2b)

**Time complexity:**

O ((n-1) ^ 3)

**Space complexity:**

O ((n-1) ^ 3\*m\*p)

**Worst-case:**

The number of nodes is n, so the total children node number of root is n-1. So the worst case is run maximum times, when the iterator method is called, its loop will run n-1 times and recursion method also run n-1 times. In addition, because the root has n-1 children nodes, and the n-1 nodes can concise with root, so the root will check the values of its children nodes are equal to the values of summary list, this loop also run n-1 times. Therefore, the time complexity is ((n-1) ^ 3) times, the space complexity is number of types (m) \* maximum length of string (p) \* times ((n-1) ^ 3).

3b)

**Time complexity:**

O ((n-1) ^ 2)

**Space complexity:**

O ((n-1) ^ 2\*m\*logn(base 2))

**Worst-case:**

The number of nodes is n, so total children node number of root is n-1. So the worst case is run maximum times, when the iterator2 method is called, its loop will run n-1 times and recursion method also run n-1 times. Therefore, the time complexity is ((n-1) ^ 2) times, the space complexity is number of types (m) \* maximum length of string (logn (base 2)) \* times ((n-1) ^ 2).

4b)

**Time complexity:**

O ((n-1) ^ 3)

**Space complexity:**

O ((n-1) ^ 3\*m\*logn(base 2))

**Worst-case:**

The number of nodes is n, so total children node number of node is n-1. So the worst case is run maximum times, when the iterator3 method is called, its loop will run n-1 times and recursion method also run n-1 times. In addition, because the root has n-1 children nodes, and the n-1 nodes can concise with root, so the root will check the values of its children nodes are equal to the values of summary list, this loop also run n-1 times. Therefore, the time complexity is ((n-1) ^ 3), the space complexity is number of types (m) \* maximum length of string (logn (base 2)) \* times ((n-1) ^ 3).