Divide and Conquer

1. Analyze the time complexity of the following algorithm.

**Algorithm** RecursiveMultiply(x, y)
**Input:** Two binary numbers x, y
**Output:** xy
1. Write $x = x_1.2^{n/2} + x_0$, $y = y_1.2^{n/2} + y_0$
2. Compute $x_0 + x_1$ and $y_0 + y_1$
3. $p \leftarrow$ RecursiveMultiply($x_0 + x_1, y_0 + y_1$)
4. $x_1y_1 \leftarrow$ RecursiveMultiply($x_1, y_1$)
5. $x_0y_0 \leftarrow$ RecursiveMultiply($x_0, y_0$)
6. return $x_1y_1.2^n + (p - x_1y_1 - x_0y_0).2^{n/2} + x_0y_0$

**Solution:** Suppose the running time of the algorithm is determined by $T(n)$. According to the algorithm we can write:

\[ T(n) = 3T(n/2) + n \]

Since $n^{\log_23} = O(n^3)$, $2(n/2)^3 \leq cn^3$, which holds for $c > 1/4$. Therefore, it is the third case of the master method which means that $T(n) = \theta(n^3)$.

2. Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

(a) $T(n) = 2T(n/2) + n^3$.

**Solution:** Since $n^{\log_22+\epsilon} = O(n^3)$, $2(n/2)^3 \leq cn^3$, which holds for $c > 1/4$. Therefore, it is the third case of the master method which means that $T(n) = \theta(n^3)$.

(b) $T(n) = 16T(n/4) + n^2$.

**Solution:** $n^2 = \theta(n^{\log_416} = n^2)$, which is the second case of the master method. Therefore, $T(n) = n^2 \lg n$. 

1
(c) \( T(n) = 7T(n/3) + n^2. \)

**Solution:** \( n^2 = \Omega(n^{\log_7 7 + \epsilon}), \) and \( 7(n/3)^2 \leq cn^2, \) which holds for \( c > 7/9. \) Therefore, it is the third case of the master method. \( T(n) = \theta(n^2). \)

---

(d) \( T(n) = 7T(n/2) + n^2. \)

**Solution:** \( n^2 = O(n^{\log_2 7 - \epsilon}), \) which is the first case of the master method. Therefore, \( T(n) = \theta(n \lg n). \)

---

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

**Solution:** \( \sqrt{n} = \theta(n^{\log_4 2}), \) which is the second case of the master method. Therefore, \( T(n) = \sqrt{n} \lg n. \)

---

(f) \( T(n) = T(n - 1) + n. \)

**Solution:** This can be solved using the substitution method as follows.

\[
T(n) = T(n - 1) + n = T(n - 2) + n - 1 + n = T(n - 3) + n - 2 + n - 1 + n = ... 
\]

\[
T(n) = \sum_{i=1}^{n} i = \theta(n^2). \]

This part should be proven by induction.

---

(g) \( T(n) = T(\sqrt{n}) + 1. \)

**Solution:** In this case we also use the substitution method. \( T(n) = T(\sqrt{n})+1 = (T(\sqrt{n})+1) + 1 = ... \) The substitution goes on up to the point that \( n^{1/2^i} = 2 \) with simple calculations we can achieve \( i = \lg \lg n. \) Therefore, using induction we can prove \( T(n) = \theta(\lg \lg n). \)

---

3. Show that \( \Theta(n \lg n) \) is the solution to the exact recurrence obtained from the merge sort algorithm.

**Algorithm** MergeSort(A, p, r)

**Input:** a list A, indices p and r

**Output:** sorted list A

1. if \( p < r \)
2. then \( q \leftarrow \lfloor (p + r)/2 \rfloor \)
3. MergeSort(A, p, q)
4. MergeSort(A, q + 1, r)
5. Merge(A, p, q, r)

**Solution:** Suppose \( T(n) \) is the running time of the merge sort algorithm. \( T(n) = 2T(n/2)+O(n), \) which is the second case of the master method. Therefore, \( T(n) = n \lg n. \)