CSC 455+555 SoCDAI

P2 Phase-2

1. Problem Description

In Phase-1, you interpreted the human's intentions for reaching a gem based on (incomplete) instructions and the configuration of Doors, Keys, and Gems in a gridworld. Accordingly, you generated the optimal response and the sequence of actions to infer the complete instructions. In P2 Phase-2, you will implement the optimal actions that you identified in Phase-1.

To handle the movements, you may choose any optimal path finding algorithm. For instance, A* (A-star) and Breadth First Search (BFS) algorithms are two popular optimal pathfinding algorithms suited for this task. But you are free to choose any algorithm of your liking as long as you can justify using it. Your objective is to implement the decision-making process that interprets the human's instructions and executes the optimal actions accordingly.

For this task, you will be provided with two separate files: one containing the grid configuration and the other detailing the human's actions, including movements (e.g., Pick_up_RED_key), relevant actions (Unlock_BLUE_door), and an instruction (e.g., Can you unlock this door?). These actions can appear in any order.

You are given 5 test cases, with the grid configurations named as `1_grid.txt`, `2_grid.txt`, etc., and the human actions as `1_human.txt`, `2_human.txt`, etc. Additional test cases will be used for further assessment.

For a better understanding of the optimal actions, refer to the Phase-1 document and your previously generated responses.

2. Input files

The following sections (2.1 and 2.2) explain the format of the input files. Text files are used to represent both the grid configuration and the human actions.

2. 1 Grid Configuration

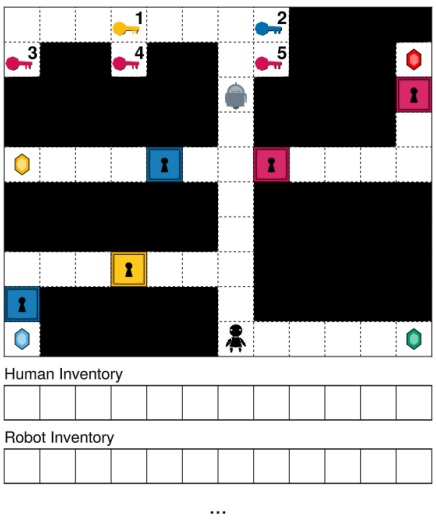

Fig: Grid Overview

- **Text grid configuration:** The grid input file will only contain the objects such as:
  - Keys: Labeled as ‘r’, ‘b’, and ‘y’.
  - Doors: Labeled as ‘R’, ‘B’, and ‘Y’.
  - Gems: Labeled as ‘g’.
  - Walls: Labeled as ‘W’.
  - Empty spaces: Labeled as ‘.’
  - Human: Labeled as ‘h’.
  - You: Labeled as ‘m’.

.	.	.	y	.	.	.	b	W	W	W	.
r	W	W	r	W	W	.	r	W	W	W	g
W	W	W	W	W	W	m	W	W	W	W	R
W	W	W	W	W	W	.	W	W	W	W	.
g	.	.	.	B	.	.	R	.	.	.	.
W	W	W	W	W	W	.	W	W	W	W	W
W	W	W	W	W	W	.	W	W	W	W	W
.	.	.	y	.	.	.	W	W	W	W	W
B	W	W	W	W	W	.	W	W	W	W	W
G	W	W	W	W	W	h	.	.	.	.	g

Fig: 1_grid.txt (Input)

2. 2 Human's Actions

- Possible Actions: The actions listed in the human's input file can be Move {directions}, Instruction, Pick_up_{COLOR}_key, and Unlock_ {COLOR}_door.

```
Move UP
Move UP
Move UP
Move UP
Move UP
Instruction: I am picking up the blue key, can you find a yellow key?
Move UP
Move UP
Move UP
Move UP
Move RIGHT
Pick_up_BLUE_key
Move LEFT
```

Fig: 1_human.txt (Input)

3. Test case samples with Solution

Here are the `1_grid.txt` and `1_human.txt` provided with the solution.

Input (1)

1_grid.txt:

.	.	.	y	.	.	.	b	W	W	W	.
r	W	W	r	W	W	.	r	W	W	W	g
W	W	W	W	W	W	m	W	W	W	W	R
W	W	W	W	W	W	.	W	W	W	W	.
g	.	.	.	B	.	.	R	.	.	.	.
W	W	W	W	W	W	.	W	W	W	W	W
W	W	W	W	W	W	.	W	W	W	W	W
.	.	.	y	.	.	.	W	W	W	W	W

B	W	W	W	W	W	.	W	W	W	W	W
G	W	W	W	W	W	h	.	.	.	.	g

Fig: 1_grid.txt

1_human.txt:

Move UP
Move UP
Instruction: I need a key for this yellow door.

Output (1)

My_position (2, 6),
UP (1, 6),
UP (0, 6),
RIGHT (0, 7),
Pick_up_BLUE_key,
LEFT (0, 6),
LEFT (0, 5),
LEFT (0, 4),
LEFT (0, 3),
Pick_up_YELLOW_key,
Locate_human: (7, 6),
Move_to_Human:
RIGHT (0, 4),
RIGHT (0, 5),
RIGHT (0, 6),
DOWN (1, 6),
DOWN (2, 6),
DOWN (3, 6),
DOWN (4, 6),
DOWN (5, 6),
DOWN (6, 6),
Drop_BLUE_key,
Drop_YELLOW_key

Here are the `2_grid.txt` and `2_human.txt` provided with the solution.

Input (2)

2_grid.txt:

W	W	b	W	W	W	r	W	W
W	r	.	r	W	b	.	b	W
W	W	.	W	W	W	.	W	W
W	W	.	.	m	.	.	W	W
W	W	W	W	.	W	W	W	g
h	.	.	.	.	.	B	B	.
W	W	W	W	.	W	W	W	g
W	W	W	W	.	W	W	W	W
W	W	W	W	R	W	W	W	W
W	W	W	W	R	W	W	W	W
g	.	.	.	.	.	.	.	g

Fig: 2_grid.txt

2_human.txt:

Move RIGHT
Move RIGHT
Move RIGHT
Move RIGHT
Instruction: Can you pass me the red keys?
Move DOWN
Move DOWN

Output (2)

My_position (3, 4),
LEFT (3, 3),
LEFT (3, 2),
UP (2, 2),
UP (1, 2),
LEFT (1, 1),
Pick_up_RED_key,
RIGHT (1, 2)

RIGHT (1, 3)
Pick_up_RED_key,
Locate_human: (7, 4),
Move_to_Human:
LEFT (1, 2)
DOWN (2, 2),
DOWN (3, 2),
RIGHT (3, 3),
RIGHT (3, 4),
DOWN (4, 4),
DOWN (5, 4),
DOWN (6, 4),
Drop_RED_key,
Drop_RED_key

4. Timesteps

The human's actions occur in single timesteps (T1, T2, etc.). For example, consider the file `2_human.txt`:

- **T1:** Move RIGHT
- **T2:** Move RIGHT
- **T3:** Move RIGHT
- **T4:** Move RIGHT
- **T5:** Instruction: Can you pass me the red keys?
- **T6:** Move DOWN
- **T7:** Move DOWN

You begin executing the instruction as soon as you receive it. In this case, the instruction arrives at **T5**. From this point onward, you must calculate the optimal actions based on the instructions:

- **T5:** My_position (3, 4),
- **T6:** LEFT (3, 3),
- **T7:** LEFT (3, 2),
- **T8:** UP (2, 2),
- **T9:** UP (1, 2),
- **T10:** LEFT (1, 1),
- **T11:** Pick_up_RED_key,
- **T12:** RIGHT (1, 2)
- **T13:** RIGHT (1, 3)
- **T14:** Pick_up_RED_key,
- **T15:** Locate_human: (7, 4),
- **T16:** Move_to_Human:....

In this example, you complete picking up the keys at **T16**. The human remains stationary after **T7**, so once you've finished picking up the keys, you must track their location and calculate the shortest path to their position. If you finished your task before the human reached **T7**, you would need to continuously update the human's location and adjust your route to meet them in the shortest possible time.

5. Execution Instructions

Please make sure that your code accepts appropriate inputs and executes on the terminal:

For Python:
python3 p2.py <grid filename> <human actions filename>
Example command: *python3 p2.py 1_grid.txt 1_human.txt*

For Java:
java p2.py <grid filename> <human actions filename>
Example command: *java p2.py 1_grid.txt 1_human.txt*

6. Code and Report

Generating Results: Ensure your code automatically creates and stores the results as `1_result.txt`, `2_result.txt`, and so on. These files should be saved in the **Results** folder within your current repository.

Comments in Code: Ensure that your code includes sufficient comments to explain each section clearly. Points will be deducted if your code lacks adequate comments explaining your logic, decisions, and actions.

Code Submission: Submit your code along with the Results folder on Moodle.

Report Requirements: In your report, you must address the following points:

- Decision-Making Process: Explain the code logic for interpreting human instructions and how it handles ambiguous or complex instructions.

- Algorithm Choice: Clearly state which algorithm (A* or BFS or any other) you used for optimizing movements within the grid. Describe how the algorithm ensures optimal performance in your solution under different conditions, such as tracking the human's position or minimal movements. Explain this using examples for clarity.

- Optimal Sequence of Actions: Detail the steps for generating the optimal sequence of actions, and explain any strategies you implemented to optimize tracking the human's position throughout the task.

Submit your report in a PDF file.

README File: Please include a **README** file with clear instructions on how to run your code from the terminal. Ensure that your README file provides the necessary steps, dependencies, and commands for smooth execution. This is critical, as any issues encountered while running your code could lead to penalties.

Your code will be *evaluated using a script* that tests it on all provided test cases, as well as additional cases. If your code cannot be properly executed from the terminal, it will result in penalties.

Generative AI Policy: You are NOT ALLOWED to use generative AI for writing or implementing your code. However, you may use it to understand concepts or conduct research.

Deadline: The submission is due by 5 pm on November 2nd.