CMPE540 – 2020-2021 - HW2: Adversarial search in some sliding tiles puzzle

Due: 04.05.2021, 23:59 (+5 bonus)

or

Due: 11.05.2021, 23:59 (with the late submission policy)

Via moodle (repeated submission is not allowed – submit once)

Late submission policy: Grade-10*day^2

You will implement adversarial search in this homework. The domain is sliding tiles domain, however, different from a standard n-tiles problem:

- There might be multiple empty cells. An empty cell is represented by dot (.)
- The numbers on the tiles are arbitrary (integer numbers, not necessarily consecutive, might be negative)
- There might be fixed tiles, represented by **x**
- Action cost corresponds to the integer value of the tile being moved.

The program will be called with:

```
python3 <id.py> <search-type> <init-file> <goal-file> <n-actions> <sln-file>
```

where

- `<id.py>` should be named with your `student id`
- `<search-type>` will be one of the following
  - `minimax`
  - `minimax_rand`
  - `alpha_beta_pruning`
- `<init-file>` and `<goal-file>` will be text files in the following format:

```
. 1 4
. 2 5
. 3 6
```

- `<n-actions>` is the number of actions in search taken by both your agent and other agents
- `<sln-file>` is the file the solution is written

General:

- There are three agents in this game:
  - Your agent, named: AGENT1. AGENT1 can move only odd tiles.
  - Another agent, named: AGENT2. AGENT2 can move only even tiles.
  - Another agent, named: AGENT3. AGENT3 can move all tiles.
- The order of play: AGENT1, AGENT2 , AGENT3.
- All agents make <n-actions>
- The board configuration is given such that there will be always available actions for all agents.
- The terminal condition is met when <n-actions> are done by all agents.
- All agents aim to maximize the utility, **unless specified otherwise**
- Utility of the AGENT1 is computed as follows:
  - For each cell, check out if the tile is correctly placed with respect to the goal board configuration.
  - If its placement is correct and if it is an odd tile, increment the utility with the corresponding number
  - If its placement is correct and if it is an even tile, decrement the utility with the corresponding number
- Utility of the AGENT2 is opposite of the utility of AGENT1, i.e. AGENT2 plays competitive game against AGENT1.
- Utility of the AGENT3 is same with the utility of AGENT1, i.e. AGENT3 cooperates with AGENT1.
- The initial and goal boards include the same set of tiles.

Example utilities:

current board:

```
. 1 4
. 2 5
. 3 6
```

goal board:

```
1 2 4
3 . 5
. . 6
```

Utility for AGENT1: `0 + 0 + (-4) + 0 + 0 + (+5) + 0 + 0 + (-6) = -5`
Utility for AGENT2: `0 + 0 + (+4) + 0 + 0 + (-5) + 0 + 0 + (+6) = +5`
Utility for AGENT3: `0 + 0 + (-4) + 0 + 0 + (+5) + 0 + 0 + (-6) = -5`

current board: goal board:

```
1 2 .          1 2 .
3 4 .          3 4 .
. . .          . . .
```

Utility for AGENT1: (+1) + (-2) + (+3) + (-4) = -2
Utility for AGENT2: (-1) + (+2) + (-3) + (+4) = +2
Utility for AGENT3: (+1) + (-2) + (+3) + (-4) = -2

Output: After running the search, you need to create solution file with the following content:

```
AGENT1 <action>
AGENT2 <action>
AGENT3 <action>
...
Value: <value>
Util calls: <n-util-calls>
```

where

- `<action>` has the form `move <tile> <direction>`
  - where `<direction>` can be one of the following: `{left, right, up, down}`
- `<value>` is the value of the root node. Report 2 decimal places. I used `fp.write("Value: "+("{:.2f}".format(value))+"\n")`
- `<n-util-calls>` is the number of calls for utility function

Example Solution File (n-actions=2):

```
AGENT1 move 3 down
AGENT2 move 2 right
AGENT3 move 2 left
AGENT1 move 3 up
AGENT2 move 2 right
AGENT3 move 4 right
Value: 4.00
Util calls: 2208
```

- Minimax search is standard minimax search.
- In minimax_rand search, assume that the AGENT2 and AGENT3 make random moves with probability of taking any action is same.
  - When reporting the action sequence for random actions, always follow the tile-action ordering described below.
- In alpha_beta_pruning search, you need to expand the nodes by giving the priority to actions as follows:
  - The order of expansion: In expanding a node, there will be multiple possible ordering choices for expansion. You need to check out whether each tile can move in different directions depending on whether the neighbouring cells in the movement directions are empty or not.
    - **Consider the following tile order:** Start from the top-left tile, then sweep the tiles in the first row one by one. After checking out all the tiles in the first row, continue with the second row. The last tile to possibly check out will be the bottom-right tile.
    - **Consider the following movement direction order**: For each tile, you need to follow a specific action/movement order:
      - left right up down

Submission details: You will submit a single file:

- <studentid>.py: your python implementation. You are allowed to use standard libraries. All search algorithms should be implemented by you.

Example input-outputs:

Attached. Generated with:

```
python3 emre.py minimax init_0.txt goal_0.txt 1 soln_0_1_minimax.txt
python3 emre.py minimax_rand init_0.txt goal_0.txt 1 soln_0_1_rand.txt
python3 emre.py alpha_beta_pruning init_0.txt goal_0.txt 1 soln_0_1_ab.txt
python3 emre.py minimax init_0.txt goal_0.txt 2 soln_0_2_minimax.txt
python3 emre.py minimax_rand init_0.txt goal_0.txt 2 soln_0_2_rand.txt
python3 emre.py alpha_beta_pruning init_0.txt goal_0.txt 2 soln_0_2_ab.txt
python3 emre.py alpha_beta_pruning init_0.txt goal_0.txt 3 soln_0_3_ab.txt
```

```
python3 emre.py minimax init_1.txt goal_1.txt 1 soln_1_1_minimax.txt
python3 emre.py minimax_rand init_1.txt goal_1.txt 1 soln_1_1_rand.txt
python3 emre.py alpha_beta_pruning init_1.txt goal_1.txt 1 soln_1_1_ab.txt
python3 emre.py minimax init_1.txt goal_1.txt 2 soln_1_2_minimax.txt
python3 emre.py minimax_rand init_1.txt goal_1.txt 2 soln_1_2_rand.txt
python3 emre.py alpha_beta_pruning init_1.txt goal_1.txt 2 soln_1_2_ab.txt
python3 emre.py alpha_beta_pruning init_1.txt goal_1.txt 3 soln_1_3_ab.txt

python3 emre.py minimax init_2.txt goal_2.txt 1 soln_2_1_minimax.txt
python3 emre.py minimax_rand init_2.txt goal_2.txt 1 soln_2_1_rand.txt
python3 emre.py alpha_beta_pruning init_2.txt goal_2.txt 1 soln_2_1_ab.txt
python3 emre.py minimax init_2.txt goal_2.txt 2 soln_2_2_minimax.txt
python3 emre.py minimax_rand init_2.txt goal_2.txt 2 soln_2_2_rand.txt
python3 emre.py alpha_beta_pruning init_2.txt goal_2.txt 2 soln_2_2_ab.txt
```