CS-182 Assignment 2

Charles Liu
cliu02@g.harvard.edu

1: Compare the performance of your pacman alpha-beta agent when it plays against the directional ghost, the random ghost and a mini-max ghost. Analyze and discuss your results.

I ran the alpha-beta agent with depth 4 on the default mediumClassic game with my better eval function 5 times each. On average the scores on the directional ghost were slightly higher (1571.4 vs. 1595.4) which seems to be due to the fact that sometimes pacman can stall on one part of the board when it is far removed from both the ghost and food. That is because either the minimax depth isn't enough plys or the evaluation function isn't quite good enough to differentiate moves. When the ghosts come closer to pacman, then the lower scoring Gamestates enter the tree and pacman subsequently moves accordingly. This happens much faster on a smarter ghost.

2: If you were to design a ghost agent to play with people, how would you go about choosing a strategy for the ghost? Think about the strategies people might use when playing this game, and how they might differ from strategies played by computer agents.

I would try to corner rather than try to catch the player. The goal for my agent would be to waste as much time as possible, because your score is constantly decreasing. If you're only one ghost then the likelihood of actually catching pacman is quite small, but if it's in a corner and you're able to move back and forth around an edge to keep him from advancing it's just as good.

3:

- **Assume alpha-beta runs depth-first from left to right. For which values of A and B, will the algorithm prune nodes C and D?**
- **Did alpha-beta have to expand the node with value 6? Explain.**

The value from the left child is 6. That means the result can be at most 6, so if both values of A and B were ≥ 6, the search could immediately stop because the right max node would return a value that is at least as large as the left node.

The node with value 6 did have to be expanded. The min chosen in the lower right is $min(9, 2) = 2$, so when the max node traverses to the right min child, it first sees a 7. Since 7 is larger than 2, the max node does not know for sure that the 2 is the optimal move. If that 7 had been a 1, then the max node would know that the next min move would output at most a 1, which is already sub-optimal to the 2 that it had achieved in the left child. Since it's not, it has to look at the node with value 6.