State: Player position: (r, c)
Boxes Position: frozenset(r, c)

Initial State : `(initial_player, frozenset(initial_boxes))`

Goal state: all box positions are the same as the goal position

Actions: Move to four directions, up, down, left, and right, but it cannot move to the wall and boxes on the wall

Result(s, a): get new positions of boxes and the player, depending on the moving directions.

Heuristic Function:
definition: Manhattan distance between boxes and goals

Code:

```
def h(self, state):
    _, boxes = state
    total_distance = 0

    for box in boxes:
        min_dist = float('inf')
        for goal in self.goals:
            dist = abs(box[0] - goal[0]) + abs(box[1] - goal[1])
            if dist < min_dist:
                min_dist = dist
        total_distance += min_dist

    return total_distance
```

Admissibility:
A* search and UCS heuristic h(n) cannot be over real cost (h*(n)). Manhattan distance is the distance of the shortest route, so the actual distance must be longer than it.

UCS:

```
        simple: Search completed in 0.00 seconds.
                Solution found with 8 moves.
                URRUULLD
        mid:    Search completed in 0.00 seconds.
                Solution found with 8 moves.
                URRUULLD
        hard:   Search completed in 14.23 seconds.
                Solution found with 34 moves.
                RURRDDDDLDRUUUULLLRDRDRDDLLDLLUUDR
```

A*:

```
        simple:Search completed in 0.00 seconds.
                Solution found with 8 moves.
        URRUULLD

mid:    Search completed in 0.75 seconds.
        Solution found with 144 moves.
        DDUURRDDLLRRDDLLULLUURDRDDRRUULLRRUULLDLLDDDRRUULR
        RRUULLDLDRDDLURUUURRDDDDLLUURLDDRRUULLDLLDRRUUULDR
        DDLLURURRRUDDDLLULURRLDDRRUULLLDLDRRRLUUULLD

hard:   Search completed in 2.30 seconds.
        Solution found with 34 moves.
        RURRDDDDLDRUUUULLLRDRDRDDLLDLLURLU
```