

## 1 Coloring Competition (Points 4)

Implement SAT based graph vertex coloring solver. Your application should take as a single command line argument a DIMACS file with a graph and find the smallest number of colors needed to color the graph. The application should output the number of colors required. The fastest solver gets [seven bonus points](#). Download and test with the following benchmark instances: <https://github.com/satlecture/kit2024/tree/main/exercises/coloring>

## 2 Sudoku Competition (Points 7)

Write an encoder for the generalized Sudoku puzzle. The generalized Sudoku puzzle of order  $n$  is an  $n^2 \times n^2$  grid, consisting of  $n^2$  sub-blocks of size  $n \times n$ , to be filled with numbers  $1, \dots, n^2$ , such that

- in each row each number occurs exactly once,
- in each column each number occurs exactly once, and
- in each sub-block each number occurs exactly once.

The well-known Sudoku problem<sup>1</sup> is the generalized Sudoku puzzle of order three. The best encoding (solving the most instances and fastest) will get a [bonus of seven points](#). Download and test with the following benchmark instances: <https://github.com/satlecture/kit2024/tree/main/exercises/sudoku>.

## 3 Pythagorean Triples (Points 6)

Find a coloring for the numbers  $1 \leq i \leq 1000$  such that no Pythagorean triple is monochromatic. Estimate the number of variables and clauses in the Pythagorean triples encoding from the lecture (as a function of  $n$ ).

## 4 Tseitin Encoding (Points 6)

Encode the following formula into CNF using the Tseitin Encoding. How would the formula look like the Plaisted-Greenbaum Encoding?

$$(\overline{x_1} \wedge (\overline{x_3} \iff x_2)) \vee ((x_3 \rightarrow \overline{x_4}) \wedge (x_1 \rightarrow (x_2 \wedge \overline{x_3})) \wedge (x_4))$$

<sup>1</sup><https://en.wikipedia.org/wiki/Sudoku>