Till now we have seen the classic game rules in which lions were stationary. They don’t care what you do but they protect their territory. Now lions are no longer stationary, they are out for a HUNT!!!

The only goal of the lion is to capture and kill you , and your goal as a adventurer is to retrieve all treasures.

Files to read through :

MultiAgents.py – To understand the basic outline of any minimax agent

game.py – to get an idea about gameState class

Files to be altered :

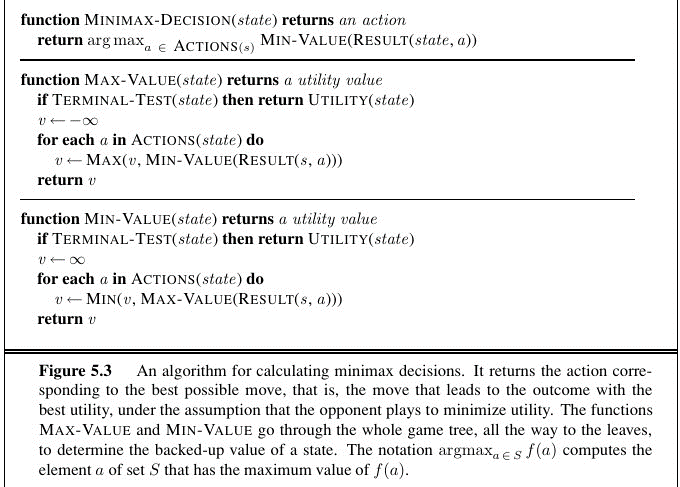
MultiAgents.py

Q1. Minimax Algorithm:

Minimax is a foundational AI strategy for adversarial decision-making, commonly used in games like chess, where it aims to minimize the worst-case outcome by considering opponent moves and selecting the optimal action to maximize one's chances of success. In this module we will treat lions as min agents and adventurer as max agent.

The gamestate stores the info about number of lions in num\_agents . In multiagents class , a depth is considered to be 1 turn of max agent followed by turns of all the min agents. We increment the depth only after all max agents and all min agents have moved once. The minimax algorithm implemented here will only look through the game search tree only till the depth of 2.

Pseudo code for minimax :



Implement the function getAction in MultiAgents.py which returns the action that the max agent has to take at any given gamestate. You can declare any function within the MinimaxAgent class but note that only getAction function will be run for grading purposes.



To implement the minimax strategy run the following command :

**python dungeon.py -p agent -method minimax -graphics True -world world.txt -omni T**

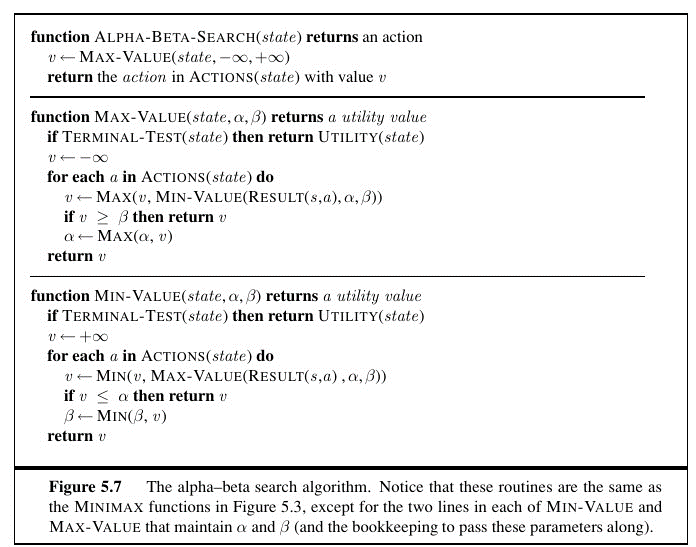
**python dungeon.py -p agent -method minimax -graphics True -world world2.txt -omni T**

Q2. Alpha beta pruned minimax :

Alpha-beta pruning is a clever optimization technique applied to the Minimax algorithm in AI, reducing the number of evaluated game states by disregarding branches that cannot affect the final decision, thereby significantly speeding up search in games and adversarial scenarios.

In this part , you need to implement getAction function in AlphaBetaMinimax class in MultiAgents.py

Pseudo code for alpha beta pruning:



To see the agent under evaluated policy run the following command:

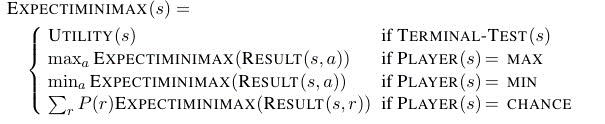
**python dungeon.py -p agent -method abminimax -graphics True -world world.txt -omni T**

**python dungeon.py -p agent -method abminimax -graphics True -world world2.txt -omni T**

Q3. Expecti-minimax :

ExpectiMinimax, an extension of the Minimax algorithm, is employed in AI for decision-making under uncertainty, modeling probabilistic outcomes to find strategies that minimize the maximum expected loss when facing uncertain or stochastic environments.

In this part , you need to implement getAction function in ExpectiMinimax class in MultiAgents.py



To see the agent under evaluated policy in a unveiled world run the following command:

**python dungeon.py -p agent -method expectiminimax -graphics True -world world.txt -omni T**

**python dungeon.py -p agent -method expectiminimax -graphics True -world world2.txt -omni T**