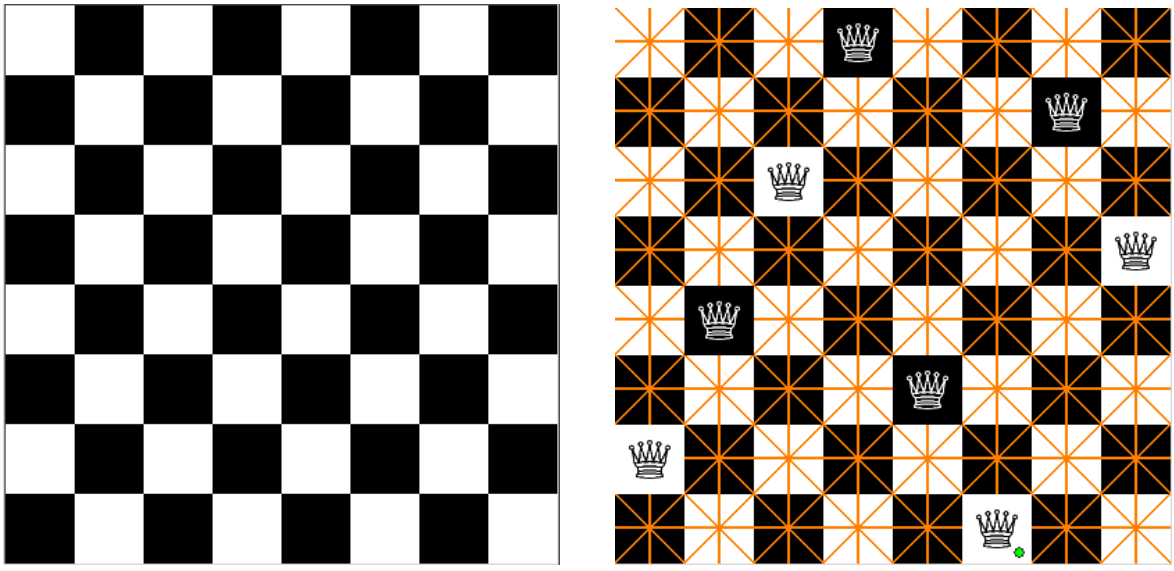
Lab 1: Search

The **eight queen's puzzle** is the problem of placing eight chess queens on an 8×8 chessboard such that none of them can capture the other using the standard chess queen's moves. The queens must be placed in such a way that no two queens attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens is an example of the more general ***n*-queens problem** of placing n queens on an $n \times n$ chessboard, where solutions exist only for $n = 1$ or $n \geq 4$.

The eight queen's puzzle can be solved by searching for a solution. Initial state is obviously an empty chess board (see figure 1(a)). Placing a queen on the board represents an action and the goal state is one where none of the queens' attacks any of the others. Note that every goal state is reached after exactly 8 actions. Formulating the eight queen's puzzle as a search problem is improved when one realizes that, in any solution, there must be exactly one queen in each of the columns (see the figure 1(b)). The possible actions can be restricted to placing a queen in the next column that does not yet contain a queen. This reduces the branching factor from (initially) 64 to 8. Note that one only needs to consider those rows in the next column that are not already attacked by a queen that was previously on the board. This is because the placing of further queens on the board can never remove the mutual attack and turn the configuration into a solution [1] and [2].

Use a programming language of your choice (preferably Python) and implement two types of search methods and compare their performance in searching the solution for 8 queens' problem.



a) Initial state b) Possible goal state

Figure 1. Eight queen's puzzle

Solve the 8-queens problem using backtracking algorithm:

- Try both depth-first (DFS) and breadth-first searches (BFS) solutions separately and compare them. Which one is better and why?
- Analyse time complexity of the search techniques.

- Does backtracking provide better solutions? If so, why? What about the time complexity of the backtracking?

We are aware that searching code on the internet in a programming language of your choice is not that difficult. Which is why we are providing some reference code to the eight queens problem: https://github.com/SadraSamadi/n_queens.

Alternatively:
https://nbviewer.org/github/dvatvani/dvatvani.github.io/blob/master/static/8-Queens/8_Queens_problem.ipynb

Additional reading:
https://en.wikipedia.org/wiki/Eight_queens_puzzle
https://medium.com/@paabrown/solving-the-n-queens-problem-6891e3d21b0

You could of course build your own code if you choose to do so. Note that the code is provided only as a reference so teaching staff are not responsible for the authenticity of the code and its functionality. We believe that is something you could figure out on your own. You will NOT pass the lab if you just run the program, produce some data but make little attempt to analyse it. Analysing and providing a thorough discussion on the different techniques is what you are expected to produce.

References

1. Eight queens' problem. http://www.aiai.ed.ac.uk/~gwickler/eightqueens.html
2. S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach, chapter 3. Prentice Hall, 2nd edition, 2003.