1. What is the minimum number of nodes that a balanced tree of height 15 can have?

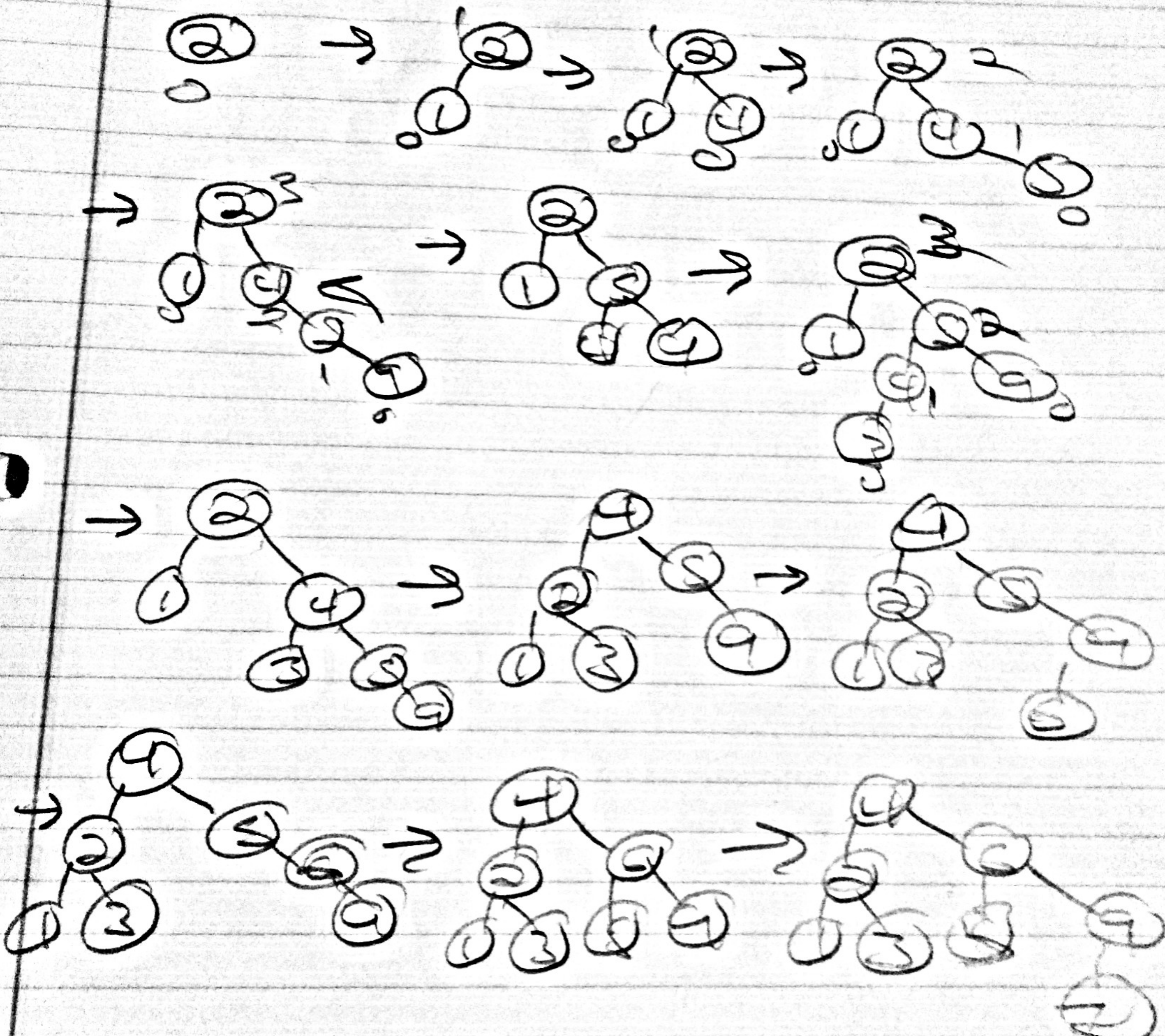
The calculation can be solved with the recurrence $\mathcal{N}(h) = \mathcal{N}(h-1)+\mathcal{N}(h-2)+1$. Which when solved for $\mathcal{N}(15)$ gives the value 2583.

2. Prove that if keys $1, 2, \ldots, 2^{k-1} - 1$ are inserted into an initially empty AVL tree, then the resulting tree is perfect.

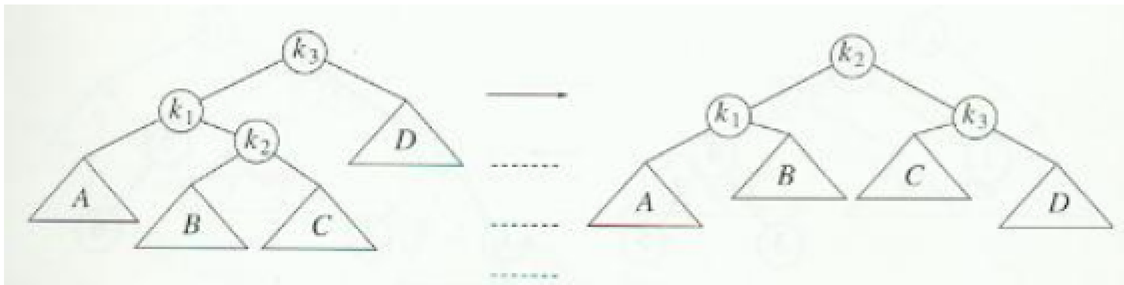
We prove by induction. $P(1)$ holds. Assume that the statement holds for all positive integers $i \leq k$ where k is also a positive integer. We show $P(k+1)$ is true. We insert the first $2^k - 1$ integers into the AVL tree. By the induction hypothesis this results in an perfect tree. The root is the 2^{k-1} element with the left and right subtree having $2^{k-1} - 1$ element each. We again add 2^{k-1} next elements to the right subtree resulting in $2^k - 1$ elements. Now the left subtree and right subtree differ by 1. Adding the next element would result in a right rotation resulting in the root now being the 2^k element and the left subtree contains $2^{k-1} + 2^{k-1} - 1 = 2^k - 1$ elements and is perfect, with the right subtree now containing 2^{k-1} nodes. We now add the remaining $2^{k-1} - 1$ keys. Ignoring the root and left subtree it is identical to the case where we have inserted the integers $2^k + 1, 2^k + 2, \ldots, 2^k + 2^{k-1} - 1$. By the induction hypothesis this result in the right subtree having $2^k - 1$ elements and is also perfect. Thus we have a perfect binary tree and $P(k+1)$ holds. By the principle of mathematical induction the result holds for all integers $n \geq 1$.

3. Insert 2,1,4,5,9,3,6,7 into an initially empty AVL tree. Redraw the tree each time a rotation is required.

2, 1, 4, 5, 9, 3, 6, 7



4. Consider the diagram on below that shows double rotation. List all of the pointers that need to be updated after the rotation. Provide the new value for each pointer.



$$
\begin{aligned}
k3.left &= k2.right \\
k2.right.parent &= k3 \\
k2.parent &= k3.parent \\
k3.parent &= k2 \\
k1.right &= k2.left \\
k2.left.parent &= k1 \\
k1.parent &= k2 \\
k2.left &= k1
\end{aligned}
$$