Theory of Computer Games 2018 - Project 6

In the series of projects, you are required to develop AI programs that play [*Threes!*](http://asherv.com/threes/), the origin of other 2048-like games.

Overview: **Join the Threes! tournament**.

1. Fine-tune the player and the antagonistic environment.
2. Make sure that the program can establish a stable internet connection with the arena.
3. Compete in the tournament.

Specification:

1. The rules are the same as the rules for Project 5:
2. Train the agent by temporal difference learning and any other possible improvements.
3. Statistic is required, and the requirement is the same as Project 2’s requirement.
   1. Average score.
   2. Maximum score.
   3. Speed (action per second).
   4. Win rate of each tiles.
4. The arena protocol is required.
5. Input arena commands (variables are marked in bold):
6. #**ID** open **AGENT**:**AGENT**  
   You will receive this message when a new game is pending, where #**ID** is the game ID, and **AGENT**:**AGENT** are the names of player and environment.
7. #**ID** ?   
   You will receive this message when your agent needs to take an action.
8. #**ID** **ACTION**   
   You will receive this message when your opponent took an action, where **ACTION** is the action code.  
   Action codes for player: #**X**, where **X** is either U, D, L, or R. The regular expression is #[UDLR].  
   Action codes for environment: **PT**+**H**, where **P** is the position of new tile; **T** is the index value of new tile; **H** is the hint of next tile. The regular expression is [0-9A-F][0-9AB]\+[1234], e.g., F3+2, 51+1, 45+3, A2+4.
9. #**ID** close **FLAG**  
   You will receive this message when a game just finished, where **FLAG** is the closing flag (the winner, or the reason of closing the game, etc.).
10. @ login  
    You will receive this message once you establish a connection to the server, and you should reply your account and agents.
11. Output arena commands:
12. #**ID** **ACTION**   
    This message should be sent when you need to perform an action. The action code definition is the same as above.
13. #**ID** open **OPEN\_REPLY**   
    This message should be sent when you decide to join a game, where **OPEN\_REPLY** should be either accept or reject.
14. @ login **ACCOUNT** **AGENT**(**ROLE**)...  
    This message should be sent when you receive the login message. If you have multiple agents, separate them with a space.
15. Internet connection is required.
    1. Send and receive arena commands over SSL connections.
16. Speed requirement: At least **10 actions** per second (over the internet).
17. Implementation details:
18. Your program should be able to compile and run under the workstation of NCTU CS.
19. Write a makefile (or CMake) for the project.
20. C++ is highly recommended for TCG.  
    You may choose other programming language to implement your project.
21. Your implementation needs to follow the statistic output format.  
    (see Project 2’s spec for details)
22. Please refer to the sample code for the details of arena protocol.
23. Your program should be able to serialize and deserialize the weight tables of N-tuple network.

Methodology:

1. There is **no minimum # of bonus tiles requirement** in Project 6.
   1. Generating all the tiles from the bag is now acceptable.
   2. **At any timestamp of an episode**, .
2. The arena protocol has already been implemented in the sample code.
   1. While you need to modify some lines. (e.g. add hint tile processing code)
   2. Also, you don’t need to handle the internet connection by yourself, just keep your program using standard input and output. We will provide you with a tool to link your program with internet.

Submission:

1. Your solution **should be archived in zip/rar/7z file**, and **named as XXXXXXX.zip**, where XXXXXXX is the student ID (e.g. 0356168.zip).
2. Pack your **source files**, **makefiles**, and other relative files in the archive.
3. Do **NOT** upload the statistic output or the network weights.
4. Provide the version control repository of your project (URL), while do **NOT** upload the hidden folder (e.g. **.git** folder).
5. Your project should be able to run under the workstations of NCTU CS.
   1. **Test your project on workstations**.
   2. Only run your project on workstations reserved for TCG. Do not occupied the normal workstations, otherwise you will get banned.
   3. Respect the rights of others. Do not occupied the resources of workstation.

Scoring Criteria:

1. Demo: **TBD**.
2. Framework: Connect to the arena server.
   1. **Your score is not counted if the program cannot connect to the server**.
3. Ranking (100 points):
   1. The score is based on the ranking, between 70 and 100.
4. Final score is calculated linearly by the participants’ credits.
   1. Round-robin scheduling, may have several rounds if we have enough time.
5. A round = a full round-robin schedule of all participants.
6. Participant with the higher player score win the match and will receive 1 credit.  
   e.g. the result of “A vs B” is determined by two matches: A’s player vs B’s evil; and B’s player vs A’s evil. The one whose player gets higher score win.
   1. Please refer to e3 discussion forum for the rules of the tournaments.
7. Maximum and minimum score (bonus):
   1. The player/environment who achieve the max/min score in the tournament will get an extra 1 point as the bonus.
8. Penalty:
   1. No special penalty.
9. Note that violates the rules in the tournament will still make you lose that game, and the credit will be your opponent’s.  
   If both player and evil violate the rule (e.g. TLE), no credit for both of them.
   1. Please refer to e3 discussion forum for any further announces in the tournaments.

Hints:

Having some problems? Feel free to ask on the Discussion of e3 platform.  
You may use [Github Student Developer Pack](https://education.github.com/pack) or [Bitbucket](https://bitbucket.org/) for the version control.  
Remember to share the sources on sharing platform, for example, [GitHub Gist](https://gist.github.com/).