Institut for Matematik og Datalogi
Syddansk Universitet, Odense

13. Feburary 2025

Exercise Sheet Week 1

DSK814

The assignments for the practice classes (also called exam classes or e-classes) are divided into two groups:

A. A set of tasks to be solved *in the practice sessions* while the instructor is available for questions.

   These tasks must not be solved in advance, just make sure you have read the material from the lecture. Please feel free to work in your study groups during the lessons as it is a great help to solve the exercises together.

B. Another set of problems similar to the first set to be solved *at home*.

   The answers to these problems are briefly reviewed at the end of the practice lesson with regard to possible solutions, but the problems must be solved at home (preferably together with your study group) before the next tutorial. These tasks are then briefly summarized at the beginning of the next practice session.

A. To be solved in the practice sessions

1. Cormen et al., 4. Edition: problem 1.1 (side 15) [Cormen et al., 3. Edition: problem 1.1 (side 14)]. However, replace "microseconds" with "nanoseconds" (i.e. 10^{-9} seconds), as this is approximately what a CPU cycle takes on a modern processor. Only fill out the columns *second* and *hour* and the rows with n, $n \log n$, n^2, n^3 and 2^n. Start with n^2.

For some inputs, the answer can be found by mathematical calculation, for others, you have to work your way towards the answer by inserting different values of n.

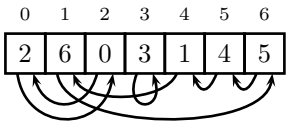
The goal of the assignment is to get a feel for the significance of growth rates of running times, simply by looking at which input sizes can be processed with a given time budget. Here, the results are more interesting than the calculation method itself.

2. Brodal's notes on puzzles, task 1.

3. Brodal's notes on puzzles, problem 2. Here, "optimal sequence of permutations" means a number of permutations as specified by Theorem 1 in the notes. The problem shows that other algorithms than "greedy algorithms" (algorithms that always put at least one element in place) can be optimal for this puzzle problem.

4. Write a Java or Python program that generates a random permutation of the integers from 0 to $n-1$ (for which n is an input parameter). In Java, you can use the type `ArrayList` as well as the method `shuffle` from the utility class `Collections`. In Python you can use lists and the function `shuffle` from the module `random`. Print the numbers in your permutation.

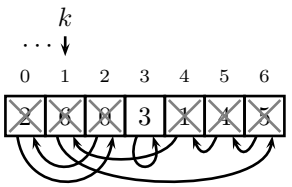
5. If an array/list contains a permutation of the numbers 0 to $n-1$, define circles similar to the puzzle in the first lecture. For each number x at index y in the array, there exists an arrow from index y to index x (i.e. if number 1 is in index 4 in the array, there is an arrow from index 4 to index 1). A collection of arrows that are connected in a cyclic chain is called a circle. Here is an example with a total of three circles in the permutation (check that you find them):



Create an algorithm that counts the number of circles in a permutation.

Hint: Have a counter k which runs from 0 to $n-1$. For each value of k, check if an undiscovered circuit starts at index k. Whenever this

is the case, the circuit is traversed and all its elements are marked as discovered. The figure below shows the situation after k has reached 1 (and has traversed the circle starting there). Crossing over an element means that it has been discovered.



What is the running time of your algorithm as a function of n? Implement your algorithm in Java or Python.

B. To be solved at home before tutorial in week 2

1. Continue with Cormen et al., 4. Edition: problem 1.1 (side 15) [Cormen et al., 3. Edition: problem 1.1 (side 14)]. Again, replace "microseconds" with "nanoseconds" (i.e. 10^{-9} seconds) and only consider the rows with n, $n \log n$, n^2, n^3 and 2^n. But this time fill out the columns *year* and *century*.

   For some inputs, the answer can be found by mathematical calculation, for others, you have to work your way towards the answer by inserting different values of n.

2. In the practice lesson, you created a program that could generate a random permutation, and a program that could count the number of circles in a permutation.

   Use these programs to generate a bunch of random permutations of length 16, and for each of them count the number of circles in them. Use data from these runs to estimate the probability that there are k circles in a random permutation with $n = 16$ for $k = 1, 2, \ldots, 16$. That is, create (data for) your own version of Figure 3 in the notes, but with n equals 16 and not 64.

   Also find the average number of circles in your experiments. Match your number with the formula on page 4 of the notes, i.e. is it close to $H_{16} = \sum_{i=1}^{16} 1/i = 1/1 + 1/2 + 1/3 + \ldots + 1/16$?