Projekat – Byte

Vladan Mihajlović
Nataša Veljković
Darko Puflović

Osnovne informacije

- Cilj projekta:
  - Formulacija problema
  - Implementacija algoritama za traženje (algoritama za igre)
  - Implementacija procene stanja korišćenjem pravila i zaključivanja
- Jezik: Lisp
- Broj ljudi po projektu: 3
- Datum objavljivanja projekta: 5.11.2019.
- Rok za predaju: 5.1.2020.

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 od vas da ozbiljno shvatite zaduženja!
  - Ukoliko ne uradite projekat u navedenom roku, naredna prilika je tek sa sledećom generacijom, po pravilima koja će biti tada definisana!

Takmičenje/turnir

- Posle predaje projekta biće organizovano takmičenje.
- Planirani termin takmičenja je sredina januara.
- Prva tri mesta na turniru donose dodatne poene: 5 za prvo mesto, 3 za drugo i 2 za treće mesto (računaju se kao dodatni poeni za angažovanje u toku semestra).

Pravila ponašanja

- Probajte da uradite projekat sami, bez pomoći kolega ili 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 uzeli od nekog navedete referencu!
- Ne dozvolite da od vas neko prepisuje, tj. da neko od kolega koristi vaš rad i vaše rezultate!
- Ako radite u timu, ne dozvolite da vaš kolega iz tima ne radi ništa! Nađite mu zaduženja koja može da uradi – ako mu nešto ne ide, nađite mu druga zaduženja.

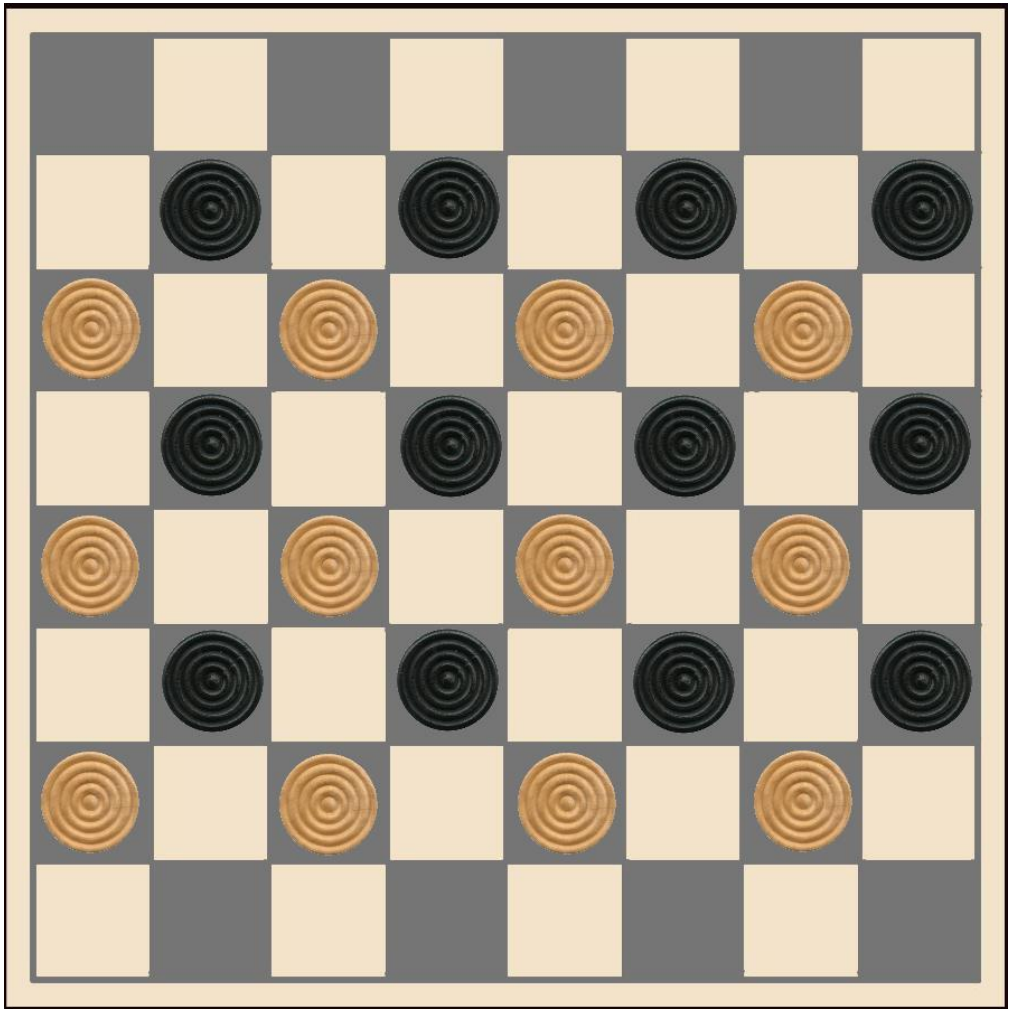
Faze izrade projekta

- Formulacija problema i implementacija interfejsa
  - Rok: 18.11.2019. godine
- Implementacija operatora promene stanja
  - Rok: 2.12.2019. godine
- Implementacija Min-Max algoritma za traženje sa alfa-beta odsecanjem
  - Rok: 16.12.2019. godine
- Definicija heuristike (procena stanja)
  - Rok: 5.1.2020. godine

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

Igra *Byte*

Početno stanje - *Byte*



Opis problema - *Byte*

- Problem je igra *Byte*
- Koristi se šahovska tabla stranice n (**paran broj**, gde broj figura na tabli mora biti **deljiv sa 8**) koju zadaje korisnik (preporučeno n=8, koliko se koristi i za igru; maksimalno n=10)
- Dva igrača crni i beli (X i O) naizmenično odigravaju po jedan potez pomerajući svoje pločice (figure) na proizvoljno polje (ili drugu pločicu, **bez obzira na njenu boju**, ukoliko takva grupa naslaganih pločica **nije veća od 8**)
- Tabla na početku ima raspored pločica sa slike na prethodnom slajdu
- Pobednik je onaj igrač čije se pločice nalaze na vrhu većeg broja naslaganih pločica. Dovoljno je na **većini stekova (2/3 ili 3/5)** i igra tada može da se **prekine**.
- Igra čovek protiv računara i moguće izabrati da prvi igra čovek ili računar

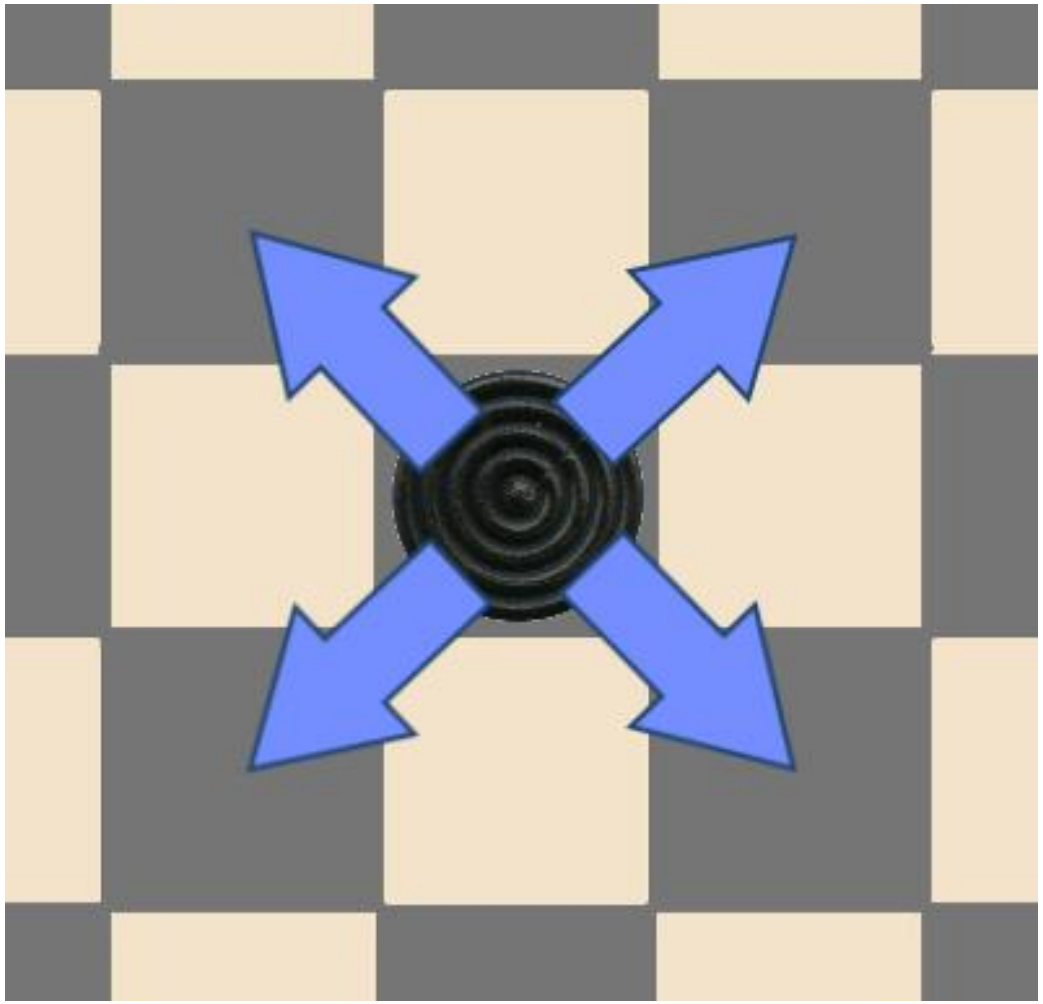
Termini - *Byte*

- Stekovi:
  - Na tabli je moguće formirati stekove od **1 - 8** pločica
  - Redosled boja pločica je nebitan
  - Stek od 8 pločica se uklanja sa table
    - Pločica **sa vrha** je indikator pobednika u tom steku (bela pločica znači da je stek osvojen od strane belog igrača i obrnuto)
  - U igri 8x8 formiraju se 3 steka, dok se u igri 10x10 formira 5. Pobednik je onaj koji ima više osvojenih stekova

Validni potezi - *Byte*

- Koriste se samo tamna polja na tabli
- Pomeranje se vrši isključivo dijagonalno, po jedno polje
  - Ukoliko je na polju stek, **na prazno polje** je moguće pomeriti samo **celokupan stek** i to samo u slučaju da je **najniža** pločica boje igrača koji je na potezu
  - Ukoliko se vrši spajanje sa drugim stekom/diskom, onda je moguće vršiti i razdvajanje postojećeg steka, od mesta gde je pločica koja pripada igraču na potezu pa naviše

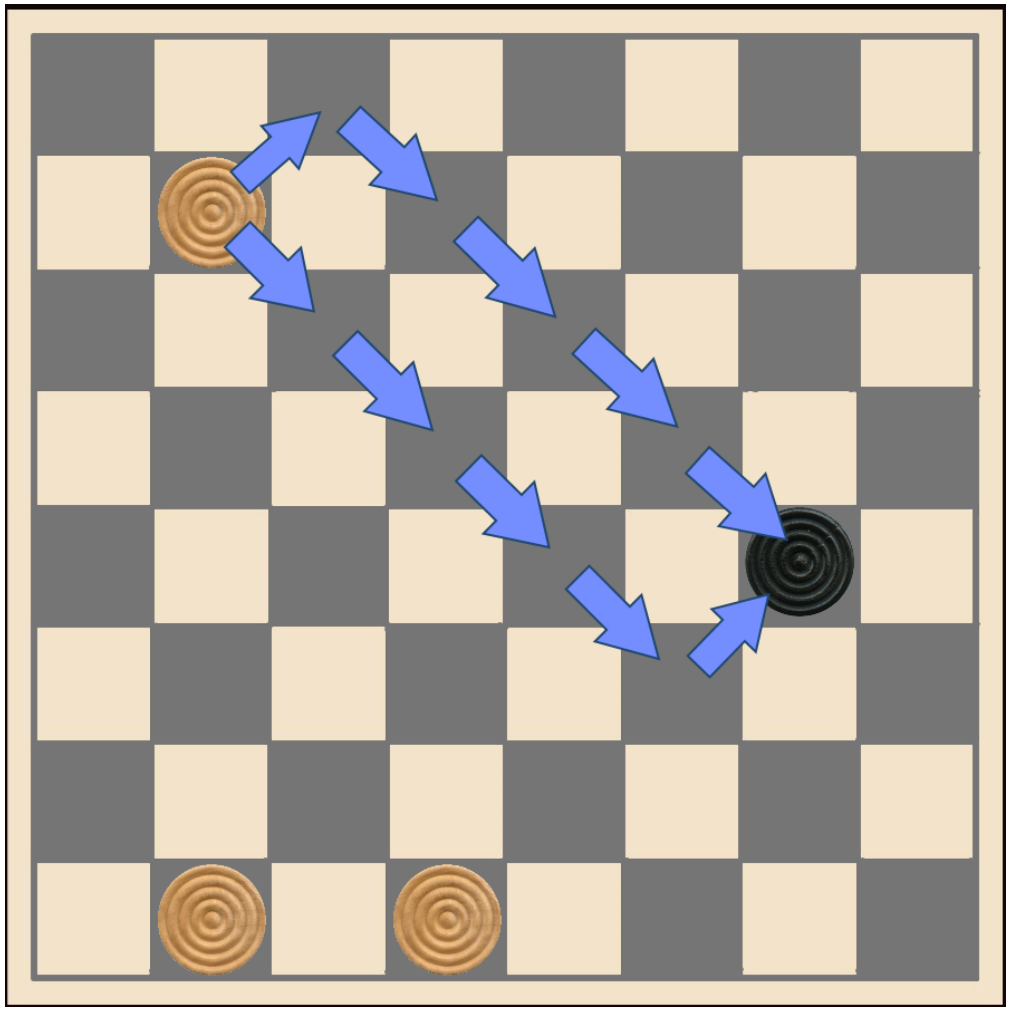
Byte – Primeri poteza



Približavanje - *Byte*

- Potez je validan samo ukoliko se pločica **približava** sebi **najbližoj** pločici ili steku
  - Udaljenost predstavlja broj poteza potreban da se stigne do tog polja
- Ukoliko postoji više polja preko kojih je moguće do najbližeg pločice/steka (ili bilo kog od najbližih pločica/steka) doći sa istom razdaljinom, onda je moguće izabrati **proizvoljno** jedno od tih polja

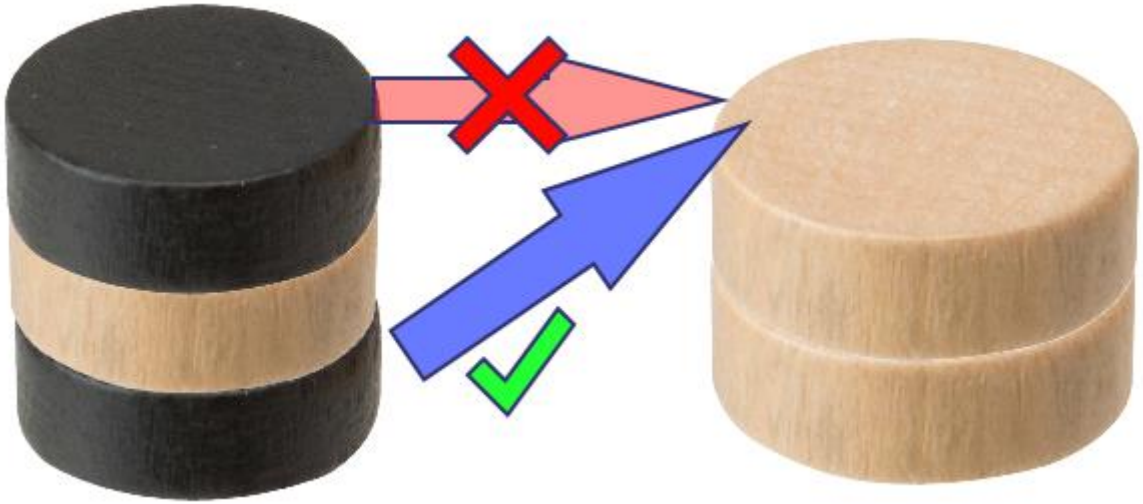
Byte - Primeri poteza



Spajanje - *Byte*

- Ukoliko postoje dva steka na rastojanju 1, moguće je podeliti stek na pločici igračeve boje i premestiti tu pločicu i **sve one koje su na njoj** na drugi stek, pod sledećim uslovima:
  1. Visina steka na koji se premešta pločica mora biti **viša (ne i jednaka)** od pločice sa koje se pomera deo steka
  2. Rezultujući stek mora da ima **manje od 8** ili **tačno 8 pločica** (kada se sklanja sa table)

Byte - Primeri poteza

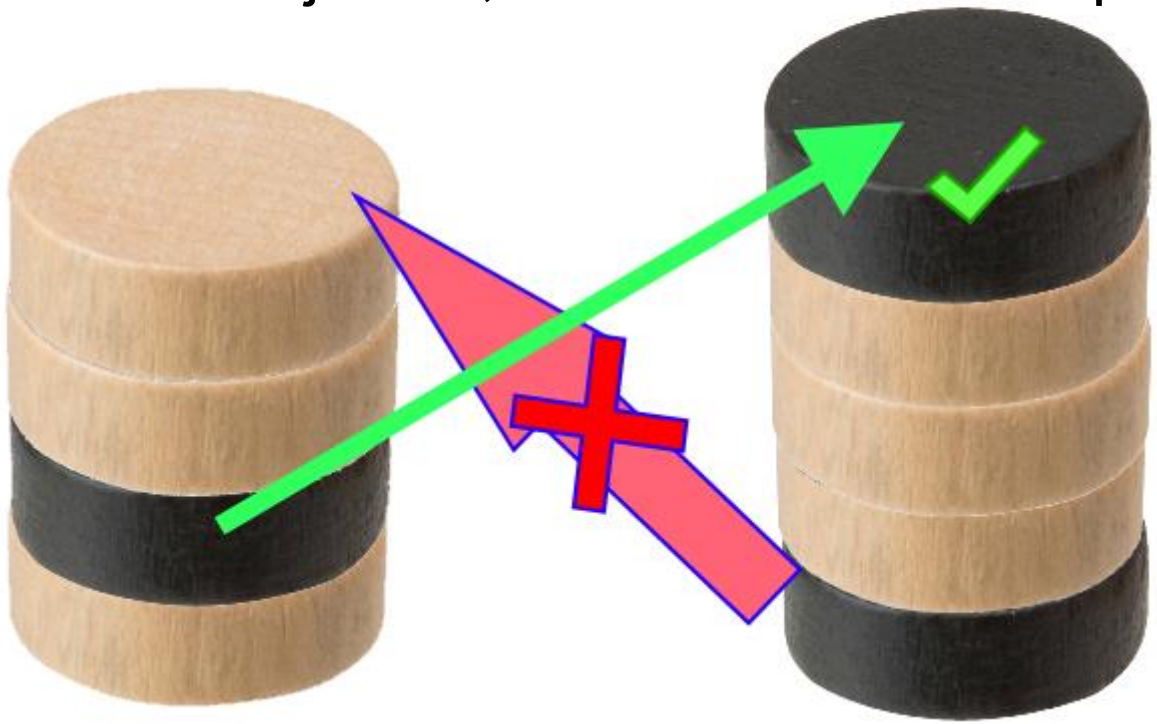


Spajanje i potezi - *Byte*

- Dva steka koja su na rastojanju 1, ne mogu se pomeriti na **prazno polje**, već samo na **polje drugog** steka (ukoliko je validan potez)
  - Igrač **ne mora** da odigra sa tog polja, može da koristi i bilo koju drugu pločicu
- Ukoliko igrač ima validan potez, **mora da odigra potez**, makar i na svoju štetu, kreirajući stek za protivnika
- Ukoliko nema validnih poteza, potez se **prepušta** protivniku

Byte - spajanje i potezi (primer)

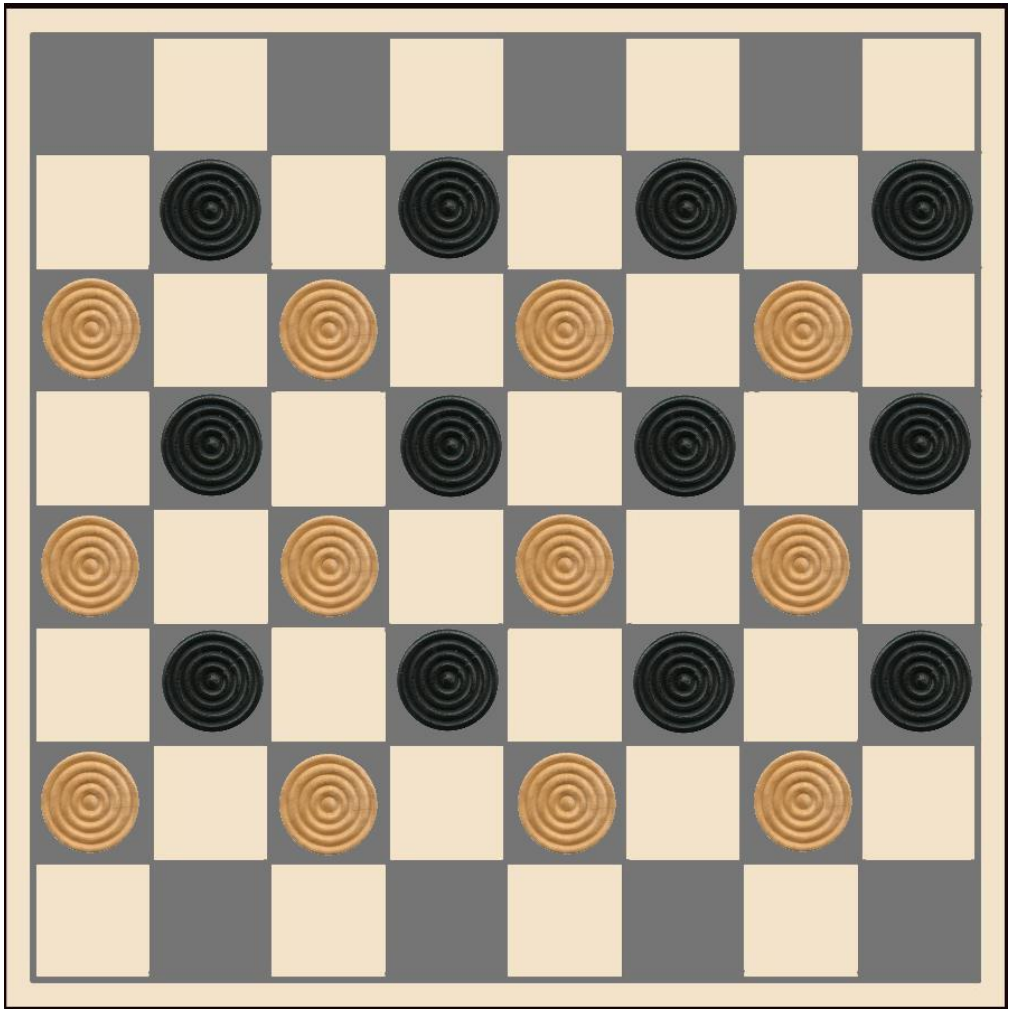
Crni ima samo jedan validan potez, koji ga primorava da kreira stek za protivnika. Crna pločica sa vrha desnog steka bi bila na istoj visini, zato ne može da se pomeri.



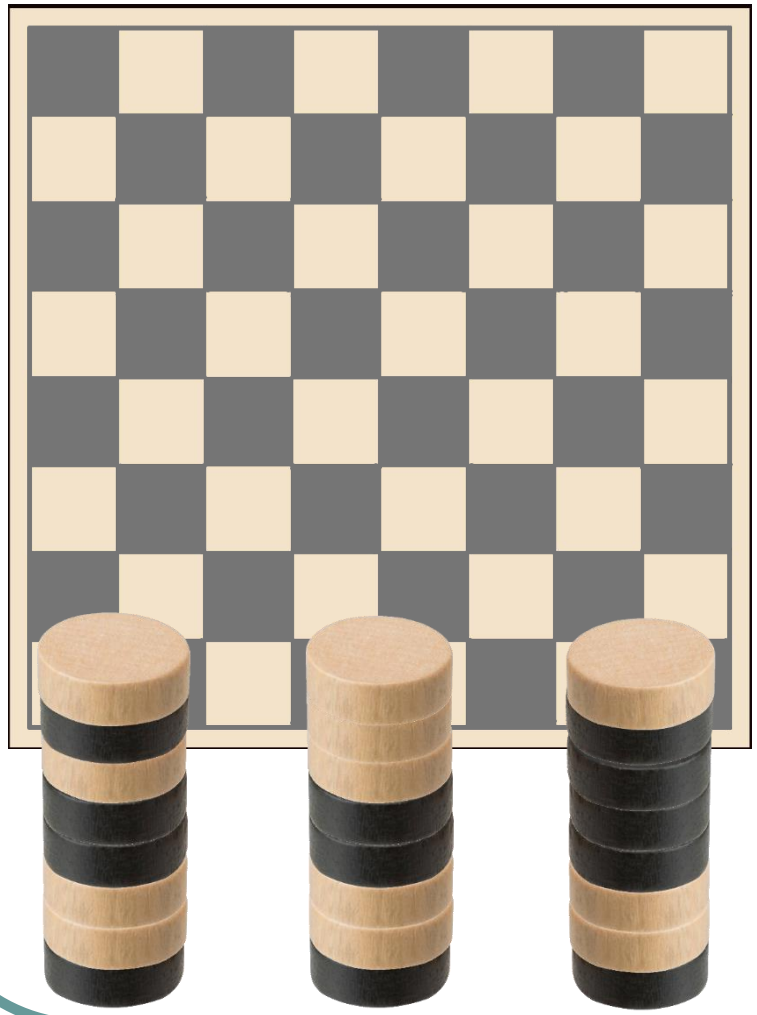
Pravila igre *Byte*

- Igrači povlače poteze naizmenično
- Nerešen rezultat je nemoguć, pobednik uvek postoji

Byte – Početak igre



Byte – Kraj igre



- Tabla je prazna (ili ima 8 (16) figura.
- Postoji 3 (5) stekova (manje ukoliko je dovoljno za pobedu). Pobednik je onaj čije su pločice na vrhu u većem broju stekova
- Primer sa slike: 3-0 za belog igrača (igra je trebalo da bude prekinuta kada je formiran drugi stek).

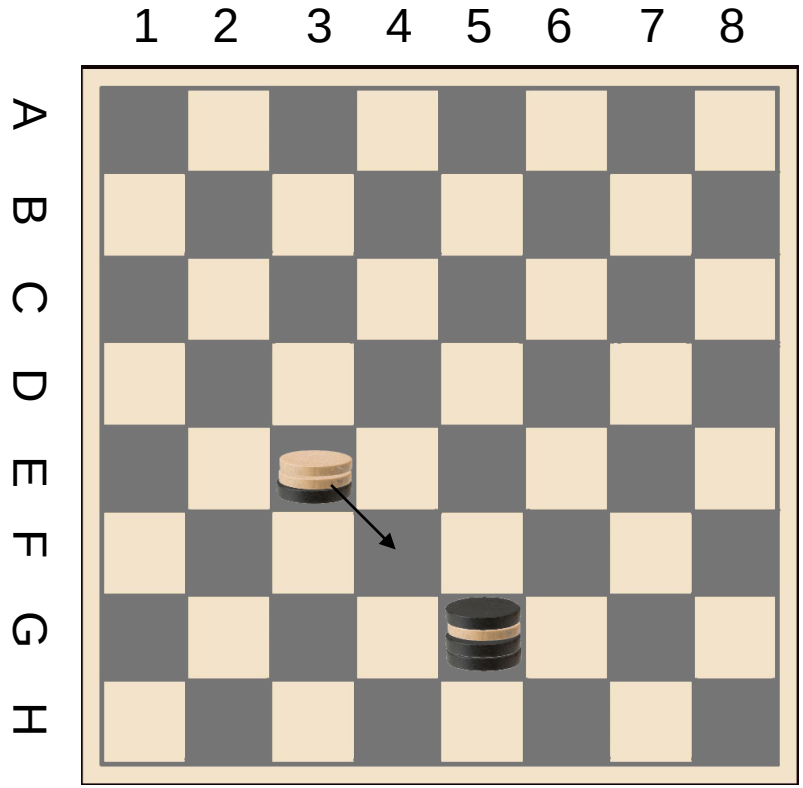
Zadatak I – Formulacija problema i interfejs (1)

- Definisati način za predstavljanje stanja problema (igre)
- Napisati funkciju za postavljanje početnog stanja na osnovu zadate veličine table
- Napisati funkcije za testiranje ciljnog stanja (pobede)

Zadatak I – Formulacija problema i interfejs (2)

- Omogućiti izbor ko će igrati prvi (čovek ili računar)
- Prvi igra uvek igrač X, a drugi igrač O
- Implementirati funkcije koje obezbeđuju prikaz proizvoljnog stanja problema (igre)
- Realizovati funkcije koje na osnovu zadatog poteza igrača, u obliku (vrsta kolona), omogućavaju:
  - proveru da li je potez valjan
  - ako jeste, promenu prosleđenog stanja problema (igre) odigravanjem poteza

Zadatak I – Interfejs



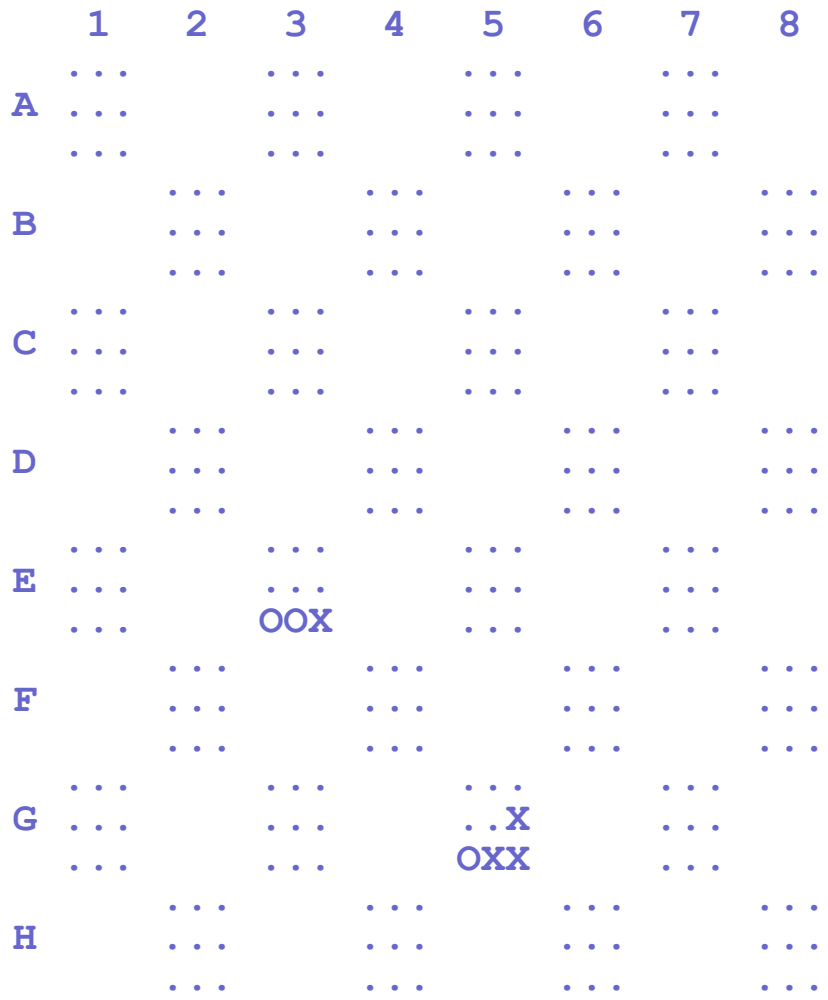
(E 3)–Pozicija sa koje se pomera disk

(F 4)–Pozicija na koju se pomera disk

'0 –Visina sa koje se pomera (ukoliko ne postoji, onda se pomera sa najnižeg diska)

Potez X:('(E 3) '(F 4) '0)

Zadatak I - Interfejs



Zadatak II – Operatori promene stanja

- Napisati funkcije za operatore promene stanja problema (igre) u opštem slučaju (proizvoljno stanje na tabli)
  - Na osnovu trenutne (proizvoljne) situacije na tabli (stanja) i zadatog (validnog) poteza formira novu situaciju na tabli (stanje). Ne menjati postojeće stanje već napraviti novo i na njemu odigrati potez.
  - Na osnovu trenutne (proizvoljne) situacije na tabli (stanja) i igrača koji je na potezu formira listu svih mogućih situacija na tabli (stanja), korišćenjem funkcije iz prethodne tačke
- Realizovati funkcije koje obezbeđuju odigravanje partije između dva igrača (dva čoveka, ne računara i čoveka)
  - unos poteza i provera da li je potez moguć
  - ukoliko nije moguć zahtevati unos novog poteza
  - ukoliko je moguć odigrati ga i promeniti trenutno stanje
  - prikazati novonastalo stanje sistema

Zadatak III – Min-max algoritam

- Implementirati Min-Max algoritam sa alfa-beta odsecanjem za zadati problem
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem ima ulazni parametar kojim se definiše dubina pretraživanja
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem vrati potez koji treba odigrati ili stanje u koje treba preći
- Funkciju za određivanje heuristike ne treba implementirati
  - Napraviti funkciju koja za odgovarajuća stanja vraća karakteristične vrednosti samo u svrhu testiranja ispravnosti napravljenog Min-Max algoritma

Zadatak IV – Heuristika

- U implementaciju Min-Max-a sa alfa-beta odsecanjem dodati funkciju za procenu stanja koja se poziva kada se dostigne zadata dubina traženja.
- Implementirati funkciju koja vrši procenu stanja na osnovu pravila zaključivanja
- Funkcija za procenu stanja kao parametre treba da ima oznaku igrača za kojeg računa valjanost stanja, kao i samo stanje za koju 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 …

Korisni linkovi

- Originalna pravila igre:
  - http://www.marksteeregames.com/Byte_rules.pdf
- Primer igre
  - http://gamesbyemail.com/Games/Byte#Preview