Katedra za računarstvo

Elektronski fakultet, Univerzitet u Nišu

Veštačka inteligencija

Projekat – Slaganje (*Byte*)

Osnovne informacije

- Cilj projekta:
  - Formulacija problema
  - Implementacija algoritma za traženje (algoritma za igru)
  - Implementacija procene stanja korišćenjem pravila i zaključivanja
- Jezik: Python
- Broj ljudi po projektu: 3
- Datum objavljivanja projekta: 13.11.2023.
- Rok za predaju: 24.12.2023.

Ocenjivanje

- Broj poena:
  - Projekat nosi maksimalno 20% od konačne ocene
  - Poeni se odnose na kvalitet urađenog rešenja, kao i na aktivnost i zalaganje studenta
- Status:
  - **Projekat je obavezan!**
  - **Minimalni broj poena koji se mora osvojiti je 5!**
  - Očekuje se od studenata da ozbiljno shvate zaduženja!
  - Ukoliko ne uradite projekat u predviđenom roku, naredna prilika je tek sa sledećom generacijom, po pravilima i na temu koja će biti definisana za novi projekat!

Takmičenje/turnir

- Posle predaje projekta biće organizovano takmičenje.
- Planirani termin takmičenja je sredina januara.
- Prva tri mesta na turniru donose dodatne bodove:
  - 5 bodova za prvo mesto,
  - 3 boda za drugo i
  - 2 boda za treće mesto
- Računaju se kao dodatni bodovi se za angažovanje u toku semestra.

Pravila ponašanja

- Probajte da uradite projekat samostalno, bez pomoći kolega iz drugih timova i prepisivanja.
- Poštujte tuđi rad! Materijal sa Web-a i iz knjiga i radova možete da koristite, ali samo pod uslovom da za sve delove koda ili rešenja koje ste preuzeli navedete referencu!
- Ne dozvolite da drugi prepisuje od vas, tj. da drugi koristi vaš rad i vaše rezultate!
- Ne dozvolite da član tima ne radi ništa! Dogovorite se i pronađite zaduženja koja on može da uradi. Ako mu ne ide, pronađite druga zaduženja.

Faze izrade projekta

- Formulacija problema i implementacija interfejsa
  - Rok: 29.11.2023. godine
- Implementacija operatora promene stanja
  - Rok: 10.12.2023. godine
- Implementacija Min-Max algoritma za traženje sa alfa-beta odsecanjem i algoritama za procenu stanja (heuristike)
  - Rok: 24.12.2023. godine

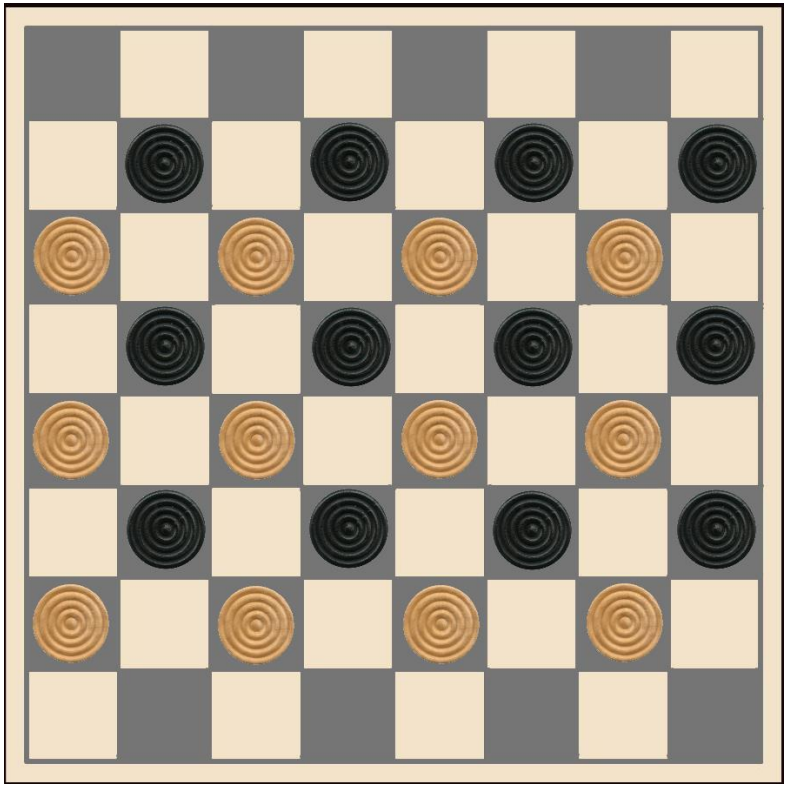
- Rezultat svake faze je izveštaj koji sadrži dokument sa obrazloženjem rešenja i datoteku (datoteke) sa kodom.

Igra Slaganje(*Byte*)

Opis problema Slaganje (*Byte*)

- Problem je igra Slaganje(*Byte*).
- Strateška igra gomilanja figura unapred postavljenih na tabli.
- Tabla je šahovska tabal kvadratnog oblika sa n×n polja tako da je **n parno** broj i **broj figura** na tabli mora biti **deljiv sa 8.**
  - Preporučena veličina je 8×8
  - Maksimalna veličina je 16×16
- Dva igrača crni i beli (X i O) naizmenično odigravaju po jedan potez.
- Figure se nalaze na crnim poljima table i kreću se samo dijagonalno za jedno polje.
- Na početku se figure jednog igrača nalaze u parnim, a drugog u neparnim redovima, pri čemu su prvi i poslednji red prazni.
- Pobednik je igrač koji složi više stekova od 8 figura na čijem je vrhu figura njegove boje.
- Igra čovek protiv računara i moguće izabrati da prvi igra čovek ili računar

Slaganje (*Byte*) – Početno stanje



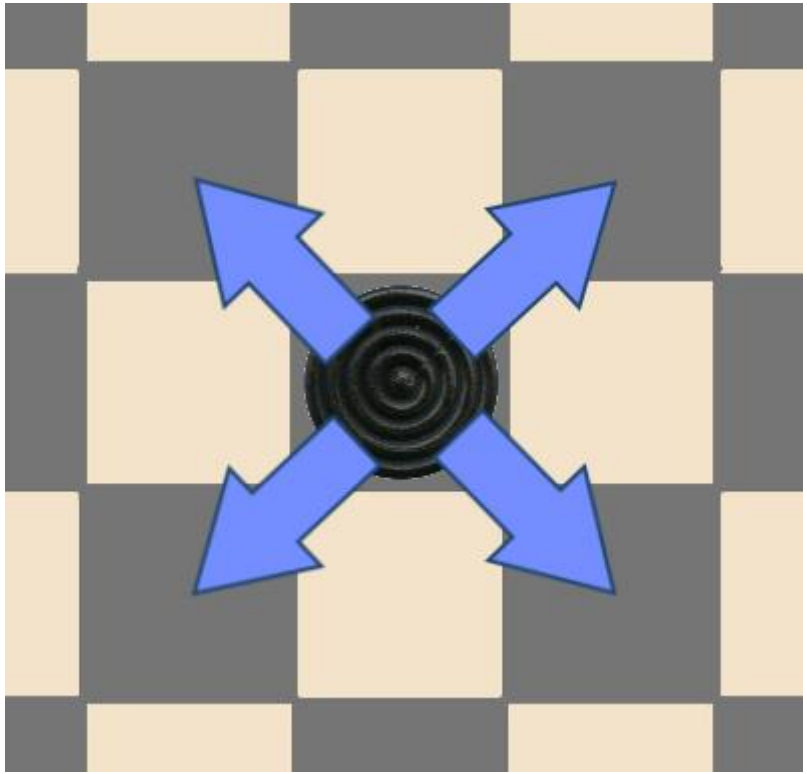
Elementi igre Slaganje (*Byte*)

- Stekovi:
  - Na tabli je moguće formirati stekove od **1 - 7** figura
  - Redosled boja figura na steku može biti proizvoljan
  - Kada se napravi stek od 8 figura on se uklanja sa table
    - Pločica **sa vrha** steka određuje vlasnika tog steka (bela pločica na vrhu znači da je vlasnik steka beli igrač i obrnuto)
  - Npr. u igri sa tablom 8x8 se mogu formirati 3 steka, dok se u igri sa tablom 10x10 može formirati 5 stekova.
- Pobednik je igrač koji ima više stekova u vlasništvu.
  - Ako je igrač vlasnik više od polovine stekova (npr. 2/3 za tablu 8x8 i 3/5 za tablu 10x10) igra može da se prekine, jer je pobednik poznat.

Potezi u igri Slaganje (*Byte*)

- Koriste se samo tamna polja na tabli
- Pomeranje figure se vrši isključivo dijagonalno za jedno polje
- Igrač u jednom potezu može pomeriti jednu figuru svoje boje bez obzira gde se nalazi na steku (na dnu, u sredini ili na vrhu), ako su ispunjena pravila pomeranja
- Ako igrač pomera neku figuru koja je u steku, onda se sa njom pomeraju i sve figure koje se nalaze na njoj i postavljaju na poslednju figuru na steku susednog polja
  - Pomeranjem se može izvršiti umanjenje steka za određeni broj figura i uvećanje njegovog suseda za isti broj figura.
  - Pomeranje celokupnog steka je moguće ako najniža figura pripada igraču koji je na potezu

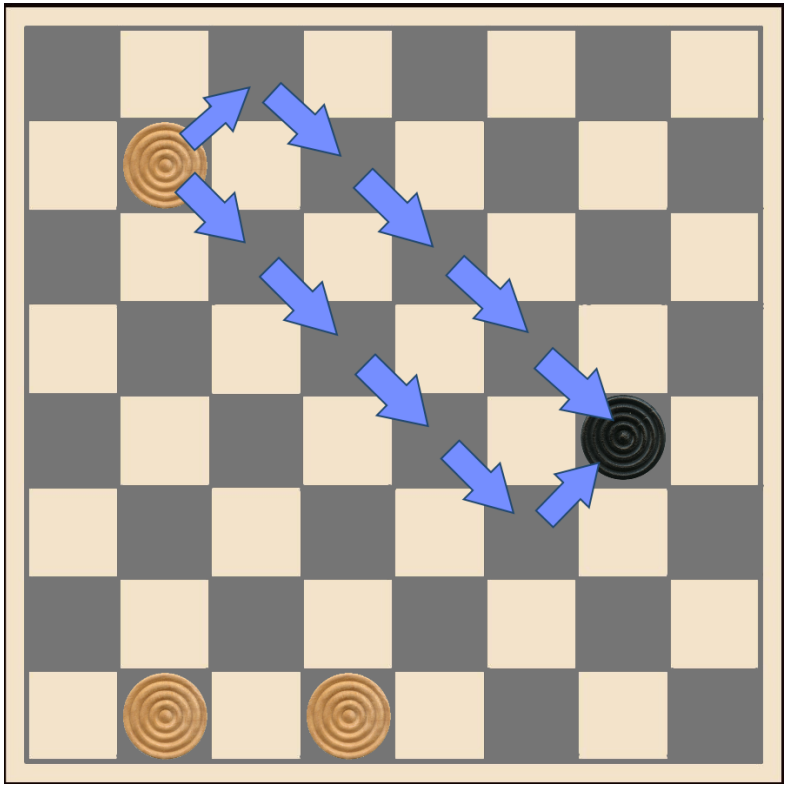
Potezi u igri Slaganje (*Byte*)



Dozvoljeni potezi u igri Slaganje (*Byte*)

- Igrač ne može pomeriti stek (figuru) na prazno polje, ako postoji bar jedno susedno polje na kome postoji stek (figura)
- Ako su sva susedna polja prazna ceo stek (figuru) je moguće pomeriti na neko od ovih polja prema sledećim pravilima:
  - Potez je **valjan** samo ukoliko se njime figura **najkraćim putem približava najbližem polju** koje **nije zauzeto**.
  - Udaljenost predstavlja broj poteza potreban da se od jednog polja stigne do drugog
  - Ukoliko postoji više susednih polja preko kojih je moguće najkraćim putem stići do najbližeg (jednog od najbližih) nezauzetih polja, onda sva ovakva susedna polja predstavljaju moguće poteze igrača.

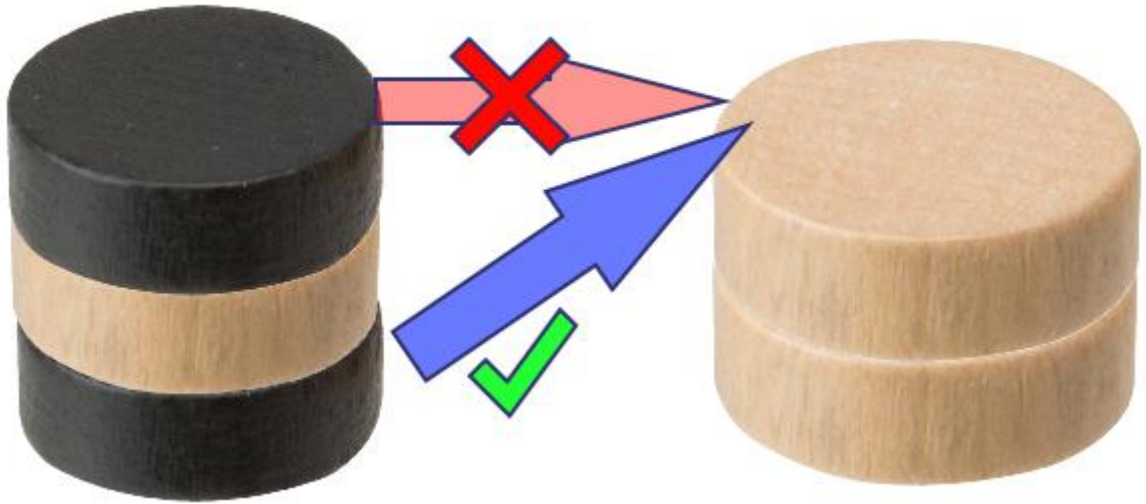
Dozvoljeni potezi u igri Slaganje (*Byte*)



Dozvoljeni potezi u igri Slaganje (*Byte*)

- Ako se na dva susedna polja nalaze stekovi (figure) moguće je premeštanje **figure i svih onih koje su na njoj** sa jednog steka na drugi
  - Figura koja se pomera na susedno polje mora se na steku naći na visini koja je **veća** (ni manja, ni jednaka) od trenutne visine na steku na trenutnom polju
  - Rezultujući stek mora da ima **8 ili manje** figura

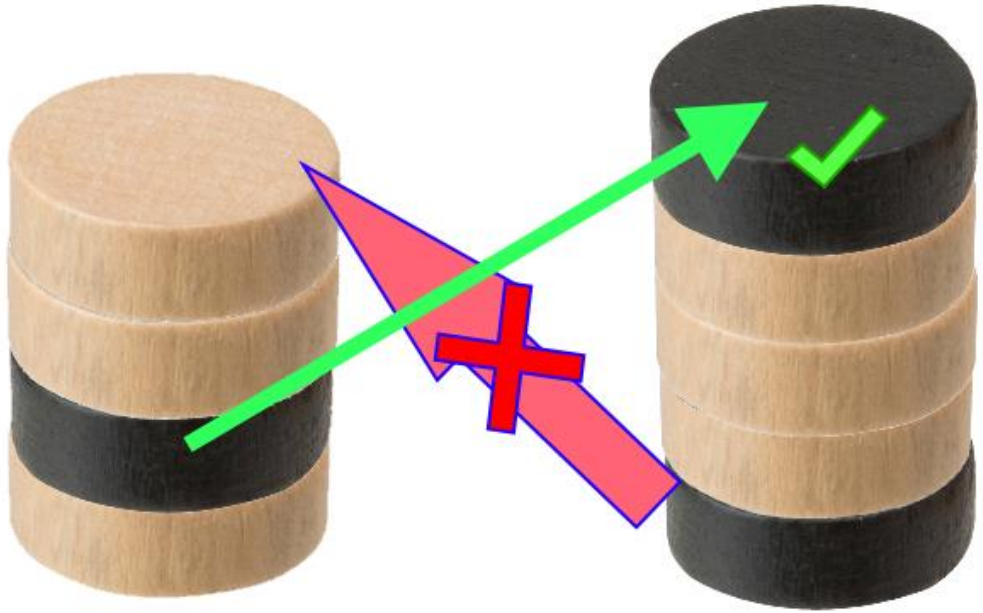
Dozvoljeni potezi u igri Slaganje (*Byte*)



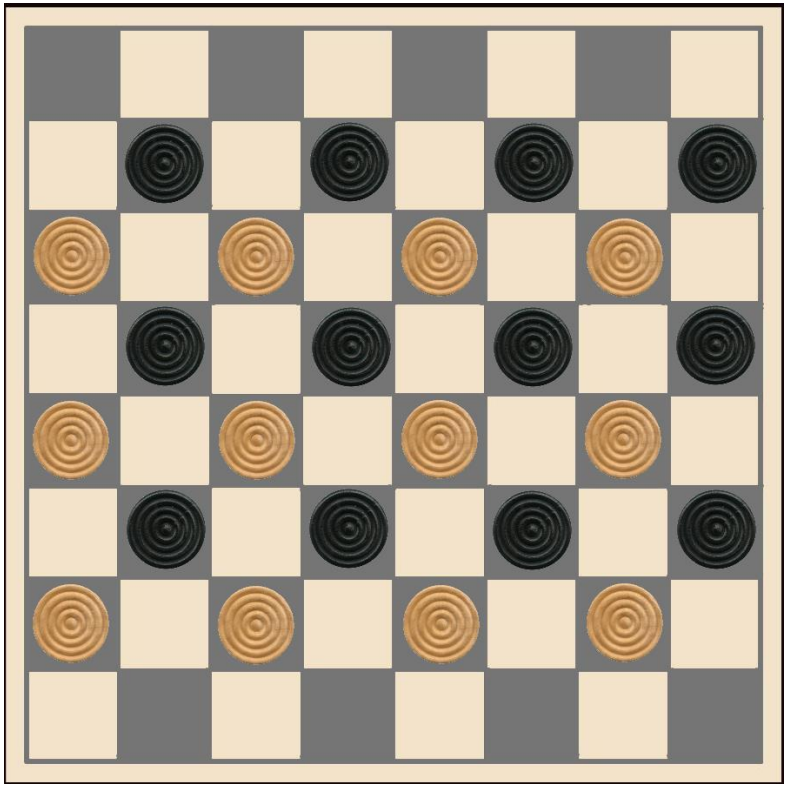
Odigravanje poteza u igri Slaganje (*Byte*)

- Igrač može izabrati bilo koju od svojih figura koje imaju valjane poteze
  - **Udaljenost se ne uzima u obzir kada se razmatraju potezi figura koje se nalaze na različitim poljima**
- Ukoliko igrač ima valjan potez, **mora da odigra potez**, makar i na svoju štetu, kreirajući stek za protivnika
- Ukoliko igrač nema nijedan valjan potez, potez se **prepušta** protivniku

Odigravanje poteza u igri Slaganje (*Byte*)

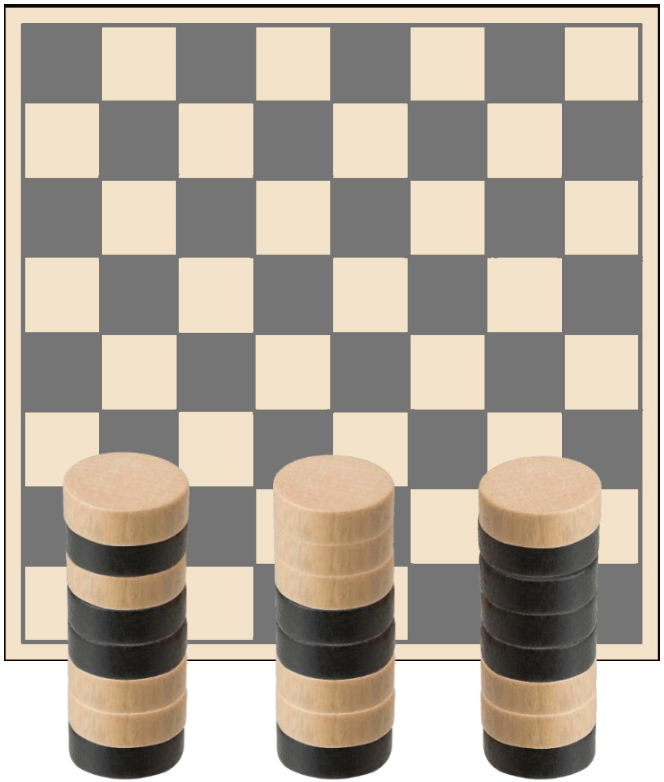


Slaganje (*Byte*) – Početak igre



Slaganje (*Byte*) – Primer stanja igre

Slaganje (*Byte*) – Primer kraja igre



- Tabla je prazna
  - Pobednik je igrač u čijem vlasništvu je više stekova
- Igrač ima više od polovine stekova u vlasništvu
- Primer sa slike:
  - 3-0 za belog igrača
  - Igra je mogla da biti prekinuta kada je formiran drugi stek belog igrača

Slaganje (*Byte*) – Korisni linkovi

- Originalna pravila igre:
  - http://www.marksteeregames.com/Byte_rules.pdf

- Web verzija igre:
  - http://gamesbyemail.com/Games/Byte#Preview

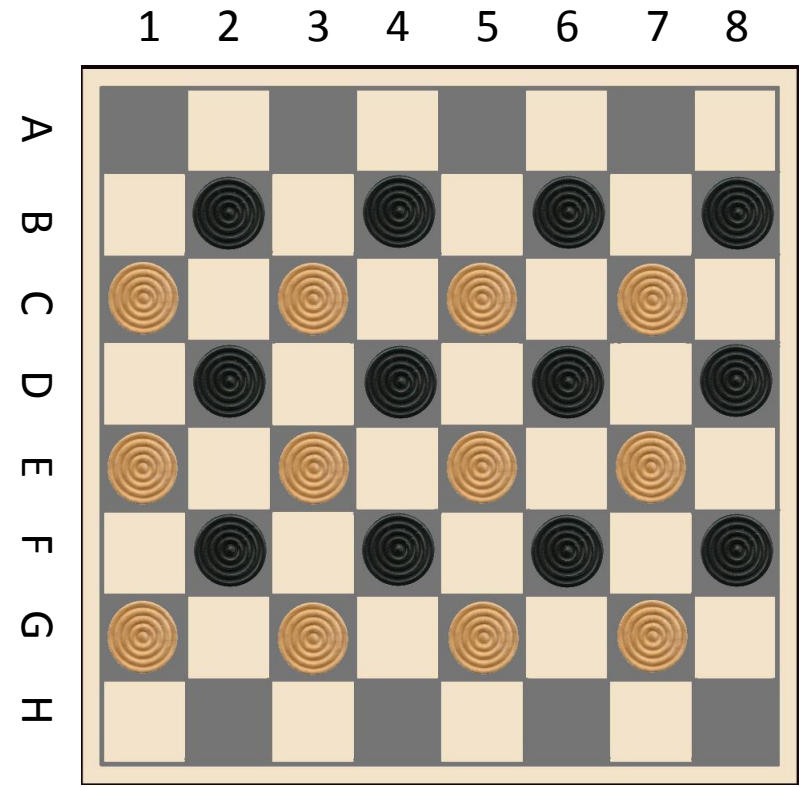
Zadatak I – Formulacija problema i interfejs

- Definisati način za predstavljanje stanja problema (igre)
  - Predstavljanje pozicija figura (stekova) na tabli
- Napisati funkciju za postavljanje početnog stanja
  - Definiše se na osnovu zadate veličine table
- Napisati funkcije za proveru kraja igre
  - Tabla je prazna ili igrač u ima u vlasništvu je više od polovine mogućih stekova
- Napisati funkcije koje proveravaju ispravnost unetog poteza
- **NIJE POTREBNO realizovati funkcije koje proveravaju valjanost poteza i odigravaju potez (faza II)**
- **NIJE POTREBNO realizovati funkcije koje obezbeđuju odigravanje partije (faza II)**

Zadatak I – Formulacija problema i interfejs

- Omogućiti izbor ko će igrati prvi (čovek ili računar)
- Prvi igra uvek igrač X, a drugi igrač O (ili omogućiti izbor koji igrač igra prvi)
- Implementirati funkcije koje obezbeđuju unos početnih parametara igre
  - Unos dimenzije table (n) i provera ispravnosti unosa
- Implementirati funkcije koje obezbeđuju pravljenje inicijalnog stanja problema (igre)
  - Pravljenje stanja igre na osnovu zadatih dimenzija (n) i postavljanje figura na početnim mestima
- Implementirati funkcije koje obezbeđuju prikaz proizvoljnog stanja problema (igre)
  - Prikaz trenutne situacije na tabli sa pozicijama figura
- Realizovati funkcije za unos poteza
  - Potez se sastoji od pozicije polja, mesta figure na steku i smer pomeranja (GL, GD, DL, DD)
- Realizovati funkcije koje proveravaju da li je unos poteza tačan
  - Proveriti da li zadato polje postoji na tabli
  - Proveriti da li postoje figure na zadatom polju
  - Proveriti da li postoji figure na zadatom mestu na steku na zadatom polju
  - Proveriti da li je smer jedan od četiri moguća

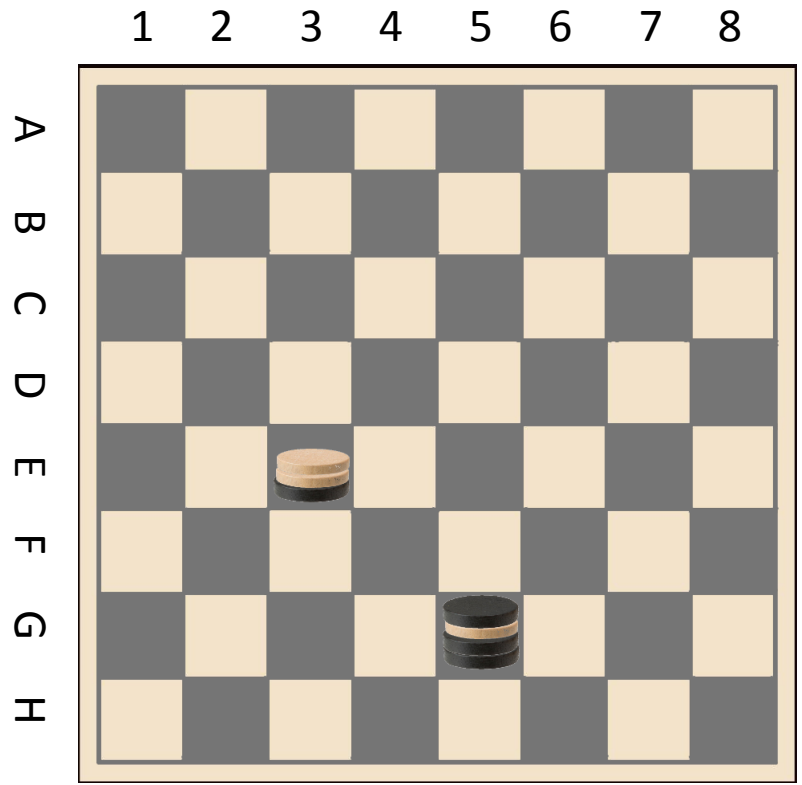
Zadatak I – Interfejs (početno stanje)




```
  1   2   3   4   5   6   7   8
A . . . . . . . . . . . . . . . .
B . . . X . . . X . . . X . . . X
C . . . O . . . O . . . O . . . O
D . . . X . . . X . . . X . . . X
E . . . O . . . O . . . O . . . O
F . . . X . . . X . . . X . . . X
G . . . O . . . O . . . O . . . O
H . . . . . . . . . . . . . . . .

X: 0   O: 0
```

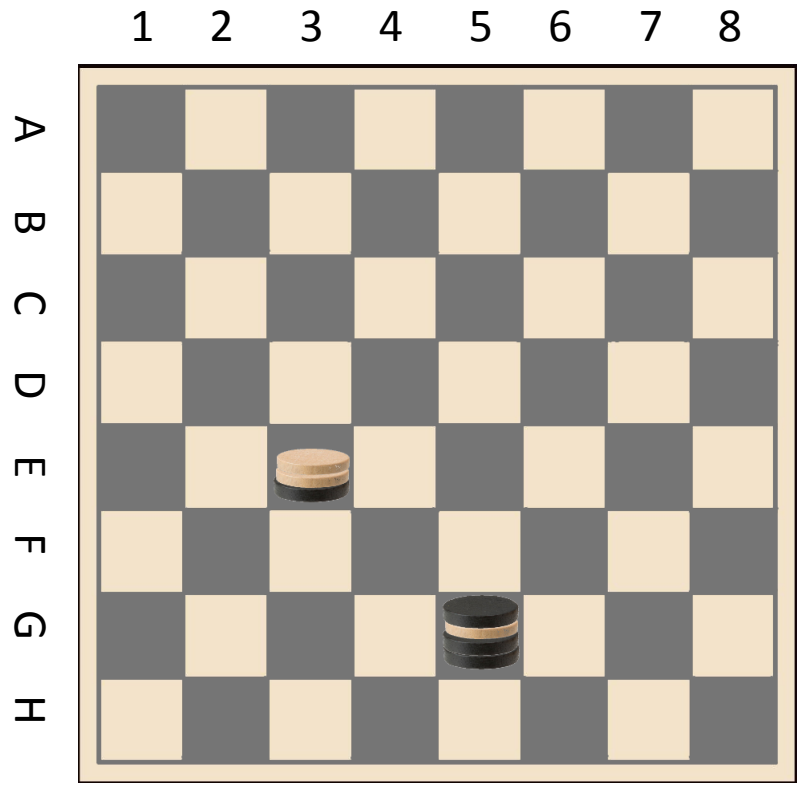
Zadatak I – Interfejs (prikaz stanja)




```
  1   2   3   4   5   6   7   8
A
B
C
D
E       XOO
F
G       X
        XXO
H

X: 0    O: 0
```

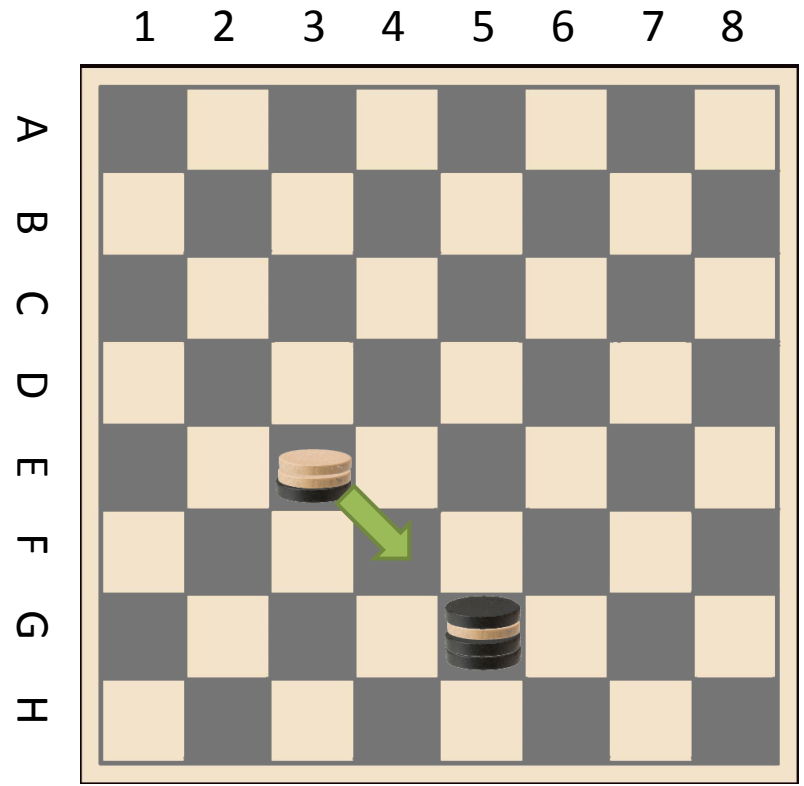
Zadatak I – Interfejs (prikaz stanja)




```
  1   2   3   4   5   6   7   8
A
B
C
D
E       XOO
F
G       X
        XXO
H

X: 0    O: 0
```

Zadatak I – Interfejs (unos poteza)




```
Potez X: [E 3 0 DD]
```

Zadatak II – Operator promene stanja

- Napisati funkcije za proveru valjanosti poteza na osnovu konkretnog poteza i trenutnog stanja problema (igre)
- Napisati funkcije koje na osnovu konkretnog poteza menjaju stanje problema (igre)
- Napisati funkcije koje obezbeđuju odigravanje partije između dva igrača (**dva čoveka, ne računara i čoveka**)
  - Unos početnih parametara i naizmenični unos poteza uz prikaz izgleda stanja igre nakon svakog poteza
- Napisati funkcije za operator promene stanja problema (igre) u opštem slučaju (proizvoljno stanje na tabli)
  - Određivanje svih mogućih poteza igrača na osnovu stanja problema (igre)

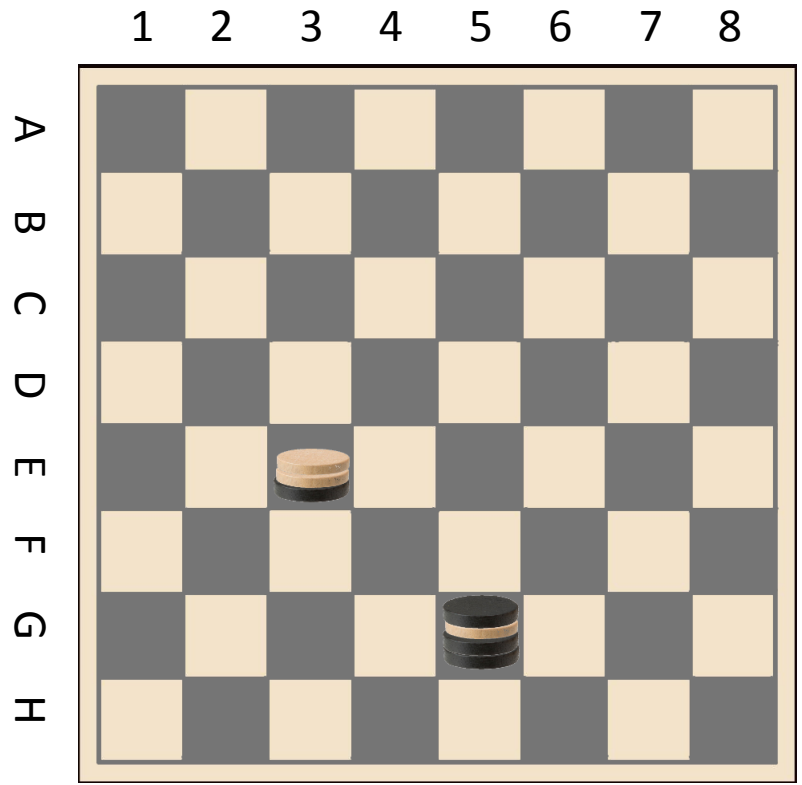
Zadatak II – Operator promene stanja

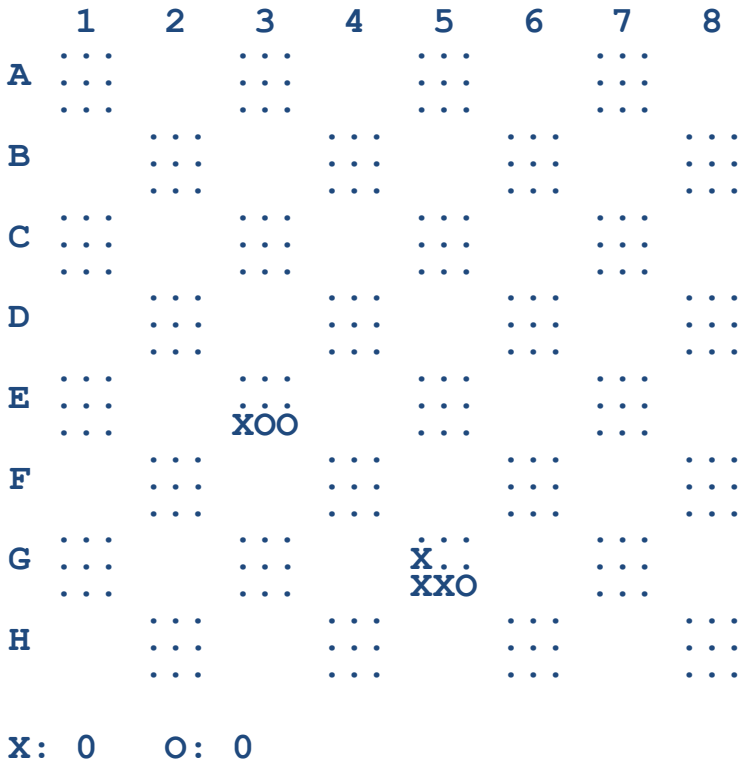
- Realizovati funkcije koje na osnovu konkretnog poteza i stanje problema (igre) proveravaju njegovu valjanost
  - Realizovati funkcije koje proveravaju da li su susedna polja prazna
  - Realizovati funkcije koje na osnovu konkretnog poteza i stanje igre proveravaju da li on vodi ka jednom od najbližih stekova (figura)
  - Realizovati funkcije koje na osnovu konkretnog poteza i stanje igre proveravaju da li se potez može odigrati prema pravilima pomeranja definisanim za stekove
- Realizovati funkcije koje na osnovu konkretnog poteza menjaju stanje problema (igre)
  - Realizovati funkcije koje na osnovu konkretnog poteza menjaju trenutno stanje igre

Zadatak II – Operator promene stanja

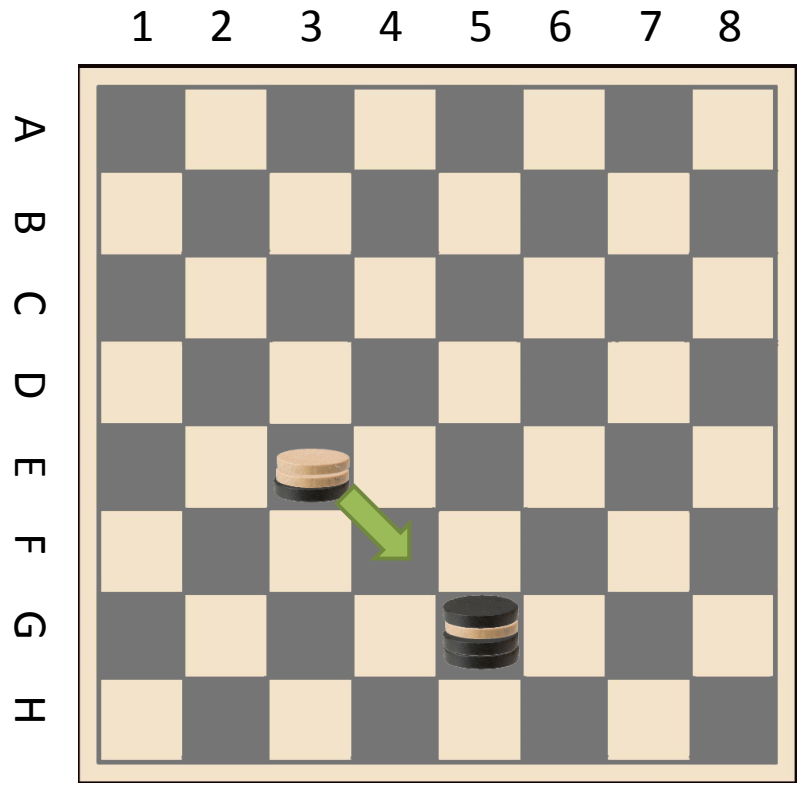
- Realizovati funkcije koje obezbeđuju odigravanje partije između dva igrača (**dva čoveka, ne računara i čoveka**)
  - Unos početnih parametara igre
  - Ponavljanje unosa novog poteza sve dok se ne unese ispravan potez
  - Odigravanje novog ispravnog poteza sa promenom trenutnog stanja igre
  - Prikaz novonastalog stanja igre nakon odigravanja poteza
  - Proveru kraja i određivanje pobednika u igri nakon odigravanja svakog poteza, odnosno promene stanja igre
- Realizovati funkcije koje implementiraju operator promene stanja problema (igre)
  - Realizovati funkcije koje na osnovu zadatog poteza i zadatog stanja igre formiraju novo stanje igre
  - Realizovati funkcije koje na osnovu zadatog igrača na potezu i zadatog stanje igre (table) formiraju sve moguće poteze
  - Realizovati funkcije koje na osnovu svih mogućih poteza formiranju sva moguća stanja igre, korišćenjem funkcija iz prethodne dve stavke

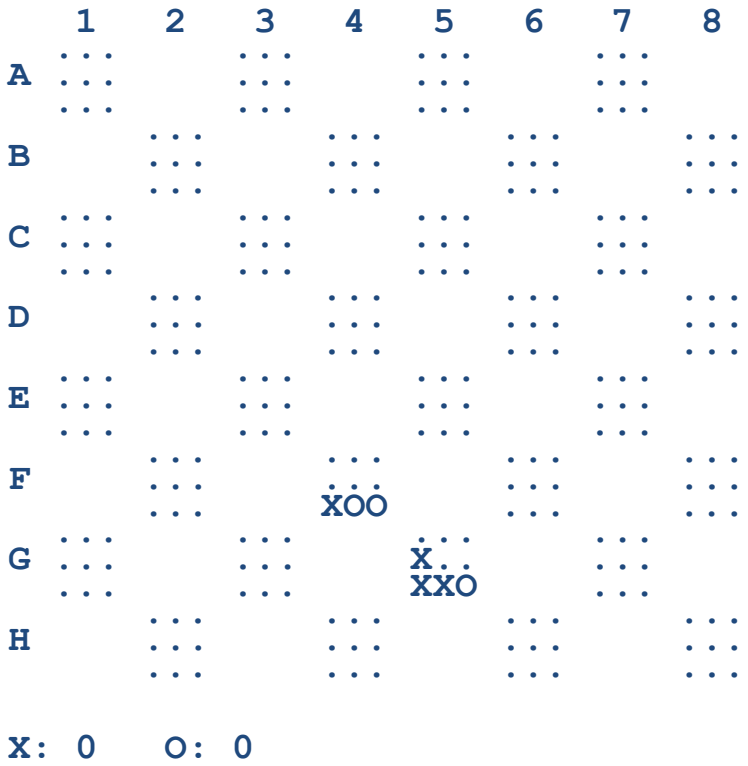
Zadatak II – Promena stanja



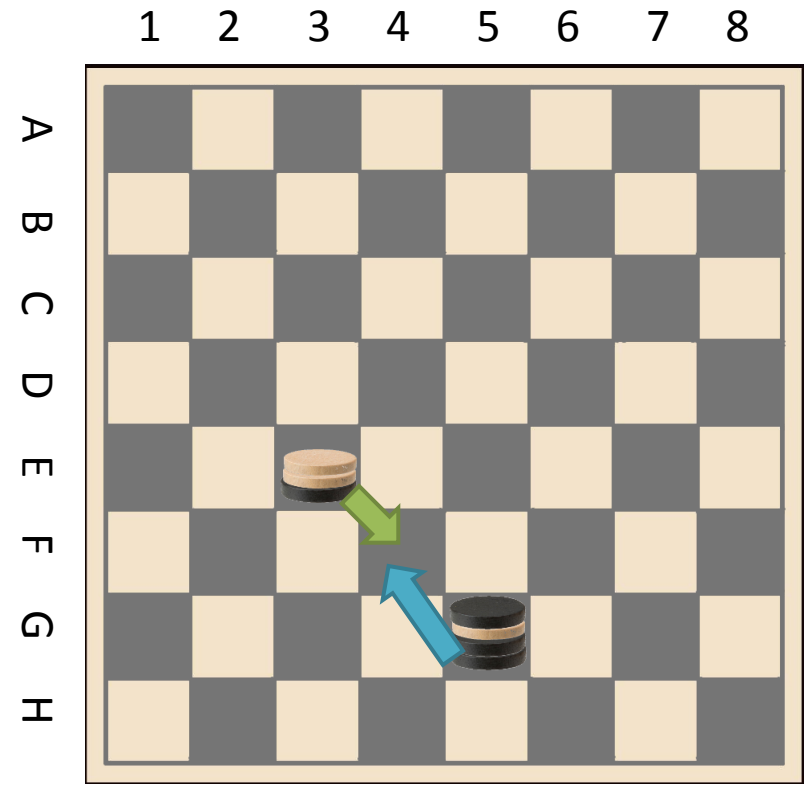


Zadatak II – Promena stanja





Zadatak II – Operator promene stanja




```
Loši potezi X:
1. [E 3 0 GL]
2. [E 3 0 GD]
3. [E 3 0 DL]
4. [G 5 0 GD]
5. [G 5 0 DL]
6. [G 5 0 DD]
7. [G 5 0 ??]
8. [G 5 0 ??]

Dobri potezi X:
1. [E 3 0 DD]
2. [G 5 0 GL]
```

Zadatak III – Min-max algoritam i heuristika

- Implementirati Min-Max algoritam sa alfa-beta odsecanjem za zadati problem (igru):
  - Vraća potez koji treba odigrati ili stanje u koje treba preći
  - Na osnovu zadatog stanja problema
  - Na osnovu dubine pretraživanja
  - Na osnovu procene stanja (heuristike) koja se određuje kada se dostigne zadata dubina traženja
- Realizovati funkcije koje obezbeđuju odigravanje partije između čoveka i računara

Zadatak III – Min-max algoritam i heuristika

- Implementirati funkciju koja vrši procenu stanja na osnovu pravila zaključivanja
- Funkcija za procenu stanja kao parametre treba da ima igrača za kojeg računa valjanost stanja, kao i samo stanje za koje se računa procena.
- Procena stanja se mora vršiti isključivo korišćenjem mehanizma zaključivanja nad prethodno definisanim skupom pravila. Zadatak je formulisati skup pravila i iskoristiti ih na adekvatan način za izračunavanje heuristike.
- Za izvođenje potrebnih zaključaka (izvršavanje upita nad skupom činjenica kojima se opisuje stanje) koristiti mašinu za zaključivanje.
- Implementirati funkciju koja prevodi stanje u listu činjenica …