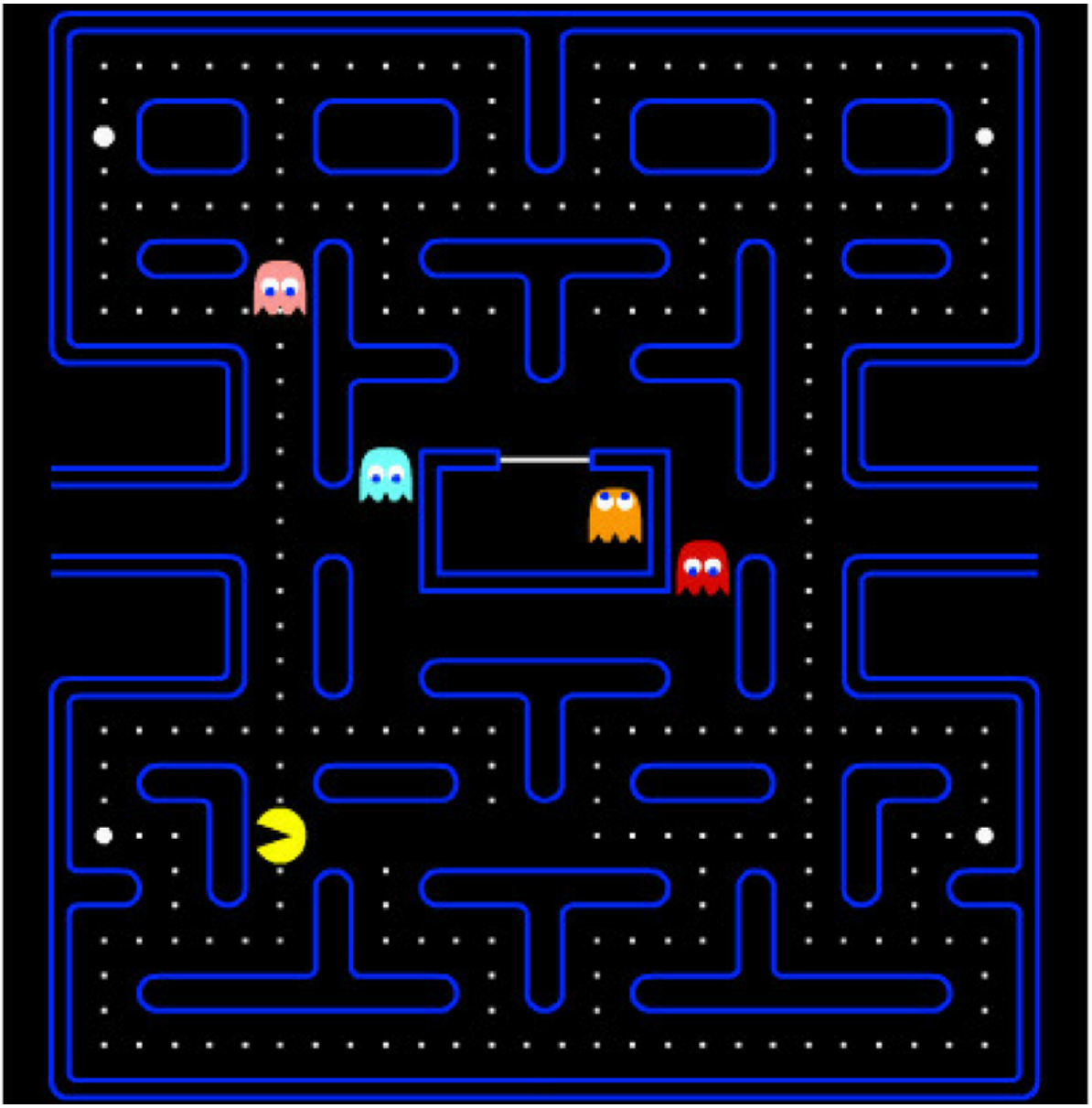
Project 2: Multi-Agent Search



REFLEX AGENT

Before starting with the question, the first thing to be done is to inspect the code of reflexAgent and understand how it works. Once that is done, the goal is to improve the agent to perform respectably and that will be done by improving the evaluation function.
The one that is provided just returns the score of the following state. The idea is to return a value to show how better the following state is compared to the previous one.
As a small reminder, on the lectures we have seen that in practice evaluation functions are weighted linear sum of features:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

So, the first thing I thought is defining which features to have:

- Feature 1: the score. Already provided by default, it only makes preferable the actions that end up in positions where there is food.
- Feature 2: distance to the ghosts. The idea is that we want the pacman as far as possible from the ghost/s. If there are multiple ghosts, we'll just track the distance to the closest ghost.
- Feature 3: distance to food. The pacman has to give priority to positions closer to food.
- Feature 4: the action to stop. Didn't even know that it exists, we won't allow the pacman to stop it doesn't provide any benefit.

Then after trying different ways of combining the features, the best I got is:

$$f = score + closest.ghost.distance - furthest.food.distance + stop$$

where stop is -1000 if the action is Stop and 0 otherwise. With this function I achieved a score of ¾ on the test. I was curious about which type of function can give the best results and found the following one:

$$f = score + (closest.ghost.distance / closest.food.distance) + stop$$

The idea here is that instead of only using the values, we have in mind the relation between them.

MINIMAX/ALPHA-BETA PRUNING

Both of these algorithms have been implemented from the following pseudocode:

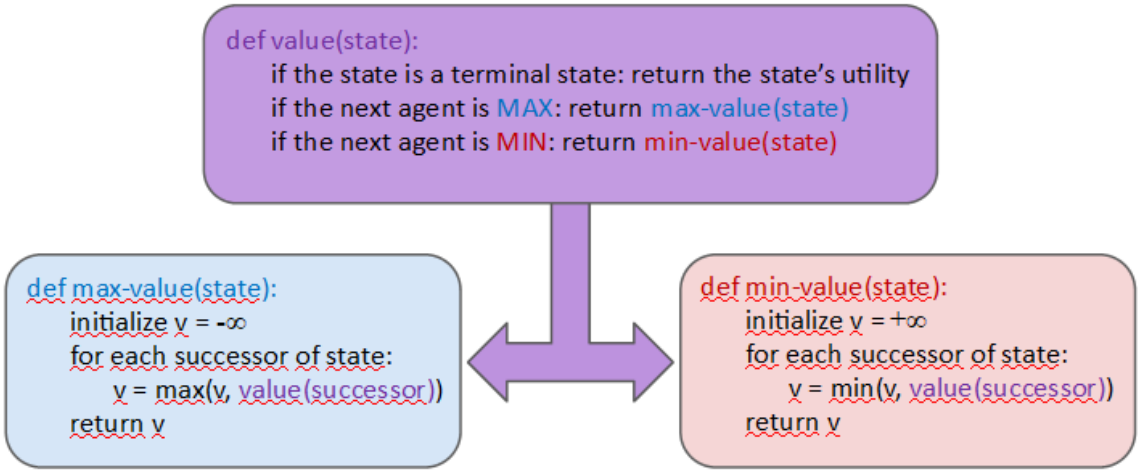


figure 1: minimax pseudocode

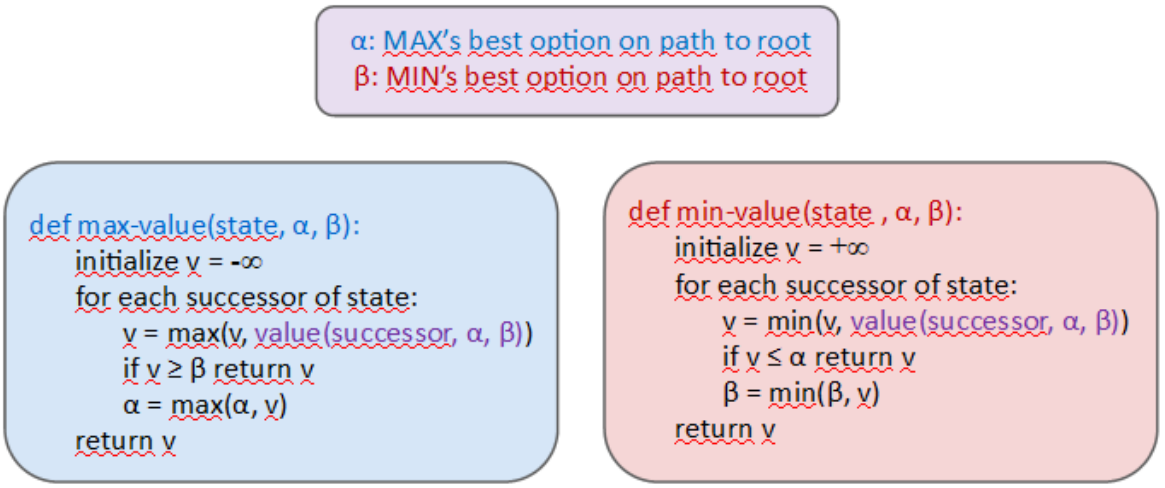
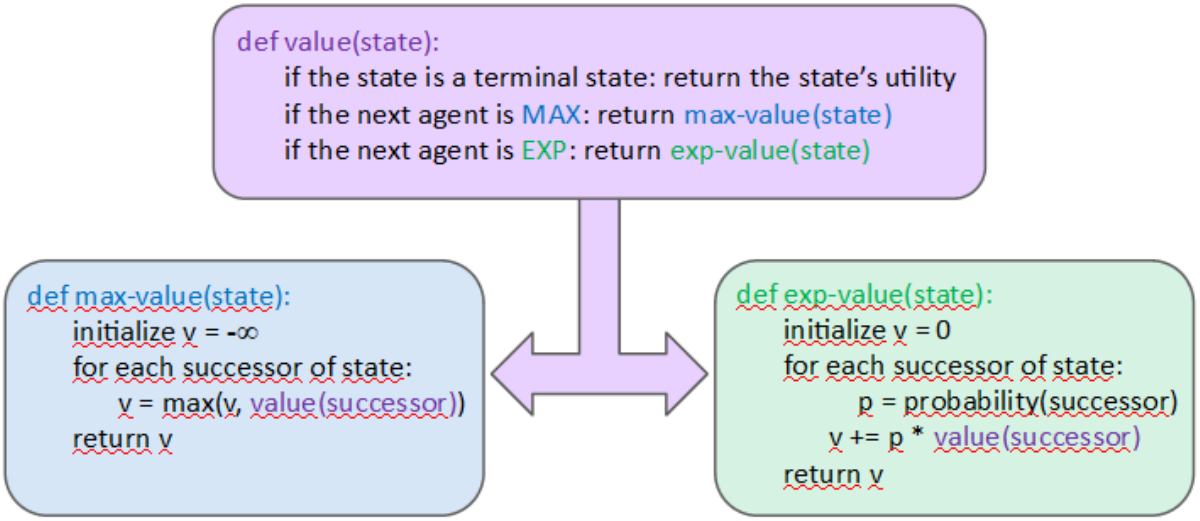


figure 2: alpha-beta pruning pseudocode

Since they both are practically identical I decided to implement max-value (as pacman-value-action) and min-value (as ghost-value-action) inside the class MultiAgentSearchAgent, which is a superclass of both MinimaxAgent and AlphaBetaAgent.
The idea was to create one function for both cases just by defining alpha and beta as optional and also the additional steps of the alpha beta pruning.

EXPECTIMAX

Expectimax has the following pseudocode:



It was supposed to be implemented inside the class ExpectimaxAgent which is also a subclass of MultiAgentSearchAgent. Therefore, since max-value is the same as in minimax I only reimplemented in this subclass the exp-value (as ghost-value-action in my code).

EVALUATION FUNCTION

For the evaluation function I started by trying the most obvious one which is

$$f = score - closest.food.distance$$

and just with this simple function you'll get the maximum score.