1. (8 points) The following pseudo-code shows a variation of the merge sort algorithm where in each iteration, the list is divided into three sublists.

```python
1: function Merge-Sort(A, p, r)
2: Return If p ≥ r
3: q1 ← ⌊(2 * p + r) / 3⌋
4: q2 ← ⌊(p + 2 * r) / 3⌋
5: Merge-Sort(A, p, q1)
6: Merge-Sort(A, q1 + 1, q2)
7: Merge-Sort(A, q2 + 1, r)
8: Merge(A, p, q1, q2, r)
9: end function
```

(a) Write down the recurrence relation that represents the running time of the above algorithm.

(b) Solve the recurrence relation using the expansion of the recurrence tree.

(c) Solve the recurrence relation using the Master method.

(d) Compare the running time of the proposed algorithm to that of the regular merge sort algorithm shown on page 34 of the textbook.

2. (4 points) Use the recurrence tree expansion method to find a tight asymptotic bound to the following recurrence relation. For simplicity, assume that \( n \) is always a power of two and \( T(1) = 1 \).

\[
T(n) = 2T(n/2) + n \lg n
\]

3. (4 points) Solve the following recurrences using the Master theorem.

(a) \( T(n) = 2T(n/8) + n \log n \)

(b) \( T(n) = 2T(n/4) + \sqrt{n} \)

(c) \( T(n) = 9T(n/3) + n \)