Institut for Matematik og Datalogi
Syddansk Universitet, Odense

27. February 2025

Exercise Sheet Week 3

DSK814

Summarizing the lectures, there are at least three natural levels to describe an algorithm:

1. as an idea described in words and drawings,
2. as pseudocode and
3. as finished code (e.g. Python or Java).

These are also the natural stages to pass through when developing algorithms for a real-world program. Level 1 already captures the main idea of an algorithm, thus we can analyze the correctness and asymptotic behavior with rough sorting of algorithms by growth rate. Level 2 adds more details to ensure the correctness of the code which can take a surprisingly long time. It is very useful to do a correctness analysis of the details, e.g. initialization and updating of variables, in your pseudocode before moving on to the next level. If level 2 is done thoroughly, level 3 is often relatively easy.

In the lectures, algorithms are primarily described on level 1 and partly on level 2. The book usually considers level 1 and level 2—with most algorithms stated in great detail at level 2. In the programming assignments for the practice classes and the project we work on level 3. This is often based on level 2 from the book and other times based on level 1 and 2, which you have to come up with yourself.

In tasks where you have to develop, describe and/or analyze algorithms, it is enough to do this at level 1 unless otherwise stated (i.e. we directly ask for pseudocode or Python/Java implementation).

A. Solve in the practice sessions

1. State the asymptotic order of the following functions:

$$\sqrt{n}, 2^n, (\log_{10} n)^2, n, \log_2 n$$

Intuitively, we assume that the asymptotic order is given by:

$$\log_2 n, (\log_{10} n)^2, \sqrt{n}, n, 2^n$$

Show that the postulated order is correct by showing $f(n) = o(g(n))$ for all neighbor pairs $f(n), g(n)$ in the list.

2. Which of the following statements are true?

   (a) $2n + 5 = \Theta(n)$
   (b) $n = o(n^2)$
   (c) $n = O(n^2)$
   (d) $n = \Theta(n^2)$
   (e) $n = \Omega(n^2)$
   (f) $n = \omega(n^2)$
   (g) $\sqrt{n}\log(n) = O(\sqrt{n})$
   (h) $\sqrt{n}\log(n) = \omega(\sqrt{n})$
   (i) $\sqrt{n} + \log(n) = O(n)$
   (j) $(\log(n))^3 = O(n \log n)$

3. Let f_1, f_2, g_1, g_2 be positive functions that satisfy $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$. Which of the following statements are true?

   (a) $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$
   (b) $g_1(n) + g_2(n) = \Omega(f_1(n) + f_2(n))$
   (c) $\frac{f_1(n)}{f_2(n)} = O\left(\frac{g_1(n)}{g_2(n)}\right)$

   Hint: If the statement holds, prove it using the definitions of O and Ω. If not, provide a counterexample.

4. For the following two algorithms, state their asymptotic running time in Θ-notation as a function of n.

```
ALGORITHM1($n$)
    $s = 0$
    for $i = 1$ to $\lfloor n/2 \rfloor$
        for $j = i$ to $n - i$
            $s = s + 1$
    return $s$
```

```
ALGORITHM2($n$)
    $s = 0$
    for $i = 1$ to $n$
        for $j = 1$ to $n$
            for $k = 1$ to $n$
                $s = s + 1$
    return $s$
```

5. Implement MERGESORT in Python or Java based on the book's pseudocode Cormen et al., 4. Edition: page 36 and 39 [Cormen et al., 3. Edition: page 31 and 34]. Let the input be a list (Python) or an array (Java) of integers.

   Test your code by generating different small arrays/lists (e.g. ascending, descending, random order), sorting them and checking the result.

   For Cormen et al., 3. Edition: Here, the value appears ∞ in the pseudocode on page 31. One quick fix is to use a really large number N and ensure that only integers smaller than N appear in the input.[1] Alternatively, use the `Merge` pseudocode from Cormen et al., 4. Edition, which does not use ∞ and can be found in the sorting slides. This is the better option, as it avoids having to take into account the value N chosen as ∞.

B. Solve at home before tutorial in week 4

1. Which of the following statements are true?

   (a) $n^2 = \Omega(n)$
   (b) $n = \Theta(n^2)$
   (c) $n \log n = o(n^2)$
   (d) $\log n = O(\sqrt{n})$
   (e) $n! = \omega(2^n)$

[1] Both Python and Java have a built-in value for ∞ (Python: `float("inf")`, Java: `Double.POSITIVE_INFINITY`). These are technically decimal numbers, but work as intended when compared to integers. In Java, a decimal number and an integer cannot appear in the same array, so it cannot be used in this implementation. In Python, however, a decimal number and an integer can appear in the same list, so your implementation can use ∞.

2. For the following two algorithms, state their asymptotic running time in Θ-notation as a function of n.

```
ALGORITHM3($n$)
    $s = 0$
    for $i = 1$ to $n$
        for $j = 1$ to $n$
            if $i == j$
                for $k = 1$ to $n$
                    $s = s + 1$
    return $s$
```

```
ALGORITHM4($n$)
    $s = 0$
    for $i = 1$ to $n$
        for $j = 1$ to $n$
            if $i\,!= j$
                for $k = 1$ to $n$
                    $s = s + 1$
    return $s$
```

3. Extend our implementation of MERGESORT as follows:

   Add the ability to time your code—only the sorting itself, not the part of the program that generates the array/list. Then run your code on input (random integers) which is sorted and reverse sorted. Do this for at least 5 different values of n (number of items to sort), choose values such that the program takes from about 100 to about 5000 milliseconds. Repeat each run three times and find the average number of milliseconds spent in the three runs. Divide the resulting numbers by $n \log_2 n$, thereby checking how well the analysis fits the practice—the resulting numbers should be constant according to the analysis.

   Hint: See the assignment sheet for week 2 about potential index pitfalls in Python/Java, as well as methods for timing and generating random integers. In Python, use `math.log(x,2)` *to calculate* $\log_2(x)$ *(import the* `math` *module). In Java, use* `java.lang.Math.log(x)` *to calculate* $\log_e(x)$*, i.e. the logarithm with base e. In order to calculate* $\log_2(x)$*, use the fact* $\log_2(x) = \log_e(x)/\log_e(2)$*, see Cormen et al., 4. Edition: Formula 3.19 page 66 [Cormen et al., 3. Edition: Formula 3.15 page 56].*

4. ($*$) Let $A[1...n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an *inversion* of A.

   Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \log n)$ worst-case time.

   Hint: Modify Mergesort.