Lab assignment-3 : Minimax algorithm

ASM Tareq Mahmood

20101073

CSE422, Section-04

### LAB ASSIGNMENT -3 ###

import random as rd;

class minimax:

    def \_\_init\_\_(self,s\_id):

        self.s\_id       = s\_id;

        self.hp       = [ self.s\_id[len(self.s\_id) - 1], self.s\_id[len(self.s\_id) - 2] ];

        self.start\_hp = int(self.hp[0])\*10 + int(self.hp[1]);

    def testcase(self):

        self.steps        = int(self.s\_id[0]);

        self.damage       = int(self.s\_id[2]);

        self.level        = 2\*self.steps;

        self.branchfactor = self.damage \*\* self.level;

        self.limit        = input("=>");

        self.limit        = self.limit.split(' ');

        self.maxrange, self.minrange = int(self.limit[0]),int(self.limit[1]);

        self.track        = [];

        iterate           = self.branchfactor;

        while (iterate):

            self.track.append(rd.randint(self.maxrange,self.minrange));

            iterate -= 1;

        return self.level,self.damage,self.branchfactor,self.track;

    def initial\_hp(self):

        return self.start\_hp;

def algo(loc,level,alpha,beta,imax,flags):

    if level == 0:

        flags += 1;

        return track[loc],flags;

    if not imax:

        minimum     = float('inf');

        for leaf in range(damage):

            count,flags = algo((loc\*damage)+leaf,level-1,alpha,beta,True,flags);

            minimum = min(minimum,count);

            beta    = min(beta,count);

            if alpha >= beta:

                flags -= 1;

                break;

        return minimum,flags;

    if imax:

        maximum     = -float('inf');

        for leaf in range(damage):

            count,flags = algo((loc\*damage)+leaf,level-1,alpha,beta,False,flags);

            maximum = max(maximum,count);

            alpha   = max(alpha,count);

            if alpha >= beta:

                flags -= 1;

                break;

        return maximum,flags;

if "\_\_main\_\_" == \_\_name\_\_:

    s\_id = input('=>');

    a = minimax(s\_id);

    level, damage, branchfactor, track = a.testcase();

    maxxed,flags = algo(0,level,float('-inf'),float('inf'),True,0);

    print("Depth and Branches ratio is {}:{}".format(level,damage));

    print('Terminal States (leaf node values) are ',end=" ");

    for x in track: print(x,end=" ");

    print('\nLeft life (HP) of the defender after maximum damage caused by the attacker is {}'.format(a.initial\_hp() - maxxed));

    print('After Alpha-Beta Pruning Leaf Node Comparisons {}'.format(flags));