

# Introduction to Artificial Intelligence

## Exercise 1: Search

Daniel Marczak

Summer 2023

### 1 Task

#### Variant 1

Write a program that solves a maze using two search algorithms: breadth-first search (BFS) and depth-first search (DFS). The maze is a 2D grid with empty space, walls, a start, and an end position. The objective is to find a path from start to end position.

The maze should be loaded from file. A step-by-step visualization of the algorithm is required. It can be done in the console and an interface may be as simple as possible (but of course it does not have to). Example solution: [https://angeluriot.com/maze\\_solver/](https://angeluriot.com/maze_solver/).

In the report discuss the differences in the obtained results between BFS and DFS. Come up with examples of the input data (a maze) that shows the strengths and weaknesses of both algorithms.

#### Variant 2

Write a program that solves a maze using greedy best-first search algorithm. The maze is a 2D grid with empty space, walls, a start, and an end position. The objective is to find a path from start to end position.

The maze should be loaded from file. A step-by-step visualization of the algorithm is required. It can be done in the console and an interface may be as simple as possible (but of course it does not have to). Example solution: [https://angeluriot.com/maze\\_solver/](https://angeluriot.com/maze_solver/).

Test multiple heuristics (at least two)  $h(n)$  and discuss the differences between the obtained results.

#### Variant 3

Write a program that solves a maze using A\* algorithm. The maze is a 2D grid with empty space, walls, a start, and an end position. The objective is to find a path from start to end position.

The maze should be loaded from file. A step-by-step visualization of the algorithm is required. It can be done in the console and an interface may be as simple as possible (but of course it does not have to). Example solution: [https://angeluriot.com/maze\\_solver/](https://angeluriot.com/maze_solver/).

Test multiple heuristics (at least two)  $h(n)$  and discuss the differences between the obtained results.

## Variant 4

Write a program that minimizes a function  $f(\mathbf{x}) : R^3 \rightarrow R$  using straight gradient descent method. The program should return the found solution  $\mathbf{x}^*$  and the value of the function  $f(\mathbf{x}^*)$ .

$$f(\mathbf{x}) = |a|(1 - x_1)^2 + |b|(x_2 - x_1^2)^2 + |c|(x_3 - x_2^2)^2$$

Parameters  $a, b, c$  are specified by the user in the runtime.

It should be possible to define the starting point for the optimization procedure in two ways: either the user can directly set the initial vector  $\mathbf{x}_{init} \in R^3$ , or its elements get generated by drawing numbers from a uniform distribution defined for the range  $[-10, 10]$ .

Test and discuss the impact of the hyperparameters (learning rate, maximum number of iterations) on the quality of the results and computation time.

## Variant 5

Write a program that minimizes a function  $f(\mathbf{x}) : R^2 \rightarrow R$  using Newton's method. The program should return the found solution  $\mathbf{x}^*$  and the value of the function  $f(\mathbf{x}^*)$ .

$$f(\mathbf{x}) = |a|(x_1 + 2x_2 - 7)^2 + |b|(2x_1 + x_2 - 5)^2$$

Parameters  $a, b$  are specified by the user in the runtime.

It should be possible to define the starting point for the optimization procedure in two ways: either the user can directly set the initial vector  $\mathbf{x}_{init} \in R^3$ , or its elements get generated by drawing numbers from a uniform distribution defined for the range  $[-10, 10]$ .

Test and discuss the impact of the hyperparameters (learning rate, maximum number of iterations) on the quality of the results and computation time.

## 2 Technical details

- The solution must be implemented in Python.
- Please ensure that your code adheres to basic standards of lean coding in accordance to PEP8. Additionally, it should contain comments in the crucial parts to help with readability and understanding.

- The clear instruction how to run and test the code should be included.
- The submission of the final report is mandatory, and the task will not be accepted without it.

### 3 Handing-in guidelines

You should submit the source code of your solution and final report via Teams not later than:

- 2023.03.26 EoD for the Monday group
- 2023.03.28 EoD for the Wednesday group
- 2023.03.30 EoD for the Friday group

Programs delivered after the deadline will not be assessed. The on-line assessment will take place during your labs. In case of questions, please contact me via Teams.

### 4 Assessment Criteria

You can get  $[0, 5]$  points for the lab. The following criteria will be used to evaluate your work:

- Proper implementation of the algorithm: 2 points
- Clean and well-documented code, with clear explanations and well-implemented logic: 1 point
- Final report including results, clear explanation of the programmed solutions and reflections on what was done well and what could be improved: 2 points

**In case of any questions contact me via MS Teams**