

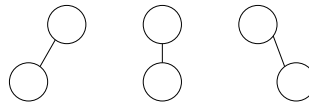
Assignment 2

CSC236, Fall 2008

1. Define the set of ternary trees, \mathcal{T}_3 , as:

- (a) The empty tree, Λ , is an element of \mathcal{T}_3 .
- (b) If trees T_L, T_M, T_R are elements of \mathcal{T}_3 having no nodes in common, and R is a node that is not in T_L, T_M or T_R , then $T = (R, T_L, T_M, T_R)$ is an element of \mathcal{T}_3 . We call T_L the left subtree, T_M the middle subtree, and T_R the right subtree, of T .

Consider two trees equivalent if (a) they are both Λ (the empty tree), or (b) they have equivalent left subtrees, equivalent middle subtrees, and equivalent right subtrees. So there is one ternary tree with zero nodes: Λ , and one ternary tree with one node: $(R, \Lambda, \Lambda, \Lambda)$. For arbitrary natural number n , find a (possibly recursive) formula for the number of non-equivalent ternary trees with n nodes. Prove your formula is correct.



Artist's impression of the three non-equivalent ternary trees with two nodes.

2. Find a recurrence that expresses the time complexity, $T(n)$ of $L(n)$, in terms of n . Find a closed form for your recurrence and prove it is correct. Your expression should assign a constant to operations that do not depend on n .

```
def L(n) :
  if n < 7 : return 0
  else : return 1 + L(n/7)
```

3. Consider a grasshopper that can hop up the stairs either 1 or 3 steps at a time. Let $G(n)$ be the number of ways it can perform the climb when there are n stairs. For example, $G(4) = 3$, since the grasshopper can jump four stairs as either $1 + 1 + 1 + 1$, or $1 + 3$, or $3 + 1$. Prove that for $n \geq 1$, $G(n) \leq F(n)$, the n th Fibonacci number.

4. For python function `bs1(x,A,b,e)`, either prove that the postcondition is correct with respect to the precondition, or else give an example of input that satisfies the precondition where the postcondition is violated. BONUS (10%) What is the running time of this function? You are expected to give as good an asymptotic bound as possible, for example $T(n) \in \Theta(n)$, $T(n) \in \Theta(2^n)$, or something of that nature, and prove your bound is correct.

```
# Precondition:
# x is an integer
# A is a non-empty list of integers sorted in non-decreasing order
# b and e are non-negative integers with b <= e < len(A)
def bs1(x,A,b,e) :
    if b == e : # we're done
        if A[b] == x : return b
        else : return 'not found'
    else :
        m = int(math.sqrt(b*e)) # int rounds toward 0
        if A[m] >= x : return bs1(x,A,b,m)
        else : return bs1(x,A,m+1,e)
# Postcondition:
# Returns n such that b <= n <= e and A[n] == x
# or returns 'not found' if no such n exists.
```