README.md

Artificial Intelligence

Intermediate report for the course of Artificial Intelligence (IART) during summer semester 2019-2020.

0. Github Repository

The project is hosted in an public github repository here.

Students / Authors

- Stylianos Tsagkarakis *up201911231*
- Vasileios Konstantaras *up201911213*

1. Specification of the work

BoxWorld 2 is a solitary type game in which the player needs to move on the board with the the **ARROW** keys to reach the exit, in order to advance to the next level. The game has several obstacles that the player needs to overcome to finish each level. There are boxes that can be pushed and holes that can be filled with boxes in order to move over them. The player starts with a specific number of lives, and if he/she gets stuck, can press the **R** key to reload the same level with the expense of one life. The progress of the game can be saved pressing the **S** key at any given time and can be loaded later by pressing the **L** key.

For this assignment we need to create a program that is able to solve different levels of the game using the following research methods:

1. Breadth-first search
2. Depth-first search
3. Iterative Deepening
4. Uniform Cost
5. Greedy Search
6. A*

Then we need to compare the different research methods based on the quality of the obtained solution, the total number of operations occured and the time taken to obtain the solution. Furthermore the program must have some kind of visualization whether text-based or graphical so the user can see the evolution of the board. Finally the program must be able to solve the game by its own using the method and the board configuration provided by the user. Optionally a game mode that the user plays and asks the program for "tips" could be implemented.

2. Research (articles, web pages, source codes)

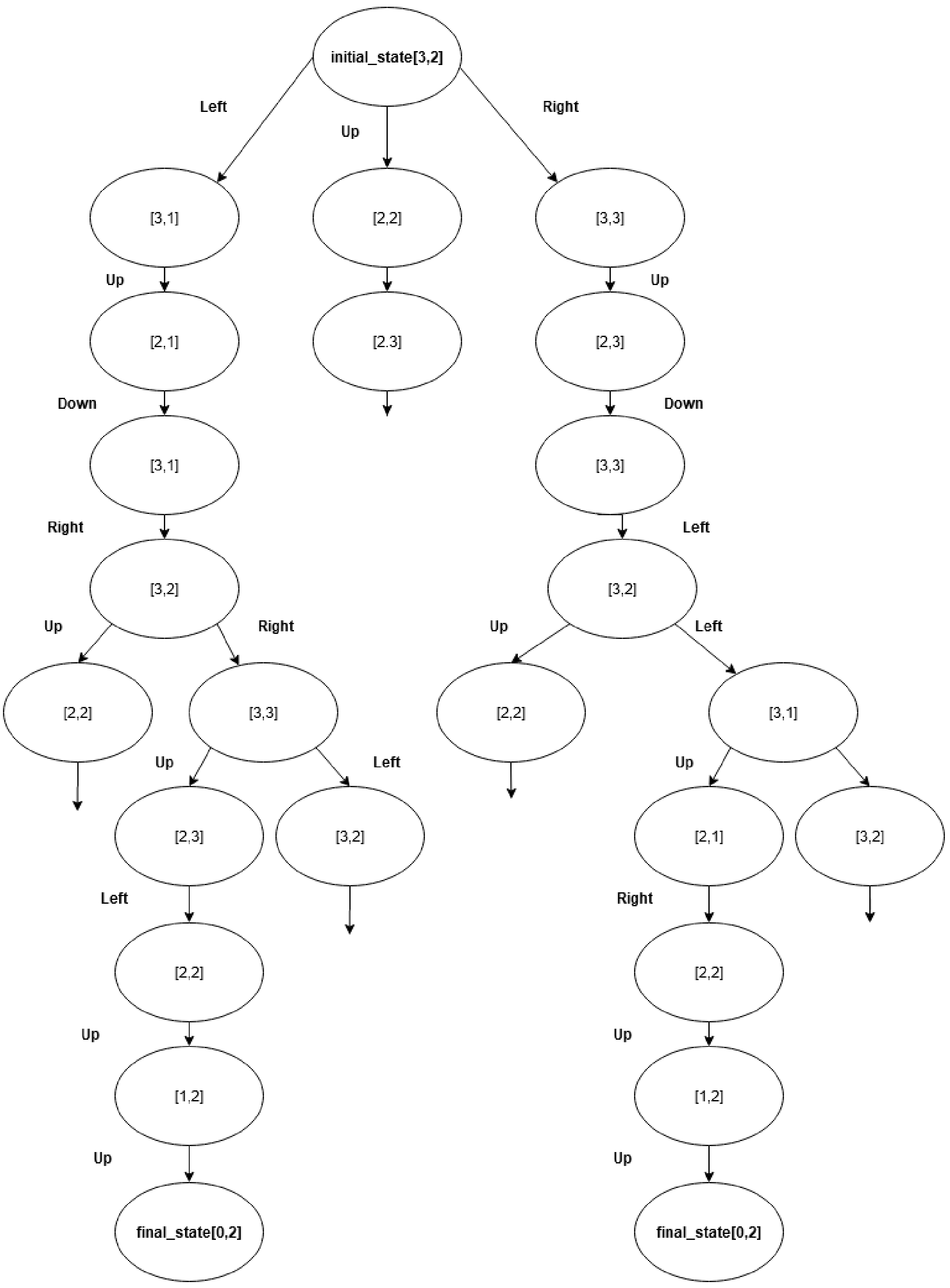
- 8 Reasons to Use Python for AI
- Create Your Own Reinforcement Learning Environment
- Pathfinding in Strategy Games and Maze Solving

3. Formulating the problem as a research problem

The initial state is the position of the user on the table when the level begins and the final state is the exit position. The operators are the keys W,A,S,D which respectively move the user Up, Left, Right and Down. The preconditions are:

1. where the player wants to be an empty space, meaning no boxes, holes or wall, then the space becomes the player's position and the previous one becomes empty.
2. To move a box, it is required the next place after the box, towards the direction the user wants to move it, to be empty or a hole. If it is a hole, when the box falls in to the hole it becomes empty space. Otherwise the box covers the next spot, the user covers the spot previously owned by the box and the place that the user was before, becomes empty.

This is the state representation:

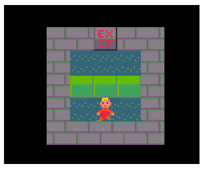
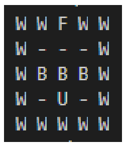


4. Implementation of the work already done

The application will be written in **Python** using The programming enviroment is Visual Studio Code as IDE.

Input

Each map will be represented by a 2D array. We transformed each level into a text file as shown below.

From this :  to this : 

```
W W F W W
W - - - W
W B B B W
W - U - W
W W W W W
```

The player has the ability to choose which level wants to play and to move around the board using the WASD and can have a visual representation of his/her movements.

Idea of implementation

We wanted to eliminate if cases as much as possible, to avoid confusion and repetitiveness. Using pythons string concatenation ability we will make a switch case like that:

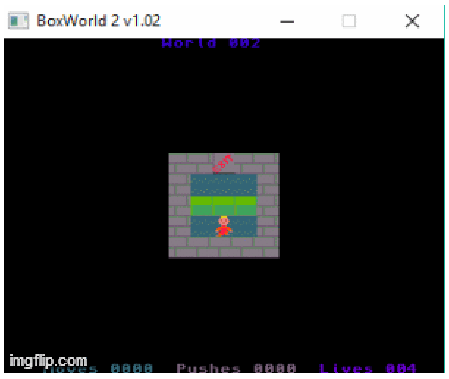
```python
def switch_case (first, second):
    switcher = {
        "user"   : 'U' ,
        "box"    : 'B' ,
        "ice"    : 'I' ,
        "hole"   : 'H' ,
        "wall"   : 'W' ,
        "finish" : 'F' ,
        "space"  : '-'
    }
    first = switcher.get(first, "error")
    second = switcher.get(second, "error")
    action_func = first+"_and_"+second
    return action_func
```

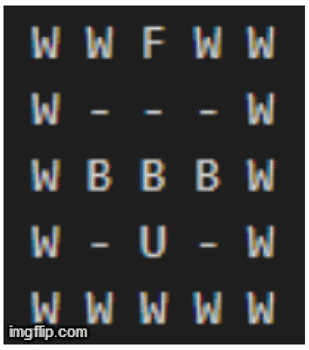
First and second are supposed to be the items that interact during the game. After getting the function name, it will be called like that:

```python
function = switch_case(arg1, arg2)
function()

# list of possible action_funcs:

def user_and_space (world, direction):
def user_and_box (world, direction):
def user_and_finish (world, direction):
def user_and_ice (world, direction):
def user_and_wall (world, direction):
def user_and_hole (world, direction):
def box_and_wall (world, direction):
def box_and_hole (world, direction):
def ice_and_wall (world, direction):
def ice_and_hole (world, direction):
```

The basic structure is the one mentioned above. The next step is to implement these functions that will use other helper function to update positions. Of course the search algorithms should also be implemented.

Side by side comparison of how the players moves between the actual game (*left*) and the program (*right*) progressing throught the level 2.





```
W W F W W
W - - - W
W B B B W
W - U - W
W W W W W
```