Algorithms for Data Science
Exercise 1

See the cheatsheet for valuable remarks!

Task 1.1: *Coin Problem*

You are given eight equally looking coins and a pair of scales. One of the coins is forged and has smaller weight, whereas all the other coins weigh the same. A single weighing on the scales consists of comparing the weight of any two (disjoint) subsets of the coins. Note that the weighting instrument does not show the actual weight.

(a) Can you find the forged coin by using the scales three times?

(b) Is it possible to find it using the scales only two times?

(c) If we are allowed to do k weighings, what is the maximum number of coins among which we can find the forged coin?

Task 1.2: *Divide by Two*

In this task, we want to divide a given integer by two without using the division operation. Consider the following algorithm.

Algorithm 1:

Input: integer $n \geq 0$
Output: the value $n/2$ (possibly fractional)
$\ell = 0$
$r = n$
while $\ell \neq r$ **do**
    $\ell = \ell + 1$
    $r = r - 1$
return ℓ

(a) How often does the while loop repeat in dependence of n?

(b) The program is not correct. What is the error? Fix the mistake!

(c) Prove the correctness of the fixed algorithm with a loop invariant. Show also that the algorithm terminates.

Bonus Tasks

Tasks for solving at home (not necessarily discussed in classes); relevant for short tests and the exam.

Task 1.3: *Smallest Element*

In this task, we want to find a smallest element in an unordered array.

(a) Consider the following algorithm.

Algorithm 4: Algorithm A

```
Input: array a[1..n]
Output: the position of a smallest element
minPos = −1
for i = 1 to n do
    isSmallest = true
    for j = 1 to n do
        if a[i] ≥ a[j] then
            isSmallest = false
    if isSmallest then
        minPos = i
return minPos
```

i. What is the output of the algorithm? Fix the error and discuss why the algorithm is now correct!

ii. How often does the (repaired) algorithm compare two elements of a (that is, $a[i] \geq a[j]$) in dependence of n in the worst case?

iii. Suppose that the running time depends only on comparing two elements of a. Further suppose that your computer can compare two elements of a in one millisecond. What is the worst case running time of the (fixed) algorithm for $n = 1000\ 000$ on your computer?

(b) Write a faster algorithm by filling the gaps of Algorithm B below.

Algorithm 5: Algorithm B

```
   Input: array a[1..n]
   Output: the position of a smallest element
1  minPos = 1
2  j = 2
3  while ..... do
4      .....
5      .....
6      j = j + 1
7  return minPos
```

i. How often does Algorithm B compare two elements of a

ii. Suppose that the running time depends only on comparing two elements of a. Further suppose that your computer can compare two elements of a in one millisecond. What is the worst case running time of the algorithm for $n = 1000\ 000$ on your computer?

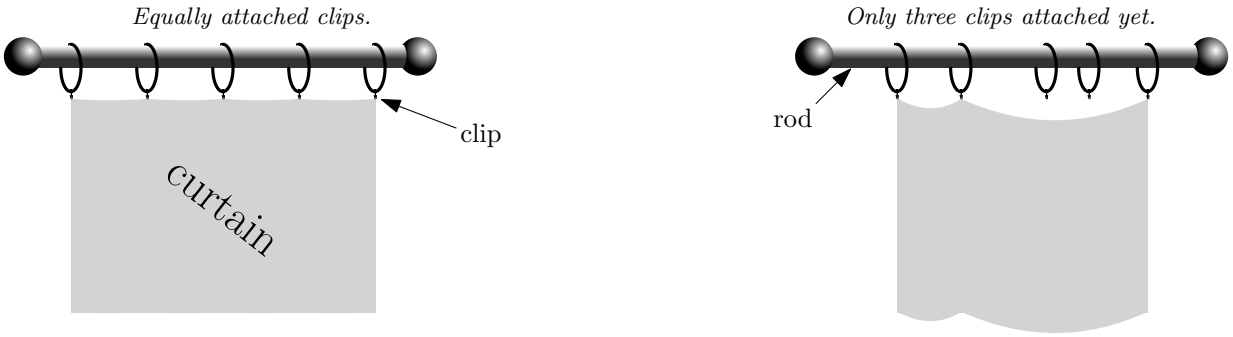
iii. Formulate an invariant depending on a, j and $minPos$. Use the invariant to argue that Algorithm B is correct.

(c) Discuss why any algorithm needs at least $n-1$ comparisons to find a minimum element in the worst case.

Task 1.4: *Curtain Problem - a real-world problem brought by the lecturer's wife*

Suppose that you washed the curtain of your window and now you want to hang it up again. To do so, you have to attach the curtain to the five ring clips lying loosely on a rod above the window. It is important that the clips are equally spaced on the curtain, otherwise it won't look nice.

Unfortunately, you don't have any tool to measure the distances in order to find the right position for attaching the clips. However, you are a smart student and you want to use gravity and the idea underlying binary search to solve this problem.



(a) Describe how to find equally spaced positions for the clips. Note that you can move the ring clips along the rod and that the curtain is obeying the laws of gravity. Your first step is to attach the top corners of the curtain to the left- and right-most clip.

(b) Does your procedure also work for seven clips?

(c) For what numbers of clips does your procedure work?

Task 1.5: *Guess my number!—Level 2*

Consider the variant of the game *Guess my number!* when there is no upper bound on the integer number x to be guessed. We only know $x \geq 8$. Propose an algorithm to solve the problem using at most $c \cdot \log_2 x$ questions where c can be any constant number that does not depend on x; e.g. $c = 8$.

Hint: First, try to find an upper bound r of x with as few questions as possible. Then, run Binary Search from the lecture. You can assume that it needs only $\log_2 n$ questions when n is the size of the given interval containing x.

Task 1.6: *Real Time Running Times*

Given are five algorithms for the same (unknown) problem that get an array of size n as the input. The number of operations of the algorithms in dependence of n are $\log_2 n$, n, $n \log_2 n$, n^2 and 2^n, respectively. Give their running times using an adequate time unit (e.g. nanoseconds,..., years) in each case for different values of n. You can round the time values to integers and ignore running times above 10 years.

(a) Suppose your computer can do 10 operations per second.

	$\log_2 n$	n	$n \log_2 n$	n^2	2^n
n=10					
n=20					
n=21					
n=1000					
n=1000 000					
n=1000 000 000					

(b) Suppose your computer can do 1000 000 operations per second.

	$\log_2 n$	n	$n \log_2 n$	n^2	2^n
n=10					
n=20					
n=21					
n=1000					
n=1000 000					
n=1000 000 000					

Advanced Tasks

Challenging tasks for motivated students, but beyond the scope of this course. Their difficulty level is comparable to that of standard computer science courses.

Task 1.7: *Computing the Square Root*

Write an algorithm that, given a nonnegative integer n, computes the value $\lfloor \sqrt{n} \rfloor$. The algorithm is not allowed to call other functions and is allowed to have only a single loop. This loop is not allowed to iterate more than $\lceil \log_2 n \rceil + 1$ times. Prove the correctnes of your algorithm.