Structural Induction

250H

Recursive Definitions for Functions

- Sometimes it is hard to define an object explicitly, but we can define it in terms of itself
- Fibonacci: {1, 1, 2, 3, 5, 8, 13, ...}
  - Recursive Definition:

$$F_n = F_{n-1} + F_{n-2}$$

  - Closed form:

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Fibonacci

$$f_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f_{n-1} + f_{n-2} & n \geq 2 \end{cases}$$

Prove $f_n = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n$.

Fibonacci Induction Proof

Base Case: Let $n = 0$,

$$f_0 = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^0 - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^0$$

$$= \frac{1}{\sqrt{5}} - \frac{1}{\sqrt{5}} = 0$$

Fibonacci Induction Proof

Let $n = 1$,

$$f_1 = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^1 - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^1$$

$$\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right) - \left(\frac{1-\sqrt{5}}{2}\right)\right)$$

$$\frac{1}{\sqrt{5}}\left(\frac{2\sqrt{5}}{2}\right) = 1$$

So our base cases holds.

Fibonacci Induction Proof

Inductive Hypothesis: Assume for some $n \geq 1$ that $f_n = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n$.

Fibonacci Induction Proof

Inductive Step: Consider, $f_{n+1} = f_n + f_{n-1}$. By our inductive hypothesis we have

$$\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n + \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\right]$$

Let $\alpha = \frac{1+\sqrt{5}}{2}$ and $\beta = \frac{1-\sqrt{5}}{2}$. So we have,

$$\frac{1}{\sqrt{5}}(\alpha^n - \beta^n + \alpha^{n-1} - \beta^{n-1})$$

$$= \frac{1}{\sqrt{5}}(\alpha^{n-1}(\alpha+1) - \beta^{n-1}(\beta+1))$$

Fibonacci Induction Proof

Inductive Step: Consider, $f_{n+1} = f_n + f_{n-1}$. By our inductive hypothesis we have

$$\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n + \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\right]$$

Let $\alpha = \frac{1+\sqrt{5}}{2}$ and $\beta = \frac{1-\sqrt{5}}{2}$. So we have,

$$\frac{1}{\sqrt{5}}(\alpha^n - \beta^n + \alpha^{n-1} - \beta^{n-1})$$

$$= \frac{1}{\sqrt{5}}(\alpha^{n-1}(\alpha+1) - \beta^{n-1}(\beta+1))$$

Note that $\alpha^2 = 1 + \alpha$ and $\beta^2 = 1 + \beta$. This comes from the fact that α and β are roots of $x^2 - x - 1$. Now we have,

$$= \frac{1}{\sqrt{5}}(\alpha^{n-1}(\alpha^2) - \beta^{n-1}(\beta^2))$$

$$= \frac{1}{\sqrt{5}}(\alpha^{n+1} - \beta^{n+1})$$

Recursively Defined Functions

Define a function with the set of nonnegative integers as its domain:

- Base Step: Specify the value of the function at zero
- Recursive Step: Give a rule for finding its value at an integer from its values at smaller integers

Give a recursive definition of a^n, where a is a nonzero real number and n is a nonnegative integer.

There are two parts: the base step and the recursive step

For n = {0, 1, 2, 3, ...}

- Base step: $a^0 = 1$
- Recursive step: $a^{n+1} = a(a^n)$

Recursive Definitions for Functions

- We can define a function with the set of nonnegative integers as its domain.
  - Basis Step: Specify the value of the function at zero
  - Recursive Step: Give a rule for finding its value at an integer from its values at smaller integers
- Recursively defined functions are well defined
- Given any positive integer,
  - We can use the two parts of the definition to find the value of the function at that integer
  - We obtain the same value no matter how we apply the two parts of the definition

Example

- f is defined recursively by
  - $f(0) = 3$
  - $f(n+1) = 2f(n) + 3$
- $f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$
- $f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$
- $f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$
- $f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$

Recursive Definitions for Sets and Structures

- Sets can also be defined recursively
- We still use the basis step and the recursive step
  - Basis Step: initial collection of elements is specified
  - Recursive Step: rules for forming new elements in the set from those already known to be in the set are provided
  - (Optional) Exclusion Rule: Specifies that a recursively defined set contains nothing other than those elements specified in the basis step or generated by applications of the recursive step

Example

Consider the subset S of the set of integers recursively defined by

- Basis Step: $3 \in S$
- Recursive Step: If $x \in S$ and $y \in S$, then $x + y \in S$

Elements in S

- 3
- 3+3 = 6
- 3+6 = 9
- 6+6 = 12
- ect

Set of Strings

The set Σ^* of strings over the alphabet Σ is defined recursively by

Basis Step: $\varepsilon \in \Sigma^*$ (where ε is the empty string containing no symbols)

Recursive Step: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$

- The basis step of the recursive definition of strings says that the empty string belongs to Σ^*
- The recursive step states that new strings are produced by adding a symbol from to the end of strings in Σ^*
- At each application of the recursive step, strings containing one additional symbol are generated

Concatenation of Strings

- Two strings can be combined via the operation of concatenation.
- Let Σ be a set of symbols and Σ^* the set of strings formed from symbols in Σ. We can define the concatenation of two strings, denoted by $\cdot$, recursively as follows.
- Basis Step: If $w \in \Sigma^*$, then $w \cdot \varepsilon = w$, where ε is the empty string
- Recursive Step: If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

Examples

- If $\Sigma = \{0, 1\}$, the strings found to be in Σ^*, the set of all bit strings
  - ε from the basis step
  - 0 and 1 from applying the recursive step once
  - 00, 01, 10, and 11 formed from applying the recursive step twice
- Concatenate w_1 = abra and w_2 = cadabra
  - w_1w_2 = abracadabra

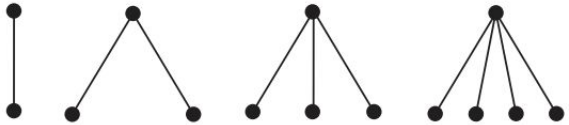
Trees

- The set of rooted trees, where a rooted tree consists of a set of vertices containing a distinguished vertex called the root, and edges connecting these vertices, can be defined recursively by these steps:
  - Basis Step: A single vertex r is a rooted tree
  - Recursive step: Suppose that T_1, T_2,...,T_n are disjoint rooted trees with roots r_1, r_2,...,r_n, respectively. Then the graph formed by starting with a root r, which is not in any of the rooted trees T_1, T_2,...,T_n, and adding an edge from r to each of the vertices r_1, r_2,...,r_n, is also a rooted tree.

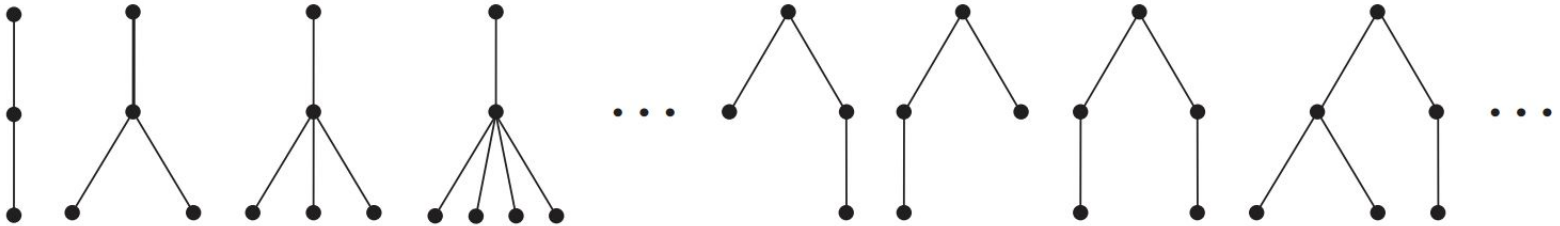
Trees

Basis step

Step 1



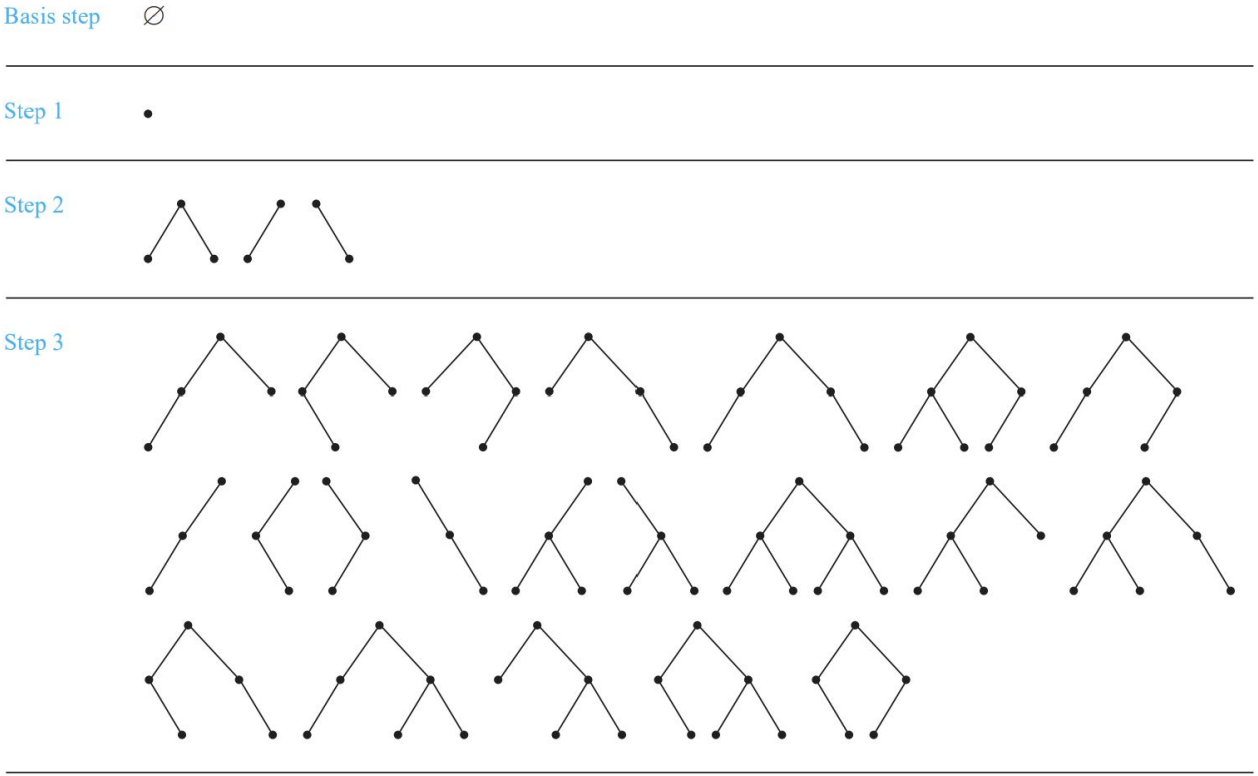
Step 2



Binary Trees

- Binary trees are a special type of rooted trees
- The set of binary trees can be defined recursively by
  - BASIS STEP: There is a full binary tree consisting only of a single vertex r RECURSIVE STEP: If T_1 and T_2 are disjoint binary trees, there is a binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2

Trees



Structural Induction

- To prove results about recursively defined sets, we use what is called Structural Induction
- Structural induction can be used to prove that all members of a set constructed recursively have a particular property

- There are three parts of structural Induction
  - Basis Case: Show that the result holds for all elements specified in the basis step of the recursive definition to be in the set
  - Inductive Hypothesis: Assume the statement is true for some elements in S.
  - Inductive Step: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements

Structural Induction Proofs: Example 1

- Define S as
  - $3 \in S$
  - If $x \in S$ and $y \in S$, then $x + y \in S$
- Prove that $(\forall x \in S)\ x \equiv 0 \bmod 3$

Proof by Structural Induction:

Base Case: Our base case is $3 \in S$. Since $3 \equiv 0 \bmod 3$, our base case holds.

Inductive Hypothesis: Assume for some elements $x,y \in S$ that $x \equiv 0 \bmod 3$ and $y \equiv 0 \bmod 3$.

Inductive Step: Consider $x,y \in S$. By our IH, $x \equiv 0 \bmod 3$ and $y \equiv 0 \bmod 3$. So, $x + y \equiv 0 + 0 \equiv 0 \bmod 3$. Therefore, all elements in S are congruent to 0 mod 3.

Structural Induction Proofs: Example 2

- Define a formal language L as
  - xyx ∈ L
  - If a string σ is in L, then xσyσx is in L
  - If a string σ is in L, then yxσxy is in L

Prove that all strings L have an even number of x's in them.

Proof by Structural Induction:

Base Case: Our base case is xyx. Since xyx, has 2 x's and 2 is even, our base case holds.

Inductive Hypothesis: Assume for some elements σ ∈ L that σ has an even number of x's.

Structural Induction Proofs: Example 2

Inductive Hypothesis: Assume for some elements $\sigma \in L$ that σ has an even number of x's.

Inductive Step: Consider $\sigma \in L$. By our IH, σ has $2k$ number of x's where $k \in \mathbf{Z}$. Consider the cases of elements generated by σ.

Case 1: $x\sigma y\sigma x$ has 2 copies of σ and 2 additional x's. So in total we have $2k + 2k + 2 = 4k + 2 = 2(2k+1)$ x's. Since $2k+1 \in \mathbf{Z}$, we have an even number of x's

Case 2: $yx\sigma xy$ has 1 σ and 2 additional x's. So in total we have $2k + 2 = 2(k+1)$ x's. Since $k+1 \in \mathbf{Z}$, we have an even number of x's.

Therefore, all elements in L have an even number of x's in them.

Structural Induction Proofs: Example 3

Define the set T which is a set containing functions as:

- $\sin(x) \in T$ (This means T contains the sine function as an element)
- If a function is in T, then f' is in T. f' is the derivative of f.

Prove that $T = \{\sin(x), \cos(x), -\sin(x), -\cos(x)\}$.

Proof by Structural Induction: Let $Trig = \{\sin(x), \cos(x), -\sin(x), -\cos(x)\}$. We will first prove $T \subseteq Trig$.

Base Case: Since sin(x) is our base case of T and $\sin(x) \in Trig$, our base case holds.

Inductive Hypothesis: Assume for some $f \in T$, that $f \in Trig$.

Structural Induction Proofs: Example 3

Proof: Let Trig = {sin(x), cos(x), -sin(x), -cos(x)}. We will first prove T $\subseteq$ Trig by structural induction.

Base Case: Since sin(x) is our base case of T and sin(x) $\in$ Trig, our base case holds.

Inductive Hypothesis: Assume for some f $\in$ T, that f $\in$ Trig.

Inductive Step: Let f $\in$ T. By inductive hypothesis, f $\in$ Trig. Consider the cases of elements generated by f,

Case 1: Let f(x) = sin(x). Then f'(x) = cos(x). So, f'(x) $\in$ Trig.

Case 2: Let f(x) = cos(x). Then f'(x) = -sin(x). So, f'(x) $\in$ Trig.

Case 3: Let f(x) = -sin(x). Then f'(x) = -cos(x). So, f'(x) $\in$ Trig.

Case 4: Let f(x) = -cos(x). Then f'(x) = sin(x). So, f'(x) $\in$ Trig.

Therefore, all elements in T are also in Trig so T $\subseteq$ Trig.

Structural Induction Proofs: Example 3

We will now prove Trig $\subseteq$ T. Consider some function f $\in$ Trig. Consider the cases from Trig,

Case 1: Let f(x) = sin(x). Then f is clearly in T.

Case 2: Let f(x) = cos(x). Since sin(x) $\in$ T, by our recursive definition, cos(x) $\in$ T.

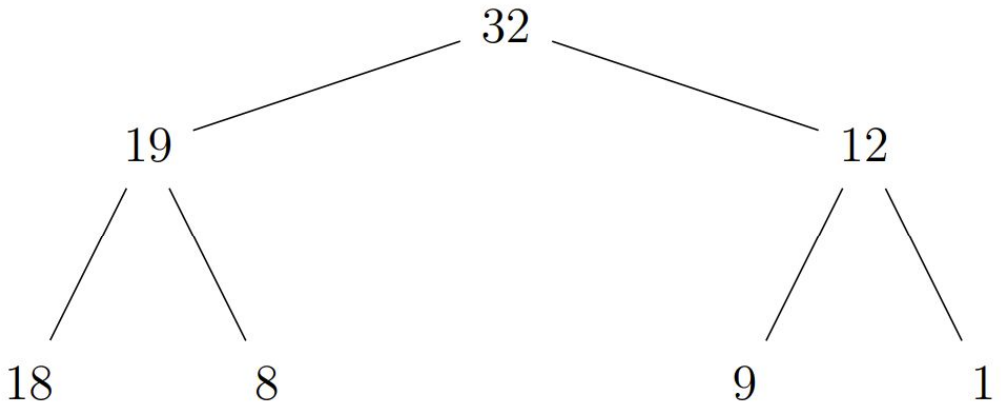
Case 3: Let f(x) = -sin(x). Since cos(x) $\in$ T, by our recursive definition, -sin(x) $\in$ T.

Case 4: Let f(x) = -cos(x). Since -sin(x) $\in$ T, by our recursive definition, -cos(x) $\in$ T.

Therefore Trig $\subseteq$ T. Since we have proved T $\subseteq$ Trig and Trig $\subseteq$ T, T = Trig.

Heap

- Lets store numbers in the nodes of a full binary tree
- The numbers obey the heap property if, for every node x in the tree, the value in x is at least as big as the value in each of x's children



Structural Induction Proofs: Example 4

Prove: If a full binary tree has the heap property, then the value in the root of the tree is at least as large as the value in any node of the tree.

Proof by Structural Induction:

Base: Our base case is that we only have one node. If we only have one node, then the value is the largest value, so our base holds.

Inductive Hypothesis: For some tree T, hen the value in the root of the tree is at least as large as the value in any node of the tree.

Structural Induction Proofs: Example 4

Inductive Hypothesis: For some tree T, hen the value in the root of the tree is at least as large as the value in any node of the tree.

Inductive Step: Consider Trees R and L that follow the heap property. Create a tree T that consists of a root node, r, and sub trees R and L and follows the heap property. Let x and y be the children of R and the roots of R and L, respectively. Since T has the heap property the value(r) ≥ value(x) and the value(r) ≥ value(y). Suppose that z is any node of T. We need to show that value(r) ≥ value(z). Consider the cases,

Case 1: z = r. Since value(r) ≥ value(z), this case holds.

Case 2: z is any node in the subtree R. By the inductive hypothesis value(x) ≥ v(z). Since value(r) ≥ value(x), value(x) ≥ value(z) and this case holds.

Case 3: z is any node in the subtree L. By the inductive hypothesis value(y) ≥ v(z). Since value(r) ≥ value(y), value(y) ≥ value(z) and this case holds.

Therefore, f a full binary tree has the heap property, then the value in the root of the tree is at least as large as the value in any node of the tree. ☽