

Prerequisites

Requires Python 3.8 to run

Install libraries

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

Run the Game

```
$ 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
```

```
$ python3 game.py --agent AStar --no_render #run game with AStar agent without rendering
```

Parameters

--play - run the game as a human player

--no_render - run the AI solver without showing the game screen

--agent [NAME] - the type of agent to use [Human, DoNothing, Random, BFS, DFS, AStar, HillClimber, Genetic, MCTS]

--level [#] - which level to test (0-99) or 'random' for a randomly selected level that an agent can solve in at most 2000 iterations.

These levels can be found in the 'assets/gen_levels/' folder (default=0)

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

--solve_speed [#] - how fast (in ms) to show each step of the solution being executed on the game screen

Agent Types

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

BFSAgent() - agent that solves the level using Breadth First Search

DFSAgent() - agent that solves the level using Depth First Search

AStarAgent() - agent that solves the level using A* Search

HillClimberAgent() - agent that solves the level using HillClimber Search algorithm

GeneticAgent() - agent that solves the level using Genetic Search algorithm

MCTSAgent() - agent that solves the level using Monte Carlo Tree Search algorithm

Code Functions

Sokoban_py

state.clone() - creates a full copy of the current state (for use in initializing Nodes or for feedforward simulation of states without modifying the original)

state.checkWin() - checks if the game has been won in this state (return type: bool)

state.update(x,y) - updates the state with the given direction in the form x,y where x is the change in x axis position and y is the change in y axis position.

Helper_py

Other functions

getHeuristic(state) - returns the remaining heuristic cost for the current state - a.k.a. distance to win condition (return type:int). Use with HillClimber Agent to compare states at the end of sequence simulations directions - list of all possible directions (x,y) the agent/player can take Use with HillClimber Agent to mutate sequences

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 and comparing Nodes (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)