Domaći zadatak iz inteligentnih sistema

Igra backgammon realizovana uz pomoc expectimax algoritma

1. Kratak prikaz expectimax algoritma:

pseudo kod algoritma:

expectimax(board,igrac,kocka1,kocka2,maxDubina,dubina):

ako je board terminalan ili ako je dubina=maxDubina

return Heuristicku vrednost boarda

izracunaj state //0,4,8,12,16 ..=max 2,6,10,14=min neparni=chance

ako je state= chance

za sve vrednosti k1 i k2 expectimax(board,igrac,k1,k2,maxDubina,dubina+1)

nadji srednju vrednost

ako je state=min

za sve moguce boardove koji mogu da nastanu iz boarda uz kocku1 i kocku2 expectimax(...)

nadji minimalnu vrednost

ako je state=max

za sve moguce boardove koji mogu da nastanu iz boarda uz kocku1 i kocku2 expectimax(...)

nadji maximalnu vrednost, ako je dubina 0 nadji i board u koji treba preci

end

//Board je objekat koji predstavlja stanje table na kojoj igramo

2. Opis klasa i resenja

Klasa Igra ima 2 igraca i tablu na kojoj se odigravaju potezi. Igrac je abstraknta klasa koja sadrzi funkciju Roll i OdigrajPotez koje implementiraju izvedene klase Covek i Racunar. Board predstavlja tablu kao niz

int-ova sa dodatnim poljima barWhite, barBlack, goalWhite i goalBlack. Crni tokeni su predstavljeni u nizu kao negativne vrednosti a beli kao pozitivne vrednosti. Klasa Tabla je jframe koji u iscrtava stanje na board-u. Klasa Menu je JFrame koji je zaduzen za inicijalizovanje igre. Igra se odvija u funkciji Game() u klasi Igra koja naizmenicno poziva igracima funkcije roll i odigrajPotez dok se igra ne zavrsi.

public class Igra {

public Igra(Igrac white, Igrac black, Board board,int IgraSeDo)

//konstruktor dodaje igrace, board i vrednost do koje se igra

public Igrac FirstRoll();

// poziva kod oba igraca firstRoll i vraca pobednika, taj pobednik igra prvi

public Igrac Game();

// naizmenicno poziva igrace da odigraju poteze dok neko ne pobedi. Kao povratnu vrednost vraca //pobednika

public Igrac Partija();

//game se poziva dok neki od igraca ne skupi dovoljno bodova za pobedu

}

public class Covek extends Igrac {

public Covek(String boja,Board board);

Board Odigraj(int kocka1, int kocka2);

// omogucavamo coveku da odigra potez na tabli, vraca novonastali board

int Roll(); // vrati random vrednost, simulira bacanje kockice

int FirstRoll() ;

}

public class Racunar extends Igrac {

public Racunar(String boja,Board board,int Tezina);

//int tezina predstavlja dubinu pretraga u expectimax algoritmu

Board Odigraj(int kocka1, int kocka2);

// omogucavamo racunaru da odigra potez na tabli, vraca novonastali board

int Roll(); // vrati random vrednost, simulira bacanje kockice

int FirstRoll() ;

double expectimax(Board board, Igrac igrac, int dice1, int dice2, int maxDubina, int dubina);

//ovu funkciju pozivamo u Odigraj i ova funkcija promeni staticki mainTabla sto predstavlja odigravanje //poteza

}

public class Board {

static Board mainTable; //u ovoj promenjivoj se nalazi stanje table

public Board(Tabla t)

// konstruktor dobijamo objekat table na koji treba da iscrtavamo board

public Board(Board board)

// pravimo kopiju objekta ali bez Tabla, ovi objekti nam sluze za kreiranje novih board-ova koji nece biti //mainTable, nego ce se koristi u expectimax algoritmu za trazenje novog stanja

public void ResetBoard()

// postavljamo board u pocetnu konfiguraciju

public boolean CheckWin(Igrac i)

// proveravamo da li je prosledjeni igrac pobednik

public int Heuristic(Igrac igrac)

// vracamo staticku procenu stanja tabla za igraca prosledjenog kao parametar

public boolean isTerminal(Igrac igrac)

// da li je board terminalan

public int terminalValue(Igrac igrac)

// vracamo vrednost terminalnog Board-a

public List<Board> sviPotezi(int d1, int d2, Igrac igrac)

// generisemo listu svih mogucih poteza za igraca igrac za kockice d1 i d2

private void odigrajPrviKocku(int d1, int d2, List<Board> lista, Igrac igrac)

// pomocna funkcija koju pozivamo u metodi SviPotezi, generise sve poteze sa jednom kockickom

private List<Board> odigrajDruguKocku(int d1, int d2, List<Board> lista, Igrac igrac)

// pomocna funkcija koju pozivamo u metodi SviPotezi, generise sve poteze kada odigramo drugu //kockicu

private boolean ReadyToTakeOff(Igrac ig)

//Da li je igrac ig ostvario uslov za sklanjanje svojih tokena van table

}