5.3 Recursive definitions + Structural induction.

Last time we saw that recursive definitions are a way to describe things in terms of themselves.

We saw the example defining $a(n)$ with

Basis step: $a(0) = 0$

Recursive step: $a(n+1) = a(n) + 2n + 1$,

found $a(1)=1$, $a(2)=4$, $a(3)=9$ and guessed that $a(n) = n^2$.

We'll prove this formula next using induction. Recursion and induction work very well together.

Example ① Use induction to prove that the recursively defined function $a(n)$ above has the explicit formula $a(n) = n^2$, for $n \geq 0$.

Solution: We use our usual induction steps.

$P(n)$ says that $a(n) = n^2$.

For the basis step, check that $P(0)$ is true: $a(0) = 0$? Yes, by the definition.

Next, assume $P(k)$ is true: $a(k) = k^2$ and try to use it to prove $P(k+1)$: $a(k+1) = (k+1)^2$

for this we need the recursive step,

$$a(k+1) = a(k) + 2k + 1$$

$$= k^2 + 2k + 1 = (k+1)^2$$

This verifies the inductive step. So by induction $a(n) = n^2$ for all $n \geq 0$.

Remember the Fibonacci numbers:

Basis step: $f_0 = 0$, $f_1 = 1$

Recursive step: $f_{n+1} = f_n + f_{n-1}$ $(n \geq 1)$.

Get 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
(f_0, f_1, f_3, f_6, f_9)

Example (2) Use induction to show that f_{3n} is always even for $n \geq 0$.

Solution: $P(n)$ says f_{3n} is even.

The basis step is good $f_0 = 0$ even ✓.

For the inductive step assume $P(k)$

f_{3k} is even

and try to show $P(k+1)$ is true:

$f_{3k+3} = f_{3(k+1)}$ is even?

$f_0, f_1, \ldots, f_{3k}, f_{3k+1}, f_{3k+2}, f_{3k+3}, \ldots$

↖ how are they ↗ related?

$$f_{3k+3} = f_{3k+2} + f_{3k+1}$$

and $f_{3k+2} = f_{3k+1} + f_{3k}$ } by the recursive step

so $f_{3k+3} = (f_{3k+1} + f_{3k}) + f_{3k+1}$

$= \underbrace{2 f_{3k+1}}_{\text{even}} + \underbrace{f_{3k}}_{\text{even by assumption}}$

and f_{3k+3} must be even.

This proves the inductive step. So by induction f_{3n} is even for all $n \geq 0$.

Structural induction

This is another form of induction that is very useful to prove things about recursively defined objects. As usual it has two parts:

- **Basis step** Show the property or statement is true for the basis objects.
- **Inductive step** Show that if the statement is true for each element used to make new elements in the recursive step, then

it must be true for these new elements too.

Example (3) We defined a set S by

Basis step: $3 \in S$
Recursive step: if $x \in S$ and $y \in S$ then $x+y \in S$.

Use structural induction to prove that every element of S is divisible by 3.

Solution: (There is no $P(n)$ now.)

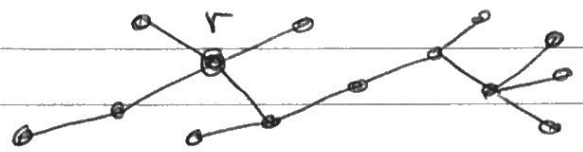
The basis object is 3 and that is divisible by 3.
Next assume that x and y are divisible by 3 so $x = 3a$, $y = 3b$. But then the new element

$$x + y = 3a + 3b = 3(a+b)$$

must be divisible by 3. Therefore, by structural induction, every element of S is divisible by 3.

We can see why structural induction works. If every new object produced in the recursive step has the property we want, then all the objects must have the property.

We will study graphs in Chapter 10 and trees in Chapter 11. A tree is a graph like

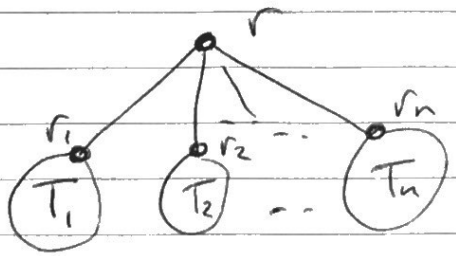


with no circuits. Also we labelled one vertex r for the root.

Rooted trees can be very complicated, but they have a simple recursive definition:

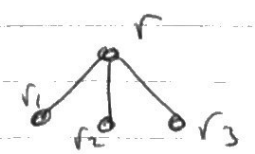
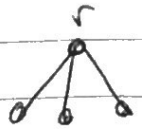
Basis step: is a rooted tree

Recursive step: If $T_1, T_2, \ldots, T_n$ are rooted trees with roots $r_1, r_2, \ldots, r_n$ then the graph made by connecting these roots to a new root r is a rooted tree.

new T = 

For example, we could take

$T_1 =$, $T_2 =$  , $T_3 =$

at the start and make  so that  is a rooted tree.

See Figure 2 p 351.

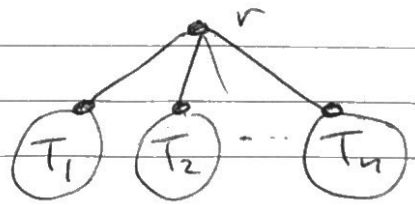
Example ④ Use structural induction to show that the number of vertices in a rooted tree is always one more than the number of edges.

Solution: The property is true for the basis object $\overset{r}{\bullet}$ with 1 vertex and 0 edges.

Now suppose the property is true for the rooted trees $T_1, T_2, \ldots, T_n$ in the recursive step:

$$\#\text{verts } T_1 = 1 + \#\text{edges } T_1$$
$$\#\text{verts } T_2 = 1 + \#\text{edges } T_2$$
$$\vdots$$
$$\#\text{verts } T_n = 1 + \#\text{edges } T_n$$

new T =



Then root r

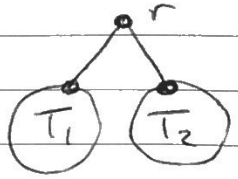
$$\#\text{verts } T = 1 + \#\text{verts } T_1 + \cdots + \#\text{verts } T_n$$
$$= 1 + (1 + \#\text{edges } T_1) + \cdots + (1 + \#\text{edges } T_n)$$
$$= 1 + n + \#\text{edges } T_1 + \cdots + \#\text{edges } T_n$$
$$= 1 + (\#\text{edges } T)$$

So the new rooted tree T has the property we want. By structural induction every rooted tree has this property.

A special kind of rooted tree is a <u>full binary tree</u>. Here is its recursive definition –

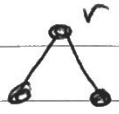
Basis step: is a full binary tree

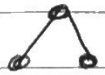
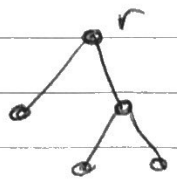
Recursive step: If T_1 and T_2 are full binary trees then the graph made by connecting their roots to a new root r is a full binary tree.



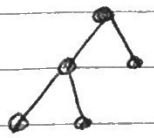
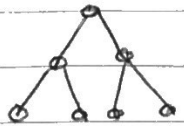
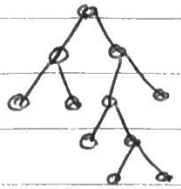
<u>Examples</u> of full binary trees

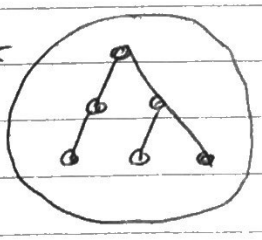
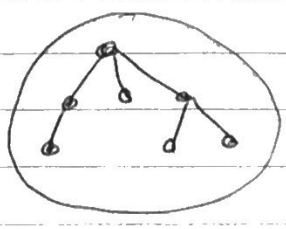
Start with

Apply the recursive step with $T_1 =$, $T_2 =$ to get 

Apply the recursive step again with $T_1 =$ and $T_2 =$  to get 

Can also get

 or  and  ....

but not  or  ← not full binary trees.