Weekplan: Analysis of Algorithms

Philip Bille   Inge Li Gørtz

Reading

Introduction to Algorithms, Cormen, Rivest, Leisersons and Stein (CLRS): Chapter 3.

Exercises

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

$$n \log n \qquad n^2 \qquad 2^n \qquad n^3 \qquad \sqrt{n} \qquad n$$

2 **Θ-notation** Write the following expressions using Θ-notation.

$$n^2 + n^3/2$$
$$2^n + n^4$$
$$\log_2 n + n\sqrt{n}$$
$$n(n-6)$$
$$4\sqrt{n}$$

$$8\log_2^7 n + 34\log_2 n + \frac{1}{1000}n$$
$$2^n 7 + 5\log_2^3 n$$
$$n(n^2 - 18)\log_2 n$$
$$n\log_2^4 n + n^2$$
$$n^3\log_2 n + \sqrt{n}\log_2 n$$

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

```
LOOP1(n)
i = 1
while i ≤ n do
    print "⋆"
    i = 2 · i
end while
```

```
LOOP2(n)
i = 1
while i ≤ n do
    print "⋆"
    i = 5 · i
end while
```

```
LOOP3(n)
for i = 1 to n do
    j = 1
    while j ≤ n do
        print "⋆"
        j = 2 · j
    end while
end for
```

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

$$\frac{1}{20}n^2 + 100n^3 = O(n^2)$$
$$\log_2 n + n = O(n)$$
$$2^{\log_2 n} = O(n)$$
$$n^3(n-1)/5 = \Theta(n^3)$$
$$\log_2^2 n + n = \Theta(n)$$

$$\frac{n^3}{1000} + n + 100 = \Omega(n^2)$$
$$2^n + n^2 = \Omega(n)$$
$$\log_4 n + \log_{16} n = \Theta(\log n)$$
$$n^{1/4} + n^2 = \Theta(n)$$
$$2^{\log_4 n} = \Theta(\sqrt{n})$$

5 Doubling Hypothesis Solve the following exercises.

5.1 [w] 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 Algorithm B runs in time respectively $5, 20, 45, 80$ and 125 seconds on input of sizes $1000, 2000, 3000, 4000$ and 5000. Give an estimate of the running time of B on a input of size 6000. Express the (estimated) running time of B using O-notation as a function of the input size n.

5.3 Algorithm C runs 3 seconds slower each time the size of the input is doubled. Express the running time of C using O-notation as a function of the input size n.

6 Asymptotic Properties Solve the following exercises.

6.1 Let $f(n)$ and $g(n)$ be asymptotically non-negative. Show that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

6.2 Explain why the statement "the running time of algorithm A is at least $O(n^2)$" does not make sense.

6.3 Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

6.4 Show that $\log_2(n!) = O(n \log n)$.

6.5 [$*$] Show that $\log_2(n!) = \Omega(n \log n)$. Combine with exercise **6.4** to conclude that $\log_2(n!) = \Theta(n \log n)$.

7 Generalized Merge Sort Professor M. Erge 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 subarrray $A[i..j]$ 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 [$\dagger$] 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 [$*\dagger$] 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.