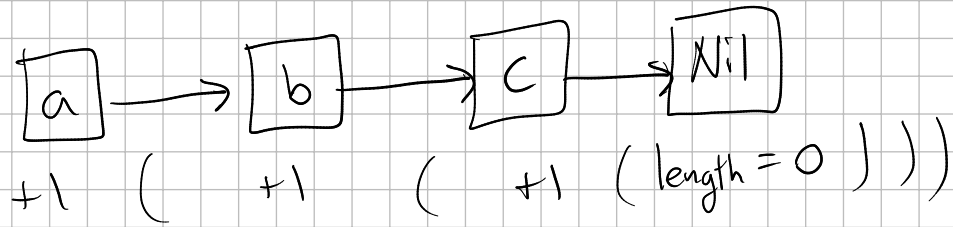


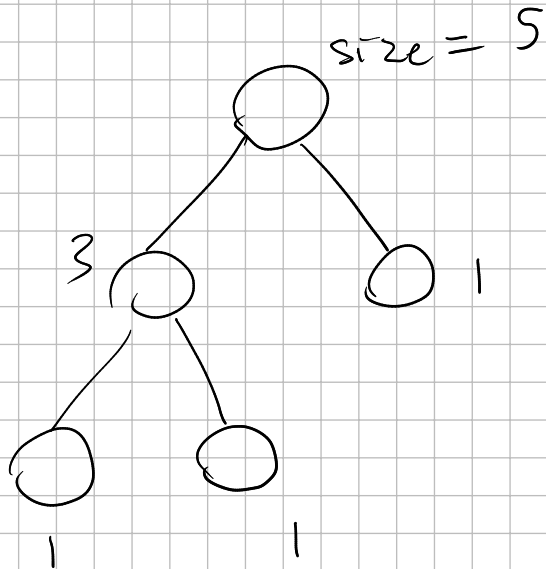
# Notes on Recursion

recursive function: function that calls itself

Why use recursion?



$$\text{len}(L) = \begin{cases} 0 & \text{if } L \text{ is empty} \\ 1 + \text{len}(\text{tail}(L)) & \text{otherwise} \end{cases}$$



$$\text{size}(T) = \begin{cases} 0 & \text{if } T \text{ is empty} \\ 1 + \text{size}(\text{left}(T)) + \text{size}(\text{right}(T)) & \text{if } T \text{ is a node} \end{cases}$$

How do I write a recursive algorithm to solve a problem?

Goal: the recursive algorithm should terminate with the answer to your problem.

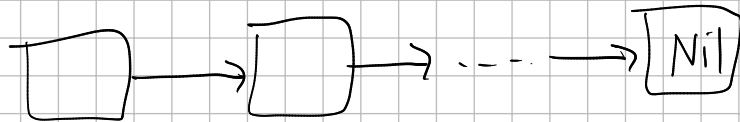
Method:

1. Categorize the possible inputs into "cases".
  2. In each case, break up the input into smaller pieces.
  3. Assuming recursive calls will solve the smaller problems, in each case, find a way to combine combine the solutions from the recursive calls to get the overall solution.
- > sometimes the order needs to be swapped here.*

Step 1: Categorize inputs into cases

Nil

*Empty list*

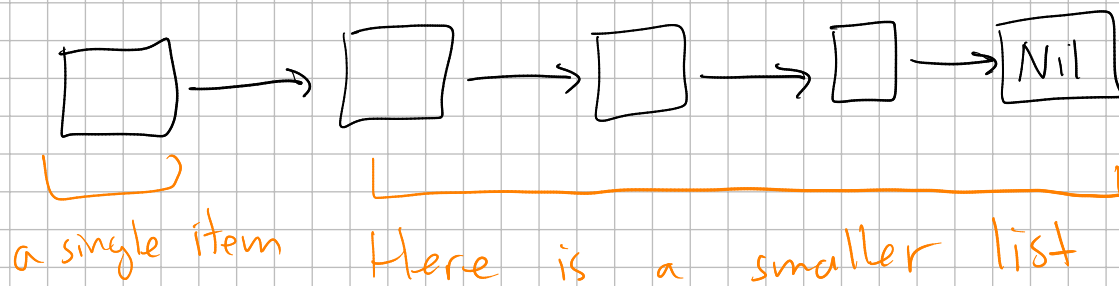


*Non-empty list*

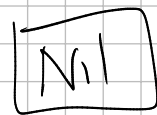
Step 2: In each case, break up the input into smaller pieces.

Nil

*Can't break up an empty list.*



Step 3: Assume that a recursive call on the smaller-sized problems will give you the desired answer. Use this to solve the overall problem in each case.

 Length of empty list is zero.

$$\text{length}(\text{Nil}) = 0$$



Assume that  $\text{length}(\text{tail}(L))$  is the length of the tail.

Then we get

$$\text{length}(L) = 1 + \text{length}(\text{tail}(L))$$

How do we know that this solution is correct?

Recursion is particularly good for defining functions over inductively defined data structures.

Linked lists

Binary trees

Strings

Natural numbers

Propositional logic

Expressions