Veštačka inteligencija - izveštaj III faza

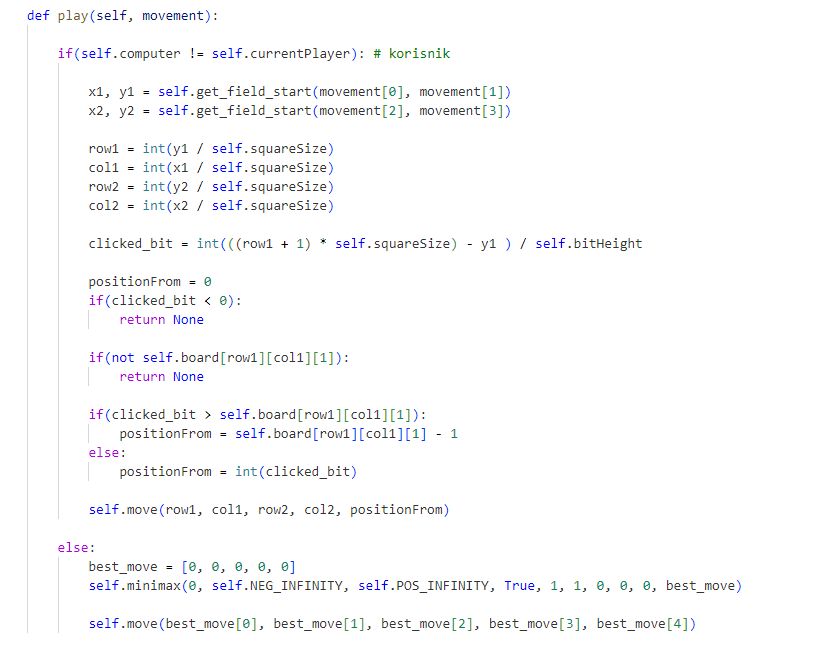
**PinkTeam**

Mijajlović Anđelija 18247

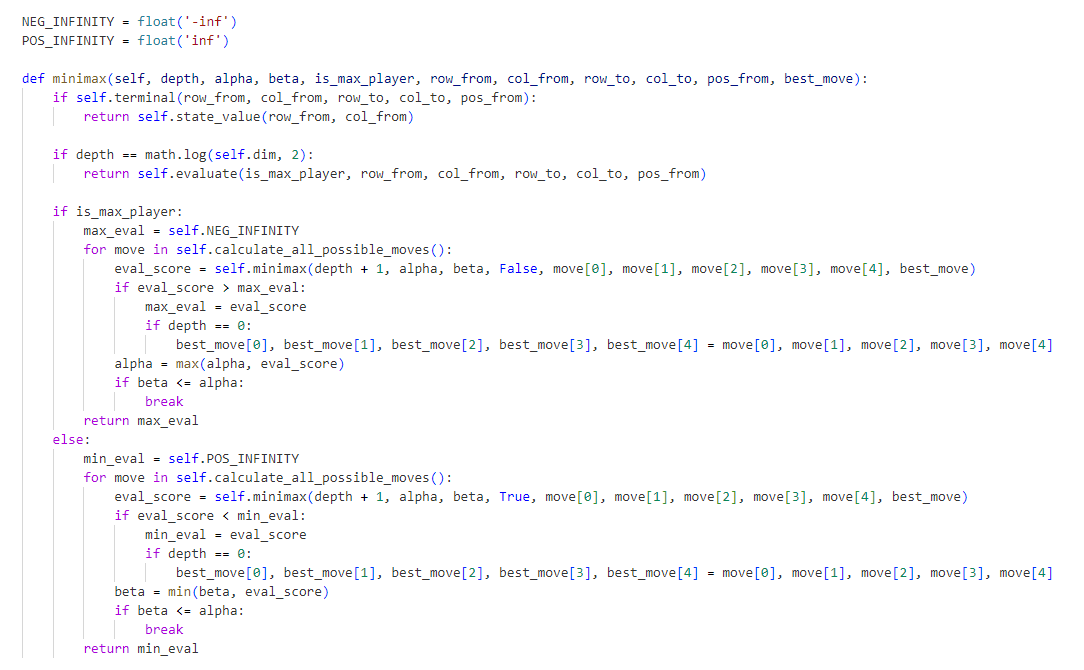
Joksimović Kristina 18203

**1. Odigravanje igre računar protiv korisnika**

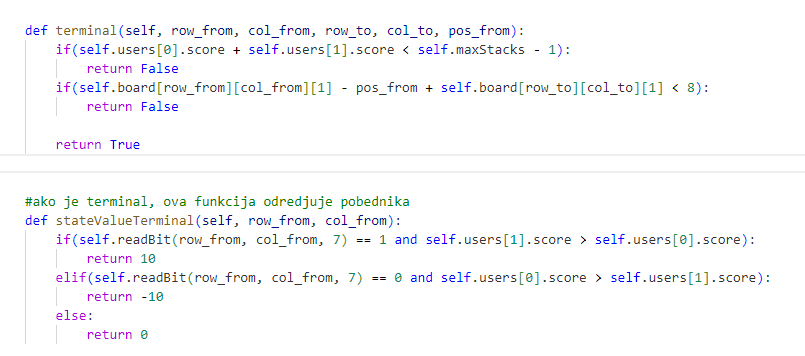
Na osnovu početnog izbora korisnika, računar igra suprotnu boju. Poziva se funkcija play, koja ukoliko je korisnik u pitanju izračunava po koordinatama odigrani potez i odigrava ga pomoću funkcija iz prethodnih faza. Ukoliko je računar na potezu, poziva se **minimax** funkcija koja vraća potez, a on se zatim odigrava.



**2. Minimax algoritam**

Na slici je prikazano kako je implementiran minimax algoritam sa alfa- beta odsecanjem i ograničenjem dubine na logaritam dimenzije table. 

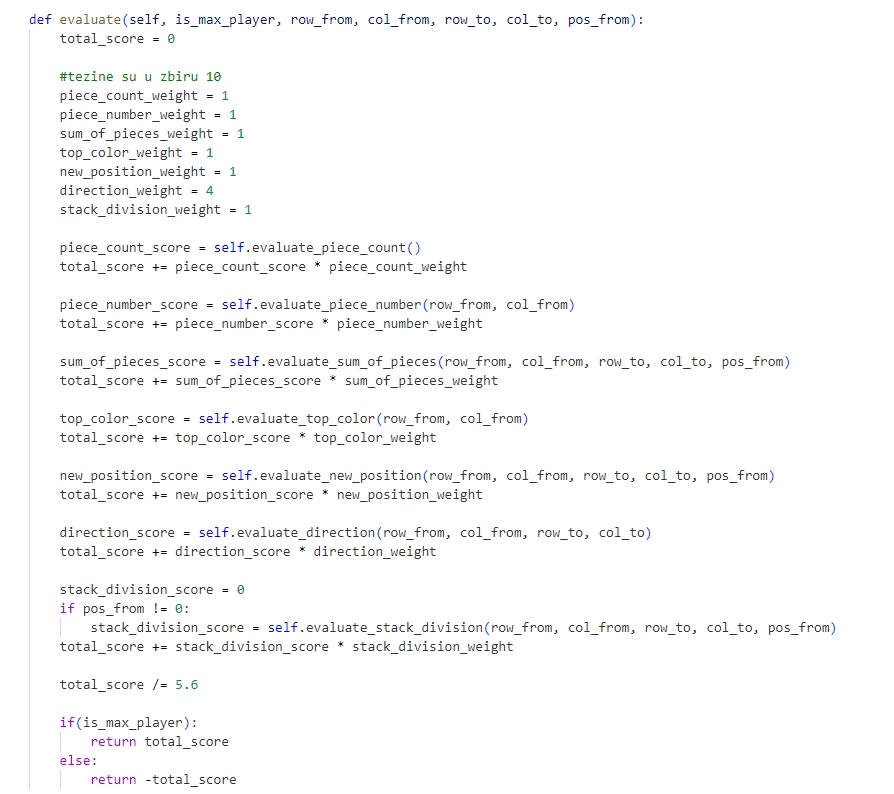
U pomoćne funkcije spada provera da li je stanje terminalno, koja određuje da li neki od igrača ima broj stekova potrebnih za pobedu. Nakon toga, određuje se vrednost terminalnog stanja:

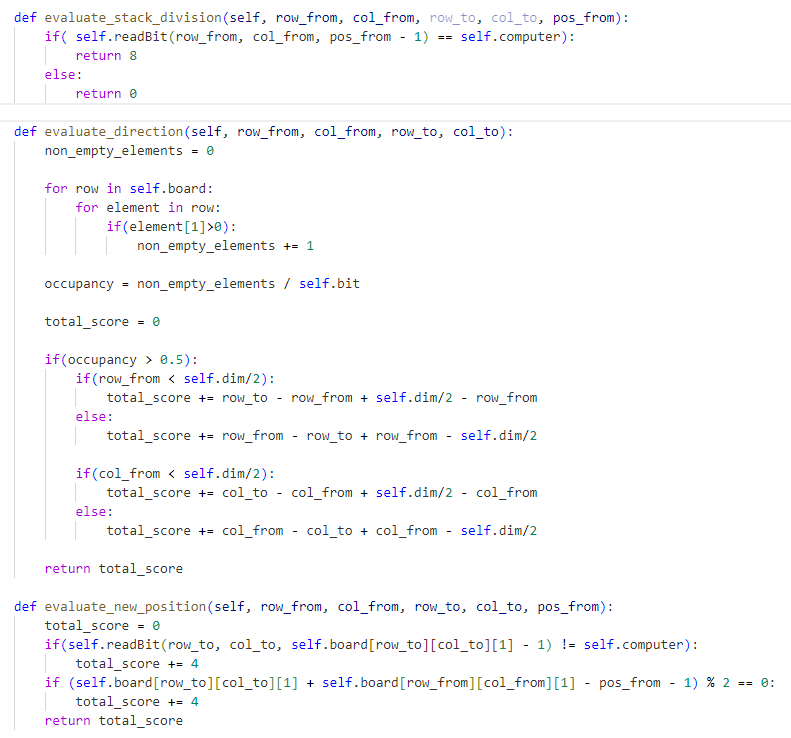


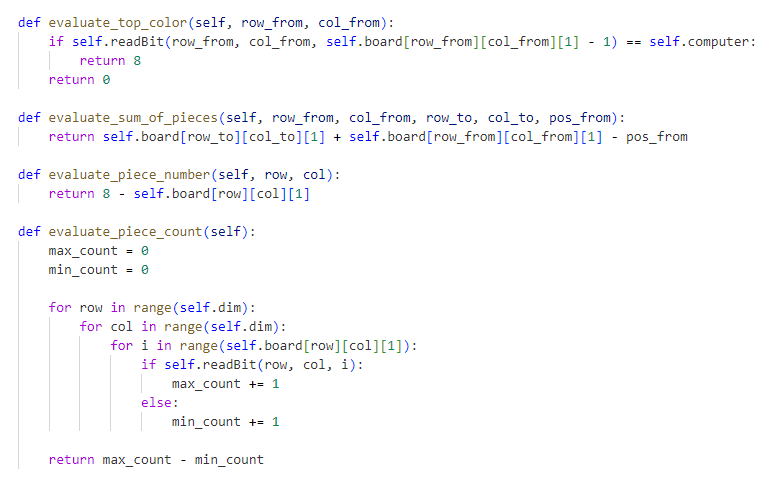
**3. Evaluacija stanja**

Na datoj ograničenoj dubini određuje se vrednost stanja koje nije terminalno pomoću funkcije evaluate. Na rezultat evaluacije utiču razni faktori. Uzeto je u obzir sledeće:

1. Stanje je bolje ukoliko ima više figurica tekućeg igrača
2. Najpre se pomeraju figurice koje su same na polju
3. Prednost se daje ukoliko je na vrhu tekućeg steka boja tekućeg igrača i ukoliko je broj figura veći u rezultujućem steku
4. Prednost se daje ako je rezultujuća pozicija iznad figurice protivnika
5. Prednost se daje ako je rezultujuća pozicija u okviru steka parna
6. Kretanje (od krajeva) ka centru, u zavisnosti od popunjenosti table
7. Ukoliko se pomeraju figurice iz steka i postoji ta mogućnost, pomeraju se tako da na vrhu početnog steka ostane figura tekućeg igrača





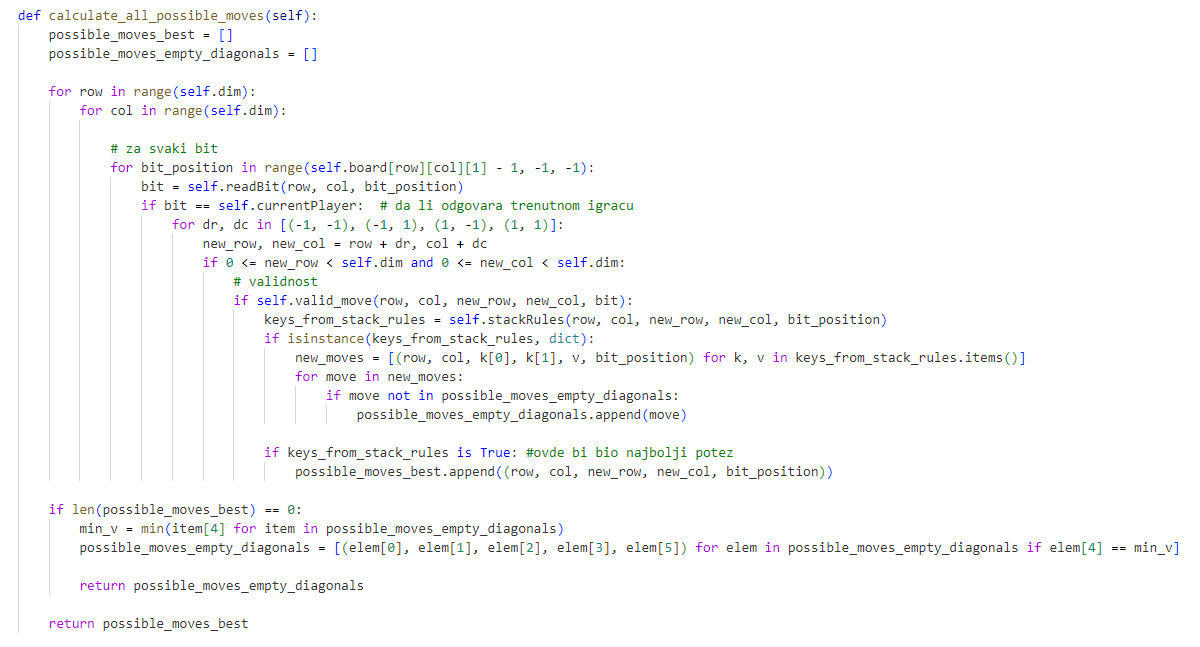


U okviru play funkcije, nakon minmax algoritma,, koričćene su funkcije iz prethodne faze projekta, koje svakako kontrolisu igru sve vreme, proveravaju moguće i validne poteze.

**4. Izračunavanje dozvoljenih poteza za sve figure konkretnog igrača na potezu.**

Funkcija prolazi kroz sve bitove svakog bolja, i trazi svog konkretnog igrača koji je na potezu i za njega, proverava njegovu validnost a nakon toga i valjanost.

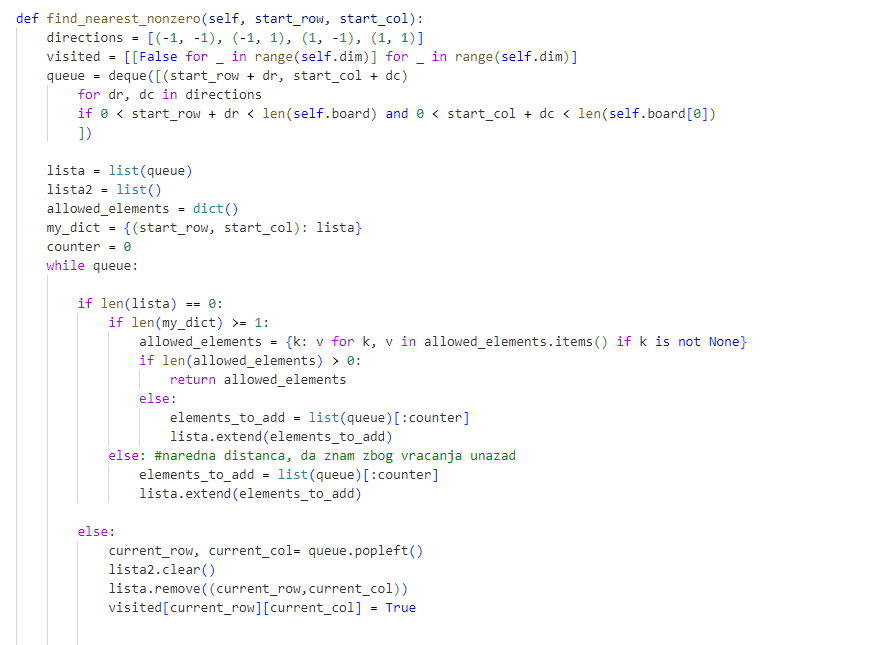
Nakon provere valjanosti, obzirom da su potezi preko dijagonala i oni dijagonalni odvojeni, postoje provere koje će vratiti tačno dozvoljene poteze, odnosno, ukoliko postoji polje koje nema prazne dijagonale oko sebe, imaće prednost nad svim ostalim potezima.

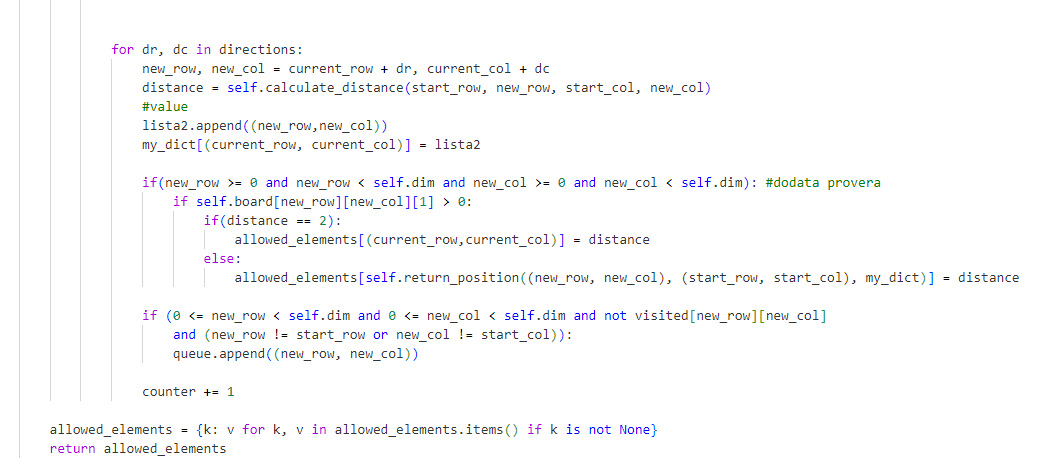
****

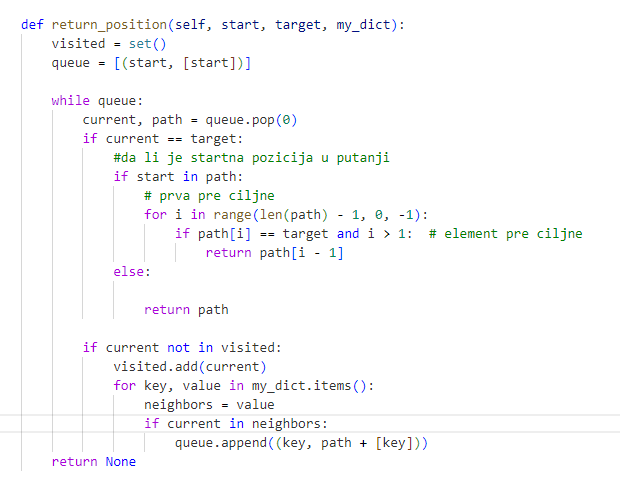
**5. Vraćanje svih dozvoljenih poteza sa datog polja.**

Pomoćna funkcija funkciji stackRules koja ce za svako obrađeno polje tražiti najbolji sledeći skok.

Funkcija predstavlja modifikovani BFS algoritam, kreće se od traženog polja i obilaziće u širinu sve dok ne naiđe na polje koje sadrži neki stek. Tada je kao pomoćna, iskorišćena funkcija return\_position koja koristi ideju algoritma traženja A\* i vraća putanju nazad do skoka koji će zapravo predstavljati najbolji izbor za to polje.

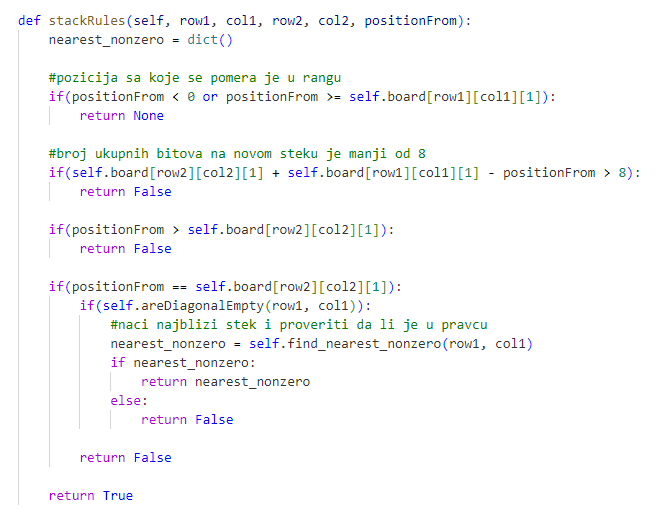
****

****

****

**6. Provera valjanosti poteza**

Funkcija stackRules će nakon što se utvrdi validnost poteza, utvrđivati valjanost isti i ukoliko su osnovni uslovi (pomeranje je u rangu, ukupan proj u steku je 8, pozicija sa koje se pomeramo je manja od one na koju skačemo) ispunjeni, poziva se find\_nearest\_nonzero.  
Ukoliko nisu prazne dijagonale, funkcija će vratiti true/false vrednosti koje će u funkciji calculate\_all\_possible\_moves biti pomoć da vrate samo te pozicije kao moguće opcije.

****