5.3 Recursion

Here is an example of a function with an <u>explicit</u> definition:

$$f(n) = n^2.$$

So we see $f(0)=0$, $f(1)=1$, $f(7)=49$ ...

Here is an example of a <u>recursively</u> defined function:

$$g(n+1) = 2g(n) + n - 1.$$

This formula is true for every integer n. When $n=2$ for instance

$$g(3) = 2g(2) + 2 - 1.$$

But what is $g(3)$ as a number? We need a starting point:

$$g(0) = 1$$ (more information about g).

Now we can find the values of g:

$n=0$ $\quad g(1) = 2g(0) + 0 - 1 = 2+0-1 = 1$

$n=1$ $\quad g(2) = 2g(1) + 1 - 1 = 2\cdot 1 = 2$

$n=2$ $\quad g(3) = 2g(2) + 2 - 1 = 2\cdot 2 + 2 - 1 = 5$

$n=3$ $\quad g(4) = 2g(3) + 3 - 1 = 2\cdot 5 + 3 - 1 = 12$

$\vdots$

Recursive definitions have two parts as we saw for g:

Example ① Define $g(n)$ with

$\boxed{\text{Basis step}}$ $g(0) = 1$

$\boxed{\text{Recursive step}}$ $g(n+1) = 2g(n) + n - 1$.

The idea of recursion is to define objects (like functions, sets, graphs) in terms of themselves. Sometimes a recursive definition is much simpler than the explicit definition. Some recursively defined functions do not have explicit definitions. Is there an explicit definition for g in example ① ?

Example ② Define $a(n)$ with

Basis step: $a(0) = 0$

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

Compute $a(1), a(2), a(3)$. Can you guess an explicit formula for $a(n)$?

Solution: We are looking for $a(1)$ first so substitute $n = 0$ into the recursive step to get

$$a(0+1) = a(0) + 2\cdot 0 + 1$$

so that $a(1) = 0 + 0 + 1 = 1$ (since $a(0)=0$).
Keep going:

$n=1$ $\quad a(2) = a(1) + 2\cdot 1 + 1$
$\quad\quad\quad = 1 + 2 + 1 = 4$

$n=2$ $\quad a(3) = a(2) + 2\cdot 2 + 1$
$\quad\quad\quad = 4 + 4 + 1 = 9.$

We've found $a(0)=0$, $a(1)=1$, $a(2)=4$, $a(3)=9$.

We can guess that $a(n) = n^2$ for all $n \geq 0$.

Example (3) Define $h(n)$ by

Basis step: $h(0) = 1$
Recursive step: $h(n+1) = 2^{h(n)}$.

Compute the first few values of $h(n)$. How far can you go?

Solution: $n=0$ $\quad h(1) = 2^{h(0)} = 2^1 = 2$

$n=1$ $\quad h(2) = 2^{h(1)} = 2^2 = 4$

$n=2$ $\quad h(3) = 2^{h(2)} = 2^4 = 16$

$n=3$ $\quad h(4) = 2^{h(3)} = 2^{16} = 65536$

$n=4$ $\quad h(5) = 2^{h(4)} = 2^{65536} = \ldots$

Example (4) The factorial function

$$F(n) = \overset{\text{notation}}{n!} = n(n-1)\cdots 3\cdot 2\cdot 1 \qquad n \geq 0$$

is the product of the first n positive integers (and $F(0) = 0! = 1$).
Give a recursive definition for $F(n)$.

Solution: To find the recursive step, compare $F(n+1)$ and $F(n)$. How are they related?

$$F(n+1) = (n+1)\underbrace{(n)(n-1)\cdots 3\cdot 2\cdot 1}_{F(n)}$$

so we want

Basis step: $F(0) = 1$
Recursive step: $F(n+1) = (n+1)F(n) \qquad (n \geq 0)$.

We can also define sequences of numbers like $a_0, a_1, a_2, a_3, \ldots$ recursively. The most famous is the Fibonacci sequence:

Example (5) Define the Fibonacci numbers with

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

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

Compute this sequence up to f_{10}.

Solution: $n=1$ $\quad f_2 = f_1 + f_0$
$$= 1 + 0 = 1$$

$n=2$ $\quad f_3 = f_2 + f_1$
$$= 1 + 1 = 2$$

$n=3$ $\quad f_4 = f_3 + f_2 = 2 + 1 = 3$

Ok, we just add two consecutive numbers in the sequence to get the next:

$$\underset{f_0}{0}, \underset{f_1}{1}, 1, 2, \underset{f_4}{3}, 5, 8, \overset{\text{add}}{13}, 21, 34, \underset{f_{10}}{55}, \ldots$$

There is an explicit formula for these numbers. Recursive definition much simpler.

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

The next example makes a recursively defined set.

<u>Example</u> 6 Define a set S by

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

Find 4 different elements of S. Can you guess what S is?

Solution: At the start we only know that 3 is in S. The recursive rule lets us make new elements from old ones.

Take $x=3$, $y=3$ then $x+y=6 \in S$

Take $x=3$, $y=6$ then $x+y=9 \in S$

Take $x=6$, $y=6$ then $x+y=12 \in S$

So 3, 6, 9, 12 are in S and we guess that S is the set of all positive multiples of 3

$$S = \{ 3m \mid \overset{\text{integer}}{m} \geq 1 \}.$$

Let Σ be a set of symbols which we'll call an alphabet. The set of strings Σ^* over the alphabet Σ has a nice recursive definition:

Basis step: the empty string $\lambda \in \Sigma^*$

Recursive step: if $w \in \Sigma^*$ and $x \in \Sigma$ then $wx \in \Sigma^*$.

means put all the symbols together.

Example (7) Find some strings in Σ^* when alphabet $\Sigma = \{0, 1\}$.

Solution: Take $w = \lambda$ (empty) and $x = 0$ in the recursive step to get

$$0 \in \Sigma^*.$$

Also $w = \lambda$, $x = 1$ shows

$$1 \in \Sigma^*.$$

Next $w = 0$, $x = 0$ shows

$$00 \in \Sigma^*.$$

Or $w = 0$, $x = 1$

$$01 \in \Sigma^*$$

Keep going

$$10, 11, 000, 001, 101, 1110, 01011011 \text{ all in } \Sigma^*$$

These are the <u>bit strings</u>.

Example (8) If $\Sigma = \{A, C, G, T\}$ then the strings in Σ^* are DNA sequences like

GTACCATGGCGATT...