Project 2 - Multiagent Search

Scott Weeden

October 29, 2025

1 Question 1 – Minimax Strategy

First thing I implemented was a minimax agent. Pacman (agent 0) is the one trying to score big, while every ghost is doing its level best to drag that score down. I cycle the agent index with `next_agent = (agent_index + 1) % num_agents`, and I only bump the depth after the last ghost in the lineup takes a swing so one depth really means a full round of moves. Any time we hit a win/lose state or run out of depth, I hand the state over to `self.evaluationFunction`. I also make sure to ask `getLegalActions` only once per node, in the exact order the game gives them. I field-tested this setup on `minimaxClassic` (depth 4) and `trappedClassic` (depth 3), then let `autograder.py -q q1` run all 33 checks—everything passed, so the depth bookkeeping looks solid.

2 Question 2 – Alpha-Beta Strategy

Next up I implemented alpha-beta pruning onto that same minimax frame. Pacman's branch keeps an eye on the best score so far (alpha) and ghosts track the worst they can force (beta). Whenever the max branch finds something bigger than beta, or the min branch finds something smaller than alpha, the algorithm bail out early—no need to keep digging. I walk the successors in the exact order `getLegalActions` hands them over so the autograder stays happy about node counts. Depth and terminal handling are copied straight from the minimax agent. The depth-3 run on `smallClassic` finishes in just a few seconds, and `autograder.py -q q2` (33 tests plus a game check) passed all checks.

3 Question 3 – Expectimax Strategy

For expectimax I didn't assume the worst case (intelligent ghosts) and instead treat them as random drifters. Pacman branches still grab the best successor, but ghost turns average the values with `sum(values)/len(values)` because RandomGhost spreads its moves evenly. Depth and terminal logic stay untouched. I ran ten depth-3 games on `trappedClassic`: AlphaBetaAgent got clobbered every time (average -501), while ExpectimaxAgent snuck out six wins and averaged about 118 points. The `autograder.py -q q3` suite (16 tests plus a game) backed up the implementation.

Advanced Questions

A1. From Minimax to Alpha-Beta. All I had to do was thread `alpha` and `beta` through the recursive helper while keeping the minimax skeleton intact. Depth tracking, terminal checks, and the evaluation function didn't need touching. The pruning just chops off branches once the bounds tell me nothing better can show up, and I leave the successor order alone.

A2. From Minimax to Expectimax. This tweak swaps the ghosts' "take the minimum" behavior for "take the average." I gather the successor scores into a list, average them, and pretend the ghosts are rolling dice. Pacman's logic, depth control, and evaluation flow stay the same, so the agent slides right into the existing framework.