AI Assignment 1 (2024)

Q. Imagine a search problem, where agent Blue (B) wants to reach the green zone in Baghdad, Iraq. The agent B is currently in a random position on a NxN grid, where N is odd. The car in which B is traveling can move in left (cost 1), right (cost 8), up (cost 1), or down (cost 10) directions. Also if B's current move (say left) is the same as the previous move (say left) then B will incur an additional cost of 5 (penalty). Further, some of the roads are blocked with cemented barriers. Solve using A* search. **You will have to submit a python file (submission_rollno.py)** containing 3 functions: (1) nextPartialPlanAStar(fringe) which should return the next partial plan from the fringe to be expanded (2) heuristicFn(state, goal) which should return the heuristic function value of a node (3) addToFringe(fringe, new_entry) which adds new states to the fringe. It can also contain any import statements, additional functions, global variables that may be needed. The base code has been provided, which uses these functions to perform A* search and displays the result. **Your objective is to build a heuristic function that performs the best by expanding the minimum number of nodes while finding the lowest cost path.** Please note that you have to use the **graph based version of the A* search** algorithm. **Also note that the heuristic function cannot be another search applied to the given problem (will be given 0 marks).**

Deliverables: Make a single report_rollno.pdf containing the screenshot of the resulting lowest cost path (in yellow) on the User Interface identified by your A* algorithm. The UI mentions the cost of lowest cost path and the number of expanded nodes. These should be clearly visible in the screenshot. **Submit the report_rollno.pdf and the submission_rollno.py python file, separately on google classroom.**

Note: Cite all the resources in the report. Not following any of the instructions will result in an appropriate penalty. Plagiarism of any kind will result in heavy penalty.

Evaluation: Your submitted code will be evaluated on different grids and barrier arrangements.

Important: You **should not edit base.py** since while evaluating the code, we will use the standard version of base.py that is shared with you.

Marks Distribution: Code + Viva = 20, Leaderboard based Marks = 10
Leaderboard based Marks Calculation: 2 + (MaxRank-Your Rank+1)*8/(MaxRank)
If more than 20 students achieve the same best performance, then the maximum marks from the leaderboard will be reduced to 8.

ExampleScreenshot