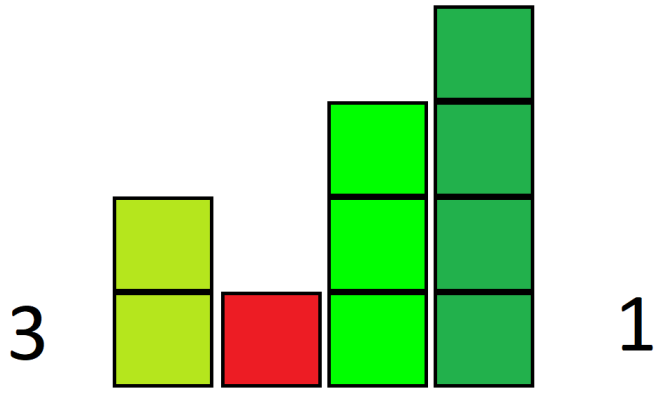
Pyramid Logic Puzzle

1. Description

This project was created to provide solutions for pyramids puzzle:

Consider a square 2D board, with a size of N. Pyramids, with a height = H:
$H \in [1, N] \cap \mathbb{N}$. Around the board, clues indicate how many pyramids are seen in a column or row. Pyramid heights in each column/row must be unique.



Visualization of pyramid visibility.

From the left side, three pyramids are seen - ‘1’ is lower than ‘2’, so it is hidden behind a higher pyramid.

From the right side, only one pyramid can be seen, as it has the height equal to the size of the board.

Board is initialized as a matrix of sets of possible values. To solve parts of the board, and reduce the number of possibilities in each cell, initial evaluation of clues is commenced.

- Clues with a value of N allow the row/column to be fully filled with subsequent numbers from 1 to N.
- Clues with a value of 1 allow the nearest cell to be filled with N.
- Normal clues reduce the possible values in certain cells by removing values which would prevent reaching specified visibility.
- Repetitions are removed if encountered.

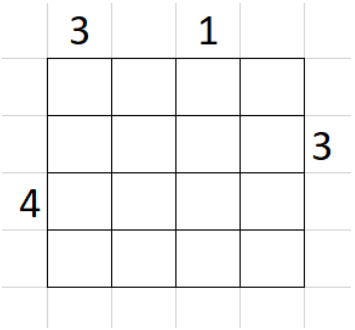
Next step is initialisation of an auxiliary board with zeros in place of unsolved values. Then, backtracking algorithm is used to find the solution on the new board.

2. Classes

1) Clues - class handling clues from input, preparing them to be passed to board
2) Board - class representing a board, produces solution for passed clues
3) SizeError - exception raised when number of clues is not correct
4) CannotSolveError - exception raised when no solution can be found
5) WrongDataError - exception raised when value of a clue is not correct
6) CluesContradictionError - exception raised when clues produce result that is not possible

3. User Manual

1) Clone the repository into a local directory
2) Run ‘ python3 -m pip install -r requirements.txt ‘.
   This will install pytest_mock and typing_extensions if they are not present.
3) Create a file, preferably .txt, containing the clues.



For clues shown on the example illustration,
file should look like this:

3 0 1 0
0 0 0 0
0 0 4 0
0 3 0 0

- first row describes clues seen from above
- second row describes clues seen from below
- third row describes clues seen from the left
- last row describes clues seen from the right

4) Run ‘ python3 pyramid_solver.py input ‘, where input is name of your clues file (if current directory does not contain pyramid_solver.py or input file, replace them with paths accordingly)
   Solved board will be displayed in the terminal.
   a) If you want to save the solution instead, add ‘--save output ‘ after the main command, where output is the name/path of the file.
   (Overwrites existing files, otherwise creates a new one)

4. Thoughts

Initially thought to be straightforward and non-problematic, the pyramid puzzle proved to be quite a challenge to complete. Translating human behavior during solving to an algorithm is something incredibly hard to achieve. While initial clue solving steps mimic real player strategies, reaching full solution in a similar manner is something I could not achieve elegantly (that is, without a great number of if statements).

Instead, I chose to implement a more brutal approach - backtracking algorithm, checking if placing a possible value on the board can lead to a solution. This method works exceptionally well for smaller boards (up to 5x5-sized), but becomes increasingly more resource-consuming the bigger the size. Because of that, program may find difficulties reaching solutions for boards 7x7 and above - causing unexpected crashes. Therefore, algorithm should be further optimized to allow for bigger boards to be solved.

Program requires a file to be specified - inputting clues directly through the terminal would confuse the user. Though implementing Graphical User Interface was briefly considered, I rejected the idea because of the varying board size - if the board would be too big, user would not be able to comfortably input the data.

All things considered, I believe this project is a great base for further development. All components are ready for improvements, and will prove useful in reaching optimal solution of solving the board quickly and in a stable process.