Weekplan: Analysis of Algorithms
The 02105+02326 DTU Algorithms Team

Reading


Exercises

1  **Asymptotic Growth**  Arrange the following functions in increasing asymptotic order, i.e., if \( f(n) \) precedes \( g(n) \) then \( f(n) = O(g(n)) \).

\[
\begin{align*}
n \log n & \quad n^2 & \quad 2^n & \quad n^3 & \quad \sqrt{n} & \quad n \\
\end{align*}
\]

2  **\( \Theta \)-notation**  Write the following expressions using \( \Theta \)-notation.

\[
\begin{align*}
n^2 + n^3/2 & \quad 2^n + n^4 & \quad 8 \log_2^n n + 34 \log_2 n + \frac{1}{1000} n \\
\log_2 n + n \sqrt{n} & \quad n(n-6) & \quad 2^n7 + 5 \log_3^n n \\
4 \sqrt{n} & \quad n(n^2 - 18) \log_2 n & \quad n \log_4^n n + n^2 \\
\end{align*}
\]

3  **Loopy Loops**  Analyze the running time of the following loops as a function of \( n \) and express the result in \( \Theta \)-notation.

\[
\begin{align*}
\text{LOOP1}(n) & \quad \text{LOOP2}(n) & \quad \text{LOOP3}(n) \\
\text{for } i = 1 \text{ to } n \text{ do } & \quad \text{for } i = 1 \text{ to } n \text{ do } & \quad \text{for } i = 1 \text{ to } n \text{ do } \\
\text{while } i \leq n \text{ do } & \quad \text{while } i \leq n \text{ do } & \quad \text{for } i = 1 \text{ to } n \text{ do } \\
\text{print "*"} & \quad \text{print "*"} & \quad \text{for } j = 1 \text{ to } n \text{ do } \\
i = 2 \cdot i & \quad i = 5 \cdot i & \quad \text{while } j \leq n \text{ do } \\
\text{end while} & \quad \text{print "*"} & \quad \text{end while} \\
\end{align*}
\]

4  **Asymptotic Statements**  Which of the following statements are true?

\[
\begin{align*}
\frac{1}{20} n^2 + 100n^3 & = O(n^2) & \frac{n^3}{1000} + n + 100 & = \Omega(n^2) \\
\log_2 n + n & = O(n) & 2^n + n^2 & = \Omega(n) \\
2^{\log_2 n} & = O(n) & \log_4 n + \log_{16} n & = \Theta(\log n) \\
n^3(n-1)/5 & = \Theta(n^3) & n^{1/4} + n^2 & = \Theta(n) \\
\log_2 n + n & = \Theta(n) & 2^{\log_4 n} & = \Theta(\sqrt{n})
\end{align*}
\]
5 Doubling Hypothesis  Solve the following exercises.

5.1 [w] The algorithm $A$ runs in exactly $7n^3$ time on an input of size $n$. How much slower does it run if the input size is doubled?

5.2 [BEng] The algorithm $B$ runs in time respectively 5, 20, 45, 80 and 125 seconds on input of sizes 1000, 2000, 3000, 4000 and 5000. Estimate how long the running time will be of $B$ on an input of size 6000. What is the running time of $B$ expressed using $\Theta$-notation?

5.3 The algorithm $C$ runs 3 seconds slower each time the size of the input is doubled. What is the running time of $C$ expressed using $\Theta$-notation?

6 Asymptotic Properties  Solve the following exercises.

6.1 CLRS 3.1-1

6.2 CLRS 3.1-3

6.3 CLRS 3.1-4

6.4 [BSc∗] Show that $\log_2(n!) = \Theta(n \log n)$. Hint: Start by showing the upper bound.

6.5 [BSc∗] CLRS 3-2

7 Generalized Merge Sort  Professor Gørtz suggests the following variant of merge sort called 3-merge sort. 3-merge sort works exactly like normal merge sort except one splits the array into 3 parts instead of 2 that are then recursively sorted and merged. Solve the following exercises.

7.1 Show it is possible to merge 3 sorted arrays in linear time.

7.2 Analyze the running time of 3-merge sort.

7.3 [∗] Generalize the algorithm and the analysis of 3-merge sort to $k$-merge sort for $k > 3$. Is $k$-merge sort an improvement over the standard 2-merge sort?

8 Maximal Subarray  Let $A[0..n-1]$ be an array of integers (both positive and negative). A maximal subarray of $A$ is a subarray $A[i..j]$ such that the sum $A[i] + A[i+1] + \cdots + A[j]$ is maximal among all subarrays of $A$. Solve the following exercises.

8.1 [w] Give an algorithm that finds a maximal subarray of $A$ in $O(n^3)$ time.

8.2 Give an algorithm that finds a maximal subarray of $A$ in $O(n^2)$ time. Hint: Show it is possible to compute the sum of any subarray in $O(1)$ time.

8.3 [∗∗] Give a divide and conquer algorithm that finds a maximal subarray of $A$ in $O(n \log n)$ time.

8.4 [∗∗] Give an algorithm that finds a maximal subarray of $A$ in $O(n)$ time.