Homework 1

Yu Hsiao Yu-Hsuan Wu

Instructions

- Please use a stapler or paperclip to bind each assignment. Do not fold the pages.
- Use paper that is approximately A4-sized for all submissions.
- Ensure your handwriting is legible. Points will be deducted due to unintelligible handwriting.
- For problems that require to prove by induction, list your base case, induction hypothesis, and inductive step clearly.
- Thoroughly explain your thought process in your assignments. Mention which theorems you are referencing and provide detailed explanations of how you derived each formula.
- If you have any questions regarding the class, please email both TAs.

Inquires About Assignments and Grades

- Assignments will be distributed during the TA session. If you have any inquiry about the assignments or grades, please feel free to come to the TA session or TA hour that week. If you are unavailable during these times, please email the TAs. After one week from the assignment distributed, no disputes will be accepted.
- TA Hour: Thursday 17:00-18:30, Room 112-b (Software Validation and Verification Research Laboratory), Teaching and Researching Hall, College of Management.


不要瞎掰好嗎

Question 1

1. Consider *proper* binary trees, where every internal (non-leaf) node has two children. For any such tree T, let l_T denote the number of its leaves and m_T the number of its internal nodes. Prove *by induction* that $l_T = m_T + 1$.

Question 1

The proof is by induction on m_T.

[Base Case] ($m_T = 0$) There's only one node (root) in the tree, and the node is a leaf. $1 = 0 + 1$.

[Inductive Step] ($m_T > 0$) For any proper binary tree T, we can randomly select an internal node with 2 leaves and remove the leaves. Now we have a tree T' with $l'_T = l_T - 2 + 1$ (remove 2 leaves and turn the internal node to a leaf) and $m'_T = m_T - 1$. By Induction Hypothesis, $l'_T = m'_T + 1$. Finally we get:

$$l_T = l'_T + 1 = m'_T + 2 = m_T + 1$$

Question 2

2. Prove *by induction* that every natural number greater than or equal to 12 is a non-negative linear combination of 4 and 5, i.e., for every $n \in \mathbb{N}$, if $n \geq 12$, then there exist $a, b \in \mathbb{N}$ s.t. $n = 4a + 5b$ (where $\mathbb{N}$ is the set of all natural numbers, including 0).

Question 2

The proof is by induction on n.

[Base Case] $(n = 12)$ In this case, $n = 12 = 4 \times 3 + 5 \times 0$.

[Inductive Step] $(n > 12)$ For $n > 12$, we want to prove that every n is a non-negative linear combination of 4 and 5. By Induction Hypothesis, we have $n - 1 = 4a + 5b$, for some $a, b \in \mathbb{N}$. That is,

$$\begin{aligned} n &= (n-1) + 1 = (4a + 5b) + 1 \\ &= (4a + 5b) + (5 - 4) = 4(a-1) + 5(b+1). \end{aligned}$$

We should notice that $a - 1$ could be negative integer if $a = 0$, which violates our hypothesis. Thus, we should consider the case of $a = 0$. If $a = 0$, $n - 1 = 5b$ for some $b \geq 3$ and

$$n = 5b + 1 = 5b + (4 \times 4 - 5 \times 3) = 4 \times 4 + 5(b - 3).$$

Question 2

In conclusion, for every $n \in \mathbb{N}$, if $n > 12$ and $n - 1 = 4a + 5b$, then

$$n = \begin{cases} 4(a-1) + 5(b+1) & \text{if } a > 0 \\ 4 \times 4 + 5(b-3) & \text{if } a = 0 \end{cases}.$$

Hence, the claim is proved.

Question 3

3. Let $a_1, a_2, \cdots, a_n$ be positive real numbers such that $a_1 a_2 \cdots a_n = 1$. Prove *by induction* that $(1 + a_1)(1 + a_2) \cdots (1 + a_n) \geq 2^n$. (Hint: in the inductive step, try introducing a new variable that replaces two chosen numbers from the sequence.)

Question 3

The proof is by induction on n.

[Base Case] $(n = 1)$ We have $a_1 = 1$ and $(1 + a_1) = (1 + 1) \geq 2^1$.

[Inductive Step] $(n = k + 1 > 1)$ For $n = k + 1$, given $a_1 a_2 \cdots a_{k+1}$ $= 1$, we want to show $(1 + a_1)(1 + a_2) \cdots (1 + a_{k+1}) \geq 2^{k+1}$.
We know there must exist $a_i \leq 1$ and $a_j \geq 1$ where $1 \leq i, j \leq k + 1$ $\wedge\ i \neq j$. Without loss of generality, let $a_k \leq 1$ and $a_{k+1} \geq 1$.
We also introduce a new variable $p = a_k a_{k+1}$ where $p \in \mathbb{R}^+$.
Note that $a_1 a_2 \cdots a_{k-1} p = 1$. By Induction Hypothesis, we have

$$
\begin{aligned}
&(1 + a_1)(1 + a_2) \cdots (1 + a_{k-1})(1 + p) \geq 2^k \\
\Leftrightarrow &(1 + a_1)(1 + a_2) \cdots (1 + a_{k-1}) \boxed{(1 + p) \cdot 2} \geq 2^k \cdot 2 = 2^{k+1}.
\end{aligned}
$$

Recall that we want to prove

$$(1 + a_1)(1 + a_2) \cdots (1 + a_{k-1}) \boxed{(1 + a_k)(1 + a_{k+1})} \geq 2^{k+1}.$$

Question 3

That is, if we can prove $(1 + a_k)(1 + a_{k+1}) \geq (1 + p) \cdot 2$ then we are done. Notice that $a_k \leq 1$ and $a_{k+1} \geq 1$, which implies $(1 - a_k)(a_{k+1} - 1) \geq 0$.

$$
\begin{aligned}
& (1 - a_k)(a_{k+1} - 1) \geq 0 \\
\Leftrightarrow\ & a_k + a_{k+1} - a_k a_{k+1} - 1 \geq 0 \\
\Leftrightarrow\ & a_k + a_{k+1} - a_k a_{k+1} - 1 + 2 + 2a_k a_{k+1} \geq 2 + 2a_k a_{k+1} \\
\Leftrightarrow\ & 1 + a_k + a_{k+1} + a_k a_{k+1} \geq 2 + 2a_k a_{k+1} \\
\Leftrightarrow\ & (1 + a_k)(1 + a_{k+1}) \geq (1 + p) \cdot 2.
\end{aligned}
$$

Thus, we know $(1 + a_k)(1 + a_{k+1}) \geq (1 + p) \cdot 2$ is true, and by mathematical induction, the claim is proved.

Question 4

4. (2.37) Consider the recurrence relation for Fibonacci numbers $F(n) = F(n-1)+F(n-2)$. Without solving this recurrence, compare $F(n)$ to $G(n)$ defined by the recurrence $G(n) = G(n-1)+G(n-2)+1$. It seems obvious that $G(n) > F(n)$ (because of the extra 1). Yet the following is a seemingly valid proof (by induction) that $G(n) = F(n)-1$. We assume, by induction, that $G(k) = F(k)-1$ for all k such that $1 \leq k \leq n$, and we consider $G(n+1)$:

$$G(n+1) = G(n) + G(n-1) + 1 = F(n) - 1 + F(n-1) - 1 + 1 = F(n+1) - 1$$

What is wrong with this proof?

Question 4

The proof is wrong since the base case of $G(n) = F(n) - 1$ is not proved.
In fact, there is no clear definition for $G(n)$.

$$F(n) = \begin{cases} 1 & \text{if } n = 1. \text{ [Base case]} \\ 1 & \text{if } n = 2. \text{ [Base case]} \\ F(n-1) + F(n-2) & \text{if } n \geq 3. \text{ [Inductive step]} \end{cases}$$

$$G(n) = \begin{cases} g_1 & \text{if } n = 1. \text{ [Base case]} \\ g_2 & \text{if } n = 2. \text{ [Base case]} \\ G(n-1) + G(n-2) + 1 & \text{if } n \geq 3. \text{ [Inductive step]} \end{cases}$$

As a result, the induction hypothesis does not hold at the beginning.

Question 4

If we set $G(1) = g_1 = 0$ and $G(2) = g_2 = 0$, then the base cases $G(1) = F(1) - 1$ and $G(2) = F(2) - 1$.
In this case, the proof will be correct.

If we set $G(1) = g_1 = 1$ and $G(2) = g_2 = 1$, then the base cases $G(1) \neq F(1) - 1$ and $G(2) \neq F(2) - 1$.
In this case, the proof will be wrong.

Question 5 (a)

(a) (10 points) Define inductively a function Max that determines the largest of all key values of a binary tree. Let $Max(\perp) = 0$, though the empty tree does not store any key value. (Note: use the usual mathematical notations; do not write a computer program.)

Question 5 (a)

Let T be a tree, the function *Max* is defined as follows:

$$Max(T) = \begin{cases} 0 & \text{if } T = \perp. \text{ [Base case]} \\ \mathbf{max}(k, \mathbf{max}(Max(t_l), Max(t_r))) & \text{if } T = node(k, t_l, t_r), \text{ [Inductive step]} \end{cases}$$

, where $\mathbf{max}(\mathrm{x},\ \mathrm{y}) = \begin{cases} x & \text{if } x > y \\ y & \text{otherwise.} \end{cases}$

Do not write pseudo-code unless the question requires.

Question 5 (b)

(b) (5 points) Suppose, to differentiate the empty tree from a non-empty tree whose largest key value happens to be 0, we require that $Max(\perp) = -1$. Give another definition for Max that meets this requirement; again, induction should be used somewhere in the definition.

Question 5 (b)

- Every key value in the binary tree with non-negative integer key values are larger than -1.
- Hence, the result of max function won't be affected by any empty node.
- We can apply the same inductive step as the former problem.

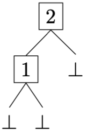
Let T be a tree, the function Max is defined as follows:

$$Max(T) = \begin{cases} -1 & \text{if } T = \perp. \text{ [Base case]} \\ \mathbf{max}(k, \mathbf{max}(Max(t_l), Max(t_r))) & \text{if } T = node(k, t_l, t_r), \text{ [Inductive step]} \end{cases}$$

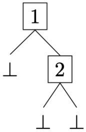
, where $\mathbf{max}(\text{x, y}) = \begin{cases} x & \text{if } x > y \\ y & \text{otherwise.} \end{cases}$

Question 5 (c)

(c) (5 points) Consider counting the number of binary trees of different shapes, ignoring the key values they store.

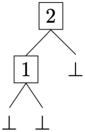


$node(2, node(1, \perp, \perp), \perp)$

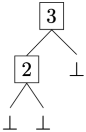


$node(1, \perp, node(2, \perp, \perp))$

The above two trees have different shapes, though they store the same key values.



$node(2, node(1, \perp, \perp), \perp)$



$node(3, node(2, \perp, \perp), \perp)$

The above two trees have the same shape, though they store different key values.

Let $b(n)$ denote the total number of distinctively-shaped binary trees with n nodes; for example, $b(0) = 1$, $b(1) = 1$, $b(2) = 2$, and $b(3) = 5$. Write a recurrence relation that defines $b(n)$, for $n \geq 0$.

Question 5 (c)

[Base Case] ($n = 0$) The tree is empty, so there is only one way to arrange it. $\Rightarrow b(0) = 1$

[Inductive Step] ($n > 0$)

For any binary tree with n nodes, we can think about how the tree is split into a left subtree(T_l) and a right subtree(T_r). For each different node count of T_l and T_r, we calculate the all the possible shapes they may form by $b(i) \times b(k)$, where i, k represents the node count of T_l and T_r, and $i + k = n - 1$.

Question 5 (c)

Let $n(T)$ denotes the node number of tree T. The number of binary trees of different shapes can be express as:

- $b(0) \times b(n-1)$, where $n(T_l) = 0$, $n(T_r) = n-1$
- $b(1) \times b(n-2)$, where $n(T_l) = 1$, $n(T_r) = n-2$

$$\vdots$$

- $b(n-1) \times b(0)$, where $n(T_l) = n-1$, $n(T_r) = 0$

Then we can derive the recurrence relation by summing up all the possible combinations. Thus, the recurrence relation is:

$$b(n) = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{i=0}^{n-1} b(i) \cdot b(n-i-1) & \text{if } n > 0 \end{cases}$$