import random

import math

class Node:

def \_\_init\_\_(self, value):

self.value = value

self.left = None

self.right = None

class AlphaBeta:

def \_\_init\_\_(self, id):

self.id = id

root = self.tree\_builder()

point = self.minimax(root, 0, -math.inf, math.inf, True)

self.print\_winner(point)

def tree\_builder(self):

min\_point = int(self.id[4])

max\_point = math.ceil(int(self.id[-1] + self.id[-2]) \* 1.5)

array = self.random\_array(min\_point, max\_point)

print(f"Generated 8 random points between the minimum and maximum point limits: {array}")

root, \_= self.build\_binary\_tree(array)

return root

def random\_array(self, min, max):

arr = []

for i in range(8):

random\_num = random.randint(min, max)

arr.append(random\_num)

return arr

def build\_binary\_tree(self, array, depth=0, i=0):

if depth != 3:

if depth % 2 == 0:

node = Node(-math.inf)

node.left, i = self.build\_binary\_tree(array, depth+1, i)

node.right, i = self.build\_binary\_tree(array, depth+1, i)

return node, i

elif depth % 2 != 0:

node = Node(math.inf)

node.left, i = self.build\_binary\_tree(array, depth + 1, i)

node.right, i = self.build\_binary\_tree(array, depth + 1, i)

return node, i

if depth == 3:

node = Node(array[i])

return node, i + 1

def minimax(self, node, depth, alpha, beta, maximizingPlayer):

if depth == 3 or (node.left == None and node.right == None):

return node.value

if maximizingPlayer:

maxEval = -math.inf

children = [node.left, node.right]

for child in children:

eval = self.minimax(child, depth + 1, alpha, beta, False)

maxEval = max(maxEval, eval)

alpha = max(alpha, eval)

if alpha >= beta:

break

return maxEval

else:

minEval = math.inf

children = [node.left, node.right]

for child in children:

eval = self.minimax(child, depth + 1, alpha, beta, True)

minEval = min(minEval, eval)

beta = min(beta, eval)

if alpha >= beta:

break

return minEval

def print\_winner(self, point):

points\_needed = int(self.id[-1] + self.id[-2])

print(f"Total points to win: {points\_needed}")

print(f"Achieved point by applying alpha-beta pruning: {point}")

if point < points\_needed:

print("The Winner is Megatron")

else:

print("The Winner is Optimus")

id = input("Enter your ID: ")

AlphaBeta(id)