Path Planning Assessment

This is a 2D path-planning assessment. You are required implement and test a path planning algorithm in an environment with obstacles.

Assessment Overview

Problem Statement:

You are required to find a collision-free path from the start node to the goal node, avoiding circular obstacles. This is a continuous space problem (There is no grid or predefined graph nodes to choose between), meaning you may choose any point in the continuous space to represent a node along your path. You are bounded by the square starting from (-0.55,0.55) to (0.55,-0.55)

What should you do?

1. **Conduct your OWN research to find a suitable algorithm:**
    - Try finding path-planning algorithms that are designed for this type of problem.
    - As long as your algorithm works, any refinements, optimizations, and improvements to the algorithm (without loss of generality) are welcome and may grant you a BONUS. For example: using a better data structure, using time-efficient algorithms, or implementing a more advanced version of the chosen algorithm.
    - While this is not a shortest-path problem. You are not allowed to exploit workarounds where the planned path isn't even converging towards the shortest one.
2. **Algorithm Implementation (C++):**
    - Implement your algorithm in `algorithm.h` (function `YourChosenAlgorithm`).
    - Read obstacles and nodes from CSV files.
    - Generate a path, new nodes, and edges.
    - Write results back to CSV files using provided I/O functions.

File Structure

- `algorithm.cpp` : Entry point.
- `algorithm.h` : Main algorithm logic.
- `IOHandler.h` : CSV reading/writing utilities.
- `CSV/` : Folder containing all input/output CSV files.
- `visualiser.py` : Python visualization tool.

Deliverables

- `CSV/path.csv` : Sequence of node IDs representing the planned path. (Example: `1,3,5,9,7,4,11,2` . Must end with 2 which is the goal point)
- `CSV/edges.csv` : List of edges (pairs of node IDs) representing connections in your graph/tree.
- `CSV/nodes.csv` : Appended with any new nodes generated by your algorithm.
- A short (1-2 mins) **screen**-recording or a video (presentation slides or whatever you prefer) explaining your algorithm and most importantly why you chose it.
- A 1-2 page report documenting your research and process.

Submission (2 Options):

1. Send it back

1. Zip your workspace folder.
2. Rename it to "FIRSTNAME_LASTNAME_GRADUATIONYEAR.zip"
3. Submit the zipped folder.

2. Github Repository

1. Create a **public** repository from the assessment workspace.
2. Submit the link to the **public** repository.

Important notes

- Please do not hesitate to reach out if you have any questions.
- The visualiser is provided for you to test your algorithm and see your output. To be able use it you need:
    - Python3
    - Pygame (Python package): Install using the following command `pip install pygame`
    - You may need to search the internet for how to set this up in a virtual environment (venv), if you are lost.
- You are not required to use the visualiser. If you find installing Python and/or Pygame to be difficult you may devise your own way of visualising your outputs but you need to maintain the format and include images of your outputs visualised.
- Don't change the workspace structure otherwise some parts of the assessment may break.
- Do your best and send us your results. Even if you weren't able to complete the task successfully, you may still be selected.
- The assessment is made for a specific type of path-planning algorithms. They are not many so please do your research on your own.