**Name: Mohammad Zain Abbas**

**Reg #: 6865**

**DE-36 (CE), Syndicate: A**

**LAB 9 JOURNEL**

**Equipment Used:** Notebook Computer, Python IDLE 3.6

**Lab Tasks:**

1. Implement minax algorithm for Tree 1 and Tree 2. Import the time module and

calculate the total time taken by your minimax algorithm.

**SOLUTION CODE:**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Dec 1 14:16:36 2017

@author: umerm

"""

import numpy

import cv2

from math import inf

import time

class Tree():

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

self.left = None

self.right = None

self.middle=None

self.data = None

def minimax(node,depth,maximizing\_player,path):

if(depth==0 or (node.left==None and node.middle==None and node.right==None)):

path=path+[node.data]

print("path = ",path)

return node.data;

if maximizing\_player:

best\_value=-1\*float("inf")

for child in [node.left , node.right , node.middle]:

value=minimax(node,depth-1,False,path)

best\_value=max(best\_value,value)

path=path+[best\_value]

return best\_value

else:

best\_value=float("inf")

for child in [node.left , node.right , node.middle]:

value=minimax(node,depth-1,True,path)

best\_value=min(best\_value,value)

path=path+[best\_value]

print("path = ",path)

return best\_value

def main():

Tree1 = Tree()

Tree1.data = -7

Tree1.left = Tree()

Tree1.left.data = -10

Tree1.right=Tree()

Tree1.right.data=-7

Tree1.left.left = Tree()

Tree1.left.left.data = 10

Tree1.left.right=Tree()

Tree1.left.right.data=-10

Tree1.left.left.left=Tree()

Tree1.left.left.left.data=10;

Tree1.left.left.right=Tree()

Tree1.left.left.right.data=5

Tree1.left.left.left.left=Tree()

Tree1.left.left.left.left.data=10;

Tree1.left.left.left.right=Tree()

Tree1.left.left.left.right.data=float("inf");

Tree1.left.left.right.left=Tree()

Tree1.left.left.right.left.data=5

Tree1.left.right.left=Tree()

Tree1.left.right.left.data=-10

Tree1.left.right.left.left=Tree()

Tree1.left.right.left.left.data=-10

Tree1.right.left=Tree()

Tree1.right.left.data=5

Tree1.right.left.left=Tree()

Tree1.right.left.left.data=5

Tree1.right.left.right=Tree()

Tree1.right.left.right.data=-1\*float("inf")

Tree1.right.left.left.left=Tree()

Tree1.right.left.left.left.data=7

Tree1.right.left.left.right=Tree()

Tree1.right.left.left.right.data=5

Tree1.right.left.right.left=Tree()

Tree1.right.left.right.left.data=-1\*float("inf")

Tree1.right.right=Tree()

Tree1.right.right.data=-7

Tree1.right.right.left=Tree()

Tree1.right.right.left.data=-7

Tree1.right.right.left.left=Tree()

Tree1.right.right.left.left.data=-7

Tree1.right.right.left.left=Tree()

Tree1.right.right.left.left.data=-5

#Tree2

Tree2 = Tree()

Tree2.data = 6

Tree2.left = Tree()

Tree2.left.data = 3

Tree2.middle = Tree()

Tree2.middle.data = 6

Tree2.right=Tree()

Tree2.right.data=5

Tree2.left.left = Tree()

Tree2.left.left.data = 5

Tree2.left.right=Tree()

Tree2.left.right.data=3

Tree2.left.left.left=Tree()

Tree2.left.left.left.data=5;

Tree2.left.left.right=Tree()

Tree2.left.left.right.data=4

Tree2.left.left.left.left=Tree()

Tree2.left.left.left.left.data=5;

Tree2.left.left.left.right=Tree()

Tree2.left.left.left.right.data=6;

Tree2.left.left.right.left=Tree()

Tree2.left.left.right.left.data=7

Tree2.left.left.right.middle=Tree()

Tree2.left.left.right.middle.data=4

Tree2.left.left.right.right=Tree()

Tree2.left.left.right.right.data=5

Tree2.left.right.right=Tree()

Tree2.left.right.right.data=3

Tree2.left.right.right.middle=Tree()

Tree2.left.right.right.middle.data=3

Tree2.middle.middle=Tree()

Tree2.middle.middle.data=6

Tree2.middle.middle.left=Tree()

Tree2.middle.middle.left.data=6

Tree2.middle.middle.left.middle=Tree()

Tree2.middle.middle.left.middle.data=6

Tree2.middle.middle.right=Tree()

Tree2.middle.middle.right.data=6

Tree2.middle.middle.right.left=Tree()

Tree2.middle.middle.right.left.data=6

Tree2.middle.middle.right.middle=Tree()

Tree2.middle.middle.right.middle.data=9

Tree2.middle.right=Tree()

Tree2.middle.right.data=7

Tree2.middle.right.right=Tree()

Tree2.middle.right.right.data=7

Tree2.middle.right.right.middle=Tree()

Tree2.middle.right.right.middle.data=7

Tree2.right.left=Tree()

Tree2.right.left.data=5

Tree2.right.left.middle=Tree()

Tree2.right.left.middle.data=5

Tree2.right.left.middle.middle=Tree()

Tree2.right.left.middle.middle.data=5

Tree2.right.middle=Tree()

Tree2.right.middle.data=8

Tree2.right.middle.middle=Tree()

Tree2.right.middle.middle.data=8

Tree2.right.middle.middle.middle=Tree()

Tree2.right.middle.middle.middle.data=9

Tree2.right.middle.middle.right=Tree()

Tree2.right.middle.middle.right.data=8

Tree2.right.middle.right=Tree()

Tree2.right.middle.right.data=6

Tree2.right.middle.right.right=Tree()

Tree2.right.middle.right.right.data=6

path=[];

start=time.clock();

print("Tree1 = ",minimax(Tree1,4,True,path));

print('Time Taken for Tree1 : ',time.clock()-start);

start=time.clock();

print("Tree2 = ",minimax(Tree2,4,True,path=[]));

print('Time Taken for Tree2: ',time.clock()-start);

main();

**OUTPUT:**





2) Implement minimax with alpha-beta pruning for both Tree 1 and Tree 2. Compare the

execution time of Question # 2 with Question # 1.

**SOLUTION CODE:**

# -\*- coding: utf-8 -\*-

"""

Created on Sat Dec 2 00:44:22 2017

@author: umerm

"""

# -\*- coding: utf-8 -\*-

"""

Created on Fri Dec 1 14:16:36 2017

@author: umerm

"""

import numpy

import cv2

from math import inf

import time

class Tree():

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

self.left = None

self.right = None

self.middle=None

self.data = None

def alphabeta(node,depth,alpha,beta,maximizing\_player,path=[]):

if(depth==0 or (node.left==None and node.middle==None and node.right==None)):

path=path+[node.data]

print("path = ",path)

return node.data;

if maximizing\_player:

value=-1\*float("inf")

for child in [node.left , node.right , node.middle]:

val=alphabeta(node,depth-1,alpha,beta,False,path)

value=max(value,val)

path=path+[value]

alpha=max(alpha,value)

if(beta<=alpha):

break

return value

else:

value=float("inf")

for child in [node.left , node.right , node.middle]:

val=alphabeta(node,depth-1,alpha,beta,True,path)

value=min(value,val)

path=path+[value]

beta=min(beta,value)

if(beta<=alpha):

break

return value

def main():

#Tree1

Tree1 = Tree()

Tree1.data = -7

Tree1.left = Tree()

Tree1.left.data = -10

Tree1.right=Tree()

Tree1.right.data=-7

Tree1.left.left = Tree()

Tree1.left.left.data = 10

Tree1.left.right=Tree()

Tree1.left.right.data=-10

Tree1.left.left.left=Tree()

Tree1.left.left.left.data=10;

Tree1.left.left.right=Tree()

Tree1.left.left.right.data=5

Tree1.left.left.left.left=Tree()

Tree1.left.left.left.left.data=10;

Tree1.left.left.left.right=Tree()

Tree1.left.left.left.right.data=float("inf");

Tree1.left.left.right.left=Tree()

Tree1.left.left.right.left.data=5

Tree1.left.right.left=Tree()

Tree1.left.right.left.data=-10

Tree1.left.right.left.left=Tree()

Tree1.left.right.left.left.data=-10

Tree1.right.left=Tree()

Tree1.right.left.data=5

Tree1.right.left.left=Tree()

Tree1.right.left.left.data=5

Tree1.right.left.right=Tree()

Tree1.right.left.right.data=-1\*float("inf")

Tree1.right.left.left.left=Tree()

Tree1.right.left.left.left.data=7

Tree1.right.left.left.right=Tree()

Tree1.right.left.left.right.data=5

Tree1.right.left.right.left=Tree()

Tree1.right.left.right.left.data=-1\*float("inf")

Tree1.right.right=Tree()

Tree1.right.right.data=-7

Tree1.right.right.left=Tree()

Tree1.right.right.left.data=-7

Tree1.right.right.left.left=Tree()

Tree1.right.right.left.left.data=-7

Tree1.right.right.left.left=Tree()

Tree1.right.right.left.left.data=-5

#Tree2

Tree2 = Tree()

Tree2.data = 6

Tree2.left = Tree()

Tree2.left.data = 3

Tree2.middle = Tree()

Tree2.middle.data = 6

Tree2.right=Tree()

Tree2.right.data=5

Tree2.left.left = Tree()

Tree2.left.left.data = 5

Tree2.left.right=Tree()

Tree2.left.right.data=3

Tree2.left.left.left=Tree()

Tree2.left.left.left.data=5;

Tree2.left.left.right=Tree()

Tree2.left.left.right.data=4

Tree2.left.left.left.left=Tree()

Tree2.left.left.left.left.data=5;

Tree2.left.left.left.right=Tree()

Tree2.left.left.left.right.data=6;

Tree2.left.left.right.left=Tree()

Tree2.left.left.right.left.data=7

Tree2.left.left.right.middle=Tree()

Tree2.left.left.right.middle.data=4

Tree2.left.left.right.right=Tree()

Tree2.left.left.right.right.data=5

Tree2.left.right.right=Tree()

Tree2.left.right.right.data=3

Tree2.left.right.right.middle=Tree()

Tree2.left.right.right.middle.data=3

Tree2.middle.middle=Tree()

Tree2.middle.middle.data=6

Tree2.middle.middle.left=Tree()

Tree2.middle.middle.left.data=6

Tree2.middle.middle.left.middle=Tree()

Tree2.middle.middle.left.middle.data=6

Tree2.middle.middle.right=Tree()

Tree2.middle.middle.right.data=6

Tree2.middle.middle.right.left=Tree()

Tree2.middle.middle.right.left.data=6

Tree2.middle.middle.right.middle=Tree()

Tree2.middle.middle.right.middle.data=9

Tree2.middle.right=Tree()

Tree2.middle.right.data=7

Tree2.middle.right.right=Tree()

Tree2.middle.right.right.data=7

Tree2.middle.right.right.middle=Tree()

Tree2.middle.right.right.middle.data=7

Tree2.right.left=Tree()

Tree2.right.left.data=5

Tree2.right.left.middle=Tree()

Tree2.right.left.middle.data=5

Tree2.right.left.middle.middle=Tree()

Tree2.right.left.middle.middle.data=5

Tree2.right.middle=Tree()

Tree2.right.middle.data=8

Tree2.right.middle.middle=Tree()

Tree2.right.middle.middle.data=8

Tree2.right.middle.middle.middle=Tree()

Tree2.right.middle.middle.middle.data=9

Tree2.right.middle.middle.right=Tree()

Tree2.right.middle.middle.right.data=8

Tree2.right.middle.right=Tree()

Tree2.right.middle.right.data=6

Tree2.right.middle.right.right=Tree()

Tree2.right.middle.right.right.data=6

start=time.clock();

print("Tree1 = ",alphabeta(Tree1,4,-1\*float("inf"),float("inf"),True));

print('Time Taken for Tree1 : ',time.clock()-start);

start=time.clock();

print("Tree2 = ",alphabeta(Tree2,4,-1\*float("inf"),float("inf"),True));

print('Time Taken for Tree2 : ',time.clock()-start);

main();

**OUTPUT:**



