DS2020– Artificial Intelligence

Lab 1

Due on 31/1/2025 11.59pm

Instructions: Upload to your moodle account one zip file containing the following. You are expected to follow the honor code of the course while doing this homework. **This lab should be completed individually.** Rename the python file using your rollnumber before submitting it to Moodle.

Yantra Hunt

In this assignment you will be experimenting with different AI search techniques that we discussed in the class on a yantra hunt problem for a single agent. This is a strategic grid-based puzzle game where the player must navigate through a labyrinth to collect a sequence of yantras in the correct order while avoiding traps and barriers. The location of the next yantra is revealed only after the previous one is collected, and after uncovering the final yantra, the exit location is revealed. The task is to implement search techniques to successfully complete yantra hunt (if a solution exists).

An example grid world is illustrated below

P . # . Y2

. # . . .

. . Y1 . .

. . # .

. . . . E

In this example, the player starts at position **P** and must collect **Y1**, after which the location of the next yantra is revealed. After collecting the final yantra, the exit **E** location is revealed.

Grid Representation
The grid consists of the following elements:

- **P**: Player's starting position.

- **Y1, Y2, ...**: Yantras to be collected sequentially (only the first yantra is known initially).

- **#**: Walls that cannot be crossed.

- **T**: Traps to be avoided by the player

- **.**: Walkable empty spaces.

- **?**: Hidden locations (yantras or exit to be revealed).

- **E**: Exit point (revealed after collecting all yantras).

The accompanying python file includes the starter code the lab.

The constructor of the `YantraCollector` Class processes the input grid and stores the details using appropriate variables. The constructor also automatically creates the sequence in which the yantras have to be collected. Helper functions for finding the positions, the yantras, revealing the location of the next yantra or exit are also provided. **Do not have to edit any of these functions**.

Your objective is to implement/complete the following functions.

- `get_neighbors(self, position)`: the child generator function that returns the list of adjacent positions reachable from the current position. The reachability of the adjacent positions are checked in the North, East, South and West order.
- `goal_test(self, position)`: returns `True i`f the goal condition is satisfied in the current position, else `False`.
- `bfs(self, start, goal)`: that implements the breadth first search strategy to reach the goal position from the start position. **You are not allowed to use any inbuild Python data structure except for List to maintain the frontier and explored lists. Always select the element at index 0 of the frontier list for expansion; accordingly implement the necessary functionality to manage the frontier list for performing BFS.** The function should return the determined path from the start position to the goal position, if one exists or return none, along with the *number* of nodes in the frontier list and the explored list at the time of exiting the function.
- `dfs(self, start, goal)`: that implements the depth first search strategy to reach the goal position from the start position. **You are not allowed to use any inbuild Python data structure except for List for maintain the frontier and explored lists. Always select the element at index 0 of the frontier list for expansion; accordingly implement the necessary functionality to manage the frontier list for performing DFS.** The function should return the determined path from the start position to the goal position, if one exists or return none. along with the number of nodes in the frontier list and the explored list at the time of exiting the function.
- `solve(self, strategy)`: complete the function to call bfs and dfs functions appropriately and consecutively for completing the yantra hunt. Ensure to combine the path from the different consecutive calls, as well the total number of nodes in the frontier and explored list for the entire search. The function should return the total number of nodes in the frontier and explored list, as well the full path for completing the yantra hunt.

Your code should work with any grid size, any number of yantras, and arbitrary yantra pickup sequence.