CSE422 Lab03

Alpha Beta Pruning

Name: Subhi Bhuiyan

ID: 17201116

turns = int(input("Enter number of turns: "))

depth = turns \* 2

branch = int(input("Enter number of branches per node: "))

#value for the range of leaf nodes

minimum = int(input("Minimum value: "))

maximum = int(input("Maximum Value: "))

#leaf nodes

terminal\_states = branch \*\* depth

score = []

for i in range(terminal\_states):

y = int(input("Enter node value: "))

if y>=minimum and y<=maximum:

score.append(y)

alpha = 1000

beta = -1000

pruneCount = 0

def alpha\_beta(depth, node, score, maximizingPlayer, alpha, beta):

global pruneCount

if depth==0 or node==None:

return score[node]

if maximizingPlayer:

maxEval = -100000

for child in range(depth):

n = (node\*2)+child

value = alpha\_beta(depth-1, n, score, False, alpha, beta)

maxEval = max(maxEval,value)

alpha = max(alpha,value)

if beta <= alpha:

pruneCount += 1

break

return maxEval

else:

minEval = 100000

for child in range(depth):

n = (node\*2)+child

value = alpha\_beta(depth-1, n, score, True, alpha, beta)

minEval = min(minEval,value)

beta = min(beta,value)

if beta <= alpha:

pruneCount += 1

break

return minEval

#Sample Output

print("Depth:",depth)

print("Branch:",branch)

print("Terminal States (Leaf Nodes):",terminal\_states)

x = alpha\_beta(depth, node, score, True, alpha, beta)

print("Maximum amount:",x)

print("Comparisons Before Alpha-Beta Pruning: ",terminal\_states)

y = terminal\_states - pruneCount

print("Comparisons After Alpha-Beta Pruning: ",y)

#print(pruneCount)