DMA

– Week 3 –

Work instructions

In this week, we will start on the math part of DMA! This week we focus math fundamentals needed later on. We will discuss asymptotic growth of functions which is the basis for analyzing and comparing runtimes of algorithms. We will return to asymptotic growth once again later on after we have seen a bit of logic.

You might soon notice that the typical math tools within computer science are quite different from the ones you already know. Formally speaking, the math you already know is mostly **continuous** while **discrete** math is essential in computer science. This means that we start at the very beginning with many of the topics. For this, we have chosen the book

B. Kolby, R.C. Busby, and S.C. Ross
Discrete mathematical structures, 3rd edition,

which we will refer to as KBR. We will use a special edition of the book in the course as you have probably already noticed. Note, however, that for this first math week, you will also be reading from the book

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein
Introduction to Algorithms, 4th edition,

which we will refer to as CLRS.

The asymptotic notation used in KBR differs from the one introduced in CLRS. We will favour the CLRS notation as it is the one commonly used in the context of computer science. The summation formulas are covered later on in the KBR book and could thus be difficult to read. We have therefore prepared some notes for you.

This week we will also discuss fundamental properties of integers related to division and talk about concepts such as divisor, multiple, prime, greatest common divisor (GCD), and least common multiple (LCM). In particular, we will go over Euclid's algorithm, one of the first algorithmic discoveries of mankind, which lets us calculate GCD of two integers efficiently. We will also discuss different ways to represent integers: first, in terms of their prime factorization and, second, using the base-b positional notation. Note that positional notation with $b = 10$ is our standard way of writing numbers

while $b = 2$ is used to represent numbers on a computer. All of this is nicely covered in KBR Section 1.4, and quite likely you are already familiar with most of it.

Assigned reading

1. [KBR] Read KBR 1.1–1.3. We will only discuss sets and set operations briefly in class on Monday but it will be useful going forward.

2. Notes for Week 3 (available on Absalon). We will discuss this material on Monday (sequences) and Tuesday (functions and asymptotic growth).

3. [KBR] Read section 1.4. Base-b notation will be covered during the exercise session on Monday. The rest of this section will be relevant for the lecture on Friday.

   Make sure you can follow the steps of the proof of Theorem 4, as this is a good example of a more advanced proof. Expect that you will need to pause and think after reading every sentence or two. You will probably need to read this proof more than once to understand it. In general, a good test for whether you have fully understood a proof is whether you can explain it (or at least its main steps) without looking in the book.

4. [CLRS] Skim Chapter 3 (pages 49–57). This will be useful for our discussion on Tuesday. Don't worry if not all of it makes sense just yet. We will read this again in later on.

Software for calculations and visualisation

The math part of the course will not contain many numerical calculations (similarly, the algorithm part will not contain that much programming). Our approach in DMA is foremost theoretical. We will, however, occasionally ask you to calculate some values or draw a graph to visualize a function. You can use whatever tools you are most comfortable with. It does not matter whether this is an app on your phone or math software on your computer.

Lecture plan

Monday Sept. 18th, 09:15–10:00

Welcome and introduction. Sets. Sequences, series, and summation formulas.

Tuesday Sept. 19th 13:15–15:00

Functions relevant for computer science: polynomials, exponentials, logarithms, $\lfloor x \rfloor$, $\lceil x \rceil$, $n!$ etc. Introduction to big-O and asymptotic growth of functions.

Friday Sept. 22nd, 09:15–10:00

Division with remainders, divisor and multiple, greatest common divisor (GCD), least common multiple (LCM), and Euclid's algorithm. Primes.

Exercise plan

At the end of the KBR book you can find solution to all odd-numbered exercises. However, it is advisable not to look at the solution until after you have solved the exercise or if you are completely stuck.

Monday Sept. 18th, 10:15–12:00

(1) Instructor introduces the procedure for obtaining a base b expansion of an integer n. The steps of this procedure are explained in the proof of Theorem 7 from KBR 1.4.

   (a) Solve KBR exercises 1.4.41, 1.4.43, and 1.4.44

(2) Solve KBR exercises 1.1.4, 1.2.1, 1.2.14, 1.3.7, 1.3.10 , 1.3.27 and 1.3.38.

(3) Use summation formulas to calculate

$$\sum_{k=1}^{100}(k^2+k)+\sum_{k=0}^{10}5^k$$

(4) [*] Find an explicit expression for the series

$$\sum_{k=n}^{2n}(k+3)$$

(5) [extra, **] Find an explicit expression for the series

$$\sum_{k=1}^{n} \frac{k}{2^k}$$

Tuesday Sept. 19th, 15:15–17:00

(1) Two teams of computer science students compete in writing the most efficient algorithm to solve a task on an array with N entries. Team 1 has constructed an algorithm that runs in

$$f_1(N) = 10 \cdot N^{10}$$

steps. Team 2 has constructed an algorithm that runs in

$$f_2(N) = 2^N$$

steps. One of the following statements must hold:

(A) Team 1's algorithm runs with fewer steps for all N.

(B) Team 2's algorithm runs with fewer steps for all N.

(C) Team 1's algorithm runs with fewer steps for some choices of N, while Team 2's algorithm runs with fewer steps for some other choices of N.

Draw the graphs of the functions. Which of the statements (A), (B), and (C) are true?

(2) Solve the equation (you will probably want to use a computer here)

$$10 \cdot N^{10} = 2^N$$

What does the solution mean in terms of the previous exercise?

(3) Team 1 and Team 2 have access to two different computers. It takes $c_1 > 0$ seconds to compute one step for Team 1 and $c_2 > 0$ seconds to compute a step for Team 2. One of the following statements must hold:

(A') Team 1's algorithm finishes faster for all N.

(B') Team 2's algorithm finishes faster for all N.

(C') Team 1's algorithm runs finishes faster for some choices of N, while Team 2's algorithm finishes faster for some other choices of N.

Is it possible to identify any of the statements (A'), (B'), and (C') as false without knowing the exact value of c_1 and c_2? If yes, then which one(s)?

(4) Consider the following pseudo-code

```
rec(n,str):
    if n=0
        print(str)
    else
        rec(n-1, "0" + str)
        rec(n-1, "1" + str)
```

Find a recursive expression for the running time, $T(n)$, of `rec(n," ")`. Find an explicit expression for $T(n)$. What does `rec(n," ")` print out?

(5) Use the rules from this week's notes to order the following functions according to their asymptotic growth

$$2^n, \quad 1000, \quad n^3 + \log_2(n)$$

(6) Assume that $a, b, c > 0$. Use the properties of the logarithms from this week's notes to show that

- $(\log_a b)\,(\log_c d) = (\log_a d)\,(\log_c b)$.
- $\log_b a = (\log_a b)^{-1}$

where, in each equation above, logarithm bases are not 1.

Friday Sept. 22nd, 10:15–12:00

(1) Instructor explains the Euclidean algorithm and uses it to compute GCD(210, 45).

(2) Solve KBR exercises 1.4.10–13 (remember to make use of the Euclidean algorithm!), 1.4.16, 1.4.25, 1.4.35.

(3) Consider the algorithm from KBR page 26 for testing if an integer n is a prime. Let's call this algorithm $\mathcal{A}$.

(a) Argue that the running time of $\mathcal{A}$ is $O(\sqrt{n})$.

(b) Argue that the running time of $\mathcal{A}$ is not $O(1)$[1].

(c) [*] Let t_n be the maximum running time of $\mathcal{A}$ over all inputs $k \in \mathbb{Z}$, where $1 \leq k \leq n$. Argue that t_n is $\Theta(\sqrt{n})$.

[1]Recall that $O(1)$ means constant.

Extra exercises

We want to find an efficient algorithm which can output a list of all primes between 2 and n.

(a) [*] Argue that the algorithm, which checks the primality of each integer between 2 and n using the algorithm from KBR (page 26) has runtime $\Theta(n^{3/2})$.

(b) [*] Read about the sieve of Erathosthenes online and produce a pseudo code for it.

(c) [***] Show that the sieve-algorithm is asymptotically faster[2] than the naive algorithm discussed above. *[Hint: Show and then use that* $\sum_{k=1}^{n} \frac{n}{k}$ *is* $\Theta(n \log n)$*.]*

[2]More formally: we want to show that $f(n)$ is $o(g(n))$, where $f(n)$ and $g(n)$ are the running times of the sieve and the naive algorithm respectively.