**CMPT 310. Fall 2024, Assignment #1**

**Introduction:**

The goal in this project is to practice design and implementation of the common search algorithms used in planning for intelligent agents in a fully visible environment. The environment is a cleaning robot which moved from room to room in a grid and clean the rooms/cells if dirty. Dirty cells are specified with grey color. There are wall cells which restrict the free movement, and the robot must find its ways around the wall blocks. Each move from a cell to its neighboring cell costs 1.

The robot finds its way around the grid by pre-planning its path to the next dirty room. For the planning the robot has access to 3 uninformed (DFS, BFS, UCS) and 2 informed (Greedy, A\*) search algorithms. There is also a reflex agent style planning in which robot moves around in a random move, in no neighboring dirty room exists, otherwise moves to the dirty room and clean.

For each path to the next dirty room, the ***explored*** and *path* cells are displayer with different color. There are 2 counters at the top of the GUI that display the accumulated *explored count* and the total *path count* so far.

This assignment is provided in the form of a shell module, and you are going to fill in the specified missing parts. These parts currently have print messages saying: "For students to implement". Once you have done that part, remove the printed message. The GUI and mechanics of the application should work fine as it is, and the above functionalities work only after you implement the corresponding algorithm.

The scripts in this project contain lots of comments and instruction. Read them carefully. They are part of the assignment description.

The way implementation works is that the *path* and *explored* list are computed when you select one of the search algorithms from the menu and click on **run** button.

The agent starts from the middle of the grid. The next search is done for the next dirty room and so on, till all grid is clean.

**Cost**:

The basic *cost*, ***Step*** count, is the number of step (moving from a cell to a neighboring cell). There is a cost dropdown menu to choose other *costing* options. Available options are: ***StepTurn***, ***StayLeft***, ***StayUp***. The *TurnCost* charges extra for each turn using a formula described later bellow. StayLeft/StayUp options are to be designed such that favor cleaning the dirty cells on the left/top half of the grid first.

Turning should costs 3 units for each 90' turn of the agent. For example, for a 180 turn, the cost should be 6. This results in agent preferring the path with a smaller number of turns.

**Heuristic:**

For heuristic we have 2 choices: Manhattan and Euclid distance.

**Rubric:**

1. Each algorithm BFS, DFS, UCS, Greedy, and A\* worths **12** points.
2. Each cost function implementation is worth **10** points. The cost functions are chosen from the middle dropdown menu in the app. The basic cost function is **Step** count which is already implemented in path\_cost() function. **StepTurn** cost function is to incorporate rotation cost per each 90’. It should cost 3 units for each 90’ turns. The third cost option is **StayLeft**. It should be designed such that encourages the agent to clean the left half of the grid cells first. You must come up with a simple function formula which embodies such tendency. Similarly, **StayUp** should cause the agent to first clean the rooms in top half of the grid. For both of the last 2 cost function we only are looking for the tendency and not matching any exact cost numbers.
3. Heuristic implementation and analysis worth **10** points.

**How to Test**

You should use PyCharm to load the project and test your solutions. Once you load the folder in PyCharm, at the top of the IDE, it gives you various run configurations to test all the required search cases:

*BFS, DFS, UCS, A\*\_Euclid, A\*\_Manhattan, Greedy\_Euclid, Greedy\_Manhattan*.

The last 4 correspond to informed searches. There is also “*corner*” version of the informed searches. In these cases, there are 4 dirty cells in the 4 corners of the grid. The corner configurations are just for testing and won’t be used for grading.

**Commandline Arguments:**

You can also run the script using command line format bellow, in a python console in the same folder as the project.

The agent can be launched from command line as following:

>python VacuumSearch.py -s searchAlgorithm -c costFunction -r heuristic

Look at the function readCommand() in vacuum\_search.py for detail of the above commandline options.

PyCharm run configurations are to make it easier to test your solution. They are using the command line options to pre-set the various app’s options.

**-For 1) in the rubric**, we will be running the corresponding configuration listed in the bellow table to test your implementation and verify that it produces the numbers given (with +/- 2)

**-For 2) in the rubric**, we test your *StepTurn* implementation by checking the numbers given in the bellow table’s second half are met by +/-2. For *StayUp* and *StayLeft* we only check the behavior for correctness, meaning when *StayLeft* is chosen, the left dirty cells must be cleaned before the right ones, but the order does not matter.

-**For 3) in the rubric**, the rows 5 and 6 in test table bellow for A\* are used for verifying your solution. Matching the numbers here counts for ***5***points. The other **5** points come from your explanation of difference of path Count and explored count for the 2 A\* heuristic test cases. For this part you should attach a text file named **analysis.txt**. Here you should explain in a few paragraphs how you justify each of the difference above.

Each algorithm has total Path Count and total Explored Count numbers. Your implementation must meet these number within +/- 2.

The numbers are the followings:

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Path\_Count** | **Explored\_Count** | **Cost** |
| BFS | 59 | 478 | Step |
| DFS | 350 | 647 | Step |
| UCS | 59 | 545 | Step |
| Greedy\_Manhattan | 78 | 113 | Step |
| A\*\_Manhattan | 59 | 154 | Step |
| A\*\_Euclid | 72 | 101 | Step |
| BFS | 59 | 478 | StepTurn |
| DFS | 350 | 647 | StepTurn |
| UCS | 59 | 607 | StepTurn |
| Greedy\_Euclid | 76 | 100 | StepTurn |
| A\*\_Euclid | 72 | 103 | StepTurn |

**Important**: You should not modify any code outside of the area specified. It makes the grading difficult.

**How to Hand in:**

To hand in you must put the following 4 files in a folder, zipped it and upload it to Canvas:

1-agents.py

2-search.py

3-vacuum\_search.py

4-Analysis report for part 3 explained above (for effect of Heuristic on search as explained above). This should be only 1 full page at most.