Introduction to Artificial Intelligence

Exercise 1: Search

Valeriya Khan

Summer 2024

1 Task

Variant 1

Using the template from the files folder (lab1_v1.py), 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 spaces, walls, a start, and an end position. The objective is to return a number of steps from start to end position, and everything needed to visualize the search procedure.
You can import libraries and add auxiliary functions if needed. Come up with different test cases and corner cases. The code will be run against different test cases, so make sure it works properly.
A step-by-step visualization of the algorithm is required. It can be done in the console and the interface may be as simple as possible. Example solution: https://angeluriot.com/maze_solver/.
In the report discuss the differences in the obtained results between BFS and DFS. Come up with test cases (a maze) that shows the strengths and weaknesses of both algorithms. Explain your choice of test cases in the report.

Variant 2

Using the template from the files folder (lab1_v2.py), write a program that solves a maze using greedy best-first search algorithm. The maze is a 2D grid with empty spaces, walls, a start, and an end position. The objective is to return a number of steps from start to end position, and everything needed to visualize the search procedure. Implement at least two heuristics h(n).
You can import libraries and add auxiliary functions if needed. Come up with different test cases and corner cases. The code will be run against different test cases, so make sure it works properly.
A step-by-step visualization of the algorithm is required. It can be done in the console and the interface may be as simple as possible. Example solution: https://angeluriot.com/maze_solver/.

In the report discuss the differences of results obtained by different heuristics. Come up with test cases that show the strengths and weaknesses of the algorithm and explain your choice of test cases in the report.

Variant 3

Using the template from the files folder (lab1_v3.py), write a program that solves a maze using A* algorithm. The maze is a 2D grid with empty spaces, walls, a start, and an end position. The objective is to return a number of steps from start to end position, and everything needed to visualize the search procedure. Implement at least two heuristics h(n). You can import libraries and add auxiliary functions if needed.
Come up with different test cases and corner cases. The code will be run against different test cases, so make sure it works properly.
A step-by-step visualization of the algorithm is required. It can be done in the console and the interface may be as simple as possible. Example solution: https://angeluriot.com/maze_solver/.
In the report discuss the differences of results obtained by different heuristics and for different test cases. Explain the choice of the test cases and corner cases in the report.

Variant 4

Using the template from the files folder (lab1_v4.py), write a program that minimizes a function using Newton's method algorithm. The program should return the found solution x* and the number of iterations.
You can import libraries and add auxiliary functions if needed. Visualize the objective function for x in range [-5,5] and y in range [-5,5]. Using visualization, choose different initial points to get as many different results in the given coordinate ranges as possible.
In the report, discuss the impact of different initial vectors and step size parameters on the results.

Variant 5

Using the template from the files folder (lab1_v5.py), write a program that minimizes a function using straight gradient descent algorithm. The program should return the found solution x* and the number of iterations.
You can import libraries and add auxiliary functions if needed. Visualize the objective function for x in range [-5,5] and y in range [-5,5]. Using visualization, choose different initial points to get as many different results in the given coordinate ranges as possible.

In the report, discuss the impact of different initial vectors and learning rates on the results.

2 Technical details

- The submission should include the python file based on the provided template and the report in the format of .pdf
- Please ensure that your code adheres to basic standards of lean coding in accordance with PEP8. Additionally, it should contain comments on the crucial parts to help with readability and understanding.
- The inputs to the code should be provided inside the python code or from the file. In case of providing inputs from the file, the python code should include reading from the file part and provide instructions in the comments. In addition, the file with the inputs should be included in the submission.
- All the test cases used should be in the submission (for variants from 1 to 3). The code without proper test cases may get less points.

3 Submission guidelines

You should submit all the files via Teams not later than:

- 2024.03.17 until 6 p.m. for Monday group
- 2024.03.19 until 6 p.m. for Wednesday group
- 2024.03.21 until 6 p.m. for Friday group

The on-line assessment will take place during your labs. In case of questions, please contact me via Teams.

Submit to: valeriya.khan.dokt@pw.edu.pl to email or in teams chat
Names of all files should include be lab1_cg{class_group_number}_g{group_number}_{surname1}_{surname2}.(py,pdf,zip, etc)
E.g. lab1_cg101_g3_Smith_Jankowski.py

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: 1 point
- Final report including clear explanation of the programmed solutions and discussion of the results: 2 points
- Explaining the solution and answering questions during the online assessment: 2 points