Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Seminarski rad

**ŠAHOVSKA IGRA**

Izradili:

Dario Mihelčić Mentor:

Damian Svirac Doc. dr. sc. Mirko Köhler

Osijek, 2019.

**SADRŽAJ**

1. **Uvod**...........................................................................1
2. **Šahovska igra**...........................................................2
   1. **ChessBoard**
   2. **ChessRules**
   3. **ChessMain**
3. **Šahovski računalni program**...................................3
   1. **SillyBot**
   2. **EatingBot**
   3. **MinMaxBot**

**3.3.1. Minimax algoritam**

**3.3.2. Alpha–beta obrezivanje**

1. **Zaključak**...................................................................4
2. **Literatura**

**1. Uvod**

Cilj ovog seminara bio je izraditi model igre šaha, u kojoj su definirana sva pravila igre, funkcionalnosti figura te tekstualna reprezentacija igre i napisati šahovski računalni program koji može samostalno igrati šah. Program je napisan u programskom jeziku *Python*. Program igre šaha je napisan tako da mogu igrati dva igrača, igrač protiv računala, ili dva računala međusobno jedno protiv drugog.

**2. Šahovska igra**

Program je napisan prema objektno orijentiranim načelima i sastoji se od 4 klase, a to su: **ChessBoard, ChessRules, ChessMain i ChessAI.**

Prvi problem koji je odrađen u seminaru je kreiranje digitalne reprezentacije igre šaha, koja se nalazi u klasi ChessBoard. Nakon kreiranja digitalnog oblika ploče, figura i njihovih pokreta potrebno je definirati pravila prema kojima se igra šah, što je definirano u klasi ChessRules. To uključuje način na koji se određene figure smiju kretati, određivanja stanja kao što su šah, šah-mat, pat. Osim osnovnih pravila, bilo je potrebno implementirati i posebna pravila kao što su rokada, *en passant* i promocija pijuna. U klasi ChessMain nalaze se funkcije za pokretanje i igranje šaha koji se pokreće u command promptu. Moguće je odabrati opcije za tipove igrača: igrač protiv igrača, igrač protiv programa, program protiv programa. Šahovski računalni program (chess engine) definiran je u ChessAI klasi. Napravljena su 3 programa od kojih MinMax najbolje igra šah.

U nastavku seminara prikazan je dio koda korišten za program šahovske igre.

**2.1. ChessBoard**

Klasa ChessBoard sadrži funkcije za stvaranje šahovske ploče i funkcije vezane uz šahovsku ploču, funkciju za pomicanje figura.

A screenshot of a cell phone

Description automatically generated

Slika 1 Funkcija koja kreira početnu ploču

Šahovska ploča je prikazana kao kvadratna matrica od 8 redova I 8 stupaca. Matrica sadrži string vrijednosti, svaka vrijednost unutar matrice zastupa šahovsku figuru ili prazno polje. Figure se sastoje od slova za boju (‘b’ za crne figure, ‘w’ za bijele figure) i slova za tip figure (‘P’-pijun, ‘R’-top, ‘N’- konj, ‘B’- lovac, ‘Q’-kraljica, ‘K’- kralj). Prazno polje je prikazanom znakom ‘e’.

Matrica šahovskog polja sačuvana je u varijabli ‘state’. Prije obavljenog poteza prošlo stanje ploče se sačuva u varijablu ‘last\_state’. Klasa ChessBoard još sadrži varijable za provjeru rokade koje uključuju stanja kralja i topova na ploči.

A screenshot of a cell phone

Description automatically generated

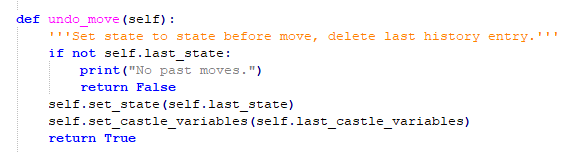
Slika 2 Funkcija koja resetira ploču na početno stanje

A screenshot of a social media post

Description automatically generated

Slika 3 Funkcija koja pomiče figure

Funkcija 'move\_piece' sadrži logiku za pomicanje figure na šahovskoj ploči. Funkcija prima jedan argument, 'move', koji se sastoji od 4 int vrijednosti, indeksi ploče. Sastoji se od, redom, broj reda i broj stupca početne pozicije, broj reda i broj stupca odredišta poteza. Indeksi se koriste za obavljanje poteza zamjenom varijabli matrice stanja. Unutar funkcije se također provjerava mogući posebni potezi: en passant, promocija pijuna i rokada.

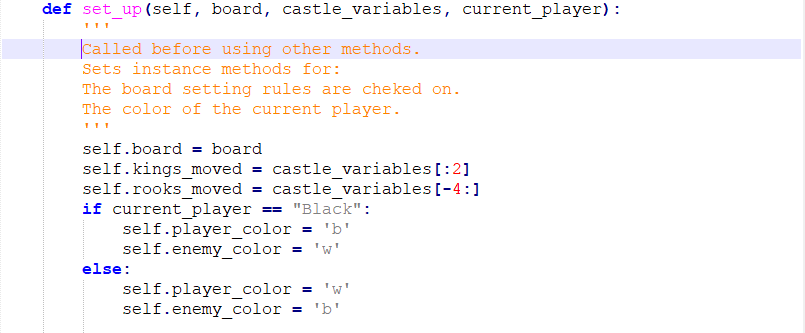


Slika 4 Funkcija koja vraća potez

**2.1. ChessRules**

Klasa ChessRules sadrži sva pravila za igranje šaha: pravila kretnje figura, provjera specijalnih stanja (šah, šah-mat, pat). Klasa sadrži i funkciju za generiranje svih mogućih poteza za određenu boju koja je potrebna kod programiranja šahovskog računalnog programa.

Prije nego što se koriste funkcije iz ChessRules klase potrebno je pozvati funkciju ‘set\_up’ koja prima i postavlja 3 argumenta: stanje ploče, varijable za rokadu i trenutnog igrača (crni ili bijeli).



Slika 5 Funkcija za postavljanje osnovnih parametara

A screenshot of a social media post

Description automatically generated

Slika 6 Funkcija koja dohvaća listu mogućih poteza

A screenshot of a cell phone

Description automatically generated

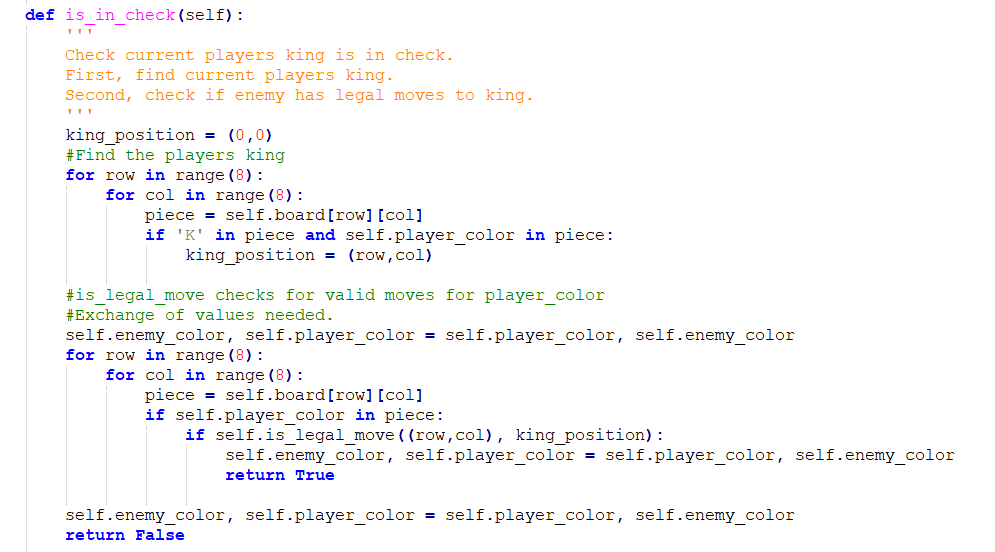
Slika 7 Funkcija koja dohvaća listu mogućih poteza za određenu figuru

A screenshot of a social media post

Description automatically generated

Slika 8 Funkcija koja provjerava je li uneseni potez ispravan

Najvažnija funkcija klase ChessRules je ‘is\_legal\_move’. U ovoj funkciji se nalaze sva pravila za pomicanje figura. Funkcija prima dva argumenta: početnu poziciju i odredište. Ime figure početne pozicije se učitava u varijablu ‘from\_piece’. Zatim se provjerava koji tip figure se nalazi na početnom položaju tako da se usporedi ime figure s mogućim tipovima figura. Funkcija može vratiti boolean vrijednosti, True ili False.



Slika 9 Funkcija koja provjerava je li šah

A screenshot of a cell phone

Description automatically generated

Slika 10 Funkcija koja provjerava je li šah mat

**2.3. ChessMain**

Klasa Chess main sadrži funkcije koje se koriste za pokretanje igre u konzoli. Kako bi se pokrenula igra potrebno je pokrenuti ChessMain.py datoteku u konzoli.

A screenshot of a cell phone

Description automatically generated

Slika 11 Funkcija koja crta ploču u konzoli

Ploča za igranje prikazana je tekstualno tako da se unutar konzole isprintaju redovi ploče. Poziva se funkcija ‘main\_loop’ koja sadrži beskonačnu petlju unutar koje se odvija igra šaha. Na početku, korisnik može izabrati koji tip igre želi pokrenuti : igrač protiv igrača, igrač protiv računala ili računalo protiv računala.

Ako je igrač korisnik, čeka se naredba za pokretanje figura koja se učitava u program. Naredba mora biti formata četiri znaka koje predstavljaju indeks figure koje se želi pokrenuti i mjesta na koje se ta figura želi postaviti. Postoje još i dodatne opcije koje korisnik može odabrati: reset, za ponovo pokretanje igre, i undo, za poništavanje zadnjeg poteza. Ako je igrač računalo, poziva se funkcija iz ChessAI ovisno koji s kojim programom se želi igrati (u slici dolje to je MinMax program). Funkcija vraća string od 4 znaka koji predstavlja potez.

Nakon što se napravi potez mijenja se igrač te se poziva funkcija ‘set\_up’ ChessRules klase. Nakon toga se provjerava da li je trenutni igrač u mat, šah-mat ili pat stanju. Ako se radi o jednom od zadnja dva slučaja igra se zaustavlja, proglašava se pobjednik i čeka se da korisnik pritisne znak ‘y’ kako bi se igra resetirala.



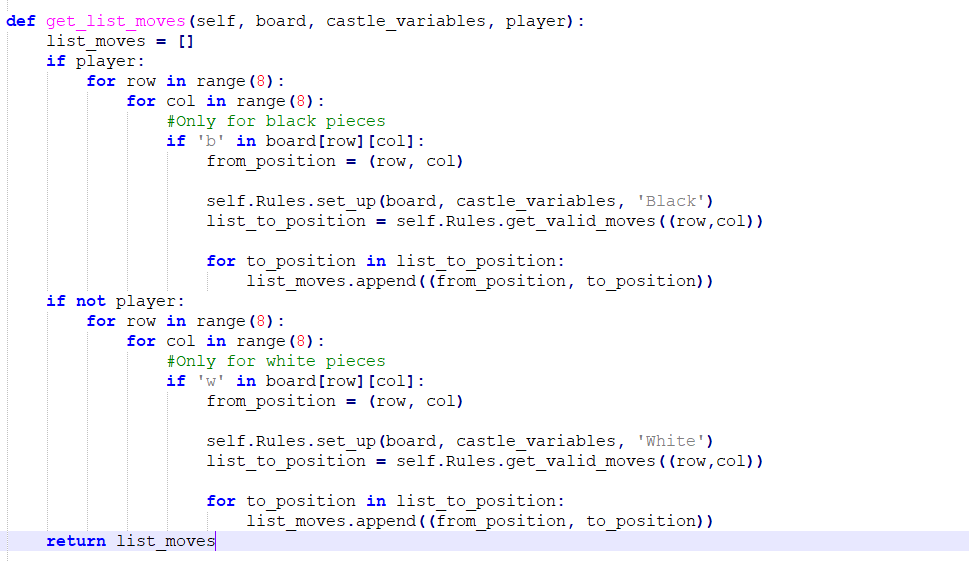
Slika 12 Glavna petlja u kojoj se odvija igra

**3. Šahovski računalni program**

Šahovski računalni program predstavlja program koji analizira moguće poteze i generira listu poteza koju smatra najboljima. Najpoznatiji su Stockfish, Komodo, Bagatur i Alphazero. Stockfish je trenutno najbolji program u svijetu i ujedno je *opensource* program.

U ovom seminaru napravljeno je više računalnih programa, pri čemu su jedni jednostavniji i brži, a drugi složeniji i sporiji u odlučivanju poteza.

U nastavku seminara prikazan je dio koda korišten za šahovski računalni program.



Slika 13 Funkcija koja vraća listu mogućih poteza

Najvažnija funkcija ChessAI klase je 'get\_list\_moves' funkcija. Potrebno je funkciji predati varijable stanja ploče, stanja varijabli za rokadu i trenutnog igrača kao bi vratila listu svih mogućih poteza trenutnog igrača. To radi pronalaženjem svih pozicija figura trenutačnog igrača i pozivanjem funkcije 'get\_valid\_moves' ChessRules klase.

**3.1. SillyBot**

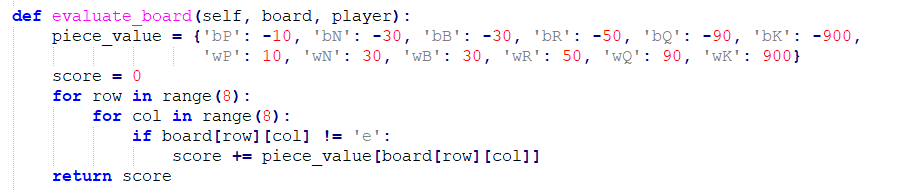
A screenshot of a cell phone

Description automatically generated

Slika 14 Jednostavni program koji radi random poteze

'silly\_bot' je jednostavni program koji vraća nasumični potez iz liste mogućih poteza. Korišten je za otkrivanje mogućih pogrešaka u dotad napisanom programu.

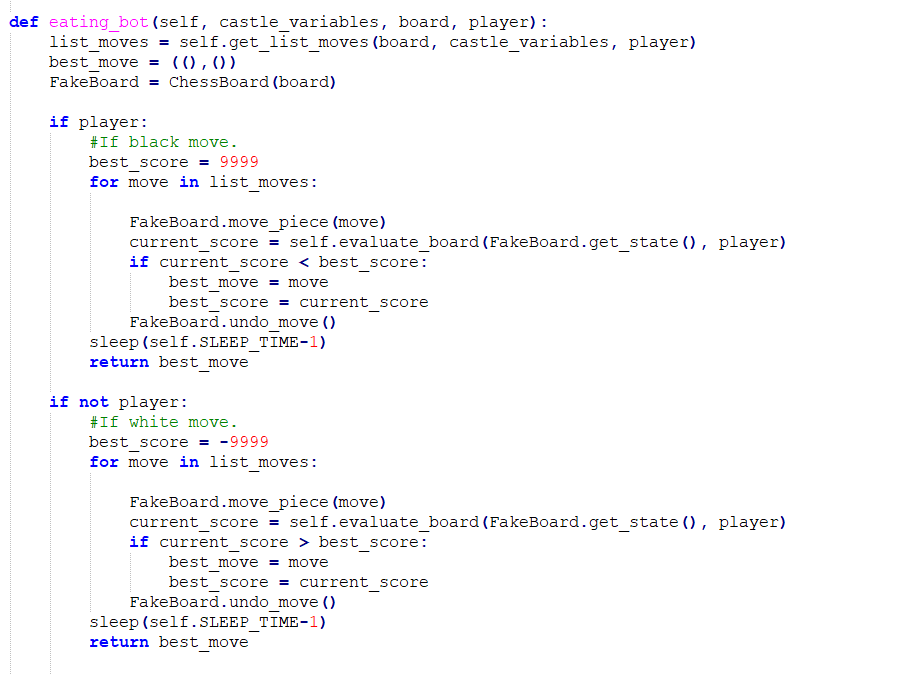
**3.2. EatingBot**

****

Slika 15 Funkcija koja procjenjuje vrijednost ploče

Kako bi računalni program mogao bolje 'igrati' šah potrebna je tehnika ocjenjivanja ploče kako bi program mogao odlučiti da li je potez dobar ili loš.

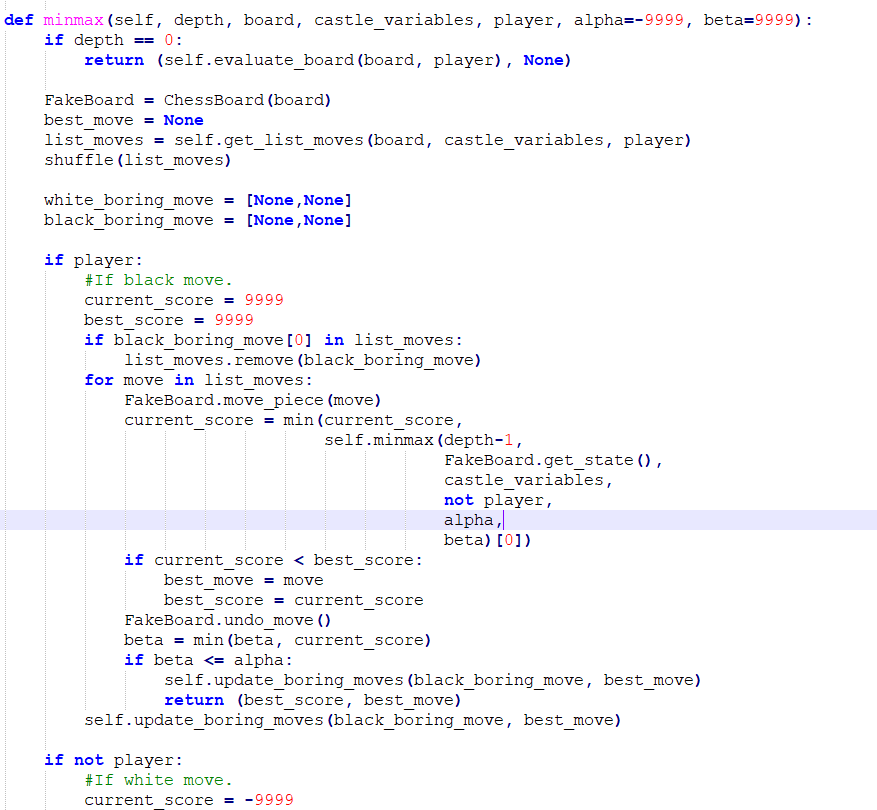
Funkcija 'evaluate\_board' daje ocjenu trenutačne ploče. Svakoj figuri pridodani su bodovi ovisno o vrijednosti figure. Figure crnog igrača imaju negativne vrijednosti, a figure bijelog igrača imaju pozitivne vrijednosti. Ukupna ocjena ploče je jednaka zbroju vrijednosti svih figura. U ovom slučaju crni igrač želi što manju ocjenu, tj. Kada pojede protivničku figuru ocjena ploče se smanjuje. Bijeli igrač želi postići što veću ocjenu.

****

Slika 16 Program koji ciljno bira poteze

‘eating\_bot’ je šahovski računalni program koji određuje najbolji potez prema ocjeni ploče. Ovaj program poziva funkciju ‘get\_list\_moves’ kako bi dobio listu svih trenutačno mogućih poteza. Svaki potez iz liste mogućih poteza se odrađuje na virtualnoj ploči (‘FakeBoard’- razčičita instanca klase ChessBoard od one u ChessMain-u). Za svako novo nastalo stanje virtualne ploče računa se ocjena te se na temelju toga bira trenutačno najbolji potez. Ovaj tip računalnog šahovskog programa ima mogućnost ’pogledati u budućnost’ za samo jedan potez. Ovaj program služi kao osnova za složenije programe.

**3.3. MinMax**

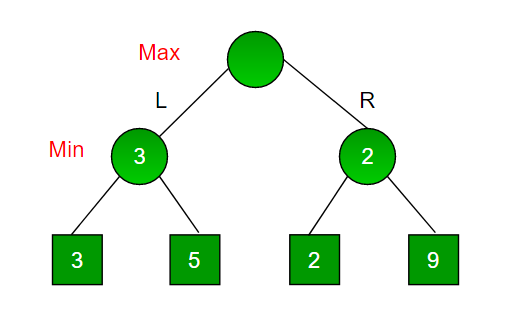


Slika 17 MinMax računalni šahovski program

Program unutar funkcije ‘minmax’ predstavlja složeniju verziju EatingBot-a. Ima mogućnosti ‘pogledati’ u moguća buduća stanja ploče. Broj poteza koje program gleda u naprijed određuje se argumentom ‘depth’. Preporučena dubina gledanja je 3, za koju programu treba prosječno 3 sekunde kako bi odigralo potez. Ovaj program koristi minimax algoritam za pronalaženje najboljeg poteza.

**3.3.1. Minimax algoritam**

Minimax algoritam često se koristi pri problemu odabiranja najboljeg poteza dva igrača. Minimax algoritam u slučaju šaha gradi stablo mogućih stanja ploče gdje svaki red prikazuje moguće poteze jednog igrača.



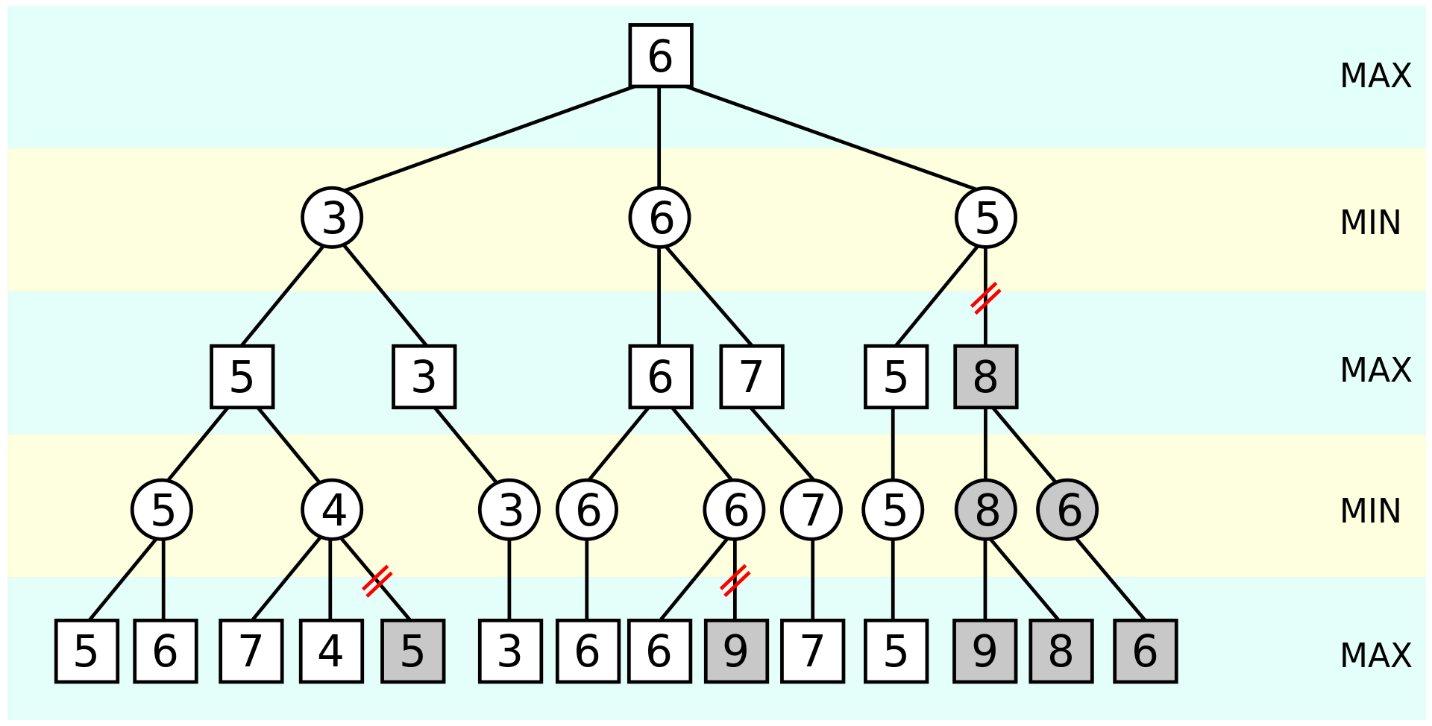
Slika 18 Jednostavno stablo

Korijen stabla je trenutačno stanje ploče. Sljedeći nivo stabla predstavlja sve moguće poteze za jednog igrača (npr. bijeli igrač) dok nivo nakon toga predstavlja sve moguće poteze koje drugi igrač (npr. crni igrač) može odigrati u odgovoru na potez prošlog igrača. Stablo se tako grana do željene dubine. Algoritam će izračunati ocjenu ploče za svaki slučaj i prema tome odlučiti najbolji potez. Izgradnja stabla unutar algoritma napravljena je for petljom i rekurzijom. Svaki poziv 'minmax' funkcije predstavlja čvor stabla a for petlja unutar algoritma predstavlja grananje slučajeva iz tih čvorova.

Ovaj pristup je komputacijski težak. Recimo da oba igrača u nekom trenutku mogu napraviti n poteza. Na dubini od 3 broj stanja za koje je potrebno izračunati ocjenu je n + n\*n + n\*n\*n. Ako bi svaki igrač imao točno 20 poteza u svakom od slučaja to bi značilo da program mora izračunati 8.420 stanja ploče. Ovaj broj se može smanjiti korištenjem alpha-beta pruning algoritmom.

**3.3.2. Alpha–beta obrezivanje**

Alpha-beta obrezivanje je algoritam koji služi za smanjivanje broja čvorova minimax stabla. Ovaj algoritam prekida daljnje pretraživanje grana nekog čvora ako pronađe potez koji je znatno bolji od onoga koji se trenutno pretražuje.



Slika 19 Alpha-beta obrezivanje

Algoritam kontrolira dvije vrijednosti, alpha i beta, koji predstavljaju minimalnu vrijednost koju max-igrač (koji pokušava imati najveću vrijednost) sigurno može imati, odnosno maksimalnu vrijednost koju min-igrač (koji želi postići što manju vrijednost) sigurno može imati. Na početku alpha ima najveću negativnu vrijednost a beta ima najveću pozitivnu vrijednost. Kada maksimalna vrijednost koju min-igrač sigurno ima postane manja od minimalne vrijednosti koju max-igrač sigurno može imati (beta <= alpha), max-igrač ne mora dalje pretraživati grane ovog čvora jer nikada neće biti postignut u pravoj igri (pod uvjetom da min-igrač odigra najbolji mogući potez na njegovom redu).

**4. Zaključak**

U ovom seminaru realizirana je šahovska igra i više različitih tipova šahovskih računalnih programa, od jednostavnih do složenijih. Šah bolje igraju složeniji programi, kao što je i očekivano, ali je potrebno više vremena programu za „razmišljanje“.

**Literatura**

<https://en.wikipedia.org/wiki/Chess_engine>

<https://en.wikipedia.org/wiki/Stockfish_(chess)>

<http://ccrl.chessdom.com/ccrl/404/>

<https://github.com/official-stockfish/Stockfish>