1

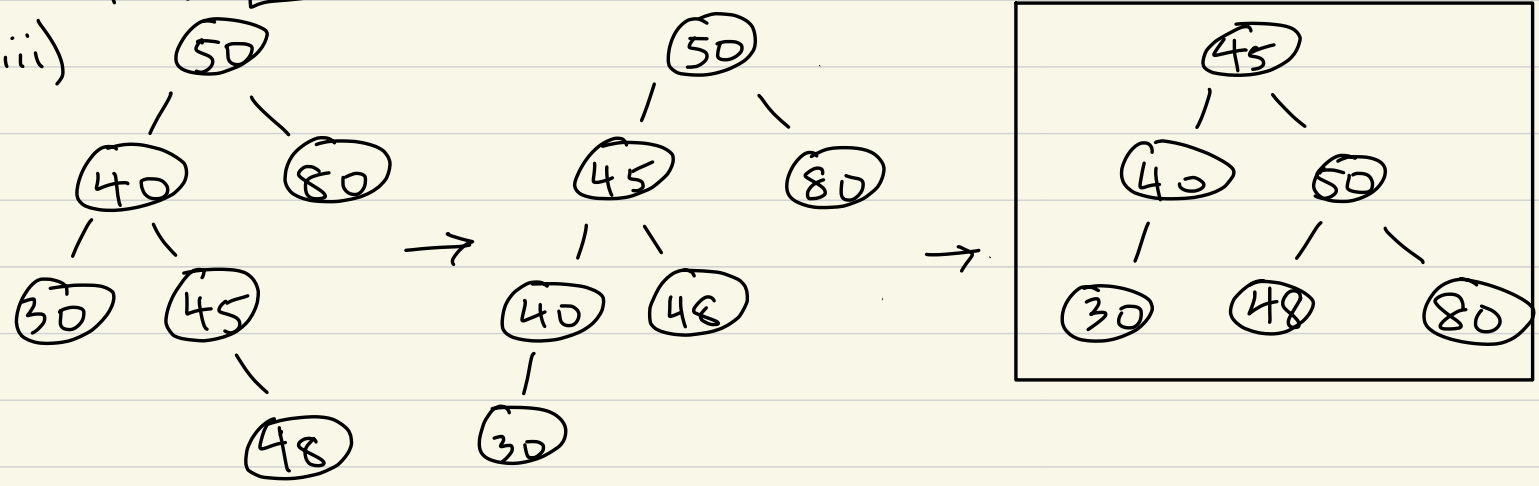
a) i) True

ii) False, the worst case for AVL Trees is $O(\log n)$ but for BST Trees, it is $O(n)$. While an unbalanced BST can have a worst-case deletion time of $O(n)$, an AVL tree always ensure $O(\log n)$ time complexity for the same operation.

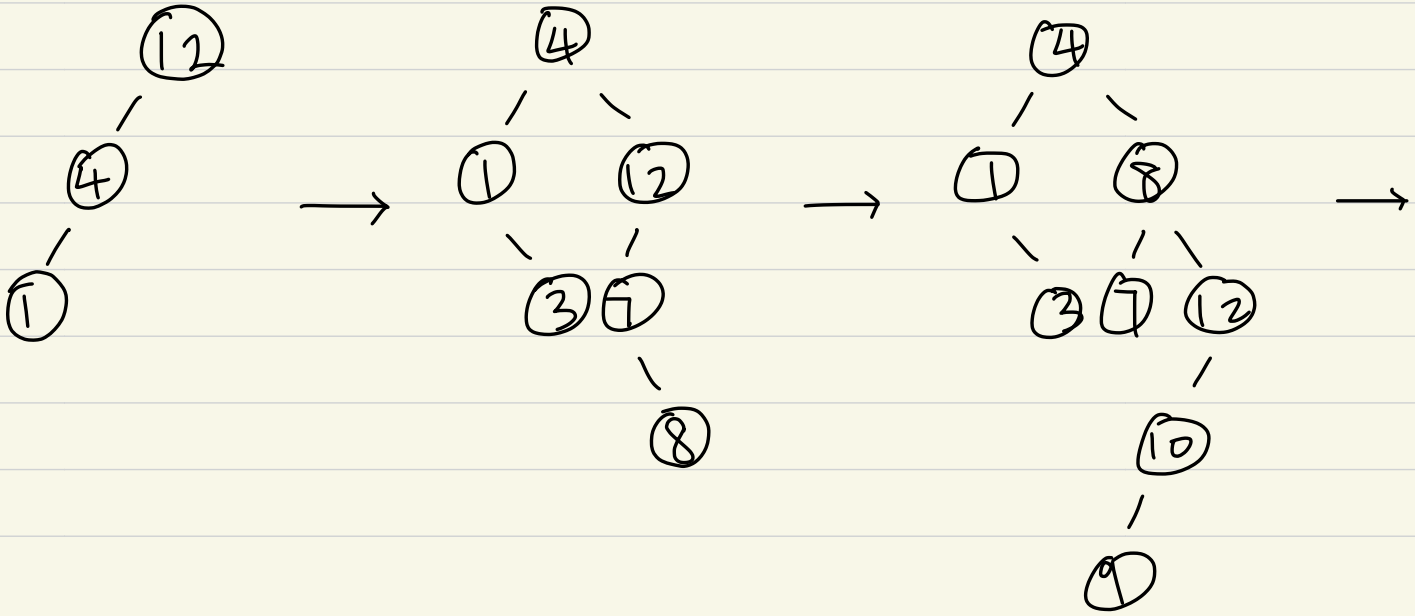
b) i) $3 - 1 = \boxed{2}$

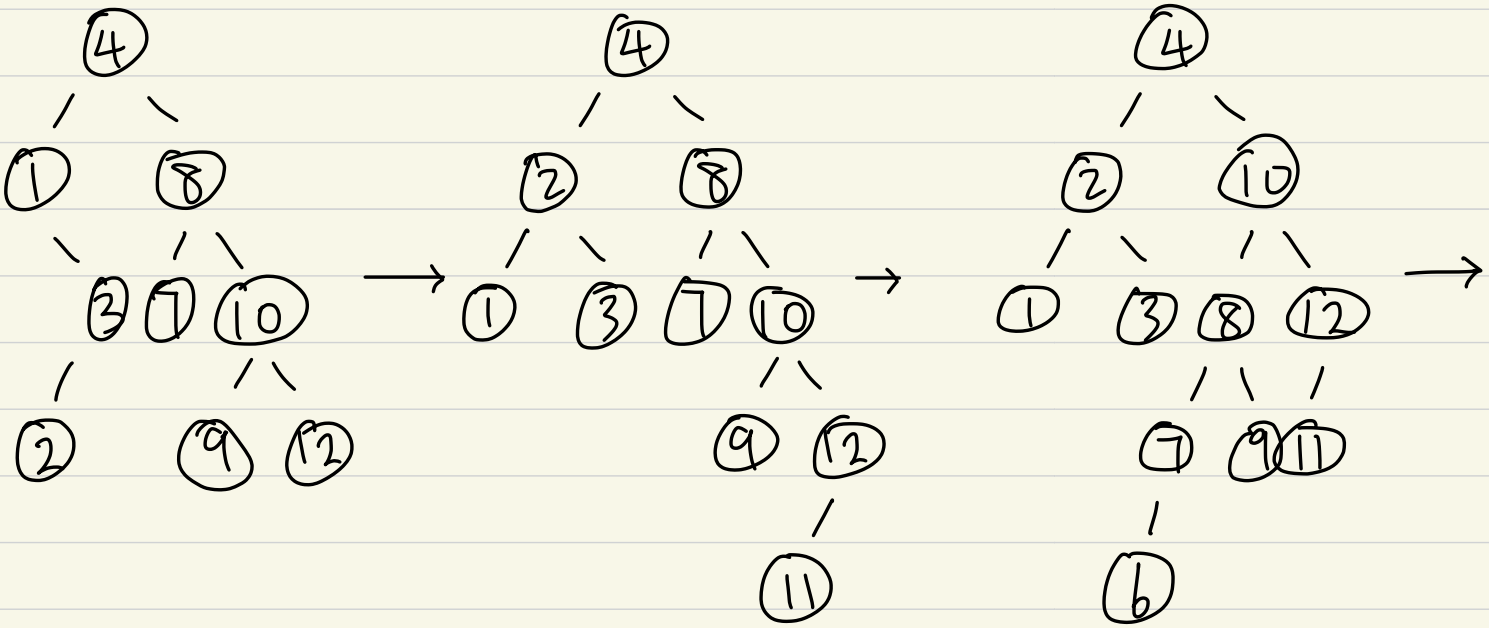
ii) $1 - 2 = \boxed{-1}$

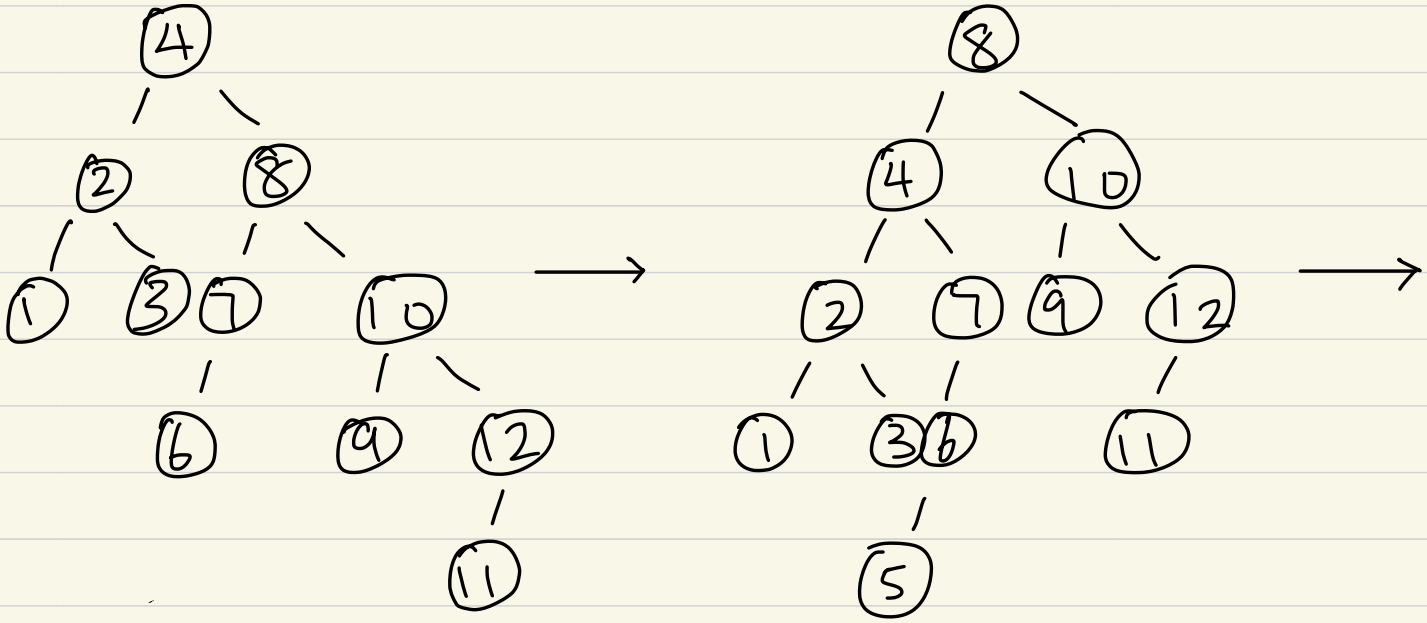
iii)

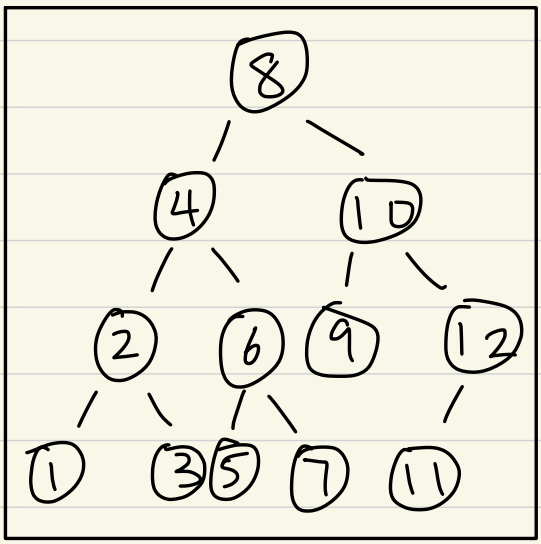


c) Insert 12, 4, 1, 3, 7, 8, 10, 9, 2, 11, 6, 5




4
1 8
3 7 10
2 9 12
4
2 8
1 3 7 10
9 12
11
4
2 10
1 3 8 12
7 9 11
6


4
2 8
1 3 7 10
6 9 12
11
8
4 10
2 7 9 12
1 3 6 11
5


8
4 10
2 6 9 12
1 3 5 7 11

2

a) i) False. Every node except root must contain at least t-1 keys and all nodes (including root) may contain at most 2t-1 keys.

ii) True.

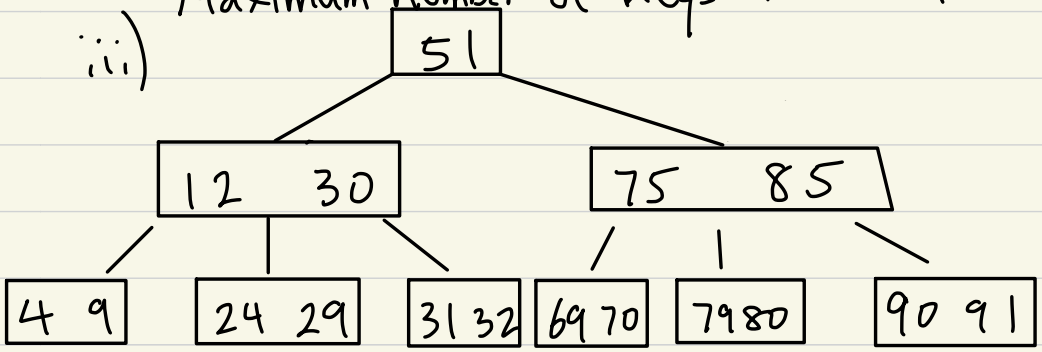
b) i) Minimum number of children: $\frac{m}{2} = 3$

Maximum number of children: $m = 5$

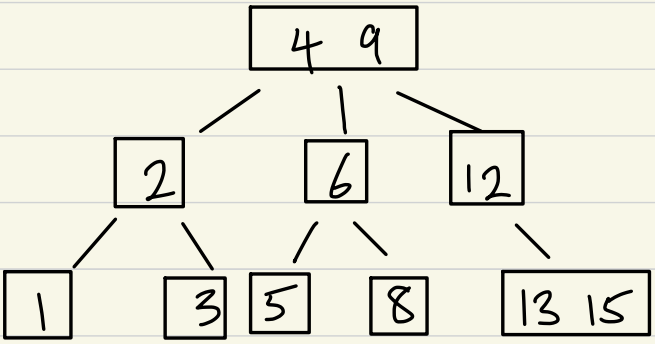
ii) Minimum number of keys: $\frac{m}{2} - 1 = 2$

Maximum number of keys: $m - 1 = 4$

iii)



c) i)



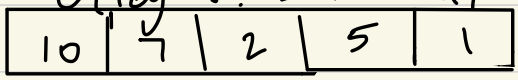
ii) $O(n \times \log n)$. To find the height of the tree, we need to traverse down the list $O(\log n)$. The time it takes to insert a key into a node and potentially split that node if it overflows. $O(n)$. Multiply those two and we get $O(n \log n)$

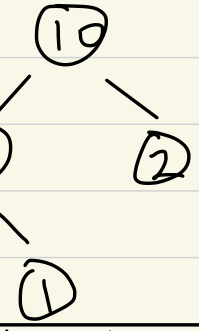
3

a)i) True

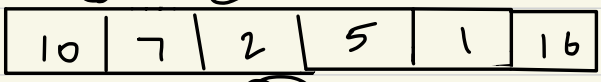
ii) False. $O(n \log n)$. Building heap = $O(n)$, Delete max operation = $O(\log n)$. In total, the complexity is $O(n \log n)$

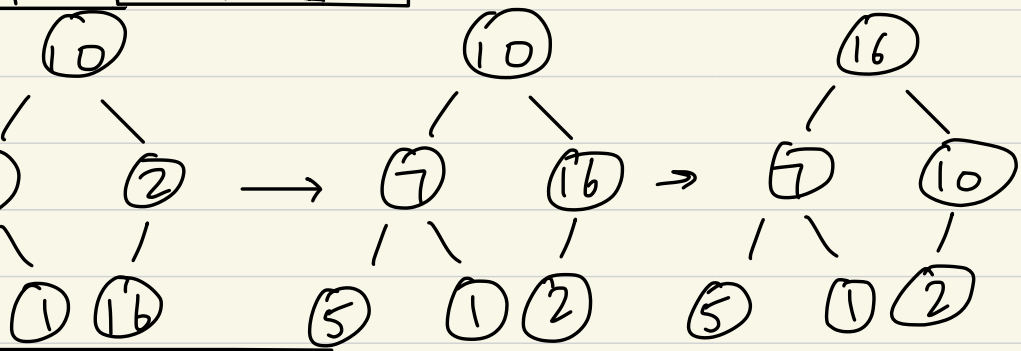
b) i)





ii)





iii)