CS3230 Tutorial 02: Recurrences and Master Theorem

Question 1. Give a tight asymptotic bound for $T(n) = 4\,T\left(\frac{n}{4}\right) + n\log n$ using telescoping.

Solution. could also use master theorem property 2.
Unroll the recurrence for i steps:

$$T(n) = 4^i T\left(\frac{n}{4^i}\right) + \sum_{j=0}^{i-1} 4^j \left(\frac{n}{4^j}\log\left(\frac{n}{4^j}\right)\right).$$

Simplify the summand:

$$4^j \cdot \frac{n}{4^j}\log\left(\frac{n}{4^j}\right) = n\left(\log n - \log(4^j)\right) = n(\log n - j\log 4).$$

Hence,

$$T(n) = 4^i T\left(\frac{n}{4^i}\right) + n\sum_{j=0}^{i-1}(\log n - j\log 4) = 4^i T\left(\frac{n}{4^i}\right) + n\left(i\log n - \log 4\sum_{j=0}^{i-1} j\right).$$

Use $\sum_{j=0}^{i-1} j = \frac{(i-1)i}{2}$:

$$T(n) = 4^i T\left(\frac{n}{4^i}\right) + n\left(i\log n - \log 4 \cdot \frac{(i-1)i}{2}\right).$$

Stop when $\frac{n}{4^i} = 1$, i.e. $i = \log_4 n$. Then $4^i = n$ and (assuming $T(1) = \Theta(1)$):

$$4^i T(1) = n \cdot \Theta(1) = \Theta(n).$$

Also, with $i = \Theta(\log n)$, the bracketed term is $\Theta((\log n)^2)$, so the sum contributes

$$n \cdot \Theta((\log n)^2) = \Theta(n\log^2 n).$$

Therefore,

$$T(n) = \Theta(n\log^2 n).$$

□

Question 2. Give a tight asymptotic bound for $T(n) = 5\,T\left(\frac{n}{3}\right) + n$.

Solution. Apply Master theorem with $a = 5$, $b = 3$, and $f(n) = n$. Compute

$$d = \log_b a = \log_3 5.$$

Then $n^d = n^{\log_3 5}$. Since $1 < \log_3 5$, we have

$$f(n) = n \in O\left(n^{d-\varepsilon}\right) \quad \text{for some } \varepsilon > 0.$$

This is Master theorem Case 1, hence

$$T(n) \in \Theta\left(n^{\log_3 5}\right).$$

Conclusion: Option (3) is correct. □

Question 3. Give a tight asymptotic bound for $T(n) = 9\,T\left(\frac{n}{3}\right) + n^3$.

Solution. Apply Master theorem with $a = 9$, $b = 3$, and $f(n) = n^3$. Compute

$$d = \log_3 9 = 2, \quad n^d = n^2.$$

Since $n^3 \in \Omega(n^{2+\varepsilon})$ with $\varepsilon = 1$, we are in Case 3, provided the regularity condition holds:

$$af\left(\frac{n}{b}\right) = 9\left(\frac{n}{3}\right)^3 = 9 \cdot \frac{n^3}{27} = \frac{1}{3}n^3 \le c\,n^3$$

for $c = \frac{1}{3} < 1$. Thus Case 3 applies and

$$T(n) \in \Theta(f(n)) = \Theta(n^3).$$

Conclusion: Option (4) is correct. □

Question 4. Give a tight asymptotic bound for $T(n) = 16\,T\left(\frac{n}{4}\right) + n^2 \log n$.

Solution. Apply Master theorem with $a = 16$, $b = 4$, and $f(n) = n^2 \log n$. Compute

$$d = \log_4 16 = 2, \quad n^d = n^2.$$

We have

$$f(n) = n^2 \log n \in \Theta(n^d \log^1 n),$$

so this is Master theorem Case 2 with $k = 1$, giving

$$T(n) \in \Theta(n^d \log^{k+1} n) = \Theta(n^2 \log^2 n).$$

Conclusion: Option (2) is correct. □

Question 5. Give a tight asymptotic bound for $T(n) = 4\,T\left(\frac{n}{2}\right) + \sqrt{n}$ using the substitution method.
Go to pdf page 86 `https://mcube.lab.nycu.edu.tw/~cfung/docs/books/cormen2001algorithms.pdf` to see full explaination.

Solution. We show $T(n) = \Theta(n^2)$.
Lower bound. Since $\sqrt{n} \ge 0$, we have $T(n) \ge 4T(n/2)$. Let $n = 2^m$. Then

$$T(2^m) \ge 4T(2^{m-1}) \ge \cdots \ge 4^m T(1) = (2^{2m})T(1) = n^2 T(1) = \Omega(n^2).$$

Upper bound. We prove by induction that for suitable constants $c > 0$ and $d > 0$,

$$T(n) \le cn^2 - d\sqrt{n} \quad \text{for all } n \ge 1.$$

Assume it holds for $n/2$. Then

$$\begin{aligned} T(n) &= 4T(n/2) + \sqrt{n} \\ &\le 4\left(c\left(\frac{n}{2}\right)^2 - d\sqrt{\frac{n}{2}}\right) + \sqrt{n} \\ &= cn^2 - 4d\frac{\sqrt{n}}{\sqrt{2}} + \sqrt{n} \\ &= cn^2 + \left(1 - 2\sqrt{2}\,d\right)\sqrt{n}. \end{aligned}$$

Choose $d \ge \frac{1}{2\sqrt{2}}$, so $1 - 2\sqrt{2}\,d \le 0$, hence $T(n) \le cn^2$ for all sufficiently large n, and we can adjust constants to satisfy the base case.
Thus $T(n) = O(n^2)$.
Combining both bounds, $T(n) = \Theta(n^2)$. □

Question 6. You are given k sorted arrays $\{A_1, \ldots, A_k\}$ with n elements each. Merge them into one sorted array of size kn using the described recursion. Give a formula for $T(k, n)$ and solve it.

Solution. When $k = 1$, no merging across arrays is needed, so take $T(1, n) = \Theta(n)$ (or $\Theta(1)$; this does not affect the final asymptotics).
For $k > 1$, the algorithm: (i) merges the first $\lceil k/2 \rceil$ arrays, (ii) merges the remaining $\lfloor k/2 \rfloor$ arrays, (iii) merges the two resulting sorted arrays.
After (i) we get a sorted array of size $\lceil k/2 \rceil n$. After (ii) we get a sorted array of size $\lfloor k/2 \rfloor n$. Merging them in (iii) takes time proportional to their total size:

$$\Theta(\lceil k/2 \rceil n + \lfloor k/2 \rfloor n) = \Theta(kn).$$

Hence the recurrence is

$$T(k, n) = T(\lceil k/2 \rceil, n) + T(\lfloor k/2 \rfloor, n) + \Theta(kn).$$

To solve it, view this as a merge-sort recursion on k items. At recursion level i, there are about 2^i subproblems, each with about $k/2^i$ arrays, so each node costs

$$\Theta\left(\frac{k}{2^i} n\right),$$

and the total cost per level is

$$2^i \cdot \Theta\left(\frac{k}{2^i} n\right) = \Theta(kn).$$

The depth is $\Theta(\log k)$, so summing over levels gives

$$T(k, n) = \Theta(kn \log k).$$

$\square$