Programming Assignment #1: Problem Setup

Assigned: 30.10.2023 Due: 17.11.2023

1 Objective

In the upcoming assignments, you will model and implement the solution of a famous problem according to the algorithms that we learn in the class. The purpose of this assignment is to get you ready for the upcoming programming assignments by implementing the infrastructure of the problem.

2 Problem Description

N Queens Problem: "This is the problem of placing N chess queens on an NxN chessboard so that no queens threaten each other; thus, a solution requires that no two queens share the same row, column, or diagonal." (Wikipedia) The most common form of this problem is 8-Queens.

3 Specification

3.1 Starter Code

The starter code is available in AI_PA1_base.ipynb file. There are four cells in the given file:

1. The first cell is a markdown cell, in which you will enter your group members' details. In the readme part, give some information about your Python version, development environment, etc. You can also add some explanations such as instructions for running, exceptional cases, etc. to the instructor, if you think it is necessary.
2. The second cell is a code cell, which is the main part that you will implement. You will implement the NQueens class that is given as a template. Read the comments in the class definition and this document carefully and implement the empty functions as described.
3. The third cell is a markdown cell that you will not touch.
4. The last cell is a code cell which includes a testing template of NQueens class.

3.2 Representation of NQueens Problem

You must use the following representation for an NQueens problem. The state of the board will be kept as a string of length N (let's call it as state). We will assume that a single queen is placed on each column of the board. So, the state string will keep the row index for the queens on each column. See the examples below (Figure 1).

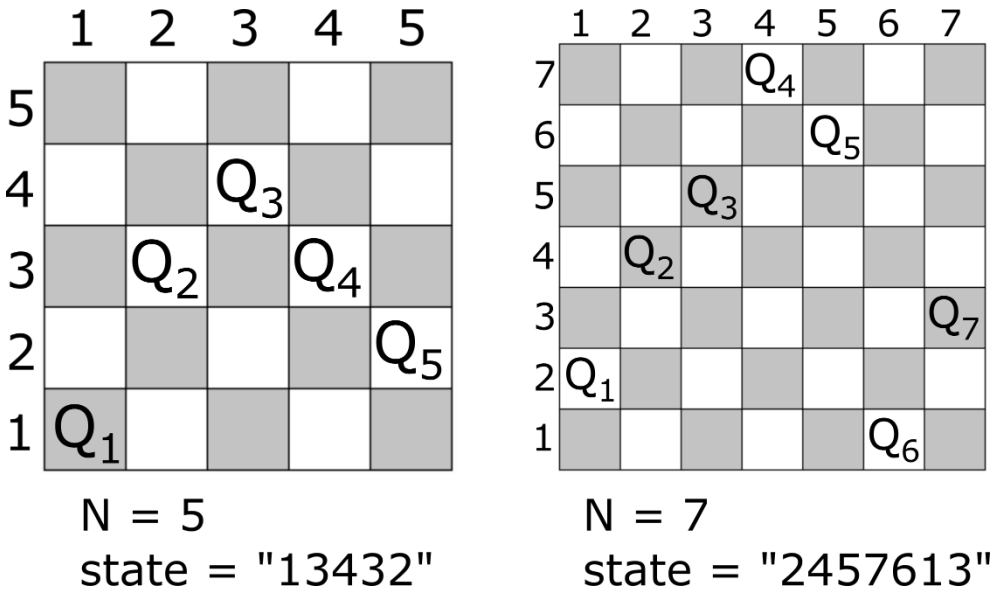

Figure 1: Examples for state representation. (Q_1 – Q_N correspond to the Queen in column 1 – N, respectively.)

3.3 Implementation Details for NQueens Class

- You don't need SimpleAI package for this assignment.
- Your task is to implement the empty functions. You are not supposed to solve the problem.
- Read the comments in the class definition carefully.
- Do not forget to delete *pass* statements when you implement the functions.
- There will be two instance attributes:
  - **N** (type: int): Determines the number of queens and board size
  - **state** (type: str): String representation of the board state (see previous section)
- You will implement six functions:
  1. ***__init__(self, N):*** This is the class constructor. It initializes the instance attributes. It takes one parameter *N* (in addition to obligatory *self*). Instance attribute *N* is initialized to the parameter. Instance attribute *state* is initialized by calling *_set_state()* member function.
  2. ***__str__(self):*** This function returns a string description of the current instance.
  3. ***_set_state(self):*** This function sets the instance attribute *state* by displaying a menu to the user. The user either enters the state manually or prompts the system to generate a random state (by calling *generate_random_state()* function). Check if the input state is a valid state (by calling *_is_valid()* function).
  4. ***generate_random_state(self):*** This function returns a valid (See Section 3.4), randomly generated state string.
  5. ***_is_valid(self, state):*** This is an internal function to check whether the given *state* is a valid state. It returns True/False. (See Section 6 for details)
  6. ***_count_attacking_pairs(self, state):*** This is the primary function of this class. It returns the number of attacking pairs of queens in the given state. (See Section 3.5 for details)

3.4 Invalid State Examples

- Case1: State string includes invalid characters (not digit). (e.g. state = “abc12”, “1xy 2”, …)
- Case2: The length of the state string is not equal to N. (e.g: N = 4, state = “12345”)
- Case3: State string includes numbers greater than N or less than 1. (e.g. N = 6, state = “12374 2” – N = 5, state = “01234”)

3.5 How to Count Attacking Pairs

According to the problem definition, a solution requires that no two queens share the same row, column, or diagonal. Our representation of the problem already restricts any two queens to be on the same column (See Section 3.2). Thus, you don’t have to check for columns. You should check the rows and diagonals in both directions. You may use *math* module for implementation. See Figure 2 for the calculation of number of attacking pairs.

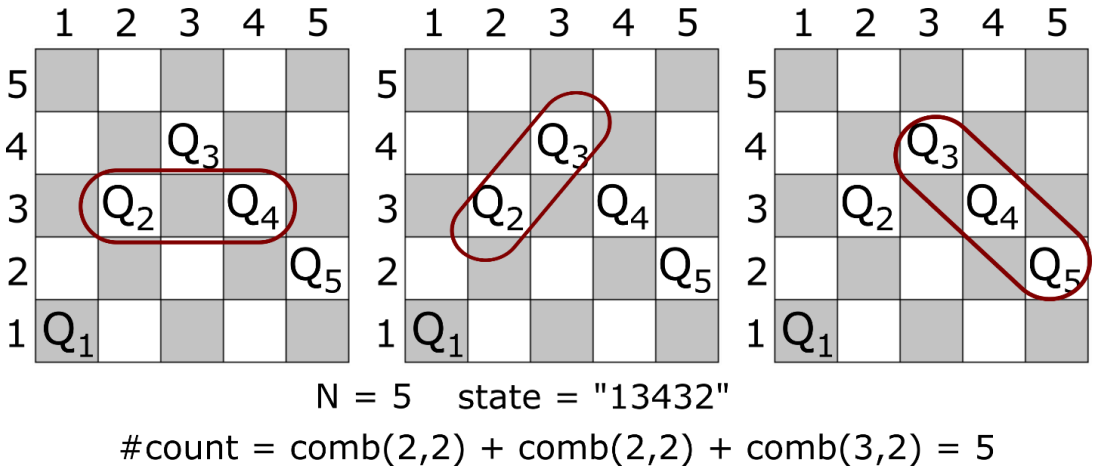

Figure 2: How to count the number of attacking pairs manually.

3.6 Sample Test Cases

In order to test your *_count_attacking_pairs()* function, you can use the test cases in the below table. Also, you may need the empty boards given in Appendix, while developing your algorithm.

Table 1: Sample test cases.

N	State	Number of attacking pairs
5	“13452”	4
5	“11143”	6
6	“666116”	7
6	“423414”	7
7	“7521161”	5
7	“3224447”	9
8	“68255482”	10
8	“82888315”	9

3.7 Sample Outputs

If you implement the NQueens class correctly, when you run the test code in Part 2 of the given .ipynb file, you will get similar results to the following:

```
problem = NQueens(6)

How do you want to set state?
1. Set state manually
2. Set state randomly
Enter selection: 1
enter state: sdgdfgfd
invalid state! try again
How do you want to set state?
1. Set state manually
2. Set state randomly
Enter selection: 1
enter state: 120345
invalid state! try again
How do you want to set state?
1. Set state manually
2. Set state randomly
Enter selection: 1
enter state: 213415

print(problem)

N: 6, state: 213415

print(problem.count_attacking_pairs())

5
```

```
problem = NQueens(5)

How do you want to set state?
1. Set state manually
2. Set state randomly
Enter selection: 2

print(problem)

N: 5, state: 14155

print(problem.count_attacking_pairs())

3
```

Figure 3: Sample runs of the program.

4 Submission

- This and the following assignments can be done individually or in teams of maximum 3. (Teams can be a mixture of normal and evening education students. You can change your team in different assignments, although it is not suggested.)
- You are free to use any Python development environment that supports Interactive Python Notebook (.ipynb). Some alternatives: Jupyter Notebook, JupyterLab, VS Code, …
- Rename the given AI_PA1_base.ipynb file to **AI_PA1_Surname1_Surname2_Surname3.ipynb** and submit using the Teams assignment module. Single submission for each team is sufficient.
- If you have further questions, you can send me an e-mail or send a message via MS Teams.

5 Academic Honesty Policy

You cannot borrow other's ideas or portions of codes, without giving proper citation. This can be an Internet source, AI chatbot, or a friend. Clearly indicate which part of your code/report/idea is borrowed and state its source. You cannot get all or most of your work from others'. Otherwise, you will be penalized.

6 Late Submission Policy

Deadline for homework submissions is **23:59** at the specified date. For each additional day, a **25% cut-off** will be applied.

7 Appendix