Recursive Definitions

Recall that sequences can be defined by writing a few terms, writing a closed formula, or *recursively*. That is, the first term(s) are defined explicitly and the n^{th} term is defined by using only term(s) before it.

1. Let $f(0) = 1$ and $f(n) = -2f(n-1)$ for $n \geq 1$.

   (a) Find $f(4)$.

   (b) Find a closed formula for $f(n)$.

2. Give a recursive definition for the sequence $\{a_n\}_{n\geq 1}$, where $a_n = 4n - 2$.

Definition 1. *Recursive definitions involve two parts (and can be applied to functions, sets, structures, and more):*

(a) Bases Step: a specific and initial definition

(b) Recursive Step: rule forming new function/set/rule dependent on facts preceding it.

3. The set of *full binary (and rooted) trees*, where a rooted tree consists of a set of vertices containing a distinguished vertex called the *root*, and eyes connecting these vertices, can be defined recursively by:

   Basis Step: A single vertex r is a full binary (and rooted) tree.
   Recursive Step: Suppose that T_1 and T_2 are disjoint full binary (and rooted) trees, there is a full binary (and rooted) tree, denoted $T_1 \cdot T_2$, consisting for a root r together with edges connecting r to the root of the left subtree T_1 and r to the root of the right subtree T_2.

   (a) Draw the basis step tree.

   (b) Draw the family of trees one step "above" the basis step tree.

   (c) Draw the family of trees two steps "above" the basis step tree.

4. Use induction to prove if T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

int factorial (int n)
if $n == 1$ **then**
    return 1;
end
else
    return $n * factorial(n-1)$
end

Algorithm 1: Sort Algorithm

5. Use induction to prove the above recursively defined program computes the factorial of n.