Deadline: 2019-10-04, 23:59
Recitation (øvingsforelesning): 2019-09-25, 16:15, F1

Minimax and Alpha-Beta Pruning

Overview

In this assignment you will implement the Minimax algorithm for adversarial search, e.g. following the pseudocode in Figure 5.3 in the textbook. Afterwards you will extend your Minimax implementation to use alpha-beta pruning, in order to speed up the search. Pseudocode for alpha-beta pruning is provided in Figure 5.7 in the textbook.

For this assignment we will follow the "Pac-Man projects" developed at UC Berkeley. The Pac-Man projects were developed for UC Berkeley's intro course to AI, and later made available for other universities to use. See `http://ai.berkeley.edu/project_overview.html` for an overview of these projects. The projects cover a variety of AI techniques, exploring them through the world of Pac-Man.

UC Berkeley has developed an extensive Python codebase to support these projects. This Python codebase provides the framework in which you will write your Minimax and alpha-beta implementations, so **for this assignment you *must* use Python**. The Python codebase implements the rules of Pac-Man and visualizes the game, so you will not have to write this code yourself. The code also includes the program `autograde.py`, that will run various tests on your Minimax and alpha-beta implementations in order to verify that you have correctly implemented all the subtleties of these algorithms.

Steps

1. Go to `http://ai.berkeley.edu/multiagent.html` and read the introduction.
2. Download the Python codebase by clicking the link "available as a zip archive" on that webpage.
3. Verify that the code works on your computer by extracting the archive, opening a terminal window, navigating to the correct directory and running `python pacman.py`. Use the arrow keys to play.
4. Complete the following questions from `http://ai.berkeley.edu/multiagent.html`:
   - **Question 2** – Minimax
   - **Question 3** – Alpha-beta pruning

   As specified in the questions, your code should be implemented by editing the classes `MinimaxAgent` and `AlphaBetaAgent` in `multiAgents.py`. Your code should then be tested for correctness by running respectively `python autograder.py -q q2` and `python autograder.py -q q3`.

   In each case the autograder requires that your code passes *all* of the tests, so you will always get either 0/5 or 5/5 points as your output from `autograder.py`. Scroll up in the output from the autograder to see which specific tests you need to fix in order to get 5/5 points.

5. Deliver the following on Blackboard, once you have achieved 5/5 points on both questions 2 and 3: [1]

   - **Code:** All of your code changes should be contained in `multiAgents.py`, so this is the only code file you need to deliver. Your changes must be well commented.
   - **Results:** Deliver a txt or a pdf file with your successful output from the autograder. The file should contain the *entire* output from running both `python autograder.py -q q2` and `python autograder.py -q q3`.

[1] If, by the deadline, you still have some tests that do not pass, submit your assignment on Blackboard anyway and we will make a judgement as to whether your assignment can still be accepted based on your code and how many of your tests passed. Make sure that your code is well commented and that you include your autograder results when you submit on Blackboard.