Artificial Intelligence
Projects 1

Overview

Files: The files are available in the class git repo (and also on Moodle). The the course page for instructions on access the git server. The name of this assignment is `contest1`. You should only edit MySolver.py and description.txt.

Grading: Each assignment will be graded as follows:

30%	Performance on test problem instances relative to reference implementations (this will be based on your final version)
40%	Code quality: correctness, use of AI methods, clarity, etc.
30%	Description of algorithm used, why it is effective and the steps to took to improve it
up to 5% bonus	Performance relative to your classmates

Submission

Due: 6am Monday, February 3.

Goal: Write an agent to solve the fifteen puzzle that finds an optimal solution as quickly as possible.

Testing: You can test your program using `Run.py`, see below.

Submission: Perform a git commit and a git push to submit your files. Submitting your files will automatically test your code and post results on the contest page. This way you know how you perform relative to your classmates and the reference implementations. Note that the final grading will be using a different set of test instances. Programs will be given a maximum time of 2 minutes to solve each puzzle.

Late penalty: Projects will receive a 10% penalty for every day that they are late. Submissions more than 3 days late will not receive detailed comments for resubmission.

Grading Note that a significant portion of your grade will come this version. You will be able to improve your program, if you like, after receiving feedback. This can improve your grade further.

Feedback and resubmission

When you are ready to submit, you should push your project to the git repository and email me with the git commit id (a hexadecimal key located on the list of files in your repository). I will provide feedback in the order received. I will try to get feedback to you as quickly as possible. However, there are a lot of projects to review. I will guaranteed a response withing 72 hours. If you choose, you can refine and improve your code and project description. These are due by 6am Monday, February 10. No late submissions will be accepted.

What to change in MySolver.py

You should carefully go through the code in MySolver, you don't have to work about the code in the other files. It's similarly to the code we discussed in class for breadth-first search but there are some differences:

- It uses a priority queue to do breadth-first search. States are inserted with the priority equal to their depth in the search tree. The priority queue will always return the item with the lowest priority value.
- Instead of storing the nodes that have been expanded, it stores all of the that have either been expanded or put in the frontier. This makes it faster to search if any new states have been seen before. (Searching a priority-queue is slow.)
- There is a check in the while loop to make sure you have not exceeded the maximum time.

You should focus on changing two things in the code to get an efficient algorithm:

- Implement a useful heuristic function (possibly the sum of the Manhattan distances between each tiles location and it's desired location.
- Change the priority that states are inserted in the priority queue with to implement A* search.

Running your program

You can test your code of various puzzles of different sizes and complexity. On the last page are some examples where the optimal solution length is known. You can run the program on one of these as follows:

```
python Run.py 3 120 -m ULLURRDDLLUURDR
```

This will try to solve a 3x3 puzzle with a known starting position and a time limit of 120 seconds. If you are running on a machine with cs1graphics installed, you can see what you solution looks like by adding an option:

```
python Run.py 3 120 -m ULLURRDDLLUURDR -g 200
```

where the 200 is the size of the graphics window. You can type `python Run.py` to see all of the options.

Hints

To do well consider the following:

- To have it run faster while you are testing use pypy3 instead of python; it's a faster implementation of python. I will be using it to test your programs. It is installed on the Linux lab computers. On hooper you can use pypy (pypy3 should be installed soon.)
- Submit agents often. Test them, find improvements, document and submit again.
- Try basic algorithms/heuristics first.
- A webpage worth looking at
  - `http://kociemba.org/themen/fifteen/fifteensolver.html`
- Ask questions and read discussions on the Moodle forum for this assignment

Examples

Here are some examples that you can run your code on to ensure that you are finding the optimal solutions. The ones with shorter solutions are usually easier to solve. You should test on those first.

Size	Moves to generate	Length
3	LDDRR	5
3	LURDLULURD	10
3	ULLURRDDLLUURDR	15
3	UULDDLUURDLDRULURRDD	20
3	ULDLUURDLDRRULLDRULURRDDL	25
4	RULDRURRDD	10
4	LUULURRDLDDLLUU	15
4	LUUULDDRDLLURRRUULDL	20
4	LUURDLULURRDDDLULDRU	20
4	LLLUUURDLURRRDDDLULDLURUU	25
4	UUULDRDDLUULURRDLURDDLDRU	25
4	UULURDDLLDLUURRURDLDLLURDRDLLU	30
4	ULUURDLLDDRRULLLURURDDDLUURULDDLDRR	35
4	LULUULDDDRRUULDDRUURULDRDLULLDRULUR	35
4	UULULDRULLDDRURULDLDDRRRULULDDLUURU	35
4	UUULDLDRDLULDRRRUULULDDLURDDRRUUULDRULLD	40
4	UUULLDDLDRRRUULLURDDLLDRUURDRUULDLDRDRUU	40
4	ULLDRRULULDRDLULURRRULLLDRDLDRULURRURDLDDLUUU	45
4	UULDLUULDRRURDDLDLULURULDDDRUURULDRDRUULDRDDL	45
4	LUURDDLULLUURDDLDRUURURDDLDLULURRDLLURURDDLUURRDLD	50
4	LLUURDDLULDRRRULLURRDDLUUULDLDRRRDLLLURRUURDDLULUR	50
4	LLLUURULDRRDRULDDRULDLURUULDDLURDRRUULDLURDRULLLDR	50