Game: A game involves a single agent or multiple agents, where each agent has its own utility to be maximized (utility-based agent).

Zero sum vs. general games: One simple form of game is the zero-sum game. Tic-tac-toe, Chess, and Go are also zero-sum games: one person wins and the other loses (or draws). It is also referred to as an adversarial game. There are general games where agents also need to collaborate, such as StarCraft, Minecraft, etc.

Type of games: There are many different types of games that correspond to the different task environments.

Formulation of Games: One possible formulation of games in a sequential setting: agents take turns in playing the game. Here, the solutions are policies since you must prescribe your agent’s next action under any possible state (due to the unpredictability of the other agents).

Search tree – revisited: Let us first look at how we can solve a game with a single agent. For example, let’s say that we want our Pacman to eat the dot while minimizing the number of steps (i.e., cost). This can be easily converted to a search problem! We only need to minimize the number of steps.

Single-agent game tree: One of the problems with such a search method is that the cost is often unknown since the utility is often determined by the end of the game state (think about a chess program). This can be addressed by introducing the notion of value function, V(s): the best achievable outcome from a state. If we can compute such a function, we should also be able to compute a policy easily by choosing the next node that leads to a better utility.

Such state values for a game tree can be constructed from the bottom up.

Adversarial game tree: An adversarial game tree can be similarly constructed with the exception that now you must also consider the actions of your opponent.

When it is your turn, you tend to choose the best valued node from among the child nodes. For your opponent, it will do just the opposite (assuming a rational opponent). In this way, you will be able to construct the game tree from the bottom up (often in a recursive manner). The two agents are often referred to as the MAX and MIN players.

You can have multiple MAX and multiple MIN players in a tree.
Minimax: An adversarial game tree is also referred to as a minimax tree, and the state value referred to as the minimax value.

Alpha-beta pruning: Computing the entire tree is often expensive. We can do better by pruning parts of the tree that do not need to be computed.

Minimax example: You may have noticed that the two leaf nodes in the middle of the minimax tree do not need to be checked since the left subtree already provides a better solution for MAX (with a value of 3).

Alpha-beta pruning: This pruning has no effect on minimax value computed but values for the intermediate nodes can be wrong. This also means that choosing actions based on these values could lead to the wrong decision! Fortunately, it is not difficult to address this issue (left as a thought exercise for you). Hint: by changing the tie-breaking behavior in your pruning.

Depth matters: Contrast the depth and evaluation function with the g and h values, respectively, for estimating the f value. The deeper you go, the more accurate g you have and hence the h becomes less important.