Assignment 3

Solve the following three exercises. Each exercise is worth five points. Solutions must be submitted by 30.4.2025. Use the link on e-ucilnica to turn in your work. The submission must be in pdf format.

Exercise 1: Amortization

You are working on an algorithm that adds rows and columns to a matrix. Each call to an *add()* function costs $i+3$ where i is the i-th call of the *add()* function. Every call where $i = k^2$ for some k costs i^2. Meaning the cost function is :

$$c_i = \begin{cases} i+3 & ; \quad i \neq k^2, k \in \mathbb{N} \\ i^2 & ; \quad i = k^2, k \in \mathbb{N} \end{cases}$$

what is the amortized cost of *add()* function?

Exercise 2: Amortization

Consider developing a dynamic table that only supports the insert operation. Instead of doubling the table size when it becomes full, you decide to increase its size by just 10%. Determine whether the amortized cost of an insertion remains constant under this strategy. Provide proof using the potential method by providing the correct potential function and showing that all amortized costs are constant.

Exercise 3: Approximation

Suppose you are working with a *symmetric 4-SAT* formula, described by 4-CNF formula F with n clauses, where each clause consists of 4 distinct literals.

For example:
$F = (x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_4 \vee \neg x_2 \vee \neg x_1 \vee x_3) \wedge (\neg x_3 \vee x_2 \vee \neg x_5 \vee x_1)$.

The common *4-SAT* accepts each clause if it evaluates to 1. *Symmetric 4-SAT* accepts a clause if it evaluates to 1 and its complement clause (the clause with all literals negated) also evaluates to 1. In other words, symmetric 4-SAT accepts each clause if it has one literal that assigns to 0 and one that assigns to 1.

A *symmetric MAX 4-SAT* is an NP-complete problem where we try to satisfy as many clauses as possible. We will use a simple approximation algorithm to solve the problem by setting each variable to 0 with a probability of 0.5 and to 1 with probability of 0.5.

Your task is to find the approximation factor for this algorithm.

Note: In 4-CNF, each clause can not have the same literal twice or have a variable x_i and its negation $\neg x_i$.