Institut for Matematik og Datalogi
Syddansk Universitet, Odense

03. April 2025

Exercise Sheet Week 8

DSK814

Unless otherwise stated, all recursion equations in Part II must be solved using both the recursion tree method (Section 4.4) and the Master Theorem (Section 4.5).

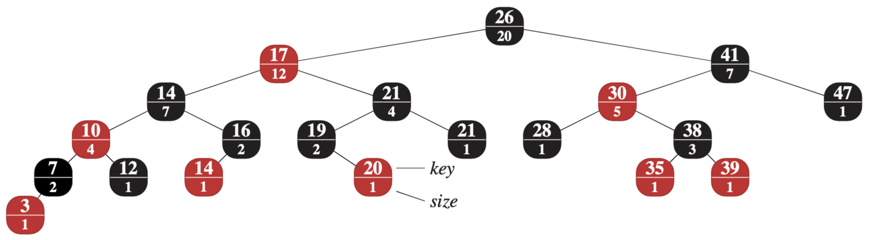
The substitution method (Section 4.3) is quite simple (it is just an induction proof) and very general, but it requires a good guess at a solution and can be a bit technical. Therefore, it is *not* part of the curriculum.

Regarding solving recursion equations, read page 78 [Cormen et al., 3. Edition: page 67] to understand which technical details can generally be omitted when working with recursion equations.

A(I). Solve in the practice sessions

1. Cormen et al., 4. Edition: Exercise 17.1-1 (page 485) [Cormen et al., 3. Edition: Exercise 14.1-1 (page 344)].

   Show how OS-SELECT($T.root, 10$) operates on the red-black tree T below.



2. Cormen et al., 4. Edition: Exercise 17.1-2 (page 485) [Cormen et al., 3. Edition: Exercise 14.1-2 (page 344)].

Show how OS-Rank(T, x) operates on the red-black tree T pictured above and the node x with $x.key = 35$.

3. $(*)$ Cormen et al., 4. Edition: Exercise 17.1-5 (page 486) [Cormen et al., 3. Edition: Exercise 14.1-5 (page 344)].

   Given an element x in an n-node order-statistic tree and a natural number i, show how to determine the i-th successor of x in the linear order of the tree in $O(\log n)$ time.

   You don't necessarily have to provide pseudocode, you can simply describe the idea of your algorithm, e.g. with figures.

4. Cormen et al., 4. Edition: Exercise 2.1-4 (page 25) [Cormen et al., 3. Edition: Exercise 2.1-3 (page 22)].

   Consider the searching problem:

   Input: A sequence of n numbers $a_1, a_2, \cdots, a_n$ stored in array $A[1:n]$ and a value x.

   Output: An index i such that x equals $A[i]$ or the special value NIL if x does not appear in A.

   Write pseudocode for linear search, which scans through the array from beginning to end, looking for x. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

5. Cormen et al., 4. Edition: Exercise 2.2-2 (page 33) [Cormen et al., 3. Edition: Exercise 2.2-2 (page 29)].

   Consider sorting n numbers stored in array $A[1:n]$ by first finding the smallest element of $A[1:n]$ and exchanging it with the element in $A[1]$. Then find the smallest element of $A[2:n]$, and exchange it with $A[2]$. Then find the smallest element of $A[3:n]$, and exchange it with $A[3]$. Continue in this manner for the first $n-1$ elements of A. Write pseudocode for this algorithm, which is known as selection sort. What loop invariant does this algorithm maintain? Why does it need to run for only the first $n-1$ elements, rather than for all n elements? Give the worst-case running time of Selection Sort in Θ-notation. Is the best-case running time any better?

A(II). Solve in the practice sessions

Solve the following recursion equations.

1. $T(n) = 3 \cdot T(n/3) + n^2$ (via Master Theorem)
2. $T(n) = T(n/2) + n^2$ (via recursion tree method and Master Theorem)
3. $T(n) = 16 \cdot T(n/2) + n^4 + n^2$ (via Master Theorem)

B(I). Solve at home before tutorial in week 9

1. ($*$) Cormen et al., 4. Edition: Exercise 17.1-7 (page 486) [Cormen et al., 3. Edition: Exercise 14.1-7 (page 345)].

   Show how to use an order-statistic tree to count the number of inversions (see Cormen et al., 4. Edition: Problem 2-4 (page 47) [Cormen et al., 3. Edition: Problem 2-4 (page 41)]) in an array of n distinct elements in $O(n \log n)$ time.

   Hint: similar to insertion sort, but think of insertions into a tree. The elements in the array are all assumed to be different.

B(II). Solve at home before tutorial in week 9

Solve the following recursion equations.

1. $T(n) = 3 \cdot T(n/3) + n^2$ (via Master Theorem)
2. $T(n) = 4 \cdot T(n/2 + 2) + n$ (via recursion tree method)