CS 173, Fall 17 Examlet 9, Part A

Name:____

NetID:____________________ **Lecture: A B**

Discussion: Thursday Friday 9 10 11 12 1 2 3 4 5 6

(18 points) Here is a grammar G, with start symbol S and terminal symbols a and b.

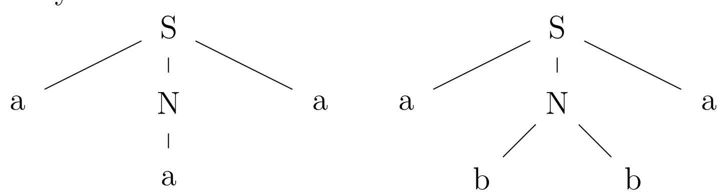
$$S \rightarrow a\,S\,a \mid S\,a\,S \mid a\,N\,a$$
$$N \rightarrow a \mid b\,b$$

Use (strong) induction to prove that any tree of height h matching (aka generated by) grammar G has at least h nodes with label a. Use $A(T)$ as shorthand for the number of a's in a tree T.

Solution:

The induction variable is named __**h**__ and it is the __**height**__ of/in the tree.

Base Case(s): The shortest trees generated by G have $h = 2$. They are as shown below and, as you can see, they both have at least two nodes labelled a.



Inductive Hypothesis [Be specific, don't just refer to "the claim"]: All trees of height h generated by G have at least h nodes labelled a, for $h = 2, 3, \ldots, k-1$. $(k \geq 3)$

Inductive Step: Suppose that T is a tree generated by G of height k. There are two cases:

Case 1: T consists of a root labelled S, with three children. The left and right children have label a. The middle child is a subtree T_1 whose root has label S. T_1 must have height $k-1$ so, by the inductive hypothesis, it contains at least $k-1$ a's. So T contains at least $(k-1)+2 = k+1$ a's.

Case 1: T consists of a root labelled S, with three children. The middle child child has label a. The left and right children are subtrees T_1 and T_2 whose roots have label S. At least one of these two subtrees has height $k-1$ so, by the indutive hypothesis, it contains at least $k-1$ a's. The middle child of T adds another a, so T must have at least k a's. (The other subtree may add additional a's.)

In either case, T has at least k nodes labelled a, which is what we needed to show.

Name:____

NetID:______________________ **Lecture:** **A** **B**

Discussion: **Thursday** **Friday** **9** **10** **11** **12** **1** **2** **3** **4** **5** **6**

(18 points) Recall that a node in a full binary tree is either a leaf or has exactly two children. A Lemon tree is a full binary tree such the two child subtrees of each internal node have heights that differ by at most one. Prove that every Lemon tree of height h contains at least F_{h+1} nodes, where F_k is the kth Fibonacci number. (Recall: $F_0 = 0$, $F_1 = F_2 = 1$)

Solution: The induction variable is named **h** and it is the **height** of/in the tree.

Base Case(s): At $h = 0$, the tree contains exactly one node and therefore exactly one leaf. Also, $F_{h+1} = F_1 = 1$, so the claim holds.

At $h = 1$, the tree must consist of three nodes: a root and its two children. So it has three nodes. $F_{h+1} = F_2 = 1$. So the number of leaves is $\geq F_{h+1}$.

Inductive Hypothesis [Be specific, don't just refer to "the claim"]: Suppose that every Lemon tree of height h contains at least F_{h+1} nodes, for $h = 0, 1, \ldots, k-1$.

Inductive Step: Let T be a Lemon tree of height k ($k \geq 2$). The root of T must have two child subtrees T_a and T_b, whose heights differ by at most one.

Case 1: T_a has height $k-1$ and T_b has height $k-2$. By the inductive hypothesis, T_a has at least F_k nodes and T_b has at least F_{k-1} nodes. So T must have at least $F_k + F_{k-1} = F_{k+1}$ nodes.

Case 2: T_a has height $k-2$ and T_b has height $k-1$. By the inductive hypothesis, T_a has at least F_{k-1} nodes and T_b has at least F_k nodes. So T must have at least $F_k + F_{k-1} = F_{k+1}$ nodes.

Case 3: T_a and T_b have height $k-1$. By the inductive hypothesis, T_a and T_b each have at least F_k nodes. So T must have at least $2F_k \geq F_k + F_{k-1} = F_{k+1}$ nodes.

In all cases, T must have at least F_{k+1} nodes, which is what we needed to prove.

Name:____

NetID:________________________ **Lecture:** **A** **B**

Discussion: **Thursday** **Friday** **9** **10** **11** **12** **1** **2** **3** **4** **5** **6**

(18 points) Suppose that G is a connected graph. A Friendly coloring of G labels each node orange or blue, such that

- If G contains only one node, it is colored orange, and
- Otherwise, every node of G is adjacent to at least one node of the opposite color.

Use (strong) induction to prove that any connected graph can be given a Friendly coloring. Hint: remove any node x (no special properties required) and color the rest of the graph. What color pattern is required if x is the only neighbor of another node?

Solution: The induction variable is named **h** and it is the **number of nodes** of/in the graph.

Base Case(s): $h = 1$. That is, the graph has a single node. Coloring it orange gives us a Friendly coloring.

Inductive Hypothesis [Be specific, don't just refer to "the claim"]: Any connected graph can be given a Friendly coloring, for graphs with number of nodes $h = 1, \ldots, k-1$, $(k \geq 2)$.

Inductive Step: Let G be a connected graph with k nodes $k \geq 2$. Pick one node x. Remove x (and all its edges) to create a smaller graph G'.

G' consists of some number of connected components $C_1, \ldots, C_p$. Each component has at most $k-1$ nodes, so it can be given a Friendly coloring by the inductive hypothesis. We need to assign a color to x to create a Friendly coloring for the full graph G.

Most nodes in G' have a neighbor of opposite color in the same component of G', and therefore they still satisfy this condition in G. So we just need to ensure that x gets a color different from one of its neighbors. Exception: if x is the only neighbor of a node y, then y was the only node in its component of G'. x must be colored blue to ensure that y (which was colored orange) has an opposite-color neighbor in G.

So there are three cases:

Case 1: All x's neighbors have the same color. Then we assign the opposite color to x.

Case 2: x is the only neighbor of a node y. Then we color x blue.

Case 3: Otherwise, we can assign either color to x.

This gives us a Friendly coloring of G, which is what we needed to construct.

CS 173, Fall 17 Examlet 9, Part A

Name:____

NetID:____________________ **Lecture:** **A** **B**

Discussion: **Thursday** **Friday** **9** **10** **11** **12** **1** **2** **3** **4** **5** **6**

(18 points) Recall that a node in a full binary tree must have either 0 or 2 children. A Happy tree is a full binary tree in which each node is colored orange or blue, such that:

- If v is a leaf node, then v may be colored orange or blue.
- If v has two children of the same color, then v is colored blue.
- If v has two children of different colors, then v is colored orange.

Use (strong) induction to show that the root of a Happy tree is blue if and only if the tree has an even number of orange leaves. You may assume basic divisibility facts e.g. the sum of two odd numbers is even.

Solution: The induction variable is named **h** and it is the **height** of/in the tree.

Base Case(s): A Happy tree with $H = 0$ consists of a single node. If it's blue, the tree contains no orange nodes, which is even. If it's orange, the tree contains one orange node, which is odd. In both cases, the claim is true.

Inductive Hypothesis [Be specific, don't just refer to "the claim"]: Suppose that the root of a Happy tree is blue if and only if the tree has an even number of orange leaves, for trees of height $h = 0, 1, \ldots, k-1$. $(k \geq 1)$

Inductive Step: Let T be a Happy tree of height k. There are two cases:

Case 1: the root of T is colored blue and it has two child subtrees whose roots are the same color. If both are blue, then both subtrees contain an even numbr of orange leaves by the inductive hypothesis. Similarly, if both are orange, then each contains an odd number of orange leaves. Since two odd numbers, or two even numbers, sum to an even number, T has an even number of orange leaves.

Case 2: the root of T is colored orange and it has two child subtrees whose roots are opposite colors. By the inductive hypothesis, the subtree with an orange root contain an odd number of orange leaves and the subtree with a blue root contains an even number of orange leaves. So T contains an odd number of orange leaves.

In both cases, T contains an even number of orange leaves if and only if its root is blue, which is what we needed to show.

Name:____

NetID:________________________ **Lecture: A B**

Discussion: Thursday Friday 9 10 11 12 1 2 3 4 5 6

(18 points) Recall that a node in a full binary tree is either a leaf or has exactly two children. A Tidy tree is a full binary tree such that leaf nodes have label 0 and an internal node with label x matches one of these patterns:

- The left subtree is a leaf and the right subtree has root label y, where $y \equiv x - 1 \pmod 3$, or
- The roots of both subtrees have label y, where $y \equiv x - 1 \pmod 3$.

Use (strong) induction to prove that the root label of every Tidy tree is congruent to $h \pmod 3$, where h is the height of the tree. You may assume basic facts about congruence and modular arithmetic (e.g. congruence is transitive).

Solution: The induction variable is named **h** and it is the **height** of/in the tree.

Base Case(s): At $h = 0$, the tree contains exactly one node, which is a leaf and therefore labelled 0. So the label is congruent to the height mod 3.

Inductive Hypothesis [Be specific, don't just refer to "the claim"]: Suppose that every Tidy tree of height h has a root label y which is congruent to $h \pmod 3$, for $h = 0, \ldots, k - 1$.

Inductive Step: Let T be a Tidy tree of height k ($k \geq 1$). Since $k \geq 1$, the root of T is an internal node. So there are two cases:

Case 1: The left subtree is a leaf and the right subtree has root label y, where $y \equiv x - 1 \pmod 3$. The right subtree must be at least as tall as the left subtree, so the right subtree has height $i - 1$. By the inductive hypothesis, we know that $y \equiv i - 1 \pmod 3$.

Case 2: The roots of both subtrees have label y, where $y \equiv x - 1 \pmod 3$. The taller subtree must have height $k - 1$. So, by the inductive hypothesis (applied to that subtree), $y \equiv h - 1 \pmod 3$.

So $y \equiv h - 1 \pmod 3$ in both cases. In both cases, we also have $y \equiv x - 1 \pmod 3$. So we must have $x - 1 \equiv k - 1 \pmod 3$. So $x \equiv k \pmod 3$, which is what we needed to prove.

Name:____

NetID:________________________ **Lecture:** **A** **B**

Discussion: **Thursday** **Friday** **9** **10** **11** **12** **1** **2** **3** **4** **5** **6**

(18 points) Recall that a node in a full binary tree is either a leaf or has exactly two children. A Trim tree is a full binary tree such the two child subtrees of each internal node have heights that differ by at most one. Prove that every Trim tree of height h contains at least $2^{h/2}$ leaves. You may use the fact that $\sqrt{2} > 1.4$.

Solution: The induction variable is named **h** and it is the **height** of/in the tree.

Base Case(s): At $h = 0$, the tree contains exactly one node and therefore exactly one leaf. $2^{h/2} = 2^0 = 1$, so the claim holds.

At $h = 1$, the tree must consist of three nodes: a root and its two children. So it has two leaves. $2^{h/2} = 2^{1/2} = \sqrt{2} < 2$, so the claim holds.

Inductive Hypothesis [Be specific, don't just refer to "the claim"]: Suppose that every Trim tree of height h contains at least $2^{h/2}$ nodes, for $h = 0, 1, \ldots, k-1$.

Inductive Step: Let T be a Trim tree of height k ($k \geq 2$). The root of T must have two child subtrees T_a and T_b, whose heights differ by at most one.

Case 1: T_a has height $k-1$ and T_b has height $k-2$. By the inductive hypothesis, T_a has at least $2^{(k-1)/2}$ leaves and T_b has at least $2^{(k-2)/2}$ leaves. So the number of leaves in T is at least

$$2^{(k-1)/2} + 2^{(k-2)/2} = (2^{-1/2} + 2^{-1})2^{k/2} = (\frac{1}{\sqrt{2}} + \frac{1}{2})2^{k/2} = \frac{\sqrt{2}+1}{2}2^{k/2} > \frac{1.4+1}{2}2^{k/2} > 2^{k/2}$$

Case 2: T_a has height $k-2$ and T_b has height $k-1$. This is exactly the same as case 1, except for swapping the roles of T_a and T_b.

Case 3: T_a and T_b have height $k-1$. By the inductive hypothesis, T_a and T_b each have at least $2(k-1)/2$ leaves. So the number of leaves in T must be at least $2 \cdot 2(k-1)/2 = 2^{(k+1)/2} > 2^{k/2}$.

In all cases, T must have at least $2^{k/2}$ leaves, which is what we needed to prove.