ACIT4420 project tasks

Project no. 1 - Website crawler

Create a python project that is able to download websites and capture sensitive data on the site. The program has to accept the following parameters:

- The start URL of the web crawling
- The depth of the crawling which means how many jumps has to be considered when downloading the website
- User defined regular expressions to find sensitive data

The program should provide the following features:

- Download the website from the provided URL, identify links inside the source code and download all website subpages that are linked until the maximum number of jumps are not reached.
- Identify email addresses and phone numbers and create a list of the captured values
- Identify comments inside the source code and make a list of them indicating the file name and line number of the comment
- Identify special data using the user provided regular expression
- Create a list of the most common words used on the crawled websites

Project no. 2 – Simplified log analyzer

Create a python project that is able to analyze and project different type of logs (e.g. apache web log)

The program has to accept the following parameters as input:

- Regular expressions for identifying different values in a log. Each log type must have a description file with the fields and regular expressions.
- Time frames to analyze a specific interval
- Input values for events to project the number of events that are occurred (e.g. how many 404 web answers were registered in a specific time frame, how many requests were initiated from a given ip)

The program should provide the following features:

- Use two different log file description
- Asks for event type from the user
- Asks for timeframe from the user
- Project the number of findings using the user provided filter

Project no. 3 – Labirinth solver

Create a python project that is able to design a 2D labyrinth and find the exit from any starting point. The program has to accept the following parameters as input:

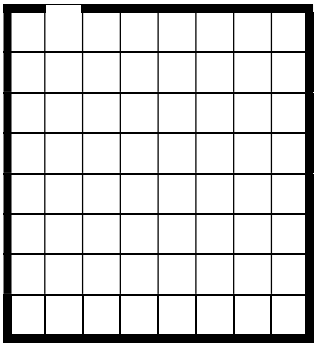
- the size of the labyrinth in both directions
- the starting point for the exit finding

The program should provide the following features:

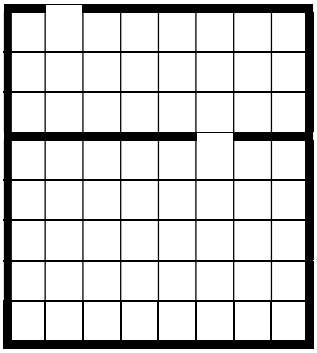
- Create a random arbitrary size labyrinth
- Find the way out using brute-force (trying all combinations) and/or using Q learning.

For the labyrinth generation, you can use e.g. the following algorithm:

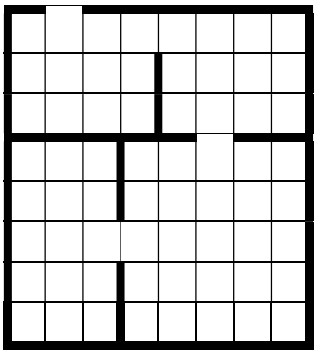
1. Set the size of the labyrinth and choose a random exit point.



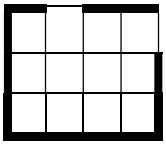
2. Split the labyrinth with a horizontal line in a random place with a hole (size=1 unit) in a random place.

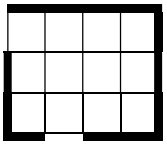


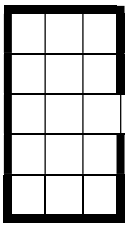
3. Split the labyrinth vertically in random places with holes.

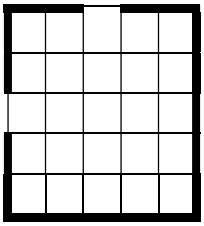


4. Recursively repeat the process for the new rectangles. If one size of the rectangle is 1 unit then stop the recursive action for that specific rectangle. Repeat the process until all rectangles have a one unit size.









Project no.4 – Chess sudoku solver

Create a python project that can solve regular sudokus, but also sudokus with additional constrains of knight and king moves. Also known as anti-knight or anti-king sudokus. See figure 1, 2 and 3 for explanation. The program must accept the following parameters as input:

- The two-dimensional 9x9 array of a sudoku
- A true/false flag if knights move constrain applies
- A true/false flag if the kings move constrain applies

The program should provide the following features:

- If solvable, it solves the sudoku and prints out the solution (hint: constraint satisfaction and depth first recursive search)
- Tell you if the sudoku is solvable, non-solvable or unknown
- Tell if a sudoku is correctly solved

You may assume the sudoku has only one unique solution.

Puzzles can be found at:
http://www.tectonicpuzzel.eu/anti-knight-sudoku-puzzle.html

http://www.tectonicpuzzel.eu/anti-king-sudoku-puzzle.html

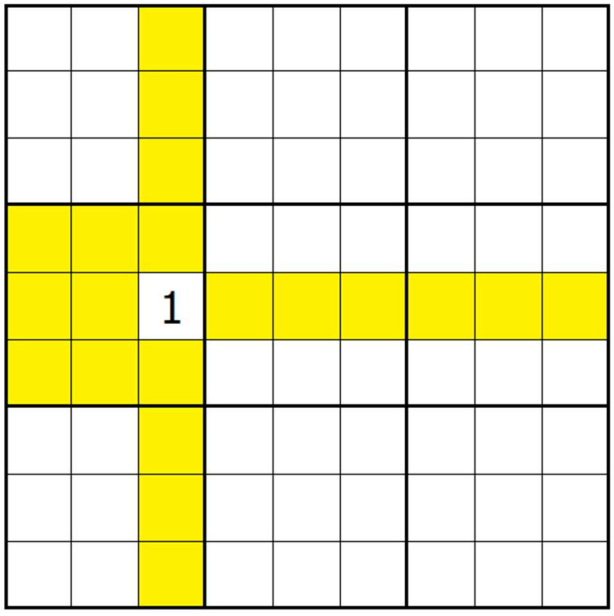


Figure 1 normal sudoku constraints marked in yellow, number 1 cannot appear in any of the yellow squares.

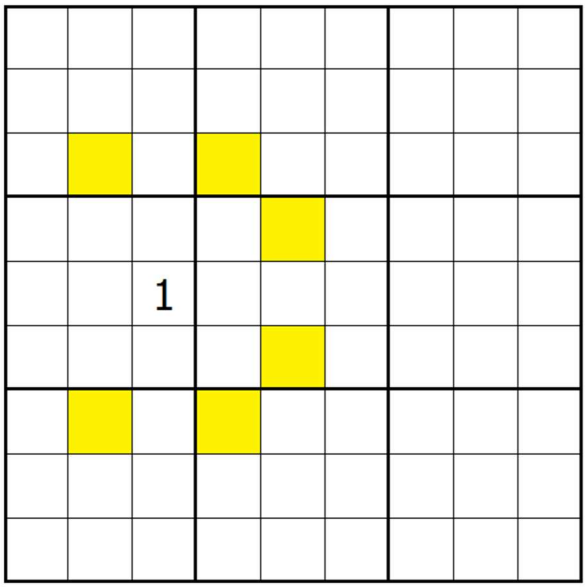


Figure 2 Additional knights move constraint marked in yellow. In this constraint no digit can appear in a cell that is a chess knights move away from the same digit.

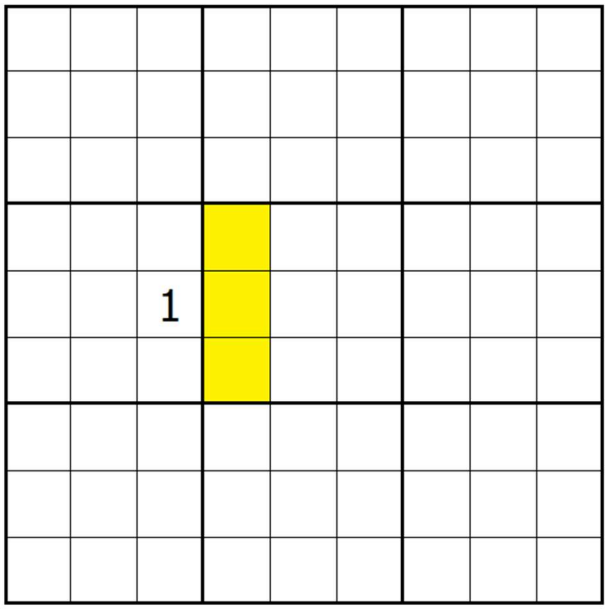


Figure 3 Additional kings move constraint marked in yellow. In this constrain no digit can appear in a cell that is a chess kings move away from the same digit