Lab01: N-queens problem

You are considering the N-queens problem, in which

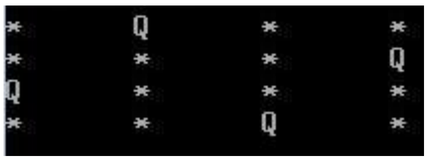
- The ***complete-state formulation*** represents a configuration, i.e., all the N queens are on the board. Furthermore, each queen is on a separate column.
- The successors of any state are *all possible configurations generated by moving a single queen* to another square in the same column.

You need to solve the above problem by implementing the following search algorithms.

- Uniform-cost search,
- Graph-search A* with MIN-CONFLICT heuristic: "the number of ATTACKING pairs of queens",
- Genetic algorithm with the objective function defined from the above heuristic.

Note that two queens attack each other if they are on the same column (which never happens due to the state representation), same row, or same diagonal.

Your program will accept the number of queens, N, and a positive integer indicating the algorithm (1: uniform-cost search, 2: A*, 3: genetic algorithm). Furthermore, it should have a simple GUI to show the output, as demonstrated below.



```
*       Q       *       *
*       *       *       Q
Q       *       *       *
*       *       Q       *
```

You then study the performance of those algorithms by measuring their **running times** and **consumed memory** when N differs and fill in the following table with your statistics.

	Running time (ms)			Memory (MB)		
Algorithms	N = 8	N = 100	N = 500	N = 8	N = 100	N = 500
UCS						
A*						
GA						

– Your computer should run nothing rather than the target application during the evaluation, and each statistical value should be an average of at least three runs to avoid bias.
– A large number of queens may cause the problem to be intractable, depending on the available computational resources. The reference configuration is the Google Colaboratory. For any scenario, write "Intractable" to the corresponding entry if you cannot solve it in an acceptable amount of time on your computer or Google Colab.

- Give meaningful observations and comments on the statistics.

This is an INDIVIDUAL assignment.
Your program should be in **Python**. Write down your report in a **PDF file**.
You can use supported data structure functions/libraries (e.g., queue, stack), yet **you must implement the search algorithms yourself**.
The user interface can be either console or graphical (you will not get extra credit for friendly UI), which is convenient enough for TA to check the validity of the output results.

Prepare a folder that includes the following subfolders
- SOURCE: a single ipynb file that includes all the requirements
- DOCUMENT: a PDF-formatted file that presents a check list of what you have/have not done and a brief description of main functions (so that the Lab Instructors do not miss any of your efforts)

Name the main folder following your Student ID and compress it in common format.