AI Project Report:

The Knights Problem

Hadile Mahmoud 40-1187

Mahmoud Ahmed Abdelwahab 43-13997

Nourhan salaheldin mahdy 43-5139

Brief Description:

The problem consists of Queens and Knights and the goal is to kill all queens in the minimum amount of steps.

Search-tree node class:

We created a public abstract class Node with 5 variables, state, parent, operator, depth, path Cost. The state is a 2d Byte array variable representing the current state of the game with all the needed information. Parent is a Node object representing the parent node to be able to backtrack the sequence of actions that led to this goal. Operator is the action that generated this node from its parent. Depth represents the depth of the node. Finally, the path Cost is a variable that will be decided by the path cost function implemented in the knights problem class. We implemented the Search-tree using a priority queue and we give it the queuing function based on the search function.

Search Problem Class:

**Problem Class:**

Is an abstract class with 3 variables and 2 functions; initial state, state space, goal state, generate state space and path cost. Initial State is the current state of the problem, state space is all the possible states from the current state, and the goal state is the state that we compare every state with. Abstract method Generate state space. Path cost takes an operator and returns its cost.

**General Search:**

Is a class that contains seen state HashMap variable and 2 functions. It is initialized with a problem and a queuing function. The 2 functions: solve and generate children. Solve consists of a while loop that is checking on the queue if the queue is empty before finding a goal state, then there is no solution for the problem, else the function will keep on generating children and adding them to the queue, but we always check that we haven’t seen this state before so we don’t go into cycles or run into an infinite loop. Generate children takes a node and generates all its unique possible states to the queue.

**Knights Problem:**

We implemented a class that consists of 3 variables: board 2d array, knights array and queens array and a all possible states function. Board array is made of bytes a negative value means there’s a queen at this cell, positive value means it is a knight and zero is empty. Knights array consists of knights and knights consists of x, y, stamina, and id. Queens array consists of queens and queens consists of x, y, and id. All possible states function takes a knight and the board and it returns all the possible states for this knight on this board. Generate state space takes the Initial State and generates all the possible states.

**Main Functions:**

1. **genBoard (int max Queens, int max Knights, int max Stamina, int board Size):** The method is in the Knights problem class a random search problem with random numbers for all variables with respect to their limits and the position of other variables so that no two variables will overlap in the same cell.
2. **Solve ():** The solve method in General Search class. It consists of a while loop that is checking on the queue if the queue is empty before finding a goal state, then there is no solution for the problem, else the function will keep on generating children and adding them to the queue, but we always check that we haven’t seen this state before so we don’t go into cycles or run into an infinite loop.
3. **genChildren (Node node):** Generate children takes a node and generates all its unique possible states to the queue.
4. **moveKnight (state, knight, movement):** it takes a knight as a parameter and moves it on the board and then it returns a new state.
5. **allPossibleStates (knight, board):** All possible states function takes a knight and the board and it returns all the possible states for this knight on this board
6. **pathCost(operator):** Path cost takes an operator and returns its cost. We prefer to kill queen over moving a knight and moving a knight over inspire.