AI - Assignment 3

Patrick Nagel: patrick.nagel@h-brs.de
Amir H. Pakdaman: amirhossein.pakdaman@smail.inf.h-brs.de

Due Date: 29.11.2020

0.1 Programming Assignment: Grid world

You will find three text files, each representing a map for the programming assignment in the maps folder. A search agent is supposed to explore each of these three maps using breadth-first search, depth-first search and iterative-deepening depth-first search. These maps can be interpreted as follows:

- Each character in the text file represents a "cell" in the map.
- (*) Cell contains a goal.
- (Space) Cell is free.
- (s) Initial position of the agent.
- Any other character represents an obstacle.

Your task is to implement the above mentioned search-algorithms to make your agent explore and find each dirt cell. The agent must follow these rules:

- The agent can move from one cell to another at each step.
- The agent can only move to the left, right, up or down from the current position. The surrounding cells should be considered as the children at the current agent's position.
- The agent does not have previous knowledge about the environment (such as dirt positions or obstacles) so it has to ”explore”.
- The agent cannot move through obstacles and the map is closed.

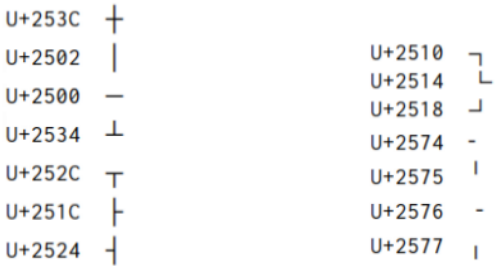
Provide terminal output, so that the functioning of your program is easily readable, e.g. by printing the map with all explored nodes marked. To circumvent compatibility issues, provide that output in text-files along with your submission for each of the implemented search algorithms.

TIP: You can re-route the output of your program to a file by using the > operator. The > operator takes the output of a command-line program and puts it into the specified file. ATTENTION: All content of that file is being overriden. To append output to a file (and not overriding its content) use the » operator. For example:

```
python3 my_python_script.py > output_file1.txt
```

- Visualize the search-tree that your algorithm spans along the map. You can use the unicode-characters given below to do so.
  To print or assign a unicode-character to a variable use the following syntax in python 3:

  ```
  my_agent_map[x][y] = ‘‘\u253C’’
  print(‘‘\u2577’’)
  ```



0.2 Implementation Details

- Implement DFS, BFS and IDDFS algorithms in the functions *depth_first_search*, *breadth_first_search*, *iterative_deepening_depth_first_search* in the respective files.
- The main function parses all the three maps and stores the data in a list.
- Implement *maze_map_to_tree* function to convert the parsed map into a search tree.
- Use the function *assign_character_for_nodes* in order to indicate the direction of traversal of the search agent through the nodes of the search tree.
- Use *write_to_file* function to write backtracked optimal path to a text file and print the traversal of the search agent on the console.
- **Important:** Please add comments explaining your code as and when required. Also fill in the doc-strings for the implemented functions.
- You may add additional functions if required and modify the parameters of the existing functions.
- Any extra functions required as per your implementation should be included in helper.py. Please add the necessary docstrings and keep your code as clean as possible.
- Please refer to the travis.yml for command line options to run your code.

0.3 Deliverables

- Add your name to the readme file.
- Add 9 text files to the folder *results*. Each file contains maps with path from starting point to each goal. For each goal add a separate map to the file.
- Use visualization symbols as described above.
- Codes should run with no errors.
- Codes should follow the theory of algorithms.