Coding challenge explanation

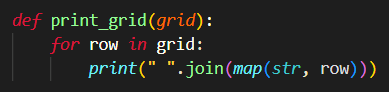
# Task 2

## Overall:

This Python script is an implementation of the A\* (A-star) pathfinding algorithm. It reads input data from files in a directory, performs the A\* algorithm to find the shortest path from a start point to a goal point on a grid, and writes the results to output files. It demonstrates the use of the A\* algorithm in a practical application. The A\* algorithm is widely used in various domains for pathfinding and optimization problems. It has applications in robotics, video games, route planning, logistics, and artificial intelligence.

1. ***Importing necessary libraries:*** The script begins by importing necessary libraries such as numpy, pandas, matplotlib, heapq, time, and os.
2. ***Defining helper functions:*** The print\_grid, heuristic, astar, is\_valid, and getNeighbours functions are defined. These are used in the main part of the script to perform the A\* algorithm.
   1. print\_grid(grid): This function prints the grid in a readable format.
   2. heuristic(node, goal): This function calculates the heuristic value for a given node based on the goal node. It uses the Manhattan distance (sum of the absolute values of the differences in the grid coordinates) as the heuristic.
   3. astar(grid, start, goal): This is the main function that implements the A\* algorithm. It takes in a grid and start and goal coordinates, and returns the shortest path and its cost.
   4. is\_valid(x, y): This function checks if a given coordinate is valid (i.e., within the grid and not an obstacle).
   5. getNeighbours(current): This function returns the valid neighbours of a given node.
3. ***Reading input files:*** The script reads input files from a directory. Each file contains information about a grid (size, start and goal points, obstacles, etc.). The script processes each file, extracts the necessary information, and constructs the grid.
4. ***Performing the A\* algorithm:*** For each input file, the script performs the A\* algorithm on the corresponding grid. It finds the shortest path from the start point to the goal point, if one exists.
5. ***Writing output files:*** The script writes the results to output files. Each output file corresponds to an input file and contains the shortest path and its cost. If no path is found, the output file contains “0”.

## Break down of each function:



***print\_grid(grid):*** This function takes a 2D list (grid) as input and prints it row by row. It converts each element of the row to a string and joins them with a space in between. This is a helper function to visualize the grid.



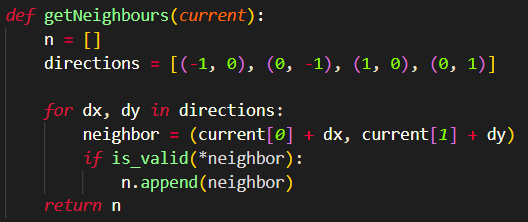
***heuristic(node, goal):*** This function calculates and returns the Manhattan distance between the current node and the goal node. The Manhattan distance is the sum of the absolute differences in the horizontal and vertical coordinates. This is used as the heuristic in the A\* algorithm.



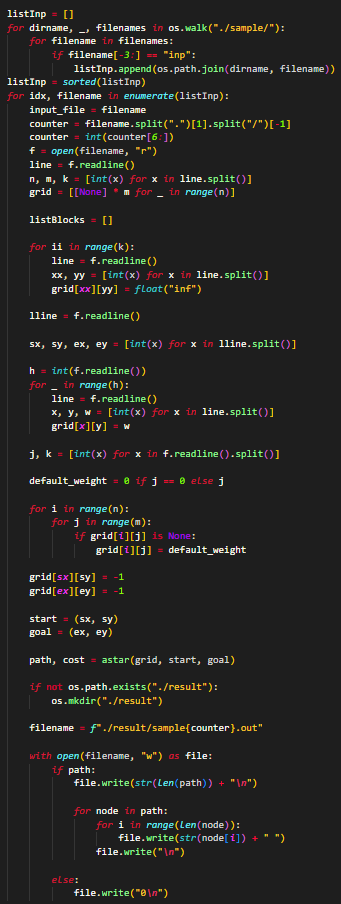
***astar(grid, start, goal):*** This is the main function implementing the A\* algorithm. It takes a grid and start and goal coordinates as input. It first initializes the open and closed sets, and the G and F dictionaries to keep track of the costs. Then it enters a loop that continues until there are no more nodes to explore (i.e., the open set is empty). In each iteration, it finds the node in the open set with the lowest F score, explores its neighbors, and updates their G and F scores. If it reaches the goal node, it constructs the path by backtracking from the goal to the start using the cameFrom dictionary.



***is\_valid(x, y):*** This is a helper function used in the A\* algorithm to check if a grid cell is valid for moving into. A cell is valid if it is within the grid boundaries and is not an obstacle (i.e., its value is not infinity).



***getNeighbours(current):*** This is another helper function used in the A\* algorithm. It returns a list of the valid neighbors of the current node. It checks all four directions (up, down, left, right) and adds the coordinates to the list if they are valid.



The rest of the code outside the functions is the main script that reads the input files, calls the astar function for each one, and writes the results to the output files. It also handles creating the output directory if it doesn’t exist.

The input and output file paths are constructed based on the file names in the input directory and a counter variable. The grid for each input file is constructed by reading the file line by line and setting the cell values according to the file contents.

The start and goal coordinates and the default cell weight are also read from the file. After the A\* algorithm is run, the path and its cost are written to the output file. If no path was found, “0” is written to the file.