 2 \cdot 2^{\max\{h(T_1)+h(T_2)\}} = 1 + 2^{1+\max\{h(T_1)+h(T_2)\}} \\
&= 1 + 2^{h(T)}
\end{aligned}$$

Hence proved!

But wait! Are we done? Is it the claim that we wanted to prove?

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$

$\boxed{\text{false claim!}}$

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$
\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\} \\
&= 1 + 2 \cdot 2^{\max\{h(T_1)+h(T_2)\}} = 1 + 2^{1+\max\{h(T_1)+h(T_2)\}} \\
&= 1 + 2^{h(T)}
\end{aligned}
$$

Hence proved!

But wait! Are we done? Is it the claim that we wanted to prove?

In fact the claim is incorrect! Find simple counter examples

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$ false claim!

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$
\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\} \\
&= 1 + 2 \cdot 2^{\max\{h(T_1)+h(T_2)\}} = 1 + 2^{1+\max\{h(T_1)+h(T_2)\}} \\
&= 1 + 2^{h(T)}
\end{aligned}
$$

Hence proved!

But wait! Are we done? Is it the claim that we wanted to prove?

In fact the claim is incorrect! Find simple counter examples

Correct Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)+1} - 1$

Binary Trees continued

Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)}$ false claim!

Basis step: For a full binary tree T with a single root node, $h(T) = 0$. $2^0 = 1$ is the number of nodes in T. Hence base case is true.

Recursive step: Assume $T = T_1 \cdot T_2$ where T_1 and T_2 are full binary trees.

$$\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{this answers last Ex: on prev. slide partly} \\
&\leq 1 + 2^{h(T_1)} + 2^{h(T_2)} && \text{by inductive hypothesis} \\
&\leq 1 + 2 \cdot \max\{2^{h(T_1)}, 2^{h(T_2)}\} && x + y \leq 2 \cdot \max\{x, y\} \\
&= 1 + 2 \cdot 2^{\max\{h(T_1)+h(T_2)\}} = 1 + 2^{1+\max\{h(T_1)+h(T_2)\}} \\
&= 1 + 2^{h(T)}
\end{aligned}$$

Hence proved!

But wait! Are we done? Is it the claim that we wanted to prove?

In fact the claim is incorrect! Find simple counter examples

Correct Claim: The number of nodes $n(T)$ for a full binary tree is $\leq 2^{h(T)+1} - 1$

Complete the proof of the correct claim – see Theorem 2, Section 5.3 [KR]

Recursive Sequences

We have already seen the fibonacci sequence in the last class.

Consider the following recursive sequence

$r_0, r_1, r_2, \ldots,$

Recursive Sequences

We have already seen the fibonacci sequence in the last class.

Consider the following recursive sequence
$r_0, r_1, r_2, \ldots,$

Basis Step: $r_0 = -1 \quad r_1 = -14$
Recursive Step: $r_n = 7r_{n-1} - 10r_{n-2} \quad n \geq 2$

Ex:

- Find $r_2,\ r_3$.

Recursive Sequences

We have already seen the fibonacci sequence in the last class.

Consider the following recursive sequence
$r_0, r_1, r_2, \ldots,$

Basis Step: $r_0 = -1 \quad r_1 = -14$
Recursive Step: $r_n = 7r_{n-1} - 10r_{n-2} \quad n \geq 2$

Ex:

- Find r_2, r_3.
- We would like a closed form expression for r_n.

Recursive Sequences

We have already seen the fibonacci sequence in the last class.

Consider the following recursive sequence

$r_0, r_1, r_2, \ldots,$

Basis Step: $r_0 = -1 \quad r_1 = -14$

Recursive Step: $r_n = 7r_{n-1} - 10r_{n-2} \quad n \geq 2$

Ex:

- Find r_2, r_3.
- We would like a closed form expression for r_n.
  Since the closed form is non-trivial, we provide some hints and ask you guess parts of it.

  Guess the values of c_1 and $f(n)$ to get a closed form for r_n :

$$r_n = c_1 \cdot 2^n - 4 \cdot 5^{f(n)}$$

We have already seen the fibonacci sequence in the last class.

Consider the following recursive sequence

$r_0, r_1, r_2, \ldots,$

Basis Step: $r_0 = -1 \quad r_1 = -14$

Recursive Step: $r_n = 7r_{n-1} - 10r_{n-2} \quad n \geq 2$

Ex:

- Find r_2, r_3.
- We would like a closed form expression for r_n.
  Since the closed form is non-trivial, we provide some hints and ask you guess parts of it.

  Guess the values of c_1 and $f(n)$ to get a closed form for r_n :

$$r_n = c_1 \cdot 2^n - 4 \cdot 5^{f(n)}$$

- Does your guess work for all of r_0, r_1, r_2, r_3?

Recursive Sequences

We have already seen the fibonacci sequence in the last class.

Consider the following recursive sequence
$r_0, r_1, r_2, \ldots,$

Basis Step: $r_0 = -1 \quad r_1 = -14$
Recursive Step: $r_n = 7r_{n-1} - 10r_{n-2} \quad n \geq 2$

Ex:

- Find r_2, r_3.
- We would like a closed form expression for r_n.
  Since the closed form is non-trivial, we provide some hints and ask you guess parts of it.

  Guess the values of c_1 and $f(n)$ to get a closed form for r_n :

$$r_n = c_1 \cdot 2^n - 4 \cdot 5^{f(n)}$$

- Does your guess work for all of r_0, r_1, r_2, r_3?
- The answer is $c_1 = 3$ and $f(n) = n$. Now prove that these values are indeed correct by using induction on n.

Summary

- Recursive Sets and proofs using induction and structure of the set.
- Recursively defined objects, specifically trees and their properties.
- Recursive sequences.
- left as reading exercise: Recursion and strings.
- Reference: Section 5.3 [KT].