Problems Solved:

41	42	43	44	45

Name:

Matrikel-Nr.:

Problem 41.

1. Consider the probability space $\Omega = \{0,1\}^n$ of all strings over $\{0,1\}$ of length n where each string occurs with the same probability 2^{-n}. Let $X : \Omega \to \mathbb{N}$ be a random variable that gives the position of the first occurrence of the symbol 1 in a string, if 1 occurs at all. For completeness, we also define that $X(0^n) = 0$. Positions are numbered from 1 to n. Give a term (not necessarily in closed form, i. e., the solution may use the summation sign) for the expected value $E(X)$ of the random variable X and justify your answer.

2. Evaluate the sum

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

in *closed form*, i.e., find a formula for the sum which does not involve a summation sign.

Hint: Take the function

$$F(z) := \sum_{k=0}^{n} \left(\frac{z}{2}\right)^k.$$

and let $F'(z)$ denote the first derivative of $F(z)$. We then have $S = F'(1)$. Why?

Thus, it suffices to compute a closed form of $F(z)$, using your high-school knowledge about geometric series. Then compute the first derivative $F'(z)$ of this form, and, finally, evaluate $F'(1)$.

You may *check* your result with the help of a computer algebra system or `https://www.wolframalpha.com`. Note, however, that simply writing down what the computer algebra system gives you is only counted, if it comes along with a proof that the function that you called gives exactly what is asked for in this problem together with a proof that this function is implemented without bugs.

Note that the index for the geometric series starts at $k = 0$.

Problem 42. Let $M = (Q, \Gamma, \sqcup, \Sigma, \delta, q_0, F)$ be a Turing machine with $Q = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$, $F = \{q_1\}$ and the following transition function δ:

δ	0	1	$\sqcup$
q_0	$q_0 0 R$	$q_1 1 R$	–
q_1	–	–	–

1. Determine the (worst-case) time complexity $T(n)$ and the (worst-case) space complexity $S(n)$ of M.
2. Determine the average-case time complexity $\bar{T}(n)$ and the average-case space complexity $\bar{S}(n)$ of M. (Assume that all 2^n input words of length n occur with the same probability, i.e., $1/2^n$.)
3. Bonus: Using results from Problem 41, express all answers in closed form, i.e., without the use of the summation symbol.

Problem 43. Let M be a Turing machine over the alphabet $\{0, 1\}$ that takes as input a string $b_1 b_2 \ldots b_n$ ($b_i \in \{0, 1\}$), prepends an additional 1 to the string and then interprets the result $1b_1 b_2 \ldots b_n$ as the binary representation of a number k. M then writes out the unary representation of k (consisting of a string of k letters 1) onto the tape and stops.
Note that in the above description it is not said *how* M computes the result. In particular M need not be the most efficient Turing machine fulfilling the above specification.

1. Give a reasonably close asymptotic lower-bound for the space- and time-complexity $S(n)$ and $T(n)$ for the execution of the task and justify these bounds (without giving a detailed construction of M). Choose adequate Landau-symbols for formulating the bounds.
2. Give an informal description of a (reasonably efficient) Turing machine M' that performs the task described above. Analyze the space and time complexity $S(n)$ and $T(n)$ and write down an upper/exact asymptotic bound for these complexities. Again choose adequate Landau symbols for formulating the bounds.

   Hint: Let M' apply the *binary powering* strategy.

Problem 44. Let X be a monoid. Device an “algorithm” (as recursive/iterative pseudo-code in the style of Chapter 6 of the lecture notes) for the computation of x^n for $x \in X, n \in \mathbb{N}$ that uses less multiplications than the naive algorithm of n times multiplying x to the result obtained so far. Determine the complexity as $M(n)$, i.e., the number of multiplications of your “algorithm” depending on the exponent n.
Hint: Note that x^8 can be computed with just 3 multiplications while the naive algorithm would use 7 multiplications. Based on this observation, the algorithm can be based on a kind of "binary powering" strategy.

Problem 45. Let $T(n)$ be given by the recurrence relation

$$T(n) = 3T(\lfloor n/2 \rfloor).$$

and the initial value $T(1) = 1$. Show that $T(n) = O(n^\alpha)$ with $\alpha = \log_2(3)$.
Hint: Define $P(n) :\iff T(n) \leq n^\alpha$. Show that $P(n)$ holds for all $n \geq 1$ by induction on n. It is not necessary to restrict your attention to powers of two.