**Veštačka inteligencija**

projekat 1 – faza 3

U ovoj fazi implementirali smo min-max algoritam sa alfa-beta odsecanjem kao i funkciju za igranje partije između računara i čoveka. Prilikom implementacije min’max funkcije korišćena je funkcija u kojoj je potrebno implementirati hieuristiku u narednoj fazi.

Funkcije koje se razlikuju u odnosu na prethodne faze su:

def minimax(stanje,dubina,moj\_potez, potez=None,alpha=(), beta=())-funkcija koja vraca tuple novog stanja(matrica) i vrednosti (koliko to stanje void igrača do pobede)

def minimax(stanje,dubina,moj\_potez, potez=None,alpha=(), beta=()):

    '''oceni- heuristika

    kraj-ispituje jek je kraj igre?

    full-puna tabla

    igraj- odigravanje potez'''

    if kraj(stanje):

        return (potez, kraj(stanje))

    if (player1):

        igrac = "o" if moj\_potez else "x"

    else:

        igrac = "x" if moj\_potez else "o"

    fja = max\_value(stanje,dubina,alpha,beta) if moj\_potez else min\_value(stanje,dubina,alpha,beta)

    lp = nova\_stanja(stanje,igrac)

    if dubina == 0:

        return (potez, proceni\_stanje(stanje))

    if lp is None or len(lp) == 0:

        return (potez, proceni\_stanje(stanje))

    return fja([minimax(x, dubina - 1, not moj\_potez, x if potez is None else potez,alfa=(x,proceni\_stanje(x)),beta=(x,proceni\_stanje(x))) for x in lp])

def nova\_stanja(stanje,igrac)-funkcija koja generiše sva moguća nova stanja na osnovu trenutnog stanja i igrača čiji je potez

def nova\_stanja(stanje,igrac

    statesOfPlayer(igrac,stanje)

    copy=np.copy(listaStates)

    listaStates.clear()

    return copy

def proceni\_stanje(stanje)-funkcija u kojoj cće se implementirati hieuristika u narednoj fazi projekta

def proceni\_stanje(stanje):

def max\_value(stanje, dubina, alpha, beta):

def max\_value(stanje, dubina, alpha, beta):   #lBodsecanje

    if dubina == 0:#jer dubnu smanjujemo svaki sledeci put

        return (stanje, proceni\_stanje(stanje))#vracamo tuple sa stanjem i njegovom vrednosti

    else:

        for s in nova\_stanja(stanje):#odredjujemo sva nova stanja i prolazimo kroz njih

            alpha = max(alpha, min\_value(s, dubina - 1, alpha, beta), key=lambda x: x[1])

            #odredjujemo alfa, kao maksimum izmedju alfa i minimalne vrednosti, vracamo kljuc cija je vrednost 1 jer vrsimo proveru po proceni ne po stanju

            if alpha[1] >= beta[1]:

                return beta

    return alpha

def min\_value(stanje, dubina, alpha, beta):

def min\_value(stanje, dubina, alpha, beta):

    if dubina == 0:

        return (stanje, proceni\_stanje(stanje))   #lBodsecanje

    else:

        for s in nova\_stanja(stanje):

            beta = min(beta, max\_value(s, dubina - 1, alpha, beta),key=lambda x: x[1])

            if beta[1] <= alpha[1]:

                return alpha

    return beta

def kraj(stanje)-funkcija koja proverava da li se u datom stanju igra završila, funkcija vraća 0 ukoliko jeste a vrednost heuristike ukoliko nije

def kraj(stanje):

    ppx1=[]

    ppx2=[]

    ppo1=[]

    ppo2=[]

    for i in range(n-1):

        for j in range(m-1):

            if( stanje[i][j]=="px1"):

                ppx1=[i,j]

            if( stanje[i][j]=="px2"):

                ppx2=[i,j]

            if( stanje[i][j]=="po1"):

                ppo1=[i,j]

            if( stanje[i][j]=="po2"):

                ppo2=[i,j]

    px1i=[x1[0],x1[1]]

    px2i=[x2[0],x2[1]]

    po1i=[o1[0],o1[1]]

    po2i=[o2[0],o2[1]]

    if (ppx1==po1i or ppx1==po2i or ppx2==po1i or ppx2==po2i ):

        return 0

    if (ppo1==px1i or ppo1==px2i or ppo2==px1i or ppo2==px2i ):

        return 0

    return proceni\_stanje(stanje)

def igraj()-funkcija koja realizuje igru izmedju računara i čoveka

def igraj():

    global tabla

    global x1

    global x2

    global o1

    global o2

    global pozicije

    global n

    global m

    global xZidovi

    global oZidovi

    global moved

    inputT()

    tabla=Tabla(n,m,x1,x2,o1,o2)

    while (not endGame()):

        if(player1):

            #while(kraj(tabla)==0):

            rez=minimax(tabla,3,"x",(tabla, 0),(tabla, 617))

            naj=rez[0]

            tabla=naj

            update()

            if cijiPotez=="o":

                cijiPotez="x"

            elif  cijiPotez=="x":

                cijiPotez="o"

            while(not moved):

                move()

            moved=False

            update()

            zid()

            update()

        else:

            while(not moved):

                move()

                moved=False

                update()

                zid()

                update()

            rez=minimax(tabla,3,"x",(tabla, 0),(tabla, 617))

            naj=rez[0]

            tabla=naj

            update()

            if cijiPotez=="o":

                cijiPotez="x"

            elif  cijiPotez=="x":

                cijiPotez="o"