FIT1045 Algorithmic Problem Solving – Workshop 7.

Objectives

The **objectives of this workshop** are:

- To implement and manipulate data structures in Python.
- To implement algorithms on these structures in Python.
- To investigate the effect of the choice of representation of a solution on the algorithm.

Important: Complete the questions from last week's workshop if you have not already.

Task 0: Prelab

When performing arithmetic on matrices, we often want to take the product of the diagonal. Using the following example write a program that calculates the product of the main diagonal of a matrix and subtracts the reverse diagonal.

$$f\begin{pmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{pmatrix} = a_0b_1c_2 - a_2b_1c_0$$

Task 1:

- Discuss some ways of representing a candidate solution to the N Queens problem.
- What are the advantages and disadvantages of each representation?
- For this workshop, you may choose any representation you like to solve Task 1

Write a function `is_solution(board, n)` which takes as input a representation of an $n \times n$ chess board with exactly n queens placed on it, and returns `True` if that board is a valid solution to the n-queens problem, and `False` otherwise. This task does not require you to take user input. You should, however, test your function on various boards. Please have those test cases stored as variables in your file so that you can show your demonstrator your tests.

Task 2

1. Write a function `bounded_lists(upper_bounds)` that accepts as argument a list of positive integers of length n and returns a list of all lists of length n consisting of non-negative integers with the following property: for a list `lst` it holds that for all indices `i`, `lst[i]` $\leq$ `upper_bound[i]`. Hint: adapt the enumeration algorithm from the lecture that enumerates bitlists in lexicographic order.
   For example, if the input list was `[1, 1, 2]`, then the output would be

```
[[0, 0, 0],
[0, 0, 1],
[0, 0, 2],
[0, 1, 0],
[0, 1, 1],
[0, 1, 2],
[1, 0, 0],
[1, 0, 1],
[1, 0, 2],
[1, 1, 0],
[1, 1, 1],
[1, 1, 2],]
```

2. Use the function from a) to write another function `brute_force_coin_exchange(amount, denominations)` that finds an optimal solution for a coin exchange problem input. Optimal meaning uses fewest total coins. The inputs to the function are the amount of money we wish to make, and a list of numbers representing the coin denominations. The output should be a list of how many of each coin we use. So the first element would be the number of times we use the first coin in `denominations`.
   Hint: What is an upper bound for the number of coins of a specific denomination that can be selected in a feasible solution?