Institut for Matematik og Datalogi
Syddansk Universitet, Odense

10. April 2025

Exercise Sheet Week 9

DSK814

A. Solve in the practice sessions

1. Cormen et al., 4. Edition: Exercise 4.5-1 (page 106) [Cormen et al., 3. Edition: Exercise 4.5-1 (page 96)].

   Use the master method to give tight asymptotic bounds for the following recurrences.

$$T(n) = 2 \cdot T(n/4) + 1$$
$$T(n) = 2 \cdot T(n/4) + \sqrt{n}$$
$$T(n) = 2 \cdot T(n/4) + \sqrt{n} \log^2 n$$
$$T(n) = 2 \cdot T(n/4) + n$$
$$T(n) = 2 \cdot T(n/4) + n^2$$

2. Cormen et al., 4. Edition: Exercise 4.5-3 (page 106) [Cormen et al., 3. Edition: Exercise 4.5-3(page 97)].

   Use the Master Theorem to show that the solution to the binary-search recurrence $T(n) = T(n/2) + \Theta(1)$ is $T(n) = \Theta(\log n)$. (For a description of binary search, see Exercise 2.3-6.)

3. Solve the following recurrence using the recursion tree method.

$$T(n) = 2 \cdot T(n-1) + 1$$

   Can it be solved via the Master Theorem?

4. Cormen et al., 4. Edition: Exercise 2.3-5 (page 44) [Cormen et al., 3. Edition: Exercise 2.3-4 (page 39)].

   You can also think of insertion sort as a recursive algorithm. In order to sort $A[1 : n]$, recursively sort the subarray $A[1 : n-1]$ and then insert $A[n]$ into the sorted subarray $A[1 : n-1]$. Write pseudocode for this recursive version of insertion sort. Give a recurrence for its worst-case running time.

   Then solve the recurrence via the recursion tree method. Can it be solved via the Master Theorem?

5. Cormen et al., 4. Edition: Exercise 4.2-1 (page 89) [Cormen et al., 3. Edition: Exercise 4.2-1 (page 82)].

   Use Strassen's algorithm to compute the matrix product

   $$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}.$$

   Calculate $S_1, \ldots, S_{10}$, $P_1, \ldots, P_7$ and C_{11}, C_{12}, C_{21} and C_{22} for the simple case that A_{ij}, B_{ij} and C_{ij} are 1×1-matrices, i.e. are just numbers. Here, we can see that Strassen's algorithm originated from discovering that the matrix product of two 2×2 matrices can be found using only 7 multiplications. Next, Strassen simply replaces numbers in 2×2 matrices with $n/2 \times n/2$ submatrices in $n \times n$ matrices.

6. Desribe how Strassen's algorithm be extended to multiplying $n \times n$ matrices, where n is not a power of two. Argue why the running time is still $\Theta(n^{\log_2 7})$.

   Hint: Fill in with zeros.

B. Solve at home before tutorial in week 10

1. Solve the following recurrences using the Master Theorem.

   $$T(n) = 8 \cdot T(n/3) + n^2$$
   $$T(n) = 9 \cdot T(n/3) + n^2$$
   $$T(n) = 10 \cdot T(n/3) + n^2$$

2. Solve the following recurrence using the Master Theorem.

   $$T(n) = 4 \cdot T(n/2) + n^2 \log(n)$$

(∗) Optional: Solve it with the recursion tree method (refer to example 4 from the slides).

3. (∗) Solve the following recurrence using the recursion tree method:

$$T(n) = 4 \cdot T(n/2 + 2) + n$$

Can it be solved via the Master Theorem?

Technical detail: For this recursion formula, the base case and thus the size of the input in the leaves must be $n \leq 5$. For example, for $n = 4$, $4/2 + 2 = 4$ shows that the recursive call does not get smaller, i.e. the recursion will never stop if the base case limit is set lower.

Hint: Get inspired by the last example in the slides (about floors and ceilings).

4. Cormen et al., 4. Edition: Exercise 4.2-4 (page 89) [Cormen et al., 3. Edition: Exercise 4.2-5 (page 82)].

V. Pan discovered a way of multiplying 68×68 matrices using $132,464$ multiplications, a way of multiplying 70×70 matrices using $143,640$ multiplications, as well as a way of multiplying 72×72 matrices using $155,424$ multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare with Strassen's algorithm?

In order to get some intuition for this exercise, refer to Exercise A.5.

5. Cormen et al., 4. Edition: Exercise 4.5-2 (page 106) [Cormen et al., 3. Edition: Exercise 4.5-2 (page 97)].

Show how to multiply the complex numbers $a + bi$ and $c + di$ using only three multiplications of real numbers. The algorithm should take a, b, c and d as input and produce the real component $ac - bd$ and the imaginary component $ad - bc$ separately.