

 README.md

Sokoban Environment - NYU CS-GY6613 - Fall 2020

Prerequisites

Requires python3 to run

Install libraries

```
$ pip install -r requirements.txt
```

Run the Game

Solve as a human

```
$ python3 game.py --play $ python3 game.py --agent Human
```

Solve with an agent

```
$ python3 game.py --agent [AGENT-NAME-HERE]
```

```
$ python3 game.py --agent BFS #run game with BFS agent
```

Parameters

`--play` - run the game as a human player

`--agent [NAME]` - the type of agent to use

`--level [#]` - which level to test (0-488) or 'random' for a randomly selected level that an agent can solve in at most 2000 iterations (default=0)

`--iterations [#]` - how many iterations to allow the agent to search for (default=3000)

Code Functions

*These are the only functions you need to concern yourselves with to complete the assignments. **WARNING: DO NOT MODIFY THESE FUNCTIONS!***

Sokoban.py

- **state.clone()** - creates a full copy of the current state (for use in initializing Nodes)

Agent.py

- **Agent()** - base class for the Agents
- **RandomAgent()** - agent that returns list of 20 random directions
- **DoNothingAgent()** - agent that makes no movement for 20 steps

Helper.py

- **Node Class**

- **__init__(state, parent, action)** - where *state* is the current layout of the game map, *parent* is the Node object preceding the state, and *action* is the dictionary XY direction used to reach the state (*return type: Node object*)
- **checkWin()** - returns if the game is in a win state where all of the goals are covered by crates (*return type: bool*)
- **getActions()** - returns the sequence of actions taken from the initial node to the current node (*return type: str list*)
- **getHeuristic()** - returns the remaining heuristic cost for the current state - a.k.a. distance to win condition (*return type: int*)
- **getHash()** - returns a unique hash for the current game state consisting of the positions of the player, goals, and crates made of a string of integers - for use of keeping track of visited states (*return type: str*)
- **getChildren()** - retrieves the next consecutive Nodes of the current state by expanding all possible directional actions (*return type: Node list*)
- **getCost()** - returns the depth of the node in the search tree (*return type: int*)