Recursive Definitions of Functions

Recursive Integer Functions

Intuitively, a **recursive function** f is one whose output can be defined for a given input by equating its associated output to an expression that includes the output values of f for inputs of smaller size. For example, if f takes as input a nonnegative integer $n \geq 0$, then recursive $f(n)$ can be defined by equating it to an expression that includes values $f(m_1), \ldots, f(m_a)$, where $a \geq 1$ and $m_1, \ldots, m_a < n$. When this equation is required for computing $f(n)$ for some n, then n is called a **recursive case**. However, what if n is already "small"? Say $n = 0$. Then since there are no smaller numbers than n, n is called a **base case**, and $f(n)$ must be defined by equating it to a constant, such as $f(0) = 5$, or $f(1) = 2$, etc..

In what follows we assume that the domain of f is the set of natural numbers $\mathcal{N} = \{0, 1, 2, \ldots\}$. Note that functions of this kind are often referred to as **sequences**, since they may also be expressed as $f_0, f_1, f_2, \ldots$, where f_i is shorthand for $f(i)$, $i = 0, 1, 2, \ldots$. $f(0) \Leftrightarrow f_0$ $f(1) \Leftrightarrow f_1$

To summarize, when defining a recursive function, there are two cases to consider:

Base Case when the input n has a size that is one of the smallest possible. In this case $f(n)$ is equated to a constant.

Recursive Case $f(n)$ can be defined by equating it to an expression that uses one or more f-values on inputs that are smaller than n.

Example 1. Consider the following recursive definition. **Base Case:** $f(0) = 1$ **Recursive Case:** for $n \geq 1$, $f(n) = 3f(n-1) + 2$. Use this definition to compute $f(1)$, $f(2)$, and $f(3)$.

$f(1) = 3f(0) + 2 = 3(1) + 2 = 5$

$f(2) = 3f(1) + 2 = 3(5) + 2 = 17$

$f(3) = 3f(2) + 2 = 3(17) + 2 = 53$

Example 2. One of the most famous recursive definitions is for the Fibonacci sequence $f_0, f_1, f_2, \ldots$.

Base Case $f_0 = 0,\ f_1 = 1.$

Recursive Case $(n \geq 2)$ $f_n = f_{n-1} + f_{n-2}.$

Compute the $f_2, f_3, \ldots, f_{10}$.

$f_2 = f_1 + f_0 = 1 + 0 = 1$

$f_3 = f_2 + f_1 = 1 + 1 = 2$

$f_4 = f_3 + f_2 = 2 + 1 = 3$

$f_5 = f_4 + f_3 = 3 + 2 = 5$

$f_6 = f_5 + f_4 = 5 + 3 = 8$

Exercise: Compute f_7, f_8, f_9, f_{10}

non-recursive formula: $f_n = c_1\left(\frac{1+\sqrt{5}}{2}\right)^n + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n$

where c_1, c_2 are constants.

Example 3. Provide a recursive definition for $f(n) = n!$.

$5! = 5 \cdot 4!$

$5! = (5)(4)(3)(2)(1)$

$= 120$

$n! = n((n-1)(n-1)\cdots(2)(1)) =$

$(n-1)!$

$n \cdot (n-1)!$

$0! = 1$

Recursive Definition

Base Case: $f(0) = 1$ $f(1) = 1$

Recursive case: $f(n) = n \cdot f(n-1)$

$f(2) = 2 \cdot f(1) = 2 \cdot 1 = 2$

$f(3) = 3 \cdot f(2) = 3 \cdot 2 = 6$

$f(4) = 4 \cdot f(3) = 4 \cdot 6 = 24$

Example 4. Provide a recursive definition for the function $f(n) = a^n$, where $a > 0$ is a real constant.

$f(0) = a^0 = 1$ $f(2) = a^2 = a \cdot a$

$f(1) = a^1$, $f(3) = a^3 = a \cdot a \cdot a$

$f(n) = a^n = a \cdot \overbrace{a \cdot a \cdots a}^{n-1}$ $a^{n-1} = f(n-1)$

n times

Base case: $f(0) = 1$

Recursive case: $f(n) = a f(n-1)$.

$f(1) = a f(0) = a \cdot 1 = a$

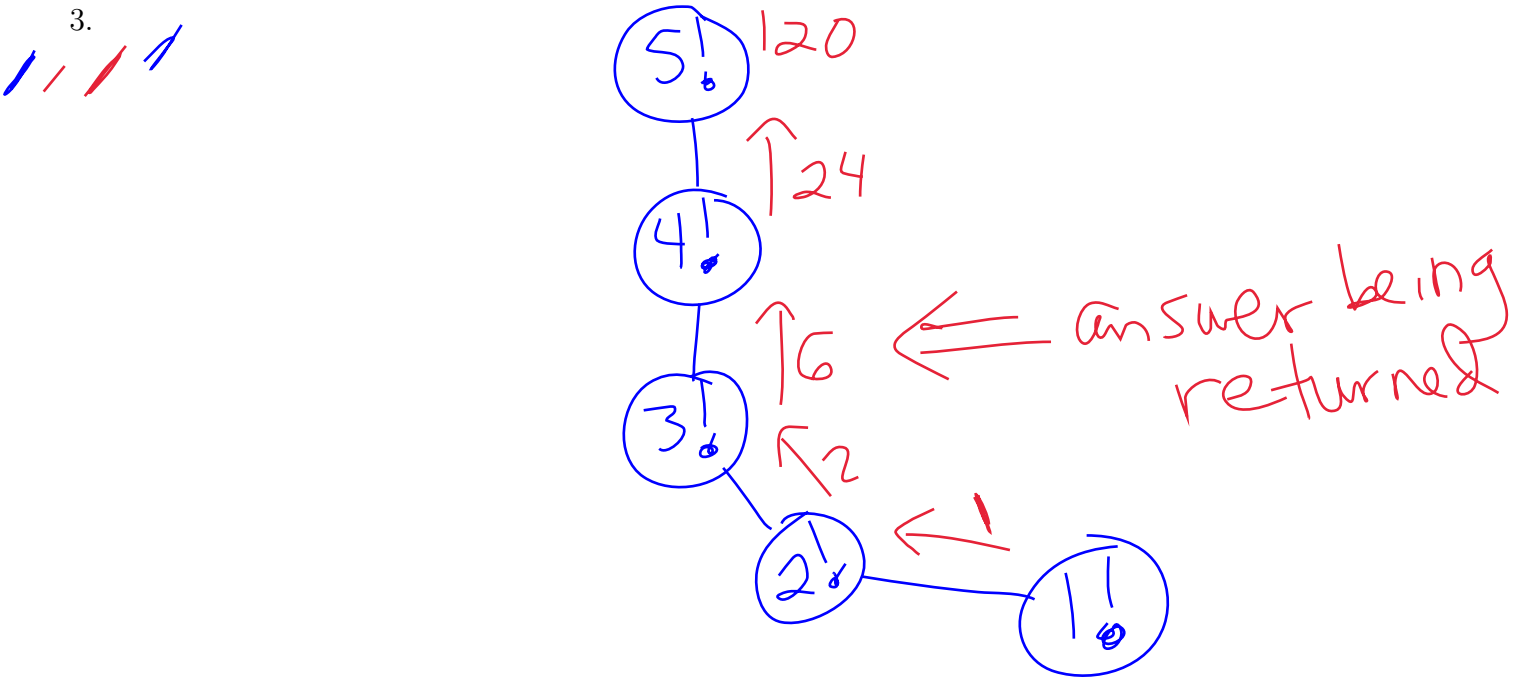
$f(2) = a f(1) = a \cdot a = a^2$

$f(3) = a f(2) = a \cdot a^2 = a^3$

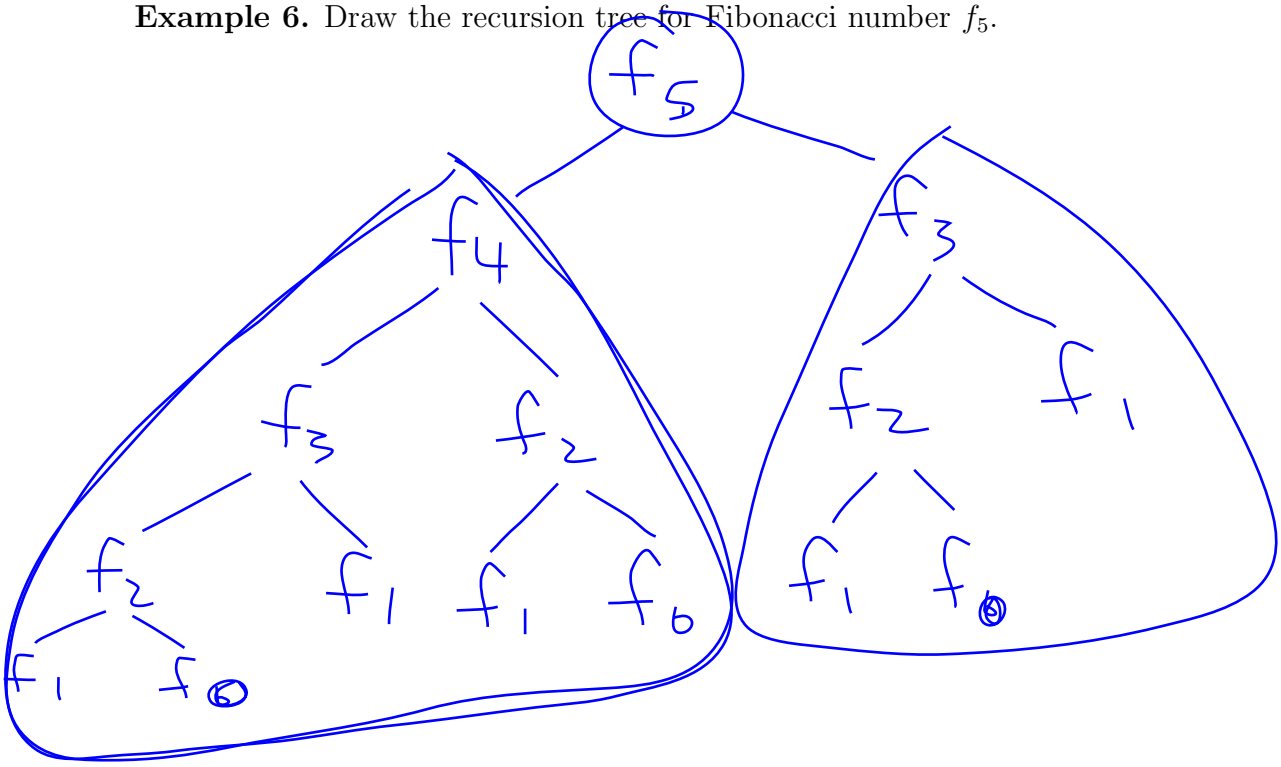
Recursion trees

Given a recursive definition for $f(n)$, suppose we use the definition to compute $f(c)$, for some $c \in \{0, 1, 2, \ldots\}$. Then a **recursion tree** for $f(c)$ is a tree whose nodes are labeled with each of the different function calls that must be made in order to compute $f(c)$. In particular, the root node is labeled with $f(c)$, and the children of the root are labeled with any direct function call $f(a)$ that is required in order to compute $f(c)$. Finally, a node of the tree is a leaf in case it is labeled with a base case.

Example 5. Draw the recursion tree for 5!, based on the recursive definition provided in Example 3.

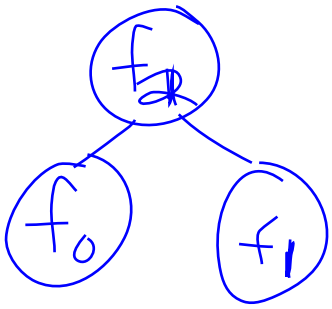


Example 6. Draw the recursion tree for Fibonacci number f_5.



One application of a recursion tree for a recursively defined function f for input n is that the number of tree nodes equals the number of calls that need to be made to function f in order to compute $f(n)$, assuming that $f(n)$ is recursively implemented. For example, below a recursive implementation of the Fibonacci sequence f_n.

```
int fib(int n)
{   //base case
    if(n <= 1)
        return n;
    //recursive case
    return fib(n-1) + fib(n-2);
}
```



Example 7. Let N_n equal the number of nodes in the recursion tree for Fibonacci sequence f_n. Show that $N_n > f_n$ for all $n \geq 0$. Conclude that a recursive implementation of f_n, such as `fib`, will require a number of function calls that exceeds f_n.

$N_0 = 1 = N_1$

$N_2 = 3 \quad N_4 = 9 \quad N_5 = 15 \quad N_3 = 5$

$N_n = N_{n-1} + N_{n-2} + 1$

$N_n \geq f_n = c_1\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n c_2$

for all $n \geq 0$

↳ grows exponentially

$n = 50 \quad (1.6)^{50} = \left(\frac{1+\sqrt{5}}{2}\right) \approx 9.6$

$= 1.6 \times 10^{10}$

The Recursion-Induction Connection

Notice how defining a recursive function has similarities with mathematical induction. When proving $P(n)$ is true for every $n \in \mathcal{N}$, we first show it is true for $n = 0$. Similarly, when defining recursive function $f(n)$, we define its value at $f(0)$. With mathematical induction we assume $P(n)$ is true for some n, and use its truth to establish that $P(n+1)$ is true. Similarly, when defining recursive function $f(n)$, we assume $f(n-1)$ is defined, and use it to compute $f(n)$. Thus, it's not surprising that mathematical induction is the most natural way of proving something to be true about a recursive function.

Example 8. Consider the recursive definition for $f(n)$, where $f(0) = 0$, $f(1) = 1$, and, for $n > 2$, $f(n) = 2f(n-1)+1$. Use math induction to prove that $f(n)$ has the nonrecursive rule $f(n) = 2^n - 1$.

Basis Step: $n=0$. $f(0) = 0 = 2^0 - 1 =$

$n=1$: $f(1) = 1 = 2^1 - 1 = 2 - 1 = 1$ ✓

Inductive Step: Assume $f(n) = 2^n - 1$ for some $n \geq 1$. Show $f(n+1) = 2^{n+1} - 1$

By rec. def.)

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

$2 \cdot 2^n - 2 + 1 = 2^{n+1} - 1$ ✓

Exercises

1. Determine $f(1)$, $f(2)$, $f(3)$, and $f(4)$ if $f(n)$ is recursively defined by $f(0) = 1$ and, for $n = 1, 2, \ldots$

   a. $f(n) = f(n-1) + 3$

   b. $f(n) = 2f(n-1)$

   c. $f(n) = 2^{f(n-1)}$

   d. $f(n) = f(n-1)^2 + 2f(n-1) + 1$

2. Determine $T(2)$, $T(3)$, $T(4)$, and $T(5)$, if $T(n)$ is recursively defined by $T(0) = 2$, $T(1) = 2$, and $T(n) = T(n-1) + 3T(n-2) + 4$.

3. Draw the recursion tree for $T(4)$, where T is the function from the previous exercise. How many nodes does it have?

4. Provide a recursive definition of the function $f(n) = 2n$.

5. Draw the recursion tree that would form when computing $f(5)$ using your recursive definition from the previous exercise. Label each node with the input being computed at that level of recursion, and next to each node write the output that will be returned for that input.

6. Provide a recursive definition of the function $f(n) = 2n + 1$.

7. Provide a recursive definition of the function $f(n) = n^2$.

8. Let $f_m(n) = mn$, where $m \geq 0$ is some integer constant. Provide a recursive defintion for f_m.

9. Provide a recursive definition of the function $f(n) = n^3$.

10. Let N_n be the sequence defined in Example 7. Use mathematical induction to prove that $N_n > f_n$ for all $n \geq 2$.

11. Define $f(n)$ recursively as $f(0) = 0$, $f(1) = 1$, and,for $n \geq 2$, $f(n) = -f(n-2)$. Compute $f(2), \ldots, f(8)$. Use mathematical induction to prove that $f(n) = \sin(\frac{\pi n}{2})$, for all $n \geq 0$.

12. Let $T(n)$ be the function defined in Exercise 2. Use mathematical induction to prove that $T(n)$ always ends with the digit 2.

Exercise Solutions

1. We have

   a. $f(1) = 4$, $f(2) = 7$, $f(3) = 10$, and $f(4) = 13$.

   b. $f(1) = 2$, $f(2) = 4$, $f(3) = 8$, and $f(4) = 16$.

   c. $f(1) = 2$, $f(2) = 4$, $f(3) = 16$, and $f(4) = 2^{16} = 65536$.

   d. $f(1) = 4$, $f(2) = 25$, $f(3) = 676$, and $f(4) = 458329$.

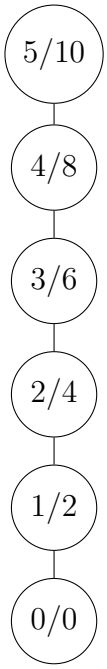
2. We have $T(2) = 2 + 3(2) + 4 = 12$, $T(3) = 12 + 3(2) + 4 = 22$, $T(4) = 22 + 3(12) + 4 = 62$, $T(5) = 62 + 3(22) + 4 = 132$.

3. The $T(4)$ recursion tree has 1 node at the root level (level 0), 2 nodes at level 1, 4 nodes at level 2, and 2 nodes at level 3, for a total of 9 nodes.

   where T is the function from the previous exercise. How many nodes does it have?

4. **Base Case.** $f(0) = 0$. **Recursive Case.** $f(n) = f(n-1) + 2$.

5. Each node is labeled as $n/f(n)$, where n is the function input at that level of recursion, and $f(n)$ is the returned value at that level.



6. **Base case:** $f(0) = 1$. **Recursive Case:** $f(n) = f(n-1) + 2$.

7. **Base case:** $f(0) = 0$. **Recursive Case:** since

$$n^2 = ((n-1)+1)^2 = (n-1)^2 + 2(n-1) + 1 = (n-1)^2 + 2n - 1,$$

   we have $f(n) = f(n-1) + 2n - 1$.

8. **Base Case.** $f_m(0) = 0$. **Recursive Case.** $f_m(n) = f_m(n-1) + m$.

9. **Base Case.** $f(0) = 0$. **Recursive Case.** Since

$$f(n-1) = (n-1)^3 = n^3 - 3n^2 + 3n - 1 = f(n) - 3n^2 + 3n - 1,$$

we have $f(n) = f(n-1) + 3n^2 - 3n + 1$.

10. From Example 7 we know that $N_0 = N_1 = 1$, and $N_n = N_{n-1} + N_{n-2} + 1$ for all $n \geq 2$.

Basis Step: $n = 2$. We have

$$N_2 = N_1 + N_0 + 1 = 1 + 1 + 1 = 3 > f_2 = 1 + 0 = 1.$$

For $n = 3$,

$$N_3 = N_2 + N_1 + 1 = 3 + 1 + 1 = 5 > f_3 = 1 + 1 = 2.$$

Inductive Step: Assume $N_n > f_n$ and $N_{n-1} > f_{n-1}$ for some $n \geq 3$. Show that $N_{n+1} > f_{n+1}$. We have

$$N_{n+1} = N_n + N_{n-1} + 1 > f_n + f_{n-1} = f_{n+1},$$

where the inequality follows from the inductive assumption. Therefore, $N_n > f_n$ for all $n \geq 2$.

11. $f(2) = 0$, $f(3) = -1$, $f(4) = 0$, $f(5) = 1$, $f(6) = 0$, $f(7) = -1$, $f(8) = 0$. **Basis Step:** $n = 0$. $f(0) = 0 = \sin(0\pi/2)$. **Basis Step:** $n = 1$. $f(1) = 1 = \sin(1\pi/2) = \sin(\pi/2)$. **Inductive Step:** Assume $f(k) = \sin(\frac{\pi k}{2})$ for all $0 \leq k \leq n$. Show $f(n+1) = \sin(\frac{\pi(n+1)}{2})$. Do this by showing that

$$\sin(\frac{\pi(n+1)}{2}) = -f((n+1)-2) = -f(n-1) = -\sin(\frac{\pi(n-1)}{2}),$$

where the last equality comes from the inductive assumption, since $n - 1 < n$. Hint: use the Sum of Angles identity for sin.