Homework 2

Yu-Chieh Kuo B07611039

This homework answers the problem set sequentially.

1. To refine the definition to include only BST, we can re-write the definition as below. If both t_l and t_r are BST and the key value in t_l and the every key value in t_r satisfy the following relationship:

$$max(t_l) < k < min(t_r)$$

where k is the key value of the root in the BST, and $max(\cdot)$ function returns the maximum value in the left hand side in the tree or 0 when the tree is empty, $min(\cdot)$ function returns the minimum value in the right hand side in the tree or infinite when the tree is empty, then we can say the node $node(k, t_l, t_r)$ is a BST.
To design a function that returns the rank of a given key value, we need to do the in-order traversal to the BST. Also, we have to design a recursive function to count the rank in the tree, and a function to check whether the given value exists or not. Let the given key value be x, the function returning the rank of the given value $Rank(t, x)$, the recursive function to count the rank in the tree $Count(t)$, and the function to check the existence of given key value $Exist(t, x)$ we have

$$Exist(t, x) = \begin{cases} false & \text{if t is an empty BST} \\ true & \text{if } t \text{ is not an empty BST} \\ Exist(t_l, n) & \text{if } x < k \\ Exist(t_r, n) & \text{if } x > k \end{cases}$$

$$Count(t) = \begin{cases} 0 & \text{if t is an empty BST} \\ Count(t_l) + Count(t_r) + 1 & \text{if } Exist(t, x) \text{ is } true \end{cases}$$

$$Rank(t, x) = \begin{cases} 0 & \text{if } Exist(t, x) \text{ is } false \\ Rank(t_l, x) & \text{if } x < k \\ Count(t_l) + 1 & \text{if } x = k \\ Count(t_l) + 1 + Rank(t_r, x) & \text{if } x > k \end{cases}$$

Note that all t above means the nodes $node(k, t_l, t_r)$.

2. In the base case, when $h = 2$, $T(2) = T(1)+T(0)+1 = 2$, and also $T(2) = F(4)-1 = 2$, which shows the property is hold obviously. By induction hypothesis, we assume that for all $h = n$, it obtains the property *i.e.* $T(h) = T(h-1) + T(h-2) + 1 = F(h+2) - 1$, when the case $h = n+1$,

$$\begin{aligned} T(n+1) &= T(n) + T(n-1) + 1 \\ &= (F(n+2) - 1) + (F(n+1) - 1) + 1 \\ &= F(n+2) + F(n+1) + 1 \\ &= F(n+3) + 1 \end{aligned}$$

Therefore, we proof the relation $T(h) = F(h+2) - 1$, where $F(n)$ is the n-th Fibonacci number, by induction.

3. Due to the property of the full binary tree, we can observe that the height increasing one layer means it creates a new root node to connect the origin full binary tree. As a result, the sum of the heights of all the nodes in T will be the sum of the sub-tree and the height of root. Denote the sum of the heights h of all nodes in T be $T(h)$, in the base case, when $h = 0$, $T(h) = 1 = 2^{1+1} - 1 - 2$, which shows we obtain the property . By induction hypothesis, we assume that for all $h = n$, it obtains the property, 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, by induction, we proof that the heights of all the nodes in T is $2^{h+1} - h - 2$.

4. In the base case *i.e.* $p = 3,\ q = 0$, the polygon is a triangle, and the area of the triangle is $\frac{1\times 1}{2} = \frac{1}{2} = \frac{3}{2} + 0 - 1$. Assume that the property holds for all simple polygons, consider the general condition for $p \geq 3,\ q \geq 0$, 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 that we need to deduct the two exception endpoints on the edge. Re-write the above formula, we have

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

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

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

Therefore, if the property is true for polygons constructed from n triangles, it is also satisfied for polygons constructed from $n+1$ triangles. As a result, we proof that the area of polygon is $\frac{p}{2} + q - 1$.

5. We denote the loop invariant assertion as below.
At the start of each iteration of the loop, the sub-list A[i+1...n] consists

of the largest elements originally in A in sorted order, and the sub-list A[0...i] contains the remaining elements in A.

In the base case, the index of last element $i = n$, the sub-list to the right is empty, which is easy to say that an empty sub-list is ordered and obeys the loop invariant.

In the inductive step, at the start of any arbitrary iteration, no element from i to n can ever be disturbed again. During the step, the inner loop will find the largest element in **A[0...i]** and will swap it for the **A[i]** element. This element is swapped into the i-th position. The new **A[i]** element cannot be larger than any element in **A[i+1...n]** because the loop invariant is true prior to the start of the iteration, so it will simultaneously be the largest element from 0 to the element of index $i - 1$ and no smaller than any element to its left. The loop invariant is preserved.

Therefore, we can prove the algorithm's correctness via stating a suitable loop invariant.