CSCB63 Assignment1

Jieying Gong
1009388066

Question 1.

1.1

Proof. For all function f, g, h in $\mathbb{N} \to \mathbb{R}^+$, assume $f \in \mathcal{O}(g)$, by definition of Big-Oh, $\exists\ c \in \mathbb{R}^+, \exists\ n_0 \in \mathbb{N}, \forall\ n \in \mathbb{N}$, $n \geq n_0 \Rightarrow f(n) \leq c \cdot g(n)$. And assume $g \in \mathcal{O}(h)$, by definition, $\exists\ d \in \mathbb{R}^+, \exists\ m_0 \in \mathbb{N}, \forall\ m \in \mathbb{N}, m \geq m_0 \Rightarrow g(m) \leq d \cdot h(m)$.
Let $k_0 = max\{n_0, m_0\}$, $\forall\ k \in \mathbb{N}, k \geq k_0$, then $k \geq n_0$ and $k \geq m_0$. We have $f(k) \leq c \cdot g(k) \leq c \cdot d \cdot h(k)$. Since $c \in \mathbb{R}^+$ and $d \in \mathbb{R}^+$, let $l = c \cdot d \in \mathbb{R}^+$. Thus for all functions $f \in \mathcal{F}$, $\exists\ k \in \mathbb{R}^+, \exists\ k_0 \in \mathbb{N}, \forall\ k \in \mathbb{N}, k \geq k_0 \Rightarrow f(k) \leq l \cdot h(k)$. By definition of Big-Oh, $f \in \mathcal{O}(h)$ as wanted. □

1.2

Proof. For all function f_1, f_2, g_1, g_2 in $\mathbb{N} \to \mathbb{R}^+$, assume $f_1 \in \mathcal{O}(g_1)$, by definition of Big-Oh, $\exists\ c \in \mathbb{R}^+, \exists\ n_0 \in \mathbb{N}$, $\forall\ n \in \mathbb{N}, n \geq n_0 \Rightarrow f_1(n) \leq c \cdot g_1(n)$. And assume $f_2 \in \mathcal{O}(g_2)$, by definition, $\exists\ d \in \mathbb{R}^+, \exists\ m_0 \in \mathbb{N}, \forall\ m \in \mathbb{N}, m \geq m_0 \Rightarrow f_2(m) \leq d \cdot g_2(m)$.
Let $k_0 = max\{n_0, m_0\}$, $\forall\ k \in \mathbb{N}, k \geq k_0$, then $k \geq n_0$ and $k \geq m_0$. We have $f(k) = f_1(k) \cdot f_2(k) \leq c \cdot g_1(k) \cdot d \cdot g_2(k) = cd \cdot g_1(k)g_2(k) = cd \cdot g(k)$. Since $c \in \mathbb{R}^+$ and $d \in \mathbb{R}^+$, let $l = c \cdot d \in \mathbb{R}^+$. Let $l = c \cdot d \in \mathbb{R}^+$. Thus for all functions $f \in \mathcal{F}$, $\exists\ k \in \mathbb{R}^+, \exists\ k_0 \in \mathbb{N}, \forall\ k \in \mathbb{N}, k \geq k_0 \Rightarrow f(k) \leq l \cdot g(k)$. By definition of Big-Oh, $f \in \mathcal{O}(g)$ as wanted. □

1.3

Proof. We will use proof by contradiction. Suppose $\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow 0 \leq 2^{2n} \leq c \cdot 2^n$.
Now we have for all $n \geq max(n_0, 1) \geq n_0$:

$$\begin{aligned} & 2^{2n} \leq c \cdot 2^n \\ \iff & 2^n \leq c && \text{divide } 2^n \text{ by both side} \\ \iff & n \leq lg(c) \end{aligned}$$

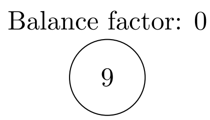
Choosing $n = max(n_0, \lceil lg(c) \rceil + 1) > lg(c)$ makes a contradiction. Therefore, $2^{2n} \notin \mathcal{O}(2^n)$ as wanted. □

1.4

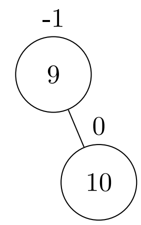
Proof. For all function f_1, f_2, g in $\mathbb{N} \to \mathbb{R}^+$, assume $f_1 \in \mathcal{O}(g)$, by definition of Big-Oh, $\exists\ c \in \mathbb{R}^+, \exists\ n_0 \in \mathbb{N}, \forall\ n \in \mathbb{N}$, $n \geq n_0 \Rightarrow f_1(n) \leq c \cdot g(n)$. And assume $f_2 \in \mathcal{O}(g)$, by definition, $\exists\ d \in \mathbb{R}^+, \exists\ m_0 \in \mathbb{N}, \forall\ m \in \mathbb{N}, m \geq m_0 \Rightarrow f_2(m) \leq d \cdot g(m)$.
Let $k_0 = max(n_0, m_0)$, $\forall k \in \mathbb{N}, k \geq k_0$, then $k \geq n_0$ and $k \geq m_0$. We have $f_{max}(k) = max(f_1(k), f_2(k)) \leq f_1(k) + f_2(k) \leq c \cdot g(k) + d \cdot g(k) = (c + d) \cdot g(k)$. Since $c \in \mathbb{R}^+$ and $d \in \mathbb{R}^+$, let $l = c + d \in \mathbb{R}^+$.Thus for all functions $f_{max} \in \mathcal{F}$, $\exists\ k \in \mathbb{R}^+, \exists\ k_0 \in \mathbb{N}, \forall\ k \in \mathbb{N}, k \geq k_0 \Rightarrow f(k) \leq l \cdot g(k)$. By definition of Big-Oh, $f \in \mathcal{O}(g)$ as wanted. □

Question 2.

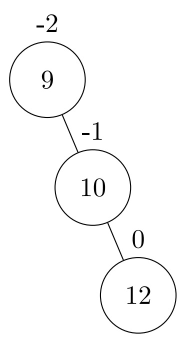
2.1



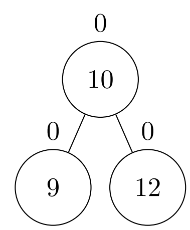
Step 1: Insert node with key 9



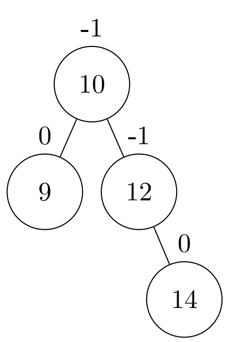
Step 2: Insert node with key 10



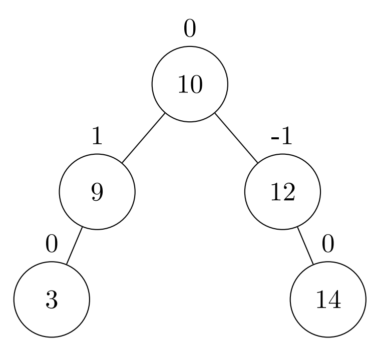
Step 3: Insert node with key 12



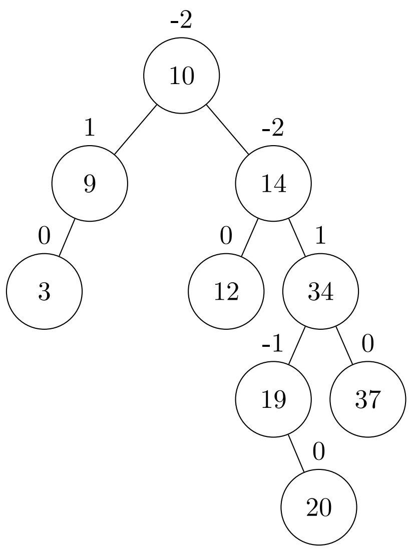
Step 4: Rebalance, single notation counter-clockwise on node with keys 9, 10, 12



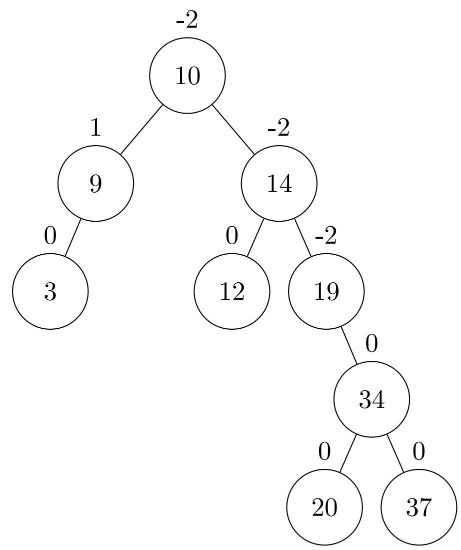
Step 5: Insert node with key 14



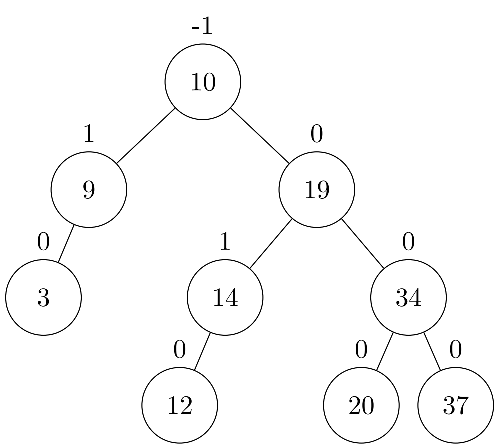
Step 6: Insert node with key 3



Step 11: Insert node with key 20. Need double notation



Step 12: Clockwise notation on node with keys 19, 20, 34, 37 centered at 34



Step 13: Counter-clockwise notation on node with keys 12, 14, 19, 20, 34 37 centered at 19

Question 4.

4. helper function

```
function updateMaxEnd(node):
if node is NULL:
    return
maxEnd = node.end
if node.left is not NULL:
    maxEnd = max(maxEnd, node.left.maxEnd)
if node.right is not NULL:
    maxEnd = max(maxEnd, node.right.maxEnd)
node.maxEnd = maxEnd
```

4.3

1. `insert` **Function**:

   - **Correctness**: The `insert` function correctly places a new interval $[x, y]$ into the tree based on the `start` value primarily and `end` value secondarily, ensuring that intervals are ordered efficiently. Duplicates are not inserted multiple times, preserving the set's integrity.
   - **Time Complexity**: Maintains the AVL tree's balanced property, ensuring the height of the tree remains logarithmic relative to the number of nodes, n, making the time complexity $O(\log n)$.

2. `delete` **Function**:

   - **Correctness**: Navigates the tree to find and remove the specific interval $[x, y]$, handling scenarios of no children, one child, and two children with the successor's substitution, thus maintaining the tree's integrity.
   - **Time Complexity**: Like insertion, it involves updating `maxEnd` and rebalancing, keeping the path traversed during deletion logarithmic in the number of nodes. Hence, deletion also achieves $O(\log n)$ complexity.

3. `meetAll` **Function**:

   - **Correctness**: Iteratively traverses the tree to check if all intervals meet the overlap condition with $[l, h]$, with an allowance for one interval not to overlap, using a flag `hasNonOverlap` and examining `maxEnd` and `start` properties efficiently.
   - **Time Complexity**: Despite iterative checks, the AVL tree's balanced nature ensures the number of nodes visited is bounded by the tree's height, $O(\log n)$. The efficient narrowing of the search space maintains the overall $O(\log n)$ complexity.