RECURRENCES AND ASYMPTOTIC BOUNDS

Question 1. Solve the following recurrences (tightest bound possible). Assume $T(n) = \Theta(1)$ for $n \le n_0$.

(a) $T(n) = 95\,T(n/10) + 223n^2 - 16.$

Solution. Apply the Master Theorem with $a = 95$, $b = 10$, and

$$f(n) = 223n^2 - 16 = \Theta(n^2).$$

Compute

$$n^{\log_b a} = n^{\log_{10} 95}.$$

Since $95 < 100 = 10^2$, we have $\log_{10} 95 < 2$, hence

$$n^{\log_{10} 95} = o(n^2).$$

So $f(n) = \Omega\big(n^{\log_{10} 95+\varepsilon}\big)$ for some $\varepsilon > 0$ (e.g. $\varepsilon = 2 - \log_{10} 95$).
Check the regularity condition:

$$af(n/b) = 95\left(223\left(\frac{n}{10}\right)^2 - 16\right) = 95\left(\frac{223}{100}n^2 - 16\right) = \frac{95}{100}\cdot 223\,n^2 - 1520.$$

For sufficiently large n, this is at most $c \cdot 223n^2$ with $c = \frac{95}{100} < 1$, hence

$$af(n/b) \le cf(n) \quad \text{for large } n.$$

Therefore Master Theorem Case 3 applies, and

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

□

(b) $T(n) = 36\,T(n/6) + 6n^2 + 4.$

Solution. Apply the Master Theorem with $a = 36$, $b = 6$, and

$$f(n) = 6n^2 + 4 = \Theta(n^2).$$

Compute

$$n^{\log_b a} = n^{\log_6 36} = n^2.$$

Thus $f(n) = \Theta\big(n^{\log_b a}\big)$, i.e. Master Theorem Case 2 with $k = 0$. Hence

$$T(n) = \Theta\big(n^{\log_b a}\log n\big) = \Theta(n^2 \log n).$$

□

(c) $T(n) = T(n/2) + T(\sqrt{n}) + 2n.$

Solution. **Lower bound.** Since $T(\sqrt{n}) \ge 0$ and $T(n/2) \ge 0$,

$$T(n) = T(n/2) + T(\sqrt{n}) + 2n \ge 2n,$$

so $T(n) = \Omega(n)$.
Upper bound (substitution). We prove $T(n) = O(n)$ by showing there exists a constant c such that

$$T(n) \le cn \quad \text{for all } n \ge n_0,$$

for a suitable constant n_0.
Assume inductively that $T(m) \le cm$ for all $m < n$. Then for $n > n_0$,

$$\begin{aligned} T(n) &= T(n/2) + T(\sqrt{n}) + 2n \\ &\le c\cdot\frac{n}{2} + c\sqrt{n} + 2n. \end{aligned}$$

For $n \geq 16$, we have $\sqrt{n} \leq \frac{n}{4}$, hence

$$T(n) \leq c \cdot \frac{n}{2} + c \cdot \frac{n}{4} + 2n = \left(\frac{3c}{4} + 2\right) n.$$

Choose $c \geq 8$, so that $\frac{3c}{4} + 2 \leq c$ (equivalently $2 \leq \frac{c}{4}$). Then $T(n) \leq cn$ holds for all $n \geq 16$, and we can enlarge c if needed to cover the constant-size base cases $n \leq 16$.
Therefore $T(n) = O(n)$.
Conclusion. Combining $\Omega(n)$ and $O(n)$ gives

$$T(n) = \Theta(n).$$

□

Question 2. For $n \geq 1$, let $H(1) = 2$, $H(n+1) = H(n) + 2$. Let $T(1) = 1$, $T(n+1) = T(n) + 1 + H(n)$. Give closed-form formulas for $H(n)$ and $T(n)$.

Solution. **For $H(n)$.** This is an arithmetic progression:

$$H(n) = H(1) + 2(n-1) = 2 + 2(n-1) = 2n.$$

For $T(n)$. Using $H(n) = 2n$,

$$T(n+1) - T(n) = 1 + H(n) = 1 + 2n.$$

Sum from $n = 1$ to $n = N - 1$:

$$T(N) - T(1) = \sum_{n=1}^{N-1} (1 + 2n) = (N-1) + 2 \cdot \frac{(N-1)N}{2} = (N-1) + N(N-1).$$

Since $T(1) = 1$,

$$T(N) = 1 + (N-1) + N(N-1) = N^2.$$

Thus, for all $n \geq 1$,

$$H(n) = 2n, \qquad T(n) = n^2.$$

□

Question 3. Euclid's algorithm (assume $m \leq n$):

$$\texttt{GCD}(m, n): \text{ if } m = 0 \text{ return } n; \text{ else return } \texttt{GCD}(n \bmod m,\, m).$$

Assume computing $n \bmod m$ takes constant time. Give the tightest upper bound possible in terms of n (the larger input).

Solution. Each recursive call does $O(1)$ work, so the running time is $\Theta(\#\text{calls})$.
Let the state be (a, b) with $0 < a \leq b$, and the next state be $(b \bmod a,\, a)$. We show that within at most *two* recursive calls, the larger argument decreases by at least a factor of 2.
Case 1: $b \geq 2a$. Then $b \bmod a < a \leq b/2$. After one call, the new pair is $(b \bmod a,\, a)$ and its maximum is $a \leq b/2$. So the maximum halves in one step.
Case 2: $a < b < 2a$. Then $b \bmod a = b - a$ and since $a > b/2$, we have

$$b - a < b - \frac{b}{2} = \frac{b}{2}.$$

After one call, we are at $(b - a,\, a)$. After the second call, the second argument becomes $b - a$, so the new maximum is at most $b - a < b/2$. Thus the maximum halves within two steps.
Therefore, every two recursive calls reduce the larger input by at least a factor of 2. Starting from at most n, after $2k$ calls the maximum is at most $n/2^k$. Once $n/2^k$ is $O(1)$, the recursion terminates, which happens for $k = \Theta(\log n)$.
Hence the number of calls is $O(\log n)$, and the running time is

$$T(n) = O(\log n).$$

This is also tight: the worst case occurs on consecutive Fibonacci numbers, giving $\Theta(\log n)$ calls. So the tight bound is

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

□

Question 3 (Alternative Solution via Fibonacci + Induction). Euclid's algorithm:

$$\texttt{GCD}(m, n): \text{ if } m = 0 \text{ return } n; \text{ else return } \texttt{GCD}(n \bmod m,\ m),$$

assume $0 < m \leq n$, and computing $n \bmod m$ is $O(1)$.

Solution. Let $C(m, n)$ be the number of recursive calls made by Euclid's algorithm on input (m, n) (including the current call). We prove a tight bound $C(m, n) = \Theta(\log n)$ using Fibonacci numbers.

Claim. If Euclid's algorithm on (m, n) makes at least t recursive calls, then

$$n \ \geq \ F_{t+1}.$$

Equivalently,

$$C(m, n) \leq t \implies n < F_{t+1}.$$

This implies $C(m, n) = O(\log n)$ since Fibonacci grows exponentially.

Proof of the Claim (by induction on t).
Base cases. For $t = 1$, the algorithm makes 1 call only in the trivial sense; then $n \geq 1 = F_2$, so $n \geq F_{t+1}$ holds. For $t = 2$, at least two calls means $m > 0$, hence $n \geq m \geq 1 = F_3$ (since $F_3 = 2$ in the convention $F_1 = 1, F_2 = 1$; if you use $F_1 = 1, F_2 = 2$ the constants shift—either way the asymptotics are unchanged). We can fix a consistent convention below.

Convention. Let $F_1 = 1$, $F_2 = 1$, and $F_{k+2} = F_{k+1} + F_k$ for $k \geq 1$.

Inductive step. Assume the claim holds for all smaller values up to $t - 1$ (where $t \geq 3$). Suppose the algorithm on (m, n) makes at least t calls. After one call, it recurses on

$$(m', n') = (n \bmod m,\ m),$$

and this recursive subcall makes at least $t - 1$ calls. By the induction hypothesis applied to (m', n'), we have

$$n' = m \ \geq \ F_t.$$

Also, since $n = qm + m'$ with $q \geq 1$ and $m' = n \bmod m$, we get

$$n \geq m + m'.$$

Now consider the second recursive step from (m', m) to (m'', m') (if $m' > 0$; otherwise the recursion ends earlier and t would not be this large). That means the call on (m', m) makes at least $t - 1$ calls, so the subcall on (m'', m') makes at least $t - 2$ calls. Applying the induction hypothesis again gives

$$m' \geq F_{t-1}.$$

Therefore,

$$n \geq m + m' \geq F_t + F_{t-1} = F_{t+1},$$

which completes the induction.
Thus the claim holds for all t.

Consequence: $O(\log n)$. If $C(m, n) = t$, then by the claim $n \geq F_{t+1}$. Using the standard bound $F_{t+1} \geq \varphi^{t-1}$ for $t \geq 2$ (where $\varphi = \frac{1+\sqrt{5}}{2}$), we get

$$n \geq \varphi^{t-1} \quad \Rightarrow \quad t - 1 \leq \log_\varphi n \quad \Rightarrow \quad t = O(\log n).$$

Tightness: $\Omega(\log n)$. Take inputs (F_k, F_{k+1}). directly use fibonacci series as input. Then (since all quotients are 1) Euclid's algorithm performs $\Theta(k)$ calls. But $F_{k+1} \leq \varphi^{k+1}$ implies $k = \Omega(\log F_{k+1}) = \Omega(\log n)$. Hence there exist inputs of size n requiring $\Omega(\log n)$ calls.

Conclusion. Combining the upper and lower bounds,

$$C(m, n) = \Theta(\log n),$$

and since each call costs $O(1)$ under the problem's assumption, the running time is

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

$\square$