CSCI-GA 1170 Homework 3

Question 1

We still need 6 rotations. Rotation only change the structure of a binary tree, i.e. the keys are not changed during rotation. So the successor/predecessor pointers of nodes are unaffected, which means that 6 pointer assignments is sufficient.

Question 2

(a)

By Lemma 6, we know $\mu(h) \geq \phi^h$. So, $100 \geq \mu(h) \geq \phi^h \Rightarrow h \leq \log_\phi 100 = 9.5699$. Thus, the maximum possible height achievable by an AVL tree of size 100 is 9.

(b)

From the textbook, we know $\mu(h) = 1 + \mu(h-1) + \mu(h-2)$ and $\mu(1) = 2, \mu(2) = 4$. So we have $\mu(3) = 7, \mu(4) = 12, \mu(5) = 20, \mu(6) = 33, \mu(7) = 54, \mu(8) = 88$ and $\mu(9) = 143 > 100$. Therefore, $h \leq 8$. On the other hand, since $M(h) = 2^{h+1} - 1$ and $M(5) < 100, M(6) > 100$, we can get $h \geq 6$. In conclusion, the range of heights for an AVL tree of size 100 is $6 \leq h \leq 8$.

Question 3

(a)

Let $t(n) = T(n) - d$, where $d = \frac{C}{1-\sum_{i=1}^{k} a_i}$ (here we assume $1 - \sum_{i=1}^{k} a_i \neq 0$). Then we have

$$\begin{aligned} t(n) &= T(n) - d \\ &= C + (\sum_{i=1}^{k} a_i T(n-i)) - d \\ &= (1 - \sum_{i=1}^{k} a_i)d + \sum_{i=1}^{k} a_i T(n-i) - d \\ &= \sum_{i=1}^{k} a_i (T(n-i) - d) \\ &= \sum_{i=1}^{k} a_i t(n-i) \end{aligned}$$

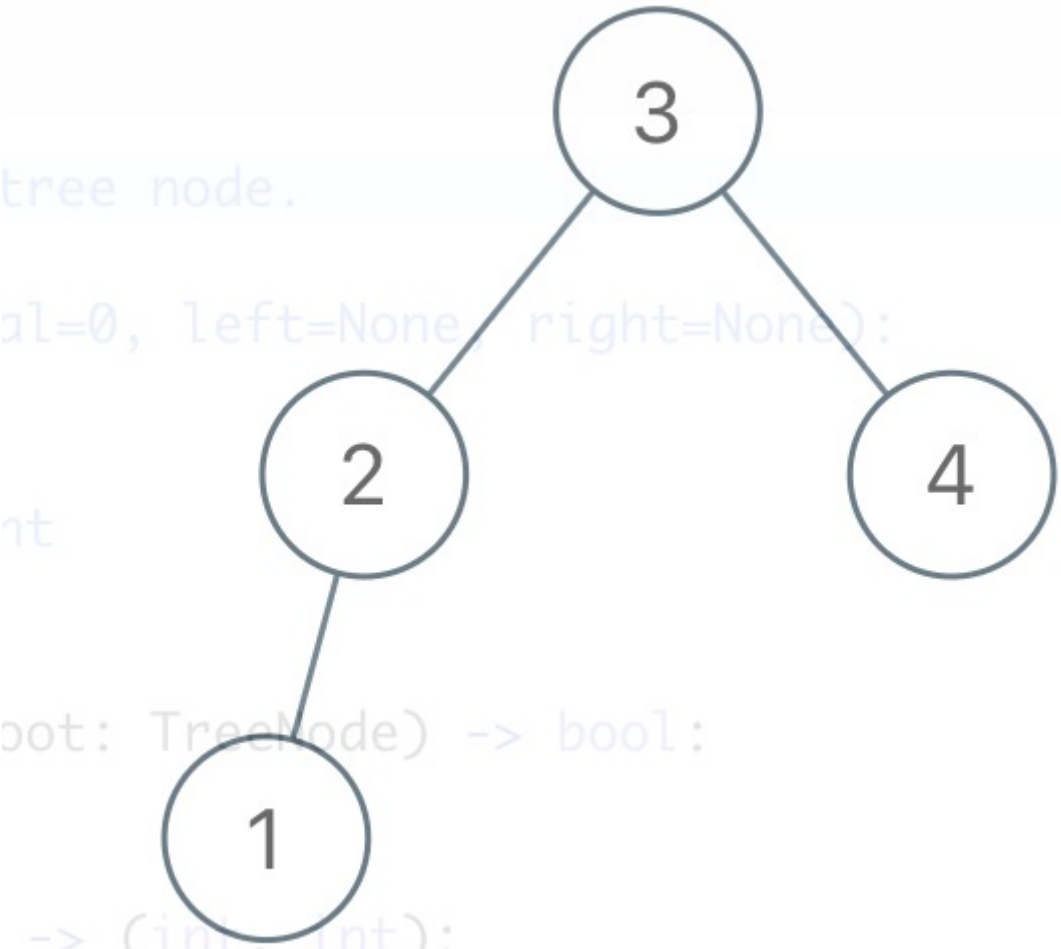
(b)

From part(a), we have $d = -1$. Let $T(h) = \mu(h) + 1$, so $T(h) = T(h-1) + T(h-2)$. Then, $x^2 - x - 1 = 0$ is the characteristic equation of the recurrence. So $T(h) = A\phi^h + B\hat{\phi}^h$ is a solution. Here we solve A and B using the initial conditions $T(0) = \mu(0) + 1 = 2$ and $T(1) = \mu(1) + 1 = 3$ and get $A = 1 + \frac{2\sqrt{5}}{5}$ and $B = 1 - \frac{2\sqrt{5}}{5}$. In conclusion, $\mu(h) = (1 + \frac{2\sqrt{5}}{5})\phi^h + (1 - \frac{2\sqrt{5}}{5})\hat{\phi}^h - 1$.

Question 4

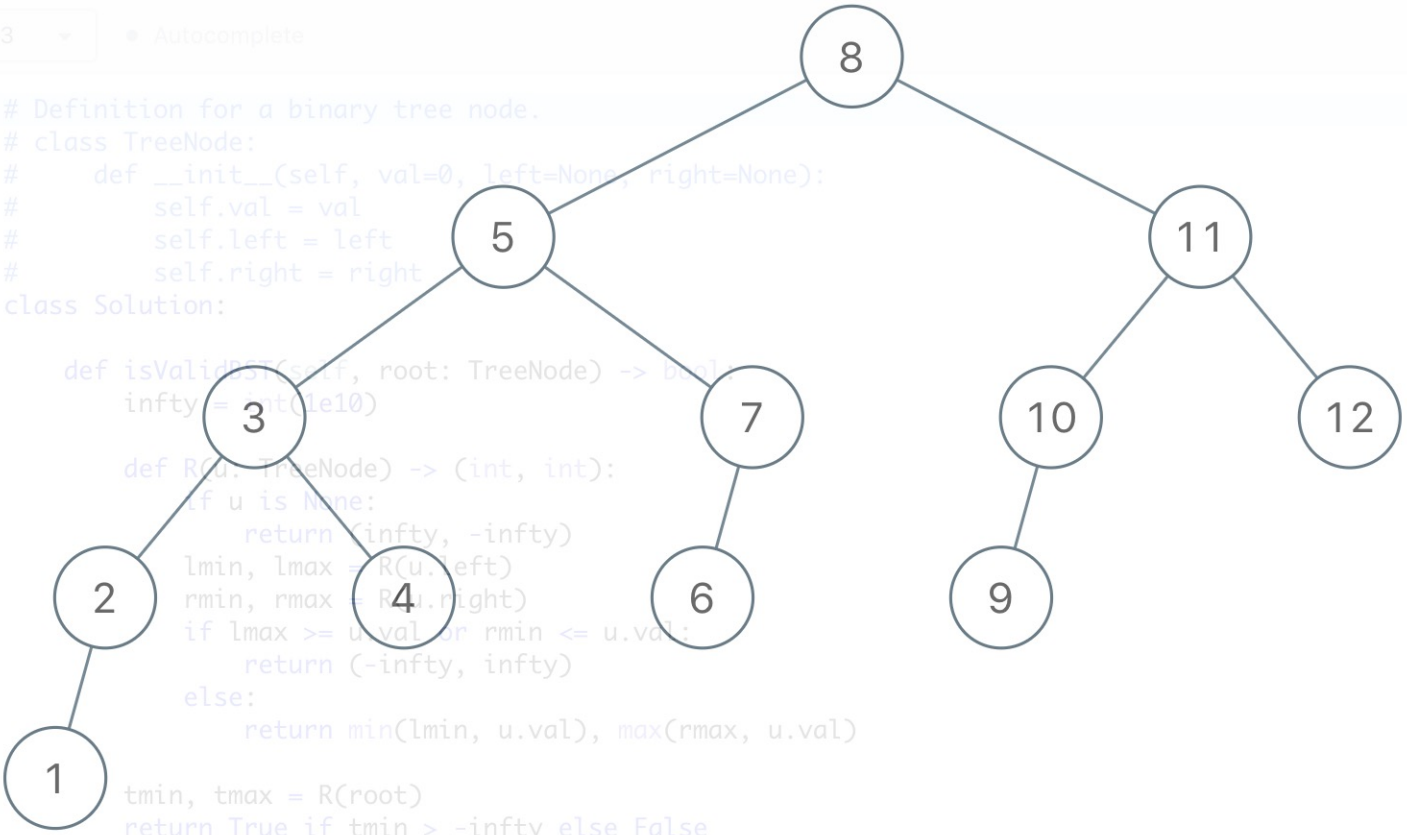
(a)

$m(1)$



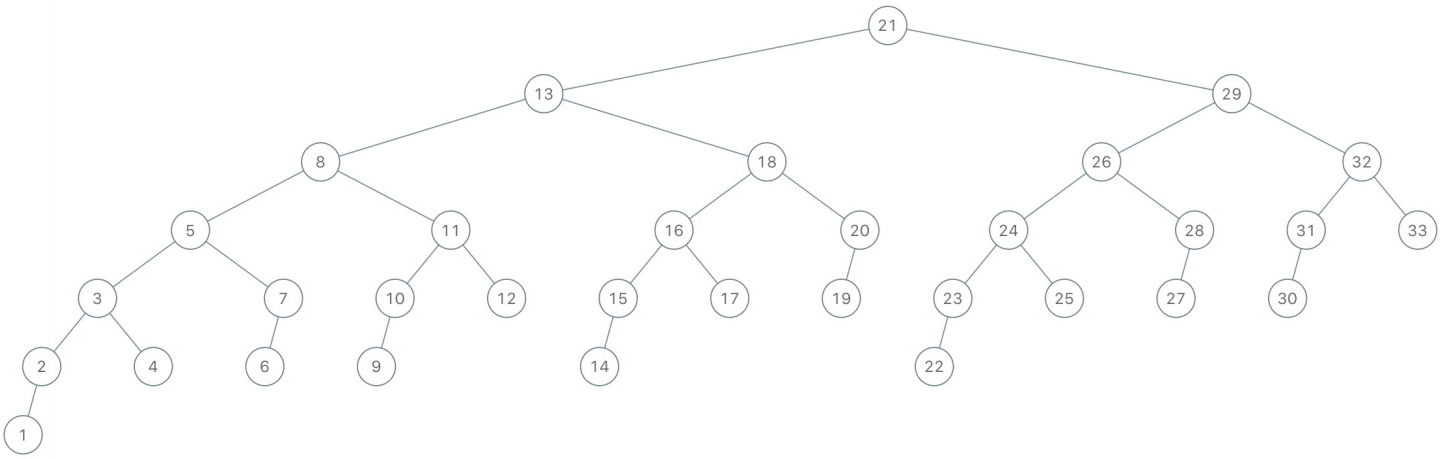
Delete Node 4 will cause 1 rebalancing act.

$m(2)$



Delete Node 12 will cause 2 rebalancing acts.

$m(3)$



Delete Node 33 will cause 3 rebalancing acts.

(b)

From part (a), we know that $m(1) = \mu(2)$. Let $h(k)$ denote the height of the minimum size AVL tree that a single deletion will cause k rebalancing acts. Then for $k \geq 2$, we have

$$\begin{aligned} m(k) &= m(k-1) + 1 + \mu(h(k-1) - 1 + 2) \\ &= \mu(2k-2) + 1 + \mu(2k-2-1+2) \\ &= 1 + \mu(2k-2) + \mu(2k-1) \\ &= \mu(2k) \end{aligned}$$

(c)

Yes. Consider an AVL tree T_k of size $m(k)$. Clearly, by definition of $m(k)$, among all AVL trees that a single deletion will cause k rebalancing acts, T_k has the minimum size. Note that among all AVL trees of height $2k$, T_k also has the minimum size. If T_k is the canonical min-size AVL tree of size $2k$, then the rebalacing acts are all single rotations, because the left subtree will always be higher than the right one, which is CASE (D.a) in the textbook. However, note that for each unbalanced node u_i we encounter, let v_i be its left child, we can exchange the roles of the siblings of v_i to CASE (D.b), in which a double rotation is applied. In conclusion, we can design the AVL tree for $m(k)$ to be any combination of k single or double rotations.

(d)

In the text, we have $\phi^h \leq \mu(h)$, so we have $n \geq m(k) = \mu(2k) \geq \phi^{2k}$. Therefore,

$$k \leq \log_{\phi^2}(n) < 0.72 \log n$$

Question 5

(a)

def isBST(u) -> bool:

- def R(u) -> (number, number):
  - if u is nil:
    - return $(+\infty, -\infty)$
  - lmin, lmax = R(u.left)
  - rmin, rmax = R(u.right)
  - if lmax >= u.key or rmin < u.key: # not a BST
    - return $(-\infty, +\infty)$
  - else:
    - return (min(lmin, u.key), max(rmax, u.key))
  - umin, umax = R(u)
  - if tmin > $-\infty$ && tmax < $+\infty$:
    - return true
  - else:
    - return false

(b)

The recursive routine $R(u)$ returns a pair of numbers $u_{\min}$ and $u_{\max}$ which represents the minimum key and the maximum key in the tree u respectively. Note that if u is nil, then $R(u)$ returns $(+\infty, -\infty)$, or if u is not a BST, then it returns $(-\infty, +\infty)$ for programming convenience. In our routine, we first check if u is nil as the base case. Then we recursively call R on the left subtree and the right subtree and get the range of keys in the subtrees. Next, we check if the maximum key in the left subtree $\geq$ the key in the root (similar for the right subtree), in which case it is not a binary search tree. Note that if the left subtree is empty, the maximum key returned is $-\infty$ so the check will be true. Finally, if u is a BST, then we just return the range of keys of it with the special case taken care of. Thus, our routine for $R(u)$ is correct.

Question 6

(a)(b)

The problem comes from small trees. For example, if a binary search tree T rooted at u is of size 2. Then ratio(u)=1/2 if u.left is nil, and ratio(u)=2 if u.right is nil. This shows that if $\rho \in [\frac{1}{2}, 1]$, T can not have any subtree of size 2, which is a severe restriction, as it implies that T has to be a perfect BST. Let $\mu(h)$ be the size of the smallest $RB[\rho]$ tree of height h, where $\rho \in (\frac{1}{2}, 1)$. We claim that $\mu(h) = 2^{h+1} - 1$. Note that have $\mu(0) = 1$ and $\mu(1) = 3$. For $h \geq 2$, $\mu(h) \geq 1 + 2\mu(h - 1) = 2^{h+1} - 1$, which means that $\mu(h) = 2^{h+1} - 1$ as it is a binary tree.

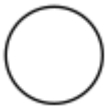
(c)

If we really want to allow values of ρ to be in $(\frac{1}{2}, 1)$, we may change the base cases of the induction in part (a), i.e. the problem comes form small trees. For example, we can change the definition of $esize(u)$ to be $2 + size(u)$ or 2 if u is nil to relax the restriction.

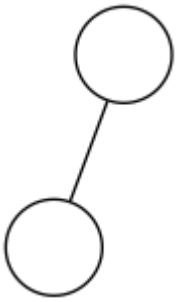
Question 7

(a)

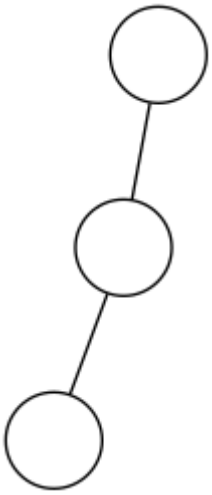
$h = 0$



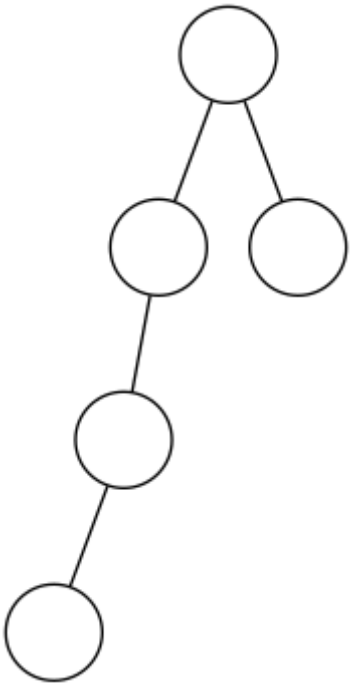
$h = 1$



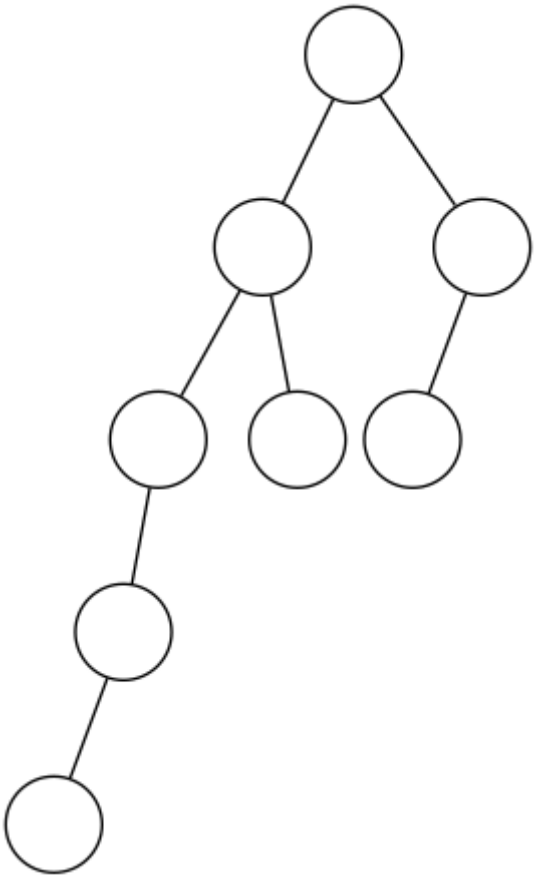
$h = 2$



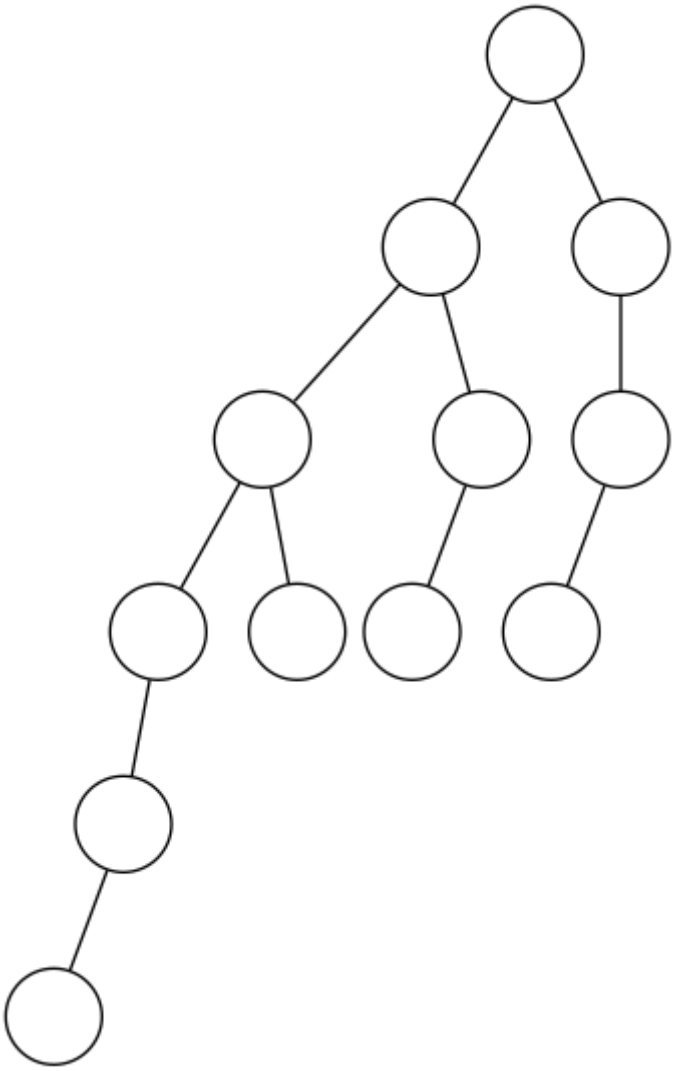
$h = 3$



$h = 4$



$h = 5$



(b)

Let $\mu(h)$ be the minimum number of nodes in any AVL[2] tree of height h. In general, $\mu(h)$ is seen to satisfy the recurrence

$$\mu(h) = 1 + \mu(h-1) + \mu(h-3)$$

Proposition: For $h \geq 0$ we have $\mu(h) \geq r^h$, where $r = \sqrt{2}$.

Proof. From part (a), we know $\mu(0) = 1 \geq r^0$, $\mu(1) = 2 > r^1$ and $\mu(2) = 3 > r^2$. For $h \geq 3$, we have

$$\mu(h) > \mu(h-1) + \mu(h-3) \geq r^{h-1} + r^{h-3} = \frac{1+r^2}{r^3} r^h = \frac{3}{2\sqrt{2}} r^h > r^h$$

.

Therefore, for an AVL[2] tree of n nodes and height h, we have $(\sqrt{2})^h < \mu(h) < n \Rightarrow \lg n > h \cdot \lg \sqrt{2} \Rightarrow h < 2 \lg n$.

(c)

UPDATE PHASE Use the standard BST insertion algorithm.

REBALANCE PHASE Go up the rebalance path and let u be the first unbalanced node we encounter. WLOG, suppose its left child is node v with height $h+2$ and its right child is node v' with height $h-1$. Let the current height of the children of v be h_L and h_R respectively. Then we have $\max(h_L, h_R) = h+1$ and $\min(h_L, h_R) = h-\delta$ where $\delta \in \{0, 1\}$.

CASE (I.a) $h_L = h+1$ and $h_R = h-\delta$. In this case, we rotate v, and the result would be balanced. Moreover, the height of the new root v is $h+2$, so this is a terminal case.

CASE (I.b) $h_L = h-\delta$ and $h_R = h+1$. In this case, let the right child of v to be w. The two children of w will have height $h-\delta'$ and $h-\delta''$ where $\delta', \delta'' \in \{0, 1, 2\}$ and $\delta'\delta'' = 0$. Now a double-rotation at w results in a balanced tree or height $h+2$ rooted at w. So this is also a terminal case.

Note that this algorithm is almost the same as the original insertion algorithm.

(d)

UPDATE PHASE Use the standard BST deletion algorithm.

REBALANCE PHASE We use the same set-up as the insertion algorithm, but now have three cases to consider:

CASE (D.a) $h_L = h+1$ and $h_R = h-\delta$. In this case, we still rotate v, but this becomes a non-terminal case.

CASE (D.b) $h_L = h-\delta$ and $h_R = h+1$. In this case, we still perform a double-rotation at w. Note that this is also a non-terminal case.

CASE (D.c) $h_L = h_R = h+1$. In this case, we rotate at v. This results in a balanced tree of height $h+3$ rooted at v. So this is a terminal case.

Note that this algorithm is almost the same as the original deletion algorithm.