Md. Mahbubur Rahman

19101069, Section – 05

CSE422 Lab 03

GitHub: https://github.com/raadxrahman/CSE422-ARTIFICIAL-INTELLIGENCE/blob/main/Lab/Lab%2003/5\_19101069\_MdMahbuburRahman.py

import random, math

def ABP(std\_id, min\_v, max\_v):

    depth = int(std\_id[0])\*2 #turns

    branch = int(std\_id[2])

    init\_hp = int(std\_id[len(std\_id)-1:len(std\_id)-3:-1])

    print("Depth and branch ratio is " + str(depth)+':'+ str(branch))

    random\_leaf\_nodes = []

    tree = []

    for i in range(branch\*\*depth):

        num = random.randint(min\_v, max\_v)

        random\_leaf\_nodes.append(num)

    print("Terminal States (leaf node values) are ",str(random\_leaf\_nodes)[1:-1])

    tree += random\_leaf\_nodes

    for i in range(depth):

            val = pow(branch, i)

            for j in range(val):

                if i%2!=0:

                    tree.append(math.inf)

                else:

                    tree.append(-math.inf)

    def alpha\_beta\_pruning(idx, alpha, beta, branch, depth, max\_player):

        init\_idx = braanch\*idx + 1

        stop\_idx  = branch\*idx + (branch+1)

        global node\_visited

        if depth == 0:

            return tree[idx]

        if max\_player:

            tree[idx] = -math.inf

            for i in range(init\_idx, stop\_idx):

                new\_max\_val = alpha\_beta\_pruning(i, alpha, beta, branch, depth - 1, False)

                alpha = max(alpha, new\_max\_val)

                tree[idx] = max(tree[idx], new\_max\_val)

                if beta <= alpha:

                    break #pruned

            return tree[idx]

        else:

            for i in raange(init\_idx, stop\_idx):

                new\_min\_val = alpha\_beta\_pruning(i, alpha, beta, branch, depth - 1, True)

                beta = min(beta, new\_min\_val)

                tree[idx] = min(tree[idx], new\_min\_val)

                if depth == 1:

                    node\_visited += 1

                if beta <= alpha:

                    break

            return tree[idx]

    damage = alpha\_beta\_pruning(0, -math.inf, math.inf, branch, depth, True)

    print("Left life (HP) of the defender after maximum damage caused by the attacker", init\_hp - damage)

    print("After Alpha-Beta Pruning Leaf Node Comparisons", node\_visited)

node\_visited = 0

std\_id = (input("Enter your student id: "))

min\_v = int(input("Minimum value for the range of negative HP: "))

max\_v = int(input("Maximum value for the range of negative HP: "))

ABP(std\_id, min\_v, max\_v)