Homework 2

Yu Hsiao Yu-Hsuan Wu

Question 1

Consider again the inductive definition in HW#1 for the set of all binary trees that store non-negative integer key values:

- The empty tree, denoted $\perp$, is a binary tree, storing no key value.
- If t_l and t_r are binary trees, then $node(k, t_l, t_r)$, where $k \in \mathbb{Z}$ and $k \geq 0$, is also a binary tree with the root storing key value k.

(a) Refine the definition to include only binary *search* trees where an inorder traversal of a binary search tree produces a list of all stored key values in *increasing* order.

(b) Further refine the definition to include only AVL trees, which are binary search trees where the heights of the left and the right children of every internal node differ by at most 1. You may reuse the function discussed in class for computing the height of a given binary tree.

In each case, the new definition should remain inductive.

Question 1

Inductive definition of all binary trees:

1. The empty tree, denoted $\perp$, is a binary tree.
2. If t_l and t_r are binary trees,
   then $node(k, t_l, t_r)$, where $k \in \mathbb{Z}$ and $k \geq 0$,
   is also a binary tree.

Question 1(a)

Inductive definition of binary **search** trees:

1. The empty tree, denoted $\perp$, is a binary search tree.
2. If t_l and t_r are binary search trees, and $Max(t_l) \leq k \leq Min(t_r)$, then $node(k, t_l, t_r)$, where $k \in \mathbb{Z}$ and $k \geq 0$, is also a binary search tree.

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

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

$$Min(T) = \begin{cases} +\infty & \text{if } T = \perp. \\ \mathbf{min}(k, \mathbf{min}(Min(t_l), Min(t_r))) & \text{if } T = node(k, t_l, t_r), \end{cases}$$

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

Question 1(b)

Inductive definition of **AVL** binary search trees:

1. The empty tree, denoted $\perp$ (with a height of -1), is an AVL binary search tree.
2. If t_l and t_r are AVL binary search trees, and $Max(t_l) \leq k \leq Min(t_r)$ and $|H(t_r) - H(t_l)| \leq 1$, then $node(k, t_l, t_r)$, where $k \in \mathbb{Z}$ and $k \geq 0$, is also an AVL binary search tree.

$$H(T) = \begin{cases} -1 & \text{if } T = \perp. \\ \textbf{max}(H(t_r), H(t_l)) + 1 & \text{if } T = node(k, t_l, t_r), \end{cases}$$

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

Question 2

Prove *by induction* that the regions formed by a planar graph all of whose vertices have even degrees can be colored with two colors such that no two adjacent regions (sharing one or more edges) have the same color.

Question 2

The proof is by strong induction on R.

[Base Case] ($R = 1$) Given a planar graph G with all of its vertices have even degrees and 1 region, we can always color the graph with two colors such that no two adjacent regions have the same color.

[Inductive Step] ($R = k + 1 > 1$) Suppose that for $R \leq k$, any planar graph G with all of its vertices have even degrees and R regions can be colored with two colors such that no two adjacent regions have the same color. We want to prove that for $R = k + 1$, the statement still holds.

Let H be a planar graph with all of its vertices have even degrees and $R = k + 1$ regions.

Question 2

Since all of its vertices have even degrees, we can always find a cycle on H. Now for simplicity, we arbitrarily pick a region, which is formed by a cycle. See Fig.1.

Note that if we remove the cycle, we get a new planar graph G with all of its vertices remaining even degrees (since removing a cycle will reduce 1 in-degree and 1 out-degree for any vertex on the cycle) and R' regions, $R' \leq k$ (since removing any edge on the cycle will merge the adjacent regions into one, so we cannot guarantee R' to be exactly k, however, we can make sure that $R' \leq k$).

That is, by Induction Hypothesis, G can be colored with two colors such that no two adjacent regions have the same color. See Fig.2.

Question 2

Moreover, when adding back the cycle and turning the graph back to $k+1$ regions, the recovered regions (separated by edges we removed at the beginning) now has the same color with its adjacent regions. See Fig.3.

Now we can turn the color of the recovered regions inside the cycle to the opposite one, and we can obtain a planar graph H with $k+1$ regions and colored with two colors such that no two adjacent regions have the same color, hence the proof is done. See Fig.4.

Question 2

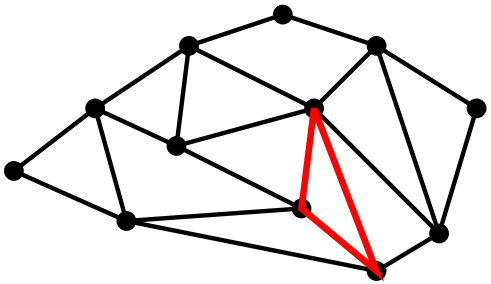

Figure 1: H with $k+1$ regions

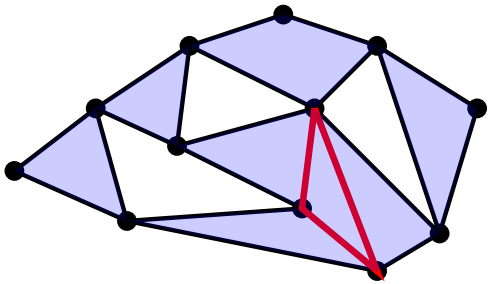

Figure 3: H with $k+1$ regions

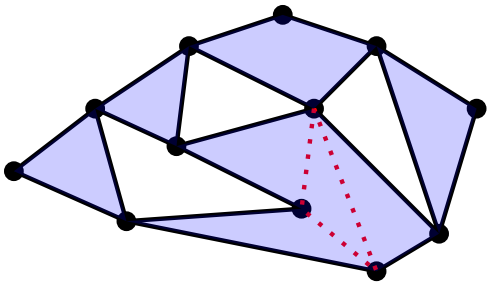

Figure 2: G with $R \leq k$ regions

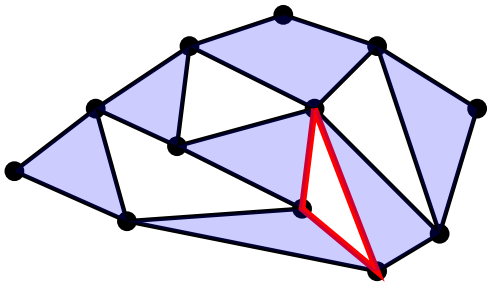

Figure 4: H with $k+1$ regions

Question 3

(2.30) A **full binary tree** is defined inductively as follows. (Note: some authors prefer using the name "perfect binary tree" or "complete binary tree", while reserving "full binary tree" for another variant of binary trees.) A full binary tree of height 0 consists of 1 node which is the root. A full binary tree of height $h+1$ consists of two full binary trees of height h whose roots are connected to a new root. Let T be a full binary tree of height h. The **height** of a node in T is h minus the node's distance from the root (e.g., the root has height h, whereas a leaf has height 0). Prove that the sum of the heights of all the nodes in T is $2^{h+1} - h - 2$.

Question 3

The proof is by induction on h.

[Claim] For a full binary tree T of height h, The sum of the heights of all nodes in T is $2^{h+1} - h - 2$.

[Base Case] $(h = 0)$ $2^1 - 0 - 2 = 0$

[Inductive Step] $(h > 0)$

The sum of the heights for a tree with height h

$= (\text{sum of the heights for } t_l \text{ and } t_r + \text{height of the root})$

$= 2 \times \left(2^h - (h-1) - 2\right) + h$

$= 2^{h+1} - 2h - 2 + h$

$= 2^{h+1} - h - 2$

Question 4

Consider the following two-player game: given a positive integer N, player A and player B take turns counting to N. In her/his turn, a player may advance the count by 1 or 2. For example, player A may start by saying "1, 2", player B follows by saying "3", player A follows by saying "4", etc. The player who eventually has to say the number N loses the game.

A game is *determined* if one of the two players always has a way to win the game. Prove *by induction* that the counting game as described is determined for any positive integer N; the winner may differ for different given integers. (Hint: think about the remainder of the number N divided by 3.)

Question 4

The proof is by induction on N.

Without loss of generality, we let A always be the starter.

[Claim] A game is always determined with $N \geq 1$ by:

$N \equiv 1 \pmod 3$: B wins

$N \equiv 2 \pmod 3$: A wins

$N \equiv 0 \pmod 3$: A wins

[Base Case] $(N = 1, 2, 3)$

$N = 1$: A is forced to count "1", so B wins.

$N = 2$: A has a winning strategy by counting "1", so A wins.

$N = 3$: A has a winning strategy by counting "1, 2", so A wins.

Question 4

[Inductive Step] ($N > 3$)

$N \equiv 1 \pmod 3$:

By induction hypothesis, we know that B has a winning strategy at $N - 3$. It implies B can always force A to say "N - 3". No matter what A says in the next round ("N - 3" or "N - 3, N - 2") B can always count up to "N", forcing A to say "N".

$N \equiv 0, 2 \pmod 3$:

By induction hypothesis, we know that A has a winning strategy at $N - 3$. It implies A can always force B to say "N - 3". No matter what A says in the next round ("N - 3" or "N - 3, N - 2") A can always count up to "N", forcing B to say "N".

Question 5

Consider the following algorithm for computing the square of the input number n, which is assumed to be a positive integer.

Algorithm mySquare(n);
begin
    // assume that $n > 0$
    $x := n$;
    $y := 0$;
    while $x > 0$ **do**
        $y := y + 2 \times x - 1$;
        $x := x - 1$
    od;
    mySquare := y
end

State a suitable loop invariant for the while loop and prove its correctness *by induction*. The loop invariant should be strong enough for deducing that, when the while loop terminates, the value of y equals the square of n.

Question 5

To start with, we first state a suitable loop invariant for the while loop. Observe that when running the while loop, the relation of x and y is as follow:

x	n	$n-1$	$n-2$	$n-3$	$\cdots$	$n-k$	$\cdots$
y	0	$2n-1$	$4n-4$	$6n-9$	$\cdots$	$(2k)n-k^2$	$\cdots$

That is, we have $x = n - k, y = (2k)n - k^2$.
Now we apply $k = n - x$ to y:

$$\begin{aligned} y &= (2k)n - k^2 \\ &= 2(n-x)n - (n-x)^2 \\ &= 2n^2 - 2xn - (n^2 - 2xn + x^2) \\ &= n^2 - x^2. \end{aligned}$$

Question 5

That is, we know that $y = n^2 - x^2$ throughout the while loop, and since $x = n$ at the beginning and $x = 0$ at the end, we know that $n \geq x \geq 0$.
Thus, we find the loop invariant to be $\{0 \leq x \leq n \wedge y = n^2 - x^2\}$.
Now, we want to prove its correctness by induction.

[Base case] (0-th iteration) $\{0 \leq n \leq n \wedge y = n^2 - n^2 = 0\}$.

[Inductive step] (k-th iteration) Suppose the loop invariant holds after the $(k-1)$-th iteration, and we want to prove the correctness of the loop invariant after the k-th iteration.

Let (x_k, y_k) be the (x, y) values at the k-th iteration. From the given algorithm, we have $(x_k, y_k) = (x_{k-1} - 1, y_{k-1} + 2x_{k-1} - 1)$.

Question 5

Recall we have the loop invariant $\{0 \leq x \leq n \wedge y = n^2 - x^2\}$.

- $0 \leq x \leq n$: $0 \leq x_k \leq n$ is true, since $x_k = x_{k-1} - 1$ and $0 < x_{k-1} \leq n$.

  Note that x_{k-1} should be larger than 0. If $x_m = 0$ for some $m \leq n$, then the program terminates at the m-th iteration. Since we are discussing the k-th iteration, we can guarantee that the program won't terminates at the $(k-1)$-th iteration.

- $y = n^2 - x^2$: We can derive this result by some substitutions:

$$\begin{aligned} y_k &= y_{k-1} + 2x_{k-1} - 1 \\ &= (n^2 - x_{k-1}^2) + 2x_{k-1} - 1 \\ &= n^2 - (x_{k-1} - 1)^2 \\ &= n^2 - x_k^2. \end{aligned}$$

Question 5

Thus, we prove the correctness of the loop invariant by induction.

Note that when the while loop terminates, which means $x = 0$, with the loop invariant $\{0 \leq x \leq n \wedge y = n^2 - x^2\}$, we have $y = n^2 - x^2 = n^2 - 0^2 = n^2$, which satisfies our assertion that the value of y equals the square of n.