CSE537: Artificial Intelligence, Fall 2018

Assignment 2

(Due: 08 October 2018)

Submission Details			
Name	SBU ID	Email Id	% Contribution
Shobhit Khandelwal	112074908	shobhit.khandelwal@stonybrook.edu	50 %
Vivek Bansal	112044493	vivek.bansal@stonybrook.edu	50 %

Answer 1) Reflex Agent

Approach:

1) If new state contains a ghost, then it is a very bad state to move for pacman so score returned is - infinity.
2) We will return more score for that state which contains food at minimum manhattan distance from current state. More will be the minimum distance, less will be the score. That's why returning inverse of minimum manhattan distance as the factor influencing score of the state.

Statistics:

Test Command Run: **python pacman.py -p ReflexAgent -l testClassic**

Result : Pacman emerges victorious with score of 564.0

Test Command Run: **python pacman.py –frameTime 0 -p ReflexAgent -k 1**

Result : Pacman emerges victorious with score of 1506.0

Test Command Run: **python autograder.py -q q1**

Result :
Pacman emerges victorious! Score: 1246
Pacman emerges victorious! Score: 1237
Pacman emerges victorious! Score: 1235
Pacman emerges victorious! Score: 1245
Pacman emerges victorious! Score: 1233
Pacman emerges victorious! Score: 1231
Pacman emerges victorious! Score: 1235
Pacman emerges victorious! Score: 1253
Pacman emerges victorious! Score: 1231
Pacman emerges victorious! Score: 1236

Average Score = 1238.2

Answer 2) Minimax Search

Approach:

1) We start with the Max node as the root and do a depth-first traversal upto a fixed depth 'depth' specified by the caller.
2) We need to keep switching between Max and min functions while we are in a recursive call to make sure they propagate the min and max values respectively.
3)We keep updating the minimum value seen by a min node so far and maximum value seen by the max node so far in order to progate these value up the tree.
4) At the terminal nodes or nodes where we reach the specified depth, we just return with the values obtained by the evaluation function at that node/state.

Statistics:
Test Command Run: **python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=2**
Average Score: -492.0
Scores: -492.0
Win Rate: 0/1 (0.00)
Record: Loss

Test Command Run: **python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=3**
Average Score: 511.0
Scores: 511.0
Win Rate: 1/1 (1.00)
Record: Win

It can be seen that Pacman both looses and win in different cases, which is expected in minmax algorithm.

Test Command Run: **python autograder.py -q q2**
All tests were passed in the autograder.

Answer 3) Alpha-Beta Pruning

Approach:

1)We start with the Max node as the root and do a depth-first traversal upto a fixed depth 'depth' specified by the caller. There are two extra variables that we keep at each state which are alpha and beta.
2) We need to keep switching between Max and min functions while we are in a recursive call to make sure they propagate the min and max values respectively.
3)We keep updating the minimum value seen by a min node so far and maximum value seen by the max node so far in order to propagate these value up the tree. While we are updating these values, we also need to check if at any point alpha value becomes greater than beta value. In such case be just break our search there and return as we will not find a better solution by looking in that part of the tree anyways.
4) At the terminal nodes or nodes where we reach the specified depth, we just return with the values obtained by the evaluation function at that node/state.

Statistics:
Test Command Run: **python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic**
Average Score: 49.0
Scores: 49.0
Win Rate: 0/1 (0.00)
Record: Loss

Test Command Run: **python pacman.py -p AlphaBetaAgent -a depth=2 -l smallClassic**
Average Score: -252.0
Scores: -252.0
Win Rate: 0/1 (0.00)
Record: Loss

Test Command Run: **python autograder.py -q q2**
All tests were passed in the autograder.

Answer 4) Expectimax

Statistics:

Test Command Run: **python pacman.py -p ExpectimaxAgent -l minimaxClassic -a depth=3**

Result : Pacman emerges victorious with score of 512.0

Test Command Run: **python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 10**

Result :
Pacman emerges victorious! Score: 531
Pacman emerges victorious! Score: 531
Pacman died! Score: -502
Pacman died! Score: -502
Pacman emerges victorious! Score: 531
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 531
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Average Score: 118.0
Scores: 531.0, 531.0, -502.0, -502.0, 531.0, 532.0, 531.0, -502.0, 532.0, -502.0
Win Rate: 6/10 (0.60)
Record: Win, Win, Loss, Loss, Win, Win, Win, Loss, Win, Loss