Problem (Analyzing nested loops)

Derive a $\Theta$-bound on the running time of the following procedure as a function of $n$.

\begin{algorithm}
\textbf{function} \text{F}(n) \begin{array}{l}
\text{array} \ A[1:n, 1:n] \\
\text{\(\Theta(n)\)} \{ \\
\text{for} \ i := 1 \text{ to } n \text{ do} \\
\quad \ A[i,i] := 0 \\
\text{for} \ l := 2 \text{ to } n \text{ do} \\
\quad \ \text{\(\Theta(l)\)} \{ \\
\quad \ \text{for} \ i := 1 \text{ to } n - l + 1 \text{ do} \begin{array}{l}
\quad \ j := i + l - 1 \\
\quad \ A[i,j] := \infty \\
\quad \ \text{\(\Theta(l)\)} \{ \\
\quad \quad \ \text{for} \ k := i \text{ to } j - 1 \text{ do} \\
\quad \quad \quad \ A[i,j] := \min \{ A[i,j], \\
\quad \quad \quad \ \ A[i,k] + A[k+1,j] + ijk \} \\
\quad \end{array} \\
\} \\
\} \\
\text{return} \ A[1,n] \\
\end{array}
\end{algorithm}

Solution We analyze the nested loops inside-out:

\[(a) \ \sum_{i \leq k \leq j-1} \Theta(i) = \Theta \left( \sum_{i \leq k \leq j-1} 1 \right) = \Theta(j-i).\]

\[(b) \ \sum_{1 \leq i \leq n-l+1} \left( \Theta(j-i) + \Theta(1) \right) = \sum_{1 \leq i \leq n-l+1} \Theta(j-i+1).\]
Solution cont'd

(b) cont'd

\[ \sum_{1 \leq i \leq n-l+1} \theta(j-i+1) = \sum_{1 \leq i \leq n-l+1} \theta((i+l-1)-i+1) \]

as \( j = i+l-1 \)

\[ = \sum_{1 \leq i \leq n+l-1} \theta(l) = \theta\left( \sum_{1 \leq i \leq n+l-1} l \right) = \theta(l(n-l+1)) \]

(c) \( \sum_{2 \leq l \leq n} \theta(l(n-l+1)) \)

\[ = \theta\left( \sum_{2 \leq l \leq n} l(n-l+1) \right) \]

\[ = \theta\left( \sum_{1 \leq l \leq n} l(n-l+1) - n \right) \]

\[ = \theta\left( \sum_{1 \leq l \leq n} (n+1)l - l^2 \right) - n \]

\[ = \theta\left( (n+1) \sum_{1 \leq l \leq n} l - \sum_{1 \leq l \leq n} l^2 - n \right) \]

\[ = \theta\left( (n+1) \left( \frac{1}{2} n^2 + \theta(n) \right) - \left( \frac{1}{3} n^3 + \theta(n^2) \right) - n \right) \]

\[ = \theta\left( \frac{1}{2} n^3 + \theta(n^2) - \frac{1}{3} n^3 - \theta(n^2) - n \right) \]

\[ = \theta\left( \frac{1}{6} n^3 + \theta(n^2) - \theta(n^2) \right) \]

\[ = \theta\left( \theta(n^3) + o(n^2) \right) \]

\[ = \theta(n^3). \]

So the entire procedure takes time \( \theta(n^3) + \theta(n) = \theta(n^3) \).
Problem: (Hybrid insertion-merge sort)

Analyze a hybrid sorting algorithm that, for a given parameter $k$, divides an input of length $n$ into $\left\lceil \frac{n}{k} \right\rceil$ lists of length $\leq k$, sorts each list using insertion sort, and merges them into one sorted list.

Solution

(a) Insertion sort on a list of length $k$ takes $\Theta(k^2)$ worst-case time. So sorting the $\left\lceil \frac{n}{k} \right\rceil$ lists takes worst-case time

$$\left\lceil \frac{n}{k} \right\rceil \Theta(k^2) = \left( \frac{n}{k} + \Theta(1) \right) \Theta(k^2)$$

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

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

$$= \Theta \left( nk \right).$$

(b) To efficiently merge all lists, we perform the merging in rounds. Each round pairs up the remaining lists and merges each pair. A round takes $\Theta(n)$ time to merge all pairs. Each round halves the number of remaining lists, so the total number of rounds is $\left\lceil \log \left( \frac{n}{k} \right) \right\rceil$. Thus the total time for merging is

$$\left\lceil \log \left( \frac{n}{k} \right) \right\rceil \Theta(n) = \Theta \left( \log \frac{n}{k} \right) \Theta(n)$$

$$= \Theta \left( n \log \frac{n}{k} \right).$$
Solution cont'd

(c) By Parts (a) and (b) the hybrid sorting algorithm runs in worst-case time

$$\Theta nk + n \log \frac{n}{k}$$  \hspace{1cm} (1)

We claim that the fastest rate of growth of \( k = k(n) \) for which expression (1) is \( \Omega(n \log n) \) is

$$k = \Theta(\log n).$$  \hspace{1cm} (2)

To see this, note that if we use expression (2) in expression (1), we get

$$\Theta(n \Theta(\log n) + n \log \frac{n}{\Theta(\log k)})$$

$$= \Theta(n \Theta(\log n) + n \log n - n \log \Theta(\log u))$$

$$= \Theta(n \Theta(\log n) - \Theta(n \log \log u))$$

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

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

Furthermore, if \( k = w(\log n) \), then the first term of expression (1) is \( w(n \log n) \), so the whole expression is not \( \Theta(n \log n) \).
Problem (Ordering functions asymptotically)

The ranking is

![Diagram showing the ranking of functions](image)

Notes:

1. \( n! = \sigma(2^{2^n}) \)
   - \( n! = \log_2 n! = \log_2 \Theta(n \log n) \) by class handout;
   - \( 2^{2^n} = \Theta(2^n) \);
   - Exponential Property with \( \Theta(n \log n) = \sigma(2^n) \).

2. \( 4^n = \sigma(n!) \)
   - \( 4^n = (\log_2 4)^n = \log_2 \Theta(n) \);
   - \( n! = \log_2 n! = \log_2 \Theta(n \log n) \) by class handout;
   - Exponential Property with \( \Theta(n) = \sigma(n \log n) \).
3. \( z^n = o(4^n) \)

\[
\lim_{n \to \infty} \frac{z^n}{4^n} = \lim_{n \to \infty} \left(\frac{z}{4}\right)^n = \lim_{n \to \infty} \frac{1}{z^n} = 0
\]

4. \( (\ln n)^{\ln n} = o(z^n) \)

\[
(\ln n)^{\ln n} = \left(2^{\ln \ln n}\right)^{\ln n} = 2^{\ln \ln n \cdot \ln n} = 2^{\ln \ln n^2} = z^n = z^{\Theta(n)}
\]

Exponential Property with \( \Theta(\log n \log \log n) = o(n) \).

5. \( (\ln n)^{\ln n} = \Theta(n^{\ln \ln n}) \)

\[
(\ln n)^{\ln n} = \left(e^{\ln \ln n}\right)^{\ln n} = (e^{\ln n})^{\ln \ln n} = n^{\ln \ln n}
\]

6. \( \ln n! = o(n^{\ln \ln n}) \)

\[
\ln n! = \Theta \left(\frac{n^{\frac{1}{2}}}{e^n} n^n\right) \text{ by Stirling's approximation}
\]

\[= o(n^n) \text{ by Polynomials vs. Exponentials;}
\]

\[\ln n! = o((\ln n)^{\ln n}) \text{ by above and}
\]

Composition Property with \( h(n) = \ln n \)

\[= o(n^{\ln \ln n}) \text{ by Note 5.}\]
Problem cont’d

7. \( n^2 = \omega \left( \ln n \right) ! \)

- \( n! = z \log n! = z \Theta(n \log n) \) by class handout;
- \( e^{2n} = (2 \log e)^{2n} = 2 \Theta(n) \);
- \((e^n)^2 = e^{2n} = \omega(\ln n!)\) by above and
  Exponential Property with \( \Theta(n) = \omega(n \log n) \);
- \( n^2 = \omega(\ln n!) \) by above and
  Composition Property with \( h(n) = \ln n \).

8. \( n \ln n = \Theta(n^2) \)

- \( \lim_{n \to \infty} \frac{n \ln n}{n^2} = \lim_{n \to \infty} \frac{\ln n}{n} = 0 \) by L’Hospital’s Rule.

9. \( n \ln n = \Theta(\ln(n!)) \)

- \( \ln(n!) = \Theta\left( \log \left( n^{\frac{1}{2}} \left( \frac{n}{e} \right)^n \right) \right) \) by Stirling’s Approximation and Logarithm Property
  \( = \Theta(n \log n) \).

10. \( n = \sigma(\ln(n!)) \)

- \( \ln(n!) = \Theta(n \log n) \) by Note 9;
- \( n = \omega(n \log n) \) by Limit Theorem.

11. \( 2 \ln n = \omega(n) \)

- \( 2 \ln n = n \frac{n^2}{e} = n^{1-\varepsilon} \) where \( \varepsilon > 0 \);
- \( n^{1-\varepsilon} = \omega(n) \) by Limit Theorem.
Problem cont'd

12. \( \ln^2 n = o(2^{\ln n}) \)
   \[ \ln^2 n = (\ln n)^2 = (2 \lg \ln n)^2 = 2 \Theta(\log \log n) \]
   - Exponential Property with \( \Theta(\log \log n) = o(\ln n) \).

13. \( \sqrt[3]{\ln n} = o(\ln^2 n) \)
   \[ \sqrt[3]{n} = o(n^2) \text{ by Limit Theorem} \]
   - Composition Property with \( h(n) = \ln n \).

14. \( \ln \ln n = o(\sqrt[3]{\ln n}) \)
   \[ n = o\left(\left(\frac{n}{\sqrt{e}}\right)^n\right) \text{ by Polynomials vs. Exponentials} \]
   - Composition Property with \( h(n) = \ln \ln n \).
Exercise

Show \( \max \{ f(n), g(n) \} = \Theta(f(n) + g(n)) \)
when \( f \) and \( g \) are asymptotically positive functions.

We must find three positive constants \( a, b, m \) such that
\[
a(f(n) + g(n)) \leq \max \{ f(n), g(n) \} \leq b(f(n) + g(n))
\]
\( \forall n \geq m. \)

Since \( f(n) \) and \( g(n) \) are asymptotically positive, there are positive constants \( m_f, m_g \) such that
\[
f(n) > 0 \quad \forall n \geq m_f, \quad \text{and} \quad g(n) > 0 \quad \forall n \geq m_g.
\]

Thus
\[
f(n) + g(n) \geq f(n) \quad \forall n \geq m_g, \quad \text{and} \quad f(n) + g(n) \geq g(n) \quad \forall n \geq m_f.
\]

So
\[
f(n) + g(n) \geq \max \{ f(n), g(n) \} \quad \forall n \geq \max \{ m_f, m_g \}.
\]

By the definition of \( \max \{ \cdot, \cdot \} \),
\[
f(n) \leq \max \{ f(n), g(n) \}, \quad \text{and} \quad g(n) \leq \max \{ f(n), g(n) \}.
\]

Adding these inequalities,
\[
f(n) + g(n) \leq 2 \max \{ f(n), g(n) \}.
\]

So
\[
\frac{1}{2}(f(n) + g(n)) \leq \max \{ f(n), g(n) \}.
\]

Combining (1) and (2),
\[
\frac{1}{2}(f(n) + g(n)) \leq \max \{ f(n), g(n) \} \leq \frac{1}{b}(f(n) + g(n)) \quad \forall n \geq \max \{ m_f, m_g \}.
\]
Problem (Gaps between orders of growth)

Proposition For all constants a and b > 1,

\[ n^{\log n} = \omega(n^a), \]

and

\[ n^{\log n} = \omega(b^n). \]

Proof Note that

\[ n^a = (2^{\log n})^a = 2^{\Theta(\log n)}, \]

\[ n^{\log n} = (2^{\log n})^{\log n} = 2^{\Theta(\log^2 n)}, \]

\[ b^n = (2^{\log b})^n = 2^{\Theta(n)}. \]

Since

\[ \Theta(\log n) = o(\log^2 n) \quad \text{(by } n = o(n^2) \text{ and the Composition Property using } \log n \text{)} \]

and

\[ \Theta(\log^2 n) = o(n) \quad \text{(by } n^2 = o(2^n) \text{ and the Composition Property using } \log n \text{)} \]

the proposition follows (by the Exponential Property).
Problem (Reducing $k^{th}$-smallest to median-finding)

Given an algorithm for finding the median of an $n$-element set in $\Theta(n)$ time, show how to find the $k^{th}$-smallest element for an arbitrary $k$ in $\Theta(n)$ time.

Solution

Do the following, where the set is stored in an array $A[1:n]$.

1. Find the median of $A[1:n]$, which is the $\lceil (n+1)/2 \rceil^{th}$-smallest element; call it $x$.


3. Let $i := \lceil (n+1)/2 \rceil$.
   - If $k = i$, return $x$;
   - else if $k < i$, recursively find the $k^{th}$-smallest element in $A[1:i]$;
   - else recursively find the $(k-i)^{th}$-smallest element in $A[i+1:n]$.

This takes time

$$T(n) = T(\frac{n}{2}) + \Theta(n),$$

which has the solution $T(n) = \Theta(n)$. \qed