1 Evaluation Function

My evaluation function is identical for both players. It is principally very simple, but is quite computationally intensive. The total value is the sum of the following 2 heuristics:

1.1 Length Difference

This is the signed difference in length between the player and its longest enemy (positive if the player is winning, negative if the player is losing). This encourages the player to eat food, while also incentivising it to kill its biggest competition, even when it is in first place.

When losing, you only care about increasing your length past your longest enemy. The difficulty of this is negligibly affected if one of the shorter enemies gets food. When winning, the shorter an enemy is, the less likely they are to kill or overtake you. In both cases, the closer the enemies scores, the more prominent the value of just eating, as the value of killing an enemy is diminished. Considering only the longest enemy is a good way to maximise effectiveness for minimal computation.

1.2 A* Distance

Here I used 1/(manhattan-distance to disk), as calculated by my own implementation of A* path-finding. I modified the algorithm to look for distance rather than paths, and also optimised it for the grid-based environment where there are often many equally-short paths to a given target.

I take one over the distance so that moving towards the food results in a higher score, but eating the food is still a favourable move. It is guaranteed that the value lost from ending up further away from food is less than the 1 gained in the length difference heuristic.

The player would naturally take the best path using only the length difference heuristic if it could look enough moves into the future to reach the food. Instead, the snake wandered around blindly, losing out on food even when it was the closest player.

2 Results and Observations

The table below shows the results from various player matchups (separated by horizontal lines) over sets of 100 games. The Mean column shows the mean score the player achieved across all the games in which they did not die. If players were tied for victory, they received one tie, and no wins.

Comparing the two minimax players, the paranoid player performs better in general, both in minimising death, and maximising length. It is able to reach deeper levels in its search tree thanks to alpha-beta pruning, and dies much less due to its paranoid outlook. The non-paranoid player's biggest weakness is assuming other players will take specific paths to reach the goal, where in reality there are many equally good paths they might take. It believes it can safely be close to enemies and dies when they do not move as it predicted. This issue could be alleviated with an evaluation function that puts more value in killing any enemy, rather than just those with the best score.

Player	**Wins**	**Ties**	**Deaths**	**Mean**
Paranoid	34	11	5	19.16
Non-Paranoid	31	11	24	17.30
A*	20	8	54	17.96
Random	0	0	33	4.81
Paranoid	75	4	5	48.82
A*	21	4	61	37.26
Non-Paranoid	65	6	10	47.91
A*	29	6	58	43.67
Paranoid	62	7	5	39.44
Non-Paranoid	31	7	26	34.50

The Paranoid player performed worse in the 4-player setup than in others. The average scores in 4-player are lower, and so random chance plays a larger role. Additionally, the paranoid player is sometimes too scared of other players, when in reality most them are very naive.

Both minimax players massively outperformed A* in terms of wins. A* still scored highly when it survive, but it often lost by dying due to its less complex method of evaluation. Minimax has proven very effective for this game, because it allows the balancing of foresight and evaluation complexity.