Assignment 2

Yu-Chieh Kuo B07611039†

†Department of Information Management, National Taiwan University

Problem 1

1.(a)

The redefinition is shown as below.

- The empty tree, denoted $\perp$, is a binary *search* tree, storing no key value.
- If t_l and t_r are binary *search* tree, every key value (of descendants) in the nodes of t_l is smaller than k, and every key value (of descendants) in the nodes of t_r is larger than k, then $node(k, t_l, t_r)$, where $k \in \mathbb{Z}$ and $k \geq 0$, is also a binary *search* tree with the root storing key value k.

1.(b)

To define AVL trees only, we need to introduce the additional auxiliary *height* and *balance* function denoted as

$$\begin{aligned} \mathcal{H}(node(k, t_l, t_r)) &:= \begin{cases} 0 & \text{if tree is empty} \\ \max\left(\mathcal{H}(t_l), \mathcal{H}(t_r)\right) + 1 & \text{if tree is nonempty} \end{cases} \\ \mathcal{B}(node(k, t_l, t_r)) &:= \mathcal{H}(t_r) - \mathcal{H}(t_l), \end{aligned}$$

where the *height* function $\mathcal{H}(node)$ calculates the height of a tree, and the *balance* function $\mathcal{B}(node)$ determines the node's balance level. A binary tree is an AVL tree if

$$\mathcal{B}(node) \in \{-1, 0, 1\}$$

holds for all nodes in the tree. Therefore, we can write the definition of AVL tree formally as

- The empty tree, denoted $\perp$, is an AVL tree, storing no key value.
- If t_l and t_r are binary *search* tree, every key value (of descendants) in the nodes of t_l is smaller than k, and every key value (of descendants) in the nodes of t_r is larger than k; moreover, $\mathcal{B}(node) \in \{-1, 0, 1\}$ holds for every node in the tree, then $node(k, t_l, t_r)$, where $k \in \mathbb{Z}$ and $k \geq 0$, is an AVL tree with the root storing key value k.

Problem 2

We begin with a simplified version by substituting $\mathbf{log_2 k}$ for $\lceil log_2 k \rceil$ and substituting **positive integer** $k = 2^i$ for any positive integer k, where $i \geq 1$. In this version, the base case is trivially satisfied since there exists Gray codes of length $\log_2^2 = 1$. Assume we can find Gray code of length $n-1$ for $k = 2^{n-1}$, $n \in \mathbb{N}$, as the inductive step $k = 2^n = 2 \times 2^{n-1}$, then we just need to add a bit and connect them together to result in the length of $n = \log_2 2^n$. We denote this proposition as Proposition 1 to ease note.

Next, we put the attention to the original statement. In base case, we set $k = 2^{n-1},\ n > 1$. By the previous proof, we can get the Gray code of length $\log_2 n$ when $n = 2^i,\ i > 0$, and the length keeps in $\log_2 n = \lceil \log_2 k \rceil$ after deleting one code of them. In the inductive step, we consider $2^{n-1} < k < 2^n - 1$. For every off $k+2$, we can find the Gray code of length $\lceil \log_2 (k+2) \rceil = n$ for $k+2$ objects, which we just need to delete the first and the last one. We denote this proposition as Proposition 2 to ease note.

To prove the Gray codes for the *even* values of k are *closed*, the base case for $k = 2$ is true by Proposition 1. In the inductive step, we consider $k = 2i,\ i > 1$. By Proposition 2, we can find an open Gray codes of length $\lceil \log_2 k \rceil$ for odd numbers, then we can add an additional bit and connect it. The procedure to prove the Gray codes for *odd* values of k are *open* remains similar by considering the case $k = 2i - 1,\ i > 1$.

Problem 3

The height increases by one when the full binary tree creates a new root node to connect the origin full binary tree. Given such property, we observe that the sum of the heights of all the nodes in T is the sum of the sub-tree and the height of root. Denote the sum of the heights h of all nodes in T by $T(h)$, the base case holds since when $h = 0$, $T(h) = 1 = 2^{1+1} - 1 - 2$. By induction hypothesis to assume that the property holds for all $h = n$, in the case $h = n+1$,

$$\begin{aligned} T(n+1) &= 2T(n) + (n+1) \\ &= 2 \cdot (2^{n+1} - n - 2) + (n+1) \\ &= 2^{n+2} - 2(n+1) - 2 \end{aligned}$$

Therefore, the heights of all the nodes in $T = 2^{h+1} - h - 2$ is proved by induction.

Problem 4

In the base case *i.e.* $p = 3,\ q = 0$, the polygon is a triangle and the corresponding area is $\frac{1 \times 1}{2} = \frac{1}{2} = \frac{3}{2} + 0 - 1$. Assume the statement holds for all simple polygons, we then consider the general condition for $p \geq 3,\ q \geq 0$, and we can divide the origin polygon into a triangle T and a smaller polygon P' with one edge connected. Let the number of lattice points on the connected edge be c, we have

$$\begin{aligned} q &= q_{P'} + q_T + (c-2) \\ p &= p_{P'} + p_T - 2(c-2) - 2 \end{aligned}$$

Notice that $c - 2$ means we need to deduct the two exception endpoints on the edge. Rewriting the above formula gives

$$\begin{aligned} q_{P'} + q_T &= q - (c-2) \\ p_{P'} + p_T &= p + 2(c-2) + 2 \end{aligned}$$

Let the area of the origin polygon, the divided polygon and the corresponding divided triangle are A_P, $A_{P'}$ and A_T, separately, then we obtain

$$\begin{aligned} A_P &= A_{P'} + A_T \\ &= (q_{P'} + \frac{p_{P'}}{2} - 1) + (q_T + \frac{p_T}{2} - 1) \\ &= q_{P'} + q_T + \frac{p_{P'} + p_T}{2} - 2 \\ &= q - (c-2) + \frac{p + 2(c-2) + 2}{2} - 2 \\ &= q + \frac{p}{2} - 1 \end{aligned}$$

Therefore, if the statement is satisfied for polygons constructed by n triangles, it is also satisfied for polygons constructed by $n+1$ triangles. Consequently, the area of polygon is $\frac{p}{2} + q - 1$.

Problem 5

The loop invariant for the main loop is

$$Inv(last, A, n) = (1 \leq last \leq n) \wedge (\forall last + 1 \leq i \leq n,\ indexofLargest(A', 1, i) = i)$$

Given an array A with the length of n and $last = n$, the array changes after the loop executes by k steps, that is,

$$Inv(n-k, A'_k, n) \rightarrow Inv(n-(k+1), A'_{k+1}, n),$$

where A'_i represents a modified array after i steps in the loop.

To prove its correctness, we start with the base case for $k = 0$ and $last = n$.

$$Inv(n, A, n) = (1 \leq n \leq n) \wedge (\forall n + 1 \leq i \leq n,\ indexofLargest(A', 1, i) = i)$$

is true for sure. Assume the inductive case is true for $1 < k \leq n-1$, $last = n-k$, by the inductive hypothesis of $last = n-k+1$, we obtain

$$\forall n - k + 2 \leq i \leq n,\ indexofLargest(A', 1, i) = i.$$

Moreover, the nature of selection sort gives that on the k-th iteration of the loop, we pick the largest element between $A'_k[1]$ and $A'_k[nk+1]$ to put in the $(n-k+1)$-th position, says $A'_{k+1}[n-k+1]$. $A'_{k+1}[n-k+1]$, therefore, is larger than elements on its left side, which shows

$$indexofLargest(A'_{k+1}[n-k+1], 1, n-k+1) = n-k+1 \iff indexofLargest(A', 1, i) = i.$$

Hence, $\forall last + 1 \leq i \leq n,\ indexofLargest(A', 1, i) = i$ holds with $last = n-k$, and the correctness of the loop invariant is proved.