Izveštaj četvrta faze projecta iz Veštačke Inteligencija

Tim: **Bots of Musk**

Projekat: **Realizacija igre Byte**

|  |  |
| --- | --- |
| **Članovi tima:** | |
| Kristijan Ilić | 16103 |
| Miloš Stoiljković | 16382 |
| Jovan Krstić | 16151 |

U četvrtoj fazi izvrsili smo implementaciju funkcije heuristike sa pratećim pomoćnim funkcijama.

U trecoj fazi izvršena je implementacija Min-Max algoritma sa alfa-beta odsecanjima.Radi implementacije ovog algoritma dodate su funkcije: alphaBeta, minPlayer, maxPlayer, whileMin, whileMax, heuristic.U daljem tekstu može se pronaći detaljniji opis funkcija.

U drugoj fazi izvresna je ažuriranje i uklanjanje ne optimalnih funkcija.Dodate su nove funkcionalnosti sto se moze videti u spisku funkcija gde su i objašnjene njihove funkcionalnosti.

**Struktura igre**

Za realizaciju ove igre prvo je bilo potrebno kreirati strukturu koja će predstavljati realno stanje table. Mi smo se odlučili da to bude graf čiji čvorovi predstavljaju polja od značaja u igri(crna polja).Čvorovi grafa sadrye sledece informacije: Indeks polja oznaka gde se polje nalazi na table, Listu pločica koje se nalaze na tom polju (stek) ukoliko nema plocica na tom polju ova lista sadrži samo E sto oznacava przano polje. Takođe cvor sadrži i listu sa indeksima polja koji su mu susedi sto olaksava dalje rad u generisanju poteza i validaciji.

**Funkcije**

Za rad sa strukturom i obezbeđivanje potrebnih funkcionalnosti kreirane su sldeće funkcije:

1. **makeCvor (n m empty)**

Ova funkcija kreira osnovni cvor grafa na osnovu prosledjenih kordinata i setuje inicijalnu plocicu na tom polju ukoliko je ima.Takođe pozivom funkcije **umetni** u cvoru se kreira i lista suseda.

1. **umetni(n m)**

Ova funkcija kreira listu susedstva za zadato polje na table.Lista se sastoji od četiri elementa gde svaki element predstavlja suseda od značaja potencijalno polje važno za potez.Ukoliko je neki element u list 0 to označava da sused nije prisutan.

1. **makeGraph(m n)**

Funkcija makeGraph na osnov zadatih dimenzija table kreira strukturu grafa pozivajući funkciju makeCvor. Ovom funkcijom se kreira osnovni graf polazno stanje na table.

1. **playMove( from to n graph)**

Funkcija koja je zadužena za odigravanje poteza. Ona poziva funkciju playAndRemoveMove i prosleđuje joj svoje ulazne parametre i pribavljeni potez.Pribavljanje poteza se realizuje funkcijom takeMove.

1. **playAndRemoveMove( from to n graph move)**

Ova funkcija vraca graf koji je kreiran na osnovu zadatog poteza i grafa koji ptedstavlja trenutno stanje na tabli.Novo nastali graf predstavlja novo nastalu situaciju na table.

1. **grabN(li n)**

Funkcija koja odseca i vraća prvih n elemenata iz liste li.Ovo je pomocna funkcija za odigravanje poteza.

1. **removeN(li n)**

Funkcija vraca odsečenu listu pocevši od n-tog elementa do kraja liste li. Ovo je pomocna funkcija za odigravanje poteza.

1. **takeMove(from n graph)**

Funkcija iz grafa pronalazi polje sa kojeg se igra i uzima sa visine n stek koji vraća. Ovo je pomocna funkcija za odigravanje poteza.

1. **grabMove(cvor n)**

Funkcija iz zadatog cvora vraca prvih n

1. **insertMove(cvor move)**

Funkcija za zadati cvor (polje na tabli) nadovezuje stek sa prosleđenim stekom (potezom) na taj nacin kreira novo stanje na table. Ovo je pomocna funkcija za odigravanje poteza.

1. **dots(n)**

Ova funkcija kreira listu od n – karaktera.Ova funkcija je pomocna funkcija za prikaz table.

1. **castBytes(li)**

Ova funkcija od liste plocica organizuje u adekvatnu listu za prikaz sa dodatim karakterima – tako da polja mogu isto izgledati nezavisno od velicine steak na polju. Ova funkcija je pomocna funkcija za prikaz table.

1. **getThree(k li)**

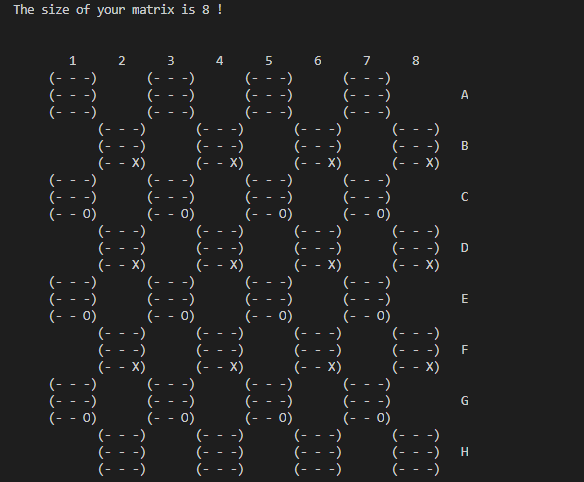
Ova funkcija organizuje listu u tri liste od po tri elementa radi lakseg prikaza.

1. **writeFirstLineOfNumbers(n)**

Ova funkcija ispisueje prvu liniju tabele sa brojevima kolona u odnosu na zadatu dimenziju table.

1. **drawTable(i j pom graph)**

Predstavlja galvnu funkciju za iscrtavanje table na osnovu prosledjenog grafa.Ona kombinacijom gore navedenih pomocnih funkcija organizuje prikaz grafa kao na sledecem primeru.



1. **distProp(obradjeni cvor n graphGlobal)**

Ova funkcija odreduje distance između zadatog čvora i suseda koji nisu obrađeni u okviru zadatog grafa (stanju). Ona predstavlja pomoćnu funkciju za generisanje poteza.

1. **propagate(obradjeni cvor n graphGlobal)**

Ova funkcija vraća listu dužina puteva za zadati čvor kako bi se odredio najkraći put u yadatom grafu (stanju). Ona predstavlja pomocnu funkciju za generisanje poteza.

1. **height(li)**

Funkcija odreduje visinu steak i predstavlja pomoćnu funkciju za generisanje svih mogucih poteza.

1. **filterPaths(paths distance ret)**

Ova funkcija na osnovu prosleđenih puteva i zadate distance vrši filtriranje puteva tako da ostanu samo oni koji su jednaki ili manji od zadate distance.

1. **distOneCheck(li)**

Ova funkcija za zadati čvor u garfu proverava njegove prve susede i vraća sve one koje pronađe.Kositi se kod funkcije generisanja svih mogućih poteza.

1. **closestNodes(li)**

Ova funkcija pronalazi duzinu puta do najbližeg steka i vraca je kako bih mogli izvrsiti filtriranje puteva prema duzini.

1. **allPossiblePlays(start paths heights graphGlobal)**

Ova funkcija proverava sve poteze koje je moguce odigrati iz datog polaznog cvora (pola na table) koji ye nalazi u zadatom grafu sa zadatim visinama na kojima je moguće sečenje steka i putevima.Ovo je takođe pomoćna funkcija za generisanje poteza.

1. **possiblePlay(start path heights graphGlobal)**

Funkcija proverava za date visine i put koje potezi predstavljaju legalne poteze u yadatom grafu(stanju) da ne bih doslo do prekoračenja steka. Ona predstavlja pomocnu funkciju za generisanje poteza.

1. **possibleHeights(stack n)**

Funcija na snovu zadatog steka i igraca koji igra daje sve moguce visine na kojima je za tog igraca moguće preseći stek. Ona predstavlja pomocnu funkciju za generisanje poteza.

1. **checkMove(from to n graphGlobal)**

Ova funcija vrsi proveru da li za zadati potez potoje ogranicenja u vidu prekoracenja steka za odigravanje poteza na zadatom grafu.Ona predstavlja pomocnu funkciju za generisanje poteza.

1. **moveGen(graph graphGlobal)**

Funkcija koja na osnovu prosleđenog grafa (trenutnog stanja na tabli) zaključuje koje je sve poteze moguće odigrati u datoj poziciji. Ona vraća listu svih mogućih validnih poteza.

1. **weArePlaying (graphGlobal)**

Ova funkcija daje informaciju igracu ko je trnutno na potezu i zahteva unos poteza. Nakon unosa poteza funkcija proverava validnost poteza.Ukoliko je potez validan odigraće se i prikazeće se novo stanje table, a ukoliko nije validan zahtevaće se ponovni unos poteza.

1. **validateMove (from to Depth)**

Ova funkcija za zadati potez vrsi proveru da li je validan. Ukoliko jeste funkcija ce vratiti sam potez, a ukoliko nije vraća nil.

1. **validateAndPlayMove(from to Depth)**

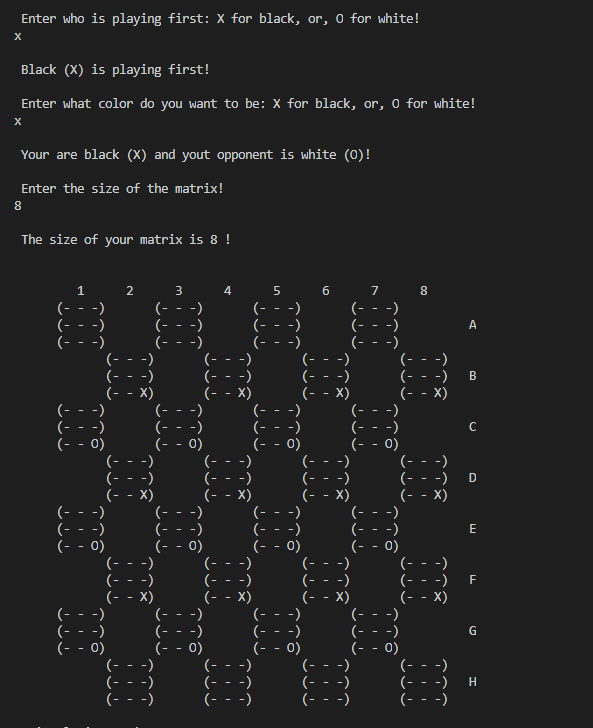
Ova funkcija na osnovu zadatog poteza vrsi proveru validnosti poyivom funkcije validateMove. Kada ustanovi da je potez validan odigraće ga u suprotnom vrati će nil i u terminal objaviti upozorenje.

1. **genStates(allMoves graphGlobal)**

Ova funkcija na osnovu svih mogucih poteza i trenutnog stanja generiše nova stanja tj. sva stanja u koja je moguće preći iz trenutnog stanja.

1. **gameSetup()**

Ovom funkcijom se konfiguriše igra na početku.Zahteva unos parametara igre kao što su ko igra prvi i velicina table.Na osnovu unetih parametara generise graf koji predstavlja tablu i prikazuje sam graf u vidu table na terminal. Primer konfiguracije:



1. **checkFinalStack()**

Funkcija koja proverava da li je prisutan zavrsni stek.Zavrsni stek je stek koji odlucuje pobednika igre.

1. **convert-me (lista)**

Ova funkcija vrsi konverziju ulaznog potza (poteza ucitanog sa terminala) u potez poznat sistemu.Pomocna je funkcija funkciji validacije poteza.

1. **checkFinalStack()**

Funkcija proverav da li je doslo do kraja igre tj. da li ima pobednika. Ako ima završava igru i proglašava pobednika.

1. **removeNils(li)**

Funkcija uređuje listu tako što iz zadate liste uklanja nil elemente. Predstavlja pomoćnu funkciju za generator poteza.

1. **alphaBeta(state move depth alpha beta maxPlayerr)**

Funkcija koja implementira min-max algoritam sa alfa-beta odsecanjima na osnovu zadate dubine i pocetnog stanjanja. Ova funkcija kao rezultat vraća potez koji je najpogodnije odigrati i huristicku vrednost do koje void konktretni potez. Argumenti alfa i beta se inicijalno postavljaju na ±∞ , a argument maxPlayerr na t ili nil u zavisnosti od igraca na potezu.

1. **minPlayer(states moves depth alpha beta)**

Ova funkcija predstavlja pomoćnu funkciju funkciji alphaBeta funkciji ona se izvršava za min igraca. Ona setuje inicijalnu evaluacionu vrednost i prosleduje sva stanja dobijena od generator stanja na osnovu prethodnog stanja.Evaluaciona vrednost se za min igraca setuje na 100000 sto se tretira koa +∞.

1. **minPlayer(states moves depth alpha beta)**

Ova funkcija predstavlja pomoćnu funkciju funkciji alphaBeta funkciji ona se izvršava za max igraca. Ona setuje inicijalnu evaluacionu vrednost i prosleduje sva stanja dobijena od generator stanja na osnovu prethodnog stanja. Evaluaciona vrednost se za max igraca setuje na 100000 sto se tretira koa +∞.

1. **whileMin(evalMin states moves depth alpha beta doWhile)**

Funkcija vrsi pretragu stable u potrazi za minimalnom evaluacionom vrednoscu u jednoj dubini kada je min plejer na potezu i vrsi alfa beta odsecanja.

1. **whileMin(evalMin states moves depth alpha beta doWhile)**

Funkcija vrsi pretragu stable u potrazi za maximalnom evaluacionom vrednoscu u jednoj dubini kada je max plejer na potezu i vrsi alfa beta odsecanja.

1. **heuristika(state player)**

Funkcija izracunava heuristicku vrednost na osnovu prosleđenog stanja I igraca koji je napotezu.Osnovna ideja heuristika je da je mnogo bole stanja gde igrac za koga se izračunava heuristika ima vise plocica na vrhu stekova vrednost je proporcijalna broju plocica na vrhu, a u slucaju stanja da postoji stek od 8 sa njegovom plocicom na vrhu ima najvecu heuristicku vrednost. Proverava se i da li na susednim pozicijama ima stekova koji bi eventualnim spajanjem u nekom trenutku doveli do stvaranja steka 8 sa pločisom na vrhu zadatog igrača.

1. **countTops(state count)**

Funkcija proverava za zadato stanej koliko ima plocica na vrhu igrača koji predstavlja računar. Za svaku nadjenu pločicu na vrhu uyima visinu steka i sabira.Ova vunkcija predstavlja pomocnu funkciju funkciji heuristike.

1. **checkEightStack(state)**

Funkcija vraća vrednost 20 ukoliko postoji stek u yadatom stanju sa visinom 8 za igraca koji predstavlja racunar.Ova funkcija predstavlja pomocnu funkciju funkciji heuristike.

1. **checkAdjisentStacks(state count graphGlobal)**

Funkcija proverava da li postoje susedstva u yadatom stanju i ukoliko postoje takva susedstva da li je na vrhu nekog od njih pločica igraca koji predstavlja računar. Za svako takvo susedstvo dodaje 2 promenjivoj count.Ova funkcija predstavlja pomoćnu funkciju funkciji heuristike.