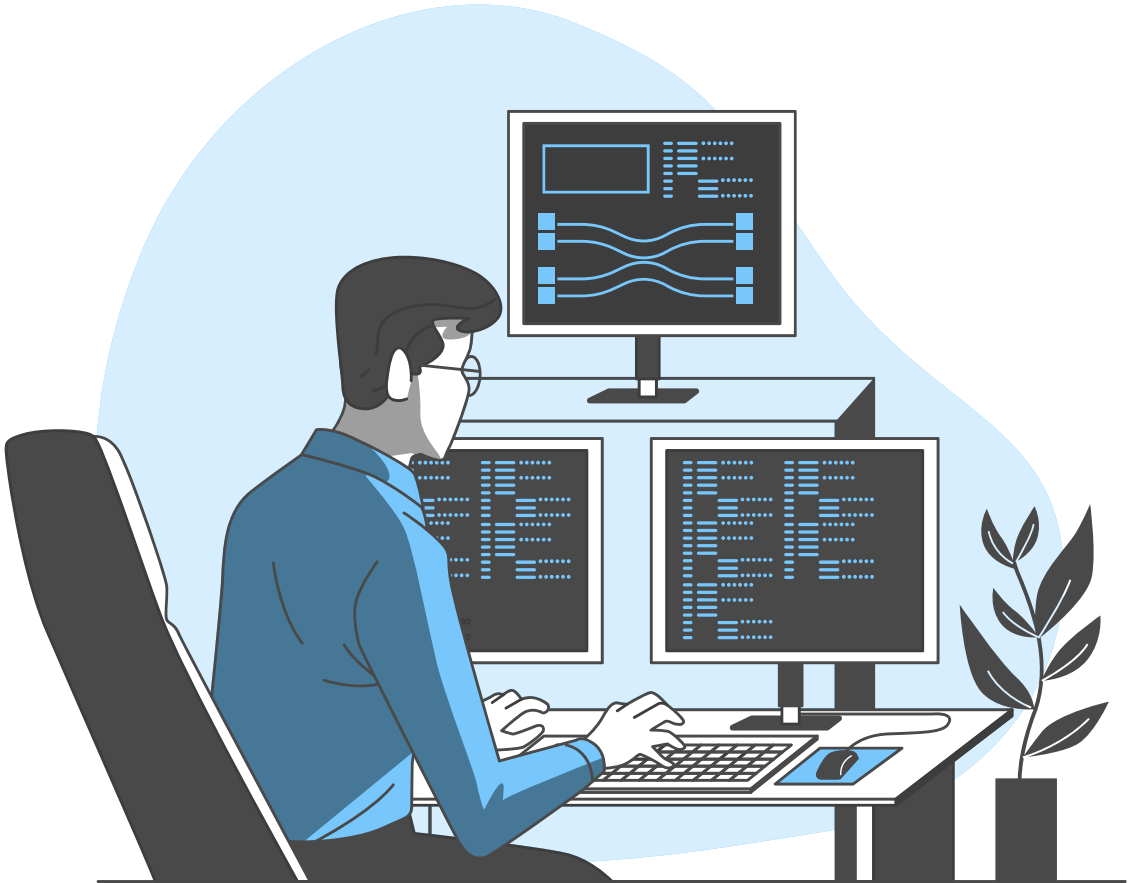

Wolf Eats Sheep AI GAME Algorithm

LUO Peiyuan
56642728

My Design Strategy

FrameWork: AB MINMAX

+

Heuristic Evaluation for Wolf/Sheep

AB(Alpha-Beta Pruning) MINMAX

Max Depth for Wolf: 4
Max Depth for Sheep: 3

Objective: To simulate several rounds of the game and use the max value of the evaluation result for my turn and find the minimum of the evaluation result for opponent's turn.

Evaluation Function: Heuristic Evaluation Function for Sheep and Wolf.

Location: I use ab minmax function inside getBestMove() function to find the best move using ab minmax method.

Ending condtion: to end the recursion function of ab minmax, the triggering condition is either game ends(one of the player wins) or currentDepth==maxDepth.
Field of the ab minmax

Wolf Heuristic function

H(Wolf) = 500 x num_of_sheep_killed + 300 x num_of_to_be_killed_sheep + 20 x shorten_distance_from_sheep_to_wolf

num_of_sheep_killed: the number of sheep that are killed between the previous taken move and current depth, calculated by the number of the sheep before this turn minus the number of sheep in current state.

- Importance: very high, because the more sheep is eaten by wolf, the higher probability the wolf would win.

num_of_to_be_killed_sheep: after all the moves made, the number of sheep that can be killed by the wolf in next turn of wolf, calculated by counting the number of sheep that are only one empty column away from the wolf.

- Importance: high, because the more sheep can be killed by wolf in next move, the higher probability the wolf would win under this situation. But the weight should be lighter than the 500.

shorten_distance_from_sheep_to_wolf: the shortened distance between all wolves and sheep after all the move is made, calculated by the distance of sheep and wolves before this turn minus the distance of sheep in current state.

- Importance: high, as the closer the wolf is to sheep, the higher probability the wolf would win.

Sheep Heuristic function

H(Sheep) = num_of_sheep_killed x (-20000) + trapped_wolf_num x 4000 + shorten_distance_from_sheep_to_wolf x (-100) + num_of_trapped_ways x 800 - num_of_to_be_killed_sheep

num_of_sheep_killed: the number of sheep that are killed between the previous taken move and current depth, calculated by the number of the sheep before this turn minus the number of sheep in current state.

- Importance: very high, because the less sheep is eaten by wolf, the higher probability the sheep would win.

trapped_wolf_num: the number of wolves that are trapped by sheep(can't move any more). Each wolf has four ways (above, below, right, left). If all four ways of a wolf are trapped, then we say the wolf is trapped.

- Importance: high, the more wolves that are trapped, then the less dangerous for sheep, and the more probability for sheep to trap the wolf and wins.

shorten_distance_from_sheep_to_wolf: the shortened distance between all wolves and sheep after all the move is made, calculated by the distance of sheep and wolves before this turn minus the distance of sheep in current state.

- Importance: high, as the farer the sheep is wolves, the higher probability the wolf would win, because the wolf would take more moves to get close to sheep.

num_of_trapped_ways:the number of ways that the wolf' move is trapped. Each wolf has four ways (above, below, right, left). If a ways is trapped then the wolf can't move in this direction.

- Importance: high, the more ways that the wolves are trapped, then the more limit for wolves' move, then the less probability for wolf to win, and the more probability for sheep to trap the wolf and wins.

num_of_to_be_killed_sheep: after all the moves made, the number of sheep that can be killed by the wolf in next turn of wolf, calculated by counting the number of sheep that are only one empty column away from the wolf.

- Importance: very low. Because for wolf's turn, it needs to wait for another turn to kill the to-be-killed sheep, which increase the uncertainty of this value. Hence, I only use this value to differ slightly.