Institut for Matematik og Datalogi
Syddansk Universitet, Odense

06. March 2025

Exercise Sheet Week 4

DSK814

A(I). Solve in the practice sessions

Note that we refer to the PARTITION variant from Cormen et al., 4. Edition: page 184 [Cormen et al., 3. Edition: page 171] throughout this section, unless otherwise specified.

1. Cormen et al., 4. Edition: Exercise 7.1-1 (page 186) [Cormen et al., 3. Edition: Exercise 7.1-1 (page 173)].

   Using Figure 7.1 as a model, illustrate the operation of PARTITION on the array $A = [13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11]$.

2. Cormen et al., 4. Edition: Exercise 7.1-2 (page 187) [Cormen et al., 3. Edition: Exercise 7.1-2 (page 174)].

   What value of q does PARTITION return when all elements in the subarray $A[p : r]$ have the same value? Modify PARTITION such that $q = \lfloor (p + r)/2 \rfloor$ when all elements in the subarray $A[p : r]$ have the same value.

   Hint: Change the second area in Cormen et al., 4. Edition: Figure 7.2 (page 186) [Cormen et al., 3. Edition: Figure 7.2 (page 173)] from '$> x$"to "$\geq x$", and ensure that the size of the two areas is as balanced as possible during the execution of PARTITION. *Convince yourself that this does not change the correctness of* QUICKSORT.

3. Cormen et al., 4. Edition: Exercise 7.2-2 (page 191) [Cormen et al., 3. Edition: Exercise 7.2-2 (page 178)].

What is the running time of QUICKSORT when all elements of array A have the same value?

Hint: in Exercise 7.1-2, we already determined what PARTITION *does on that input. Argue what happens to the running time when this happens repeatedly in* QUICKSORT.

4. Cormen et al., 4. Edition: Exercise 7.2-3 (page 191) [Cormen et al., 3. Edition: Exercise 7.2-3 (page 178)].

   Show that the running time of QUICKSORT is $\Theta(n^2)$ when the array A contains distinct elements and is sorted in decreasing/ascending order.

   Hint: start by determining what PARTITION *does on these inputs. Argue how the running time behaves when this happens repeatedly in* QUICKSORT.

5. State the best and worst-case runtime as well as the runtime for sorted input for INSERTIONSORT, MERGESORT and QUICKSORT.

6. Cormen et al., 4. Edition: Problem 7-2 b. (page 200) [Cormen et al., 3. Edition: Problem 7-2 b. (page 186)].

   The PARTITION procedure returns an index q such that each element of $A[p : q-1]$ is less than or equal to $A[q]$ and each element of $A[q+1 : r]$ is greater than $A[q]$. Modify PARTITION to a procedure PARTITION'(A, p, r), which permutes the elements of $A[p : r]$ and returns two indices q and t, where $p \leq q \leq t \leq r$, such that

   - all elements of $A[q : t]$ are equal,
   - each element of $A[p : q-1]$ is less than $A[q]$, and
   - each element of $A[t+q : r]$ is greater than $A[q]$.

   The PARTITION' procedure should run in $\Theta(r-p)$ time.

   Assume that QUICKSORT uses the new procedure PARTITION' (making its two recursive calls on the two parts of the array A where PARTITION has placed elements different from $A[q]$). Answer the questions A(I).3 and A(I).4 for this variant of QUICKSORT.

A(II). Solve in the practice sessions

1. Cormen et al., 4. Edition: Exercise 6.1-6 (page 164) [Cormen et al., 3. Edition: Exercise 6.1-5 (page 154)].

Is an array that is in sorted order a min-heap?

2. Cormen et al., 4. Edition: Exercise 6.1-7 (page 164) [Cormen et al., 3. Edition: Exercise 6.1-6 (page 154)].

   Is the array with values $[33, 19, 20, 15, 13, 10, 2, 13, 16, 12]$ a max-heap?

3. Cormen et al., 4. Edition: Exercise 6.2-1 (page 166) [Cormen et al., 3. Edition: Exercise 6.2-1 (page 156)].

   Using Figure 6.2 as a model, illustrate MAX-HEAPIFY$(A, 3)$ on the array $A = [27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$.

4. Insert 9 into the max-heap $[10, 8, 6, 3, 7, 4, 5, 1, 2]$. You may draw it as a tree instead of an array.

5. Insert 2 into the min-heap $[1, 3, 5, 4, 10, 13, 7, 6, 17]$. You may draw it as a tree instead of an array.

6. Perform MAX-HEAP-EXTRACT-MAX(A) on the max-heap A.

$A =$

1	2	3	4	5	6	7	8	9	10
21	18	10	12	8	9	4	7	5	2

7. Cormen et al., 4. Edition: Exercise 6.1-4 (page 164) [Cormen et al., 3. Edition: Exercise 6.1-4 (page 154)].

   Where in a max-heap might the smallest element reside, assuming that all elements are distinct?

B(I). Solve at home before tutorial in week 5

1. Implement QUICKSORT in Java or Python from the pseudocode in Cormen et al., 4. Edition: side 183–184 [Cormen et al., 3. Edition: side 171]. Test that your code works by generating arrays with different contents and sorting them. Add timing to your code (only the sorting itself, not the part of the program that generates the array's content).

   Then run your code with inputs that are random `int`'s. Do this for at least 5 different values of n (number of elements to sort), choosing values that cause the program to spend from about 100 to about 5000 milliseconds. Repeat each run three times and find the average of the number of milliseconds used in the three runs. Divide the resulting numbers by $n \log_2 n$ and check how well the analysis fits with practice - the resulting numbers should be constant according to the analysis.

Compare your results with the ones for MERGESORT in week 3. Is QUICKSORT or MERGESORT faster?

In Java, repeat the experiment with the method `sort` from the class `java.util.Arrays` (an even more optimized QUICKSORT implementation). In Python, do the same with the `sort` method from lists (uses Powersort, which is a variant of MERGESORT designed to take advantage of any existing order in the input). Compare the runtimes to your implementations of QUICKSORT and MERGESORT.

Extra Task: Implement a slightly optimized variant of QUICKSORT. Select the pivot element x in PARTITION from the three elements at the first, middle and last position in the partial array to be partitioned. More precisely, choose the median of the three elements as the pivot element. Do this for all calls to PARTITION with 16 elements or more, but use the book PARTITION for smaller instances. Does this version of QUICKSORT run (slightly) faster than the standard version?

B(II). Solve at home before tutorial in week 5

1. Draw all possible min-heaps containing four nodes with 1, 2, 3 and 4.

2. Cormen et al., 4. Edition: Exercise 6.5-11 (page 178) [Cormen et al., 3. Edition: Exercise 6.5-9 (page 166)].

   Give an $O(n \log k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements over all the input lists.

   Hint: Use a min-heap for k-way merging.

3. $(*)$ Cormen et al., 4. Edition: Problem 6.2 (page 179) [Cormen et al., 3. Edition: Problem 6.2 (page 167)].

   A d-ary heap is like a binary heap, but (with one possible exception) nonleaf nodes have d children instead of two children. In all parts of this problem, assume that the time to maintain the mapping between objects and heap elements is $O(1)$ per operation.

   (a) Describe how to represent a d-ary heap in an array.

(b) Using Θ-notation, express the height of a d-ary heap of n elements in terms of n and d.

(c) Give an efficient implementation of EXTRACT-MAX in a d-ary max-heap. Analyze its running time in terms of d and n.

(d) Give an efficient implementation of INCREASE-KEY in a d-ary max-heap. Analyze its running time in terms of d and n.

(e) Give an efficient implementation of INSERT in a d-ary max-heap. Analyze its running time in terms of d and n.

Hint: For part b, use e.g. Cormen et al., 4. Edition: Formula A.6 (page 1142) [Cormen et al., 3. Edition: Formula A.5 (page 1147)] and inspiration from slide analysis of height in binary heaps.