Katedra za računarstvo

Elektronski fakultet, Univerzitet u Nišu

Veštačka inteligencija

Projekat – Atol (*Atoll*)

Osnovne informacije

- Cilj projekta:
  - Formulacija problema
  - Implementacija algoritma za traženje (algoritma za igru)
  - Implementacija procene stanja
- Jezik: Python
- Broj ljudi po projektu: 3
- Datum objavljivanja projekta: 15.12.2025. godine
- Rok za predaju: 1.2.2026. godine

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 februara.
- 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! Moguće je koristiti materijal sa interneta i iz knjiga i radova, ali samo pod uslovom da za sve delove rešenja koje ste preuzeli navedete referencu!
- Ne dozvolite da drugi prepisuju od vas, tj. da drugi koriste vaš rad i vaše rezultate!
- Ako koristite skladišta (repozitorijume) na internetu obezbedite da budu privatna do završetka januarskog ispitnog roka!
- 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: 28.12.2025. godine
- Implementacija operatora promene stanja
  - Rok: 18.1.2026. godine
- Implementacija Min-Max algoritma za traženje sa alfa-beta odsecanjem i algoritama za procenu stanja (heuristike)
  - Rok: 1.2.2026. godine

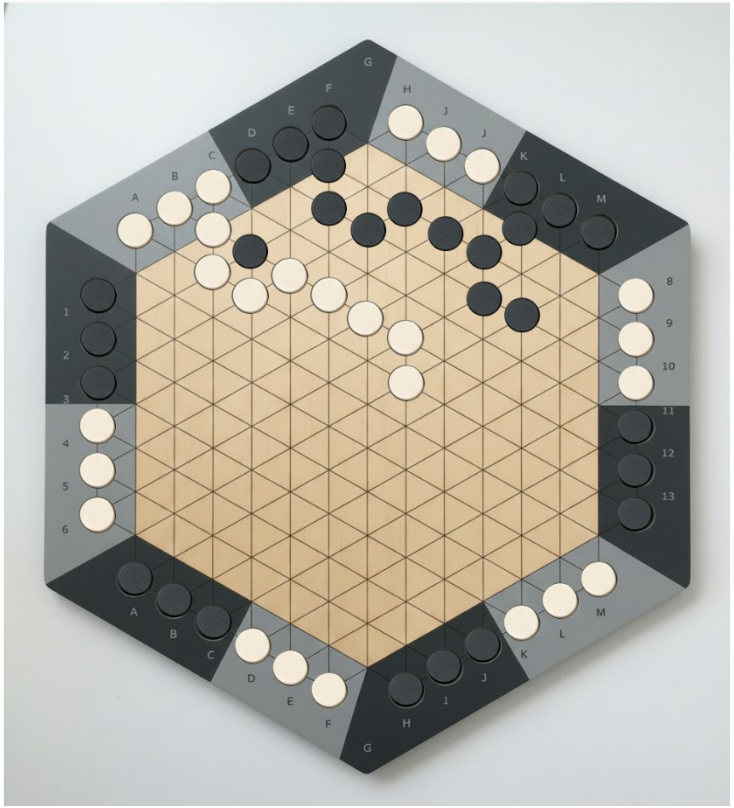
- Rezultat svake faze je izveštaj (arhiva) koji sadrži dokument sa obrazloženjem rešenja i datoteku (datoteke) sa kodom.
- Rezultat svake faze treba postavljati na odgovarajuće zadatke na portalu predmeta.

Igra Atol (*Atoll*)

Opis problema Atol (*Atoll*)

- Problem je igra Atol (*Atoll*).
- Strateška igra povezivanja ostrva na šestougaonoj tabli sa heksagonalnom mrežom.
- Tabla je šestougaonog oblika čija dužina stranice je neparan broj n.
  - Preporučena dužina stranice je 5
  - Maksimalna dužina stranice je 9
- Dva igrača zeleni i crveni (X i O) naizmenično odigravaju po jedan potez.
- Svaki igrač ima svoj komplet kamenčića određene boje.
- Igrač mora da odigra svaki svoj potez, postavljanjem jednog kamenčića.
- Tabla na početku ima samo ostrva koja okružuju šestougaonu mrežu u sredini.
- Pobednik je igrač koji spoji dva ili više svojih ostrva tako da dominira tablom.
- Igra čovek protiv računara i moguće izabrati da prvi igra čovek ili računar

Atol (*Atoll*) – Stanje u toku igre



Elementi igre Atol (*Atoll*)

- Kamenčići:
  - Svaki igrač ima svoj set kamenčića odgovarajuće boje.
  - Kamenčić je moguće postaviti na bilo koje prazno polje.
  - Crni igrač prvi postavlja kamenčić
  - U svakom potezu igrači imaju slobodno polje na koje mogu da postave kamenčić, koji moraju da postave.
  - Kamenčići koji čine ostrva mogu da budu deo puta.

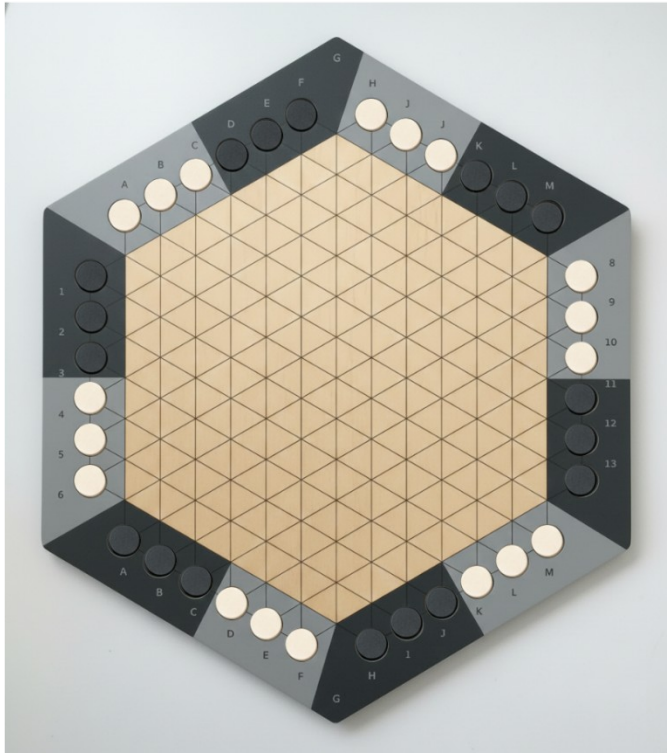
Elementi igre Atol (*Atoll*)

- Pobednik je igrač koji prvi poveže kamenčićima dva ili više svojih ostrva tako da najmanji broj ostrva povezanih sa obodnim putem bude ne manji od d = m / 2 + 1 ostrva (m = ukupan broj ostrva).
- Obodni put je niz polja duž ivica područja za igru (slobodna polja na početku igre) koji obuhvata sva ostrva koja su povezana odigranim putem.
- Najmanji broj ostrva povezanih sa obodnim putem je minimalan broj ostrva bilo koje boje u kontaktu sa obodnim putem u oba smera (u smeru kazaljke na satu ili suprotnom smeru).

Potezi u igri Atol (*Atoll*)

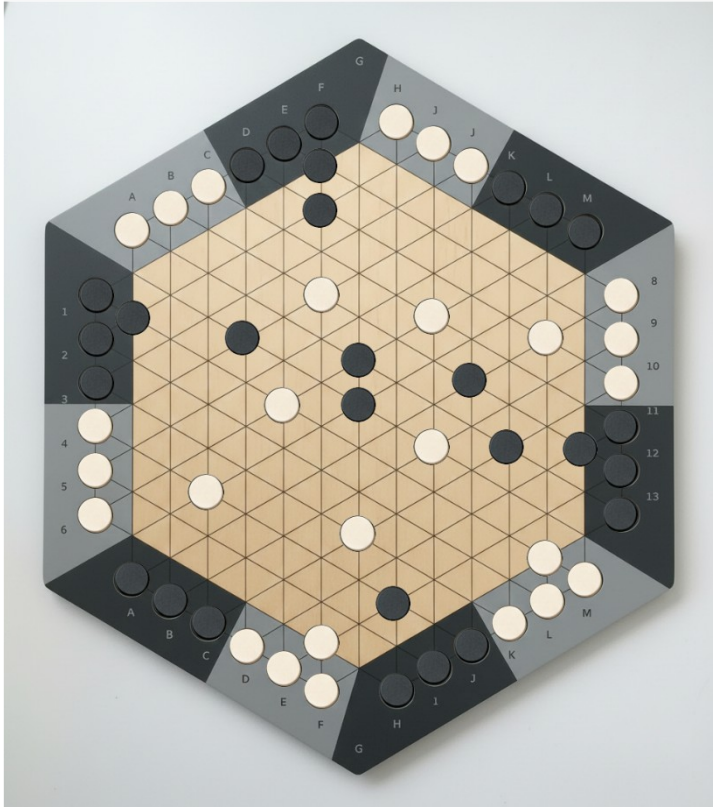
- Svaki potez se sastoji od postavljanja jednog kamenčića na tablu, na bilo koje slobodno polje.
- Igrači naizmenično odigravaju poteze.
- U svakom potezu igrač je dužan da postavi **jedan** svoj kamenčić na jedno od preostalih, nepopunjenih polja na tabli. Igrači ne smeju preskakati svoje poteze.
- Potezi se odigravaju do kraja igre.

Atol (*Atoll*) – Početak igre

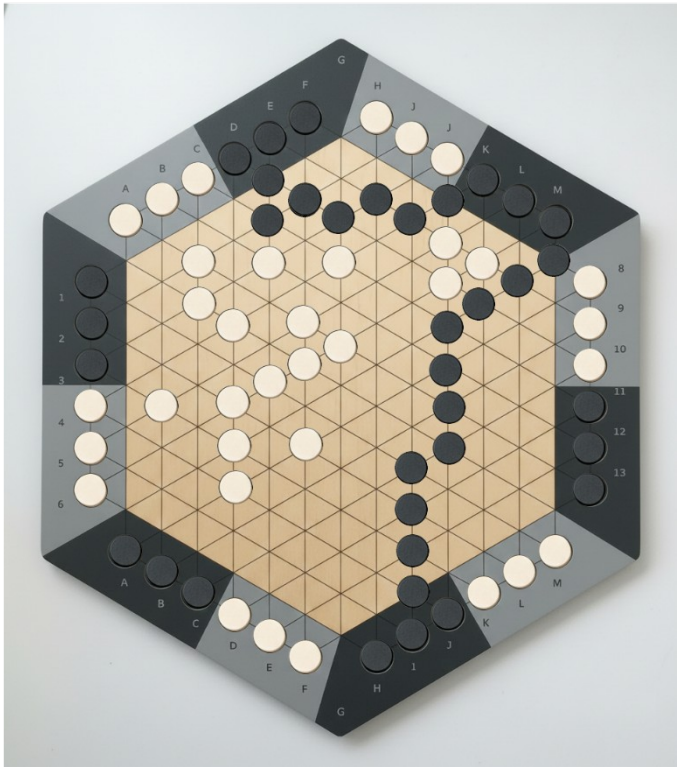


- Tabla sa dužinom stranice n kamenčića
  - Primer: n=7
- Oko table se nalaze ostrva
  - Na svakoj strani šestougla po 2 ostrva
    - 1 ostrvo crnog igrača
    - 1 ostrvo belog igrača
- Broj ostrva je m=12
- Ostrvo je niz kamenčića iste boje
  - Dužina ostrva je o=(n-1)/2 kamenčića
  - Primer: o=3

Atol (*Atoll*) – Primer stanja igre



Atol (*Atoll*) – Primer kraja igre



- Postoji put između tri ostrva
  - Pobednik je crni igrač.
  - Broj ostrva na najkraćem obodnom putu je 7
  - m=12, d=7

Atol (*Atoll*) – Korisni linkovi

- Opis igre sa pravilima:
  - https://www.marksteeregames.com/Atoll_rules.pdf

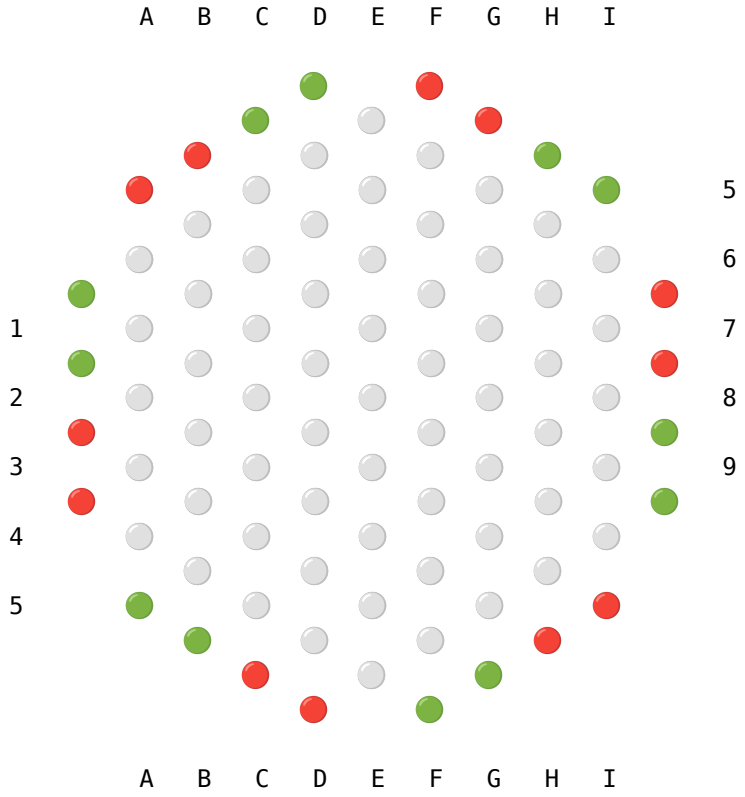
Zadatak I – Formulacija problema i interfejs

- Definisati način za predstavljanje stanja problema (igre)
  - Predstavljanje praznih polja i postavljenih kamenčića sa informacijom o igraču koji ga je postavio
- Napisati funkciju za postavljanje početnog stanja
  - Definiše se na osnovu zadate veličine table
- Napisati funkcije koje proveravaju ispravnost unosa poteza
- Napisati funkcije koje prikazuju stanje problema (igre)
- **NIJE POTREBNO realizovati funkcije koje odigravaju potez i proveravaju da li je kraj igre (faza II)**
- **NIJE POTREBNO realizovati funkcije koje obezbeđuju odigravanje partije (faza II)**

Zadatak I – Formulacija problema i interfejs

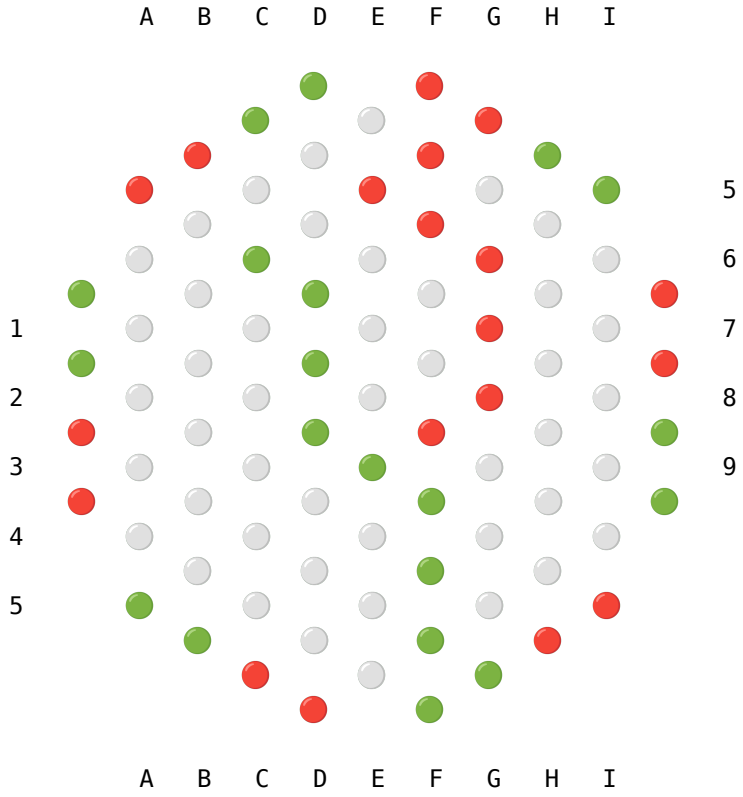
- Omogućiti izbor da li će igrati čovek protiv čoveka ili čovek protiv računara
- Omogućiti izbor ko će igrati prvi (čovek ili računar)
- Omogućiti izbor koji igrač igra prvi (X ili O)
- Implementirati funkcije koje obezbeđuju unos početnih parametara igre
  - Unos dužine stranice table (n) i provera ispravnosti unosa
- Implementirati funkcije koje obezbeđuju pravljenje inicijalnog stanja problema (igre)
  - Pravljenje stanja igre na osnovu zadatih dužine stranice table (n)
- Implementirati funkcije koje obezbeđuju prikaz proizvoljnog stanja problema (igre)
  - Prikaz trenutne (proizvoljne) situacije na tabli sa praznim poljima i kamenčićima oba igrača
- Realizovati funkcije za unos poteza
  - Potez se sastoji od pozicije polja
- Realizovati funkcije koje proveravaju da li je unos poteza tačan
  - Proveriti da li je zadata pozicija u okviru table i da se na toj pozicije ne nalazi kamenčić

Zadatak I – Interfejs (početno stanje)



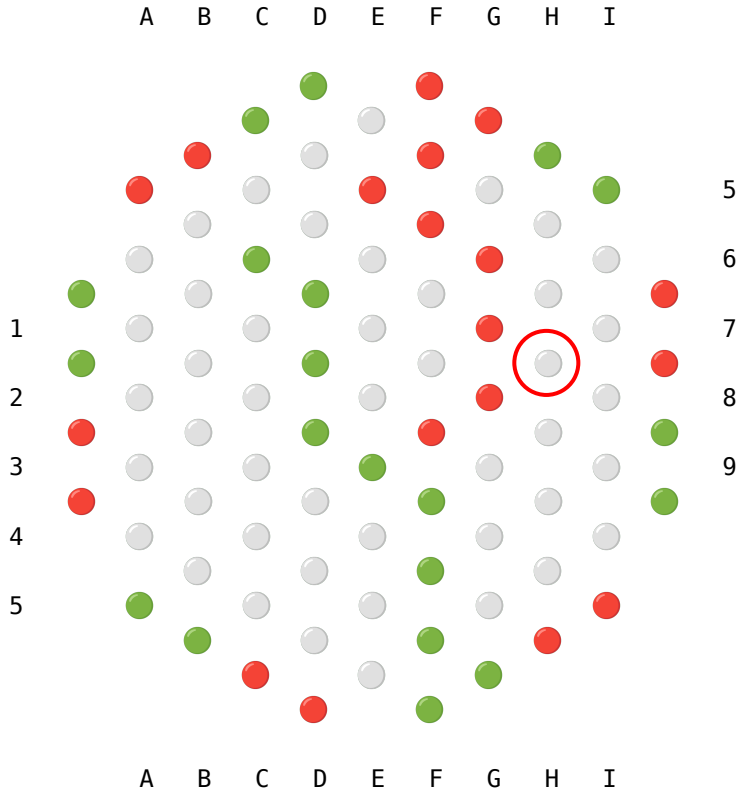

```
Dužina stranice: n = 5


Red   = '\U0001F534'  #
Green = '\U0001F7E2'  #
White = '\u26AA'      #

Crveni igrač = Beli igrač (O)
Zeleni igrač = Crni igrač (X)
```

Zadatak I – Interfejs (trenutno stanje)



Stanje u toku igre

Zadatak I – Interfejs (unos poteza)



- Potez O (crvenog) igrača:
- (’H’, 6)

Zadatak II – Operator promene stanja

- Napisati funkcije za proveru kraja igre
  - Provera postojanja puta između dva nesusedna ostrva
- 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 (za proizvoljno stanje igre)
  - Određivanje svih mogućih poteza igrača na osnovu stanja problema (igre)

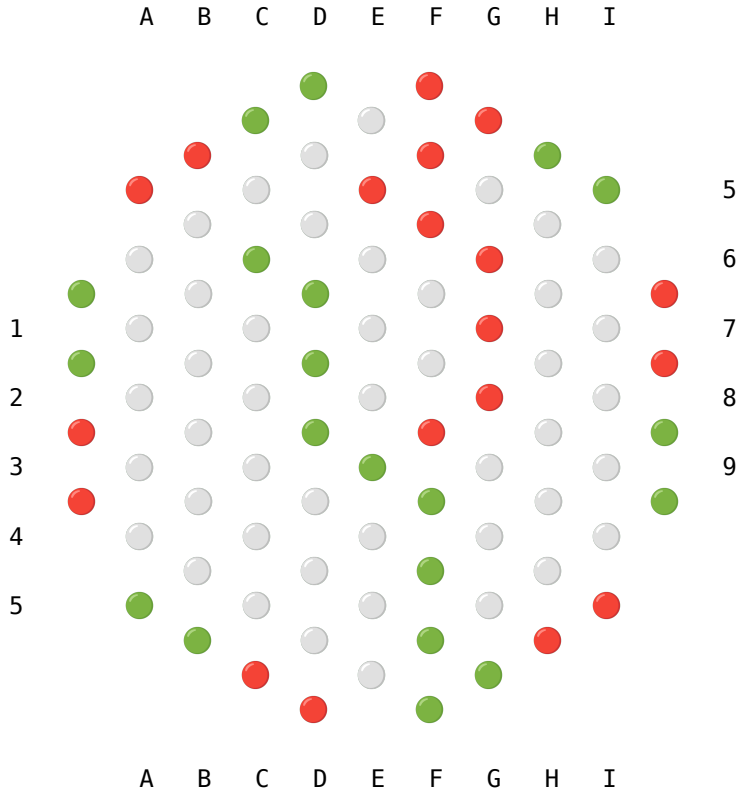
Zadatak II – Operator promene stanja

- Realizovati funkcije koje na osnovu zadatog poteza menjaju stanje problema (igre)
- Realizovati funkcije za proveru kraja igre
  - Proveriti da li postoji put između dva nesusedna ostrva
  - Proveriti da li je najkraći put između dva nesusedna svoja ostrva dužine najmanje 7 kamenčića
- 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

Zadatak II – Operator promene stanja

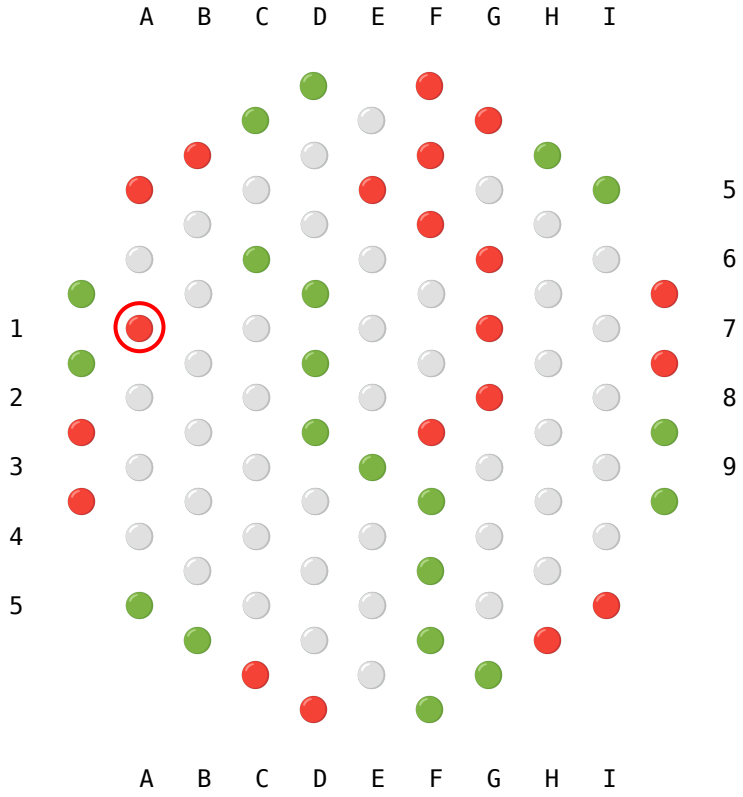
- Realizovati funkcije koje implementiraju operator promene stanja problema (igre)
  - Realizovati funkcije koje na osnovu zadatog igrača na potezu, zadatog poteza i zadatog stanja igre formiraju novo stanje igre
  - Realizovati funkcije koje na osnovu zadatog igrača na potezu i zadatog stanje igre (table) 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 – Interfejs (promena stanja)



Stanje pre
poteza O (crvenog) igrača

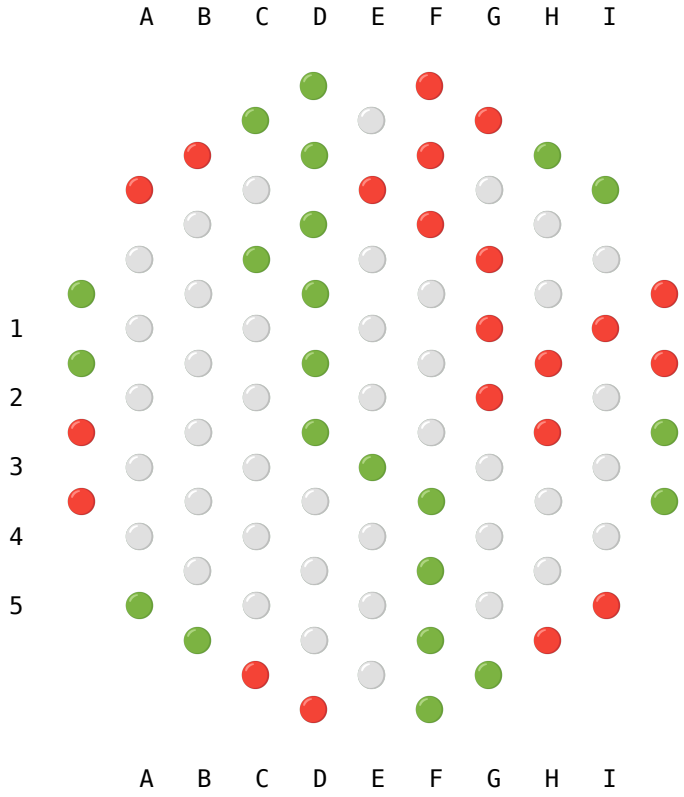
Zadatak II – Interfejs (promena stanja)



Novo stanje nakon poteza O (crvenog) igrača

('A', 2)

Zadatak II – Interfejs (krajnje stanje)

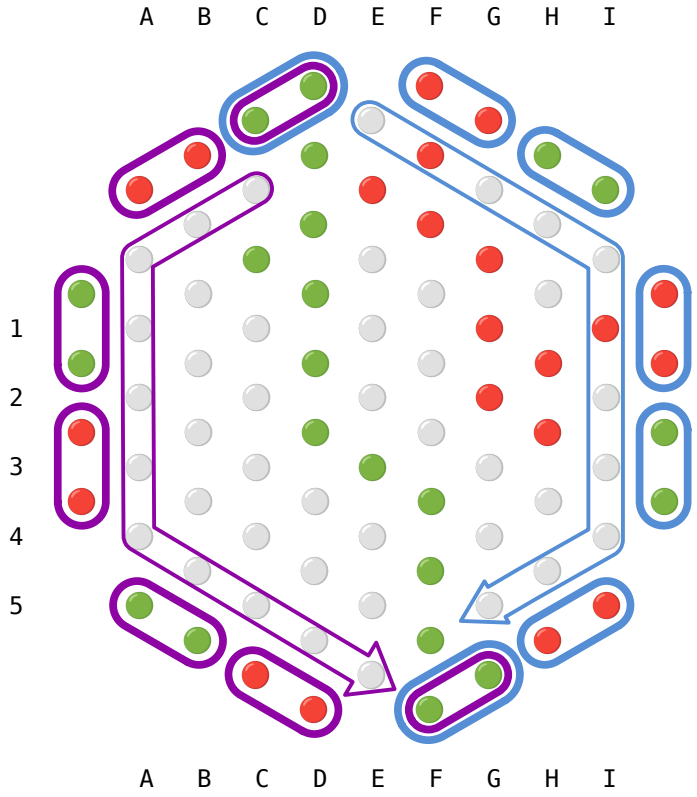


- Pobeda X igrača (zelenog igrača)

X ima dva potencijalna puta:

- U smeru kazaljke na satu: **7 ostrva**
- U smeru suprotnom od kazaljke na satu: **7 ostrva**

Zadatak II – Interfejs (krajnje stanje)

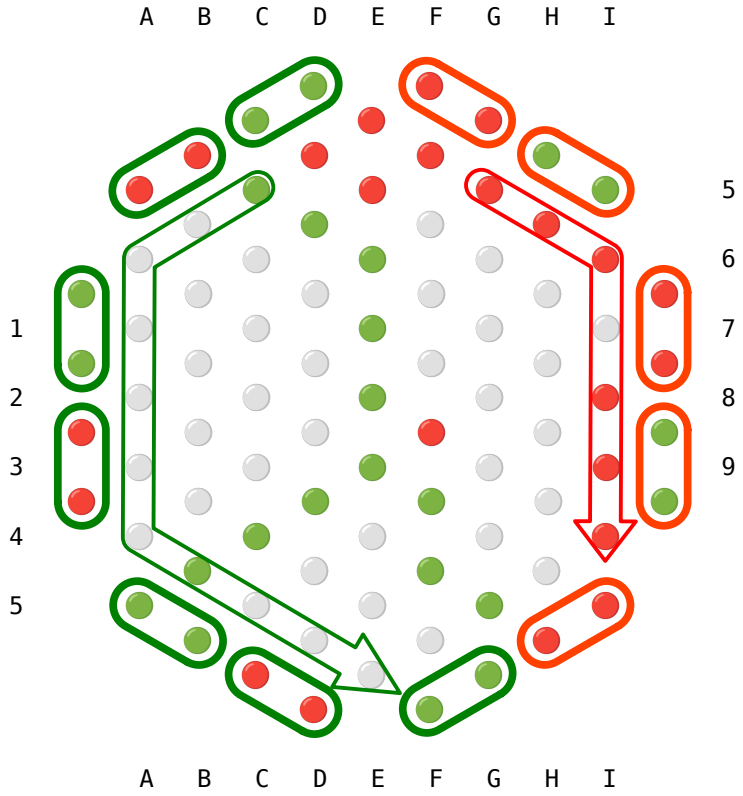


- Pobeda X igrača (zelenog igrača)

X ima dva potencijalna puta:

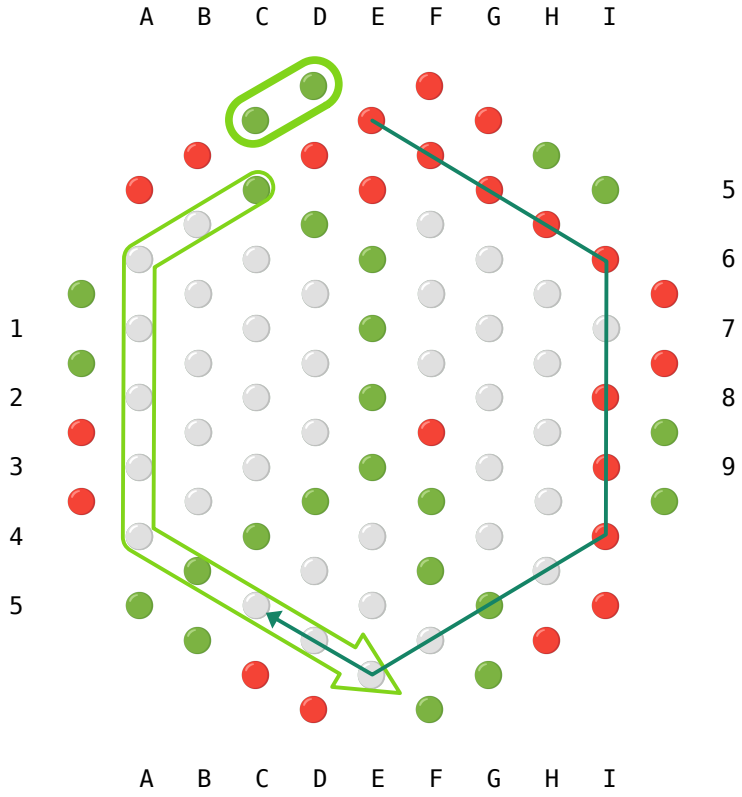
- U smeru kazaljke na satu: **7 ostrva**
- U smeru suprotnom od kazaljke na satu: **7 ostrva**

Zadatak II – Interfejs (krajnje stanje)



- X igrač (zeleni): **pobeda**
- *Najmanji broj povezanih ostrva:* ***7***
- O igrač (crveni):
- *Najmanji broj povezanih ostrva:* ***5 (< 7)***

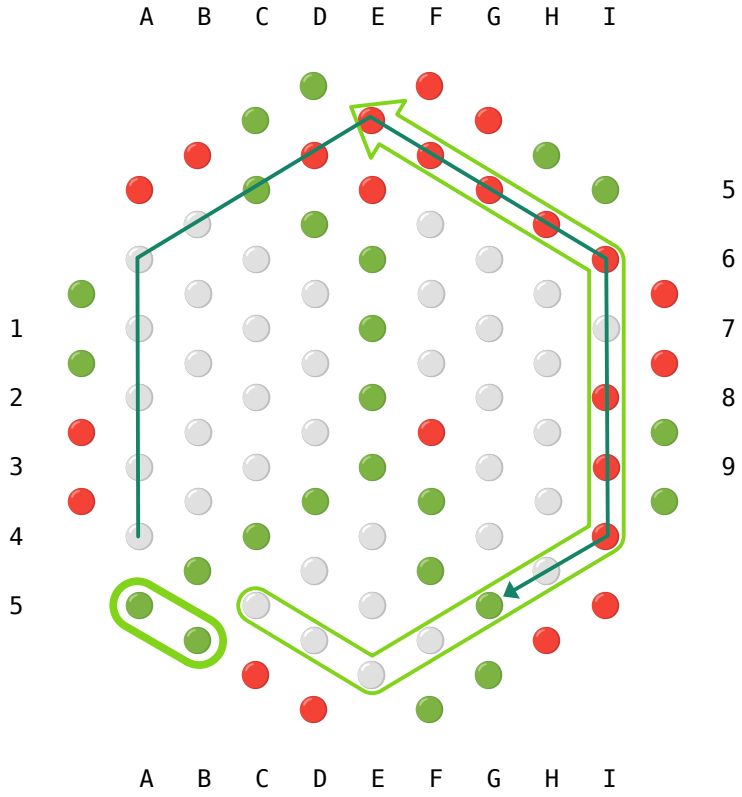
Zadatak II – Interfejs (krajnje stanje)



- Primer 1:
- Putevi od 1. ostrva:

- U smeru kazaljke na satu: **9 ostrva**
- Suprotno od kazaljke na satu: **7 ostrva**

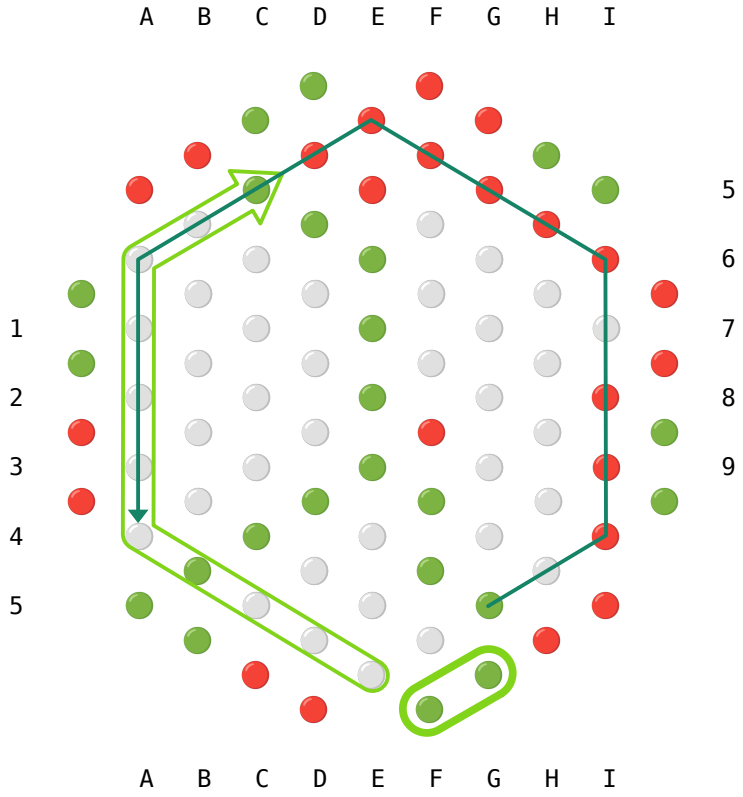
Zadatak II – Interfejs (krajnje stanje)



- Primer 1:
- Putevi od 2. ostrva:

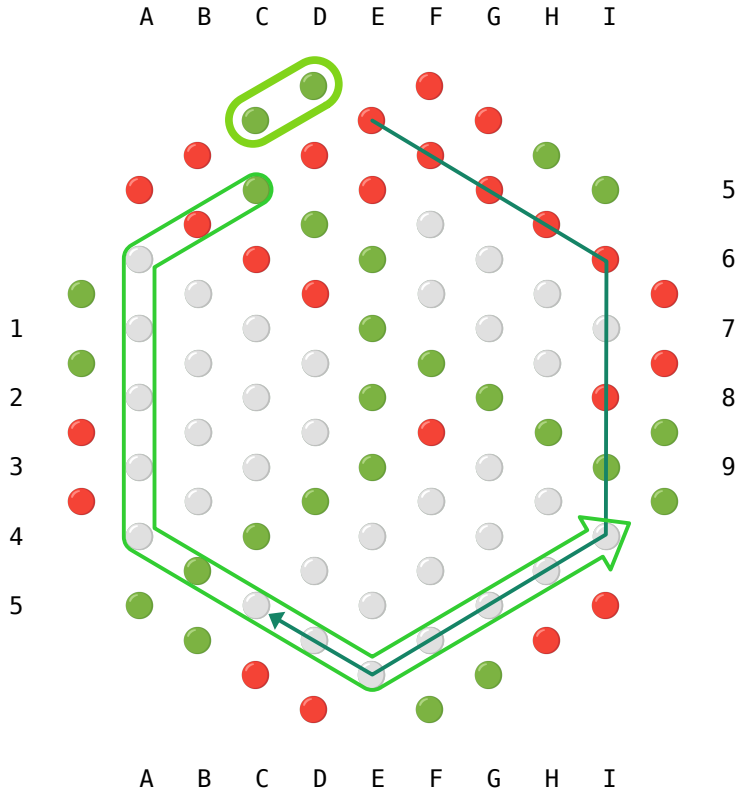
- U smeru kazaljke na satu: **11 ostrva**
- Suprotno od kazaljke na satu: **9 ostrva**

Zadatak II – Interfejs (krajnje stanje)



- Primer 1:
- Putevi od 3. ostrva:
- U smeru kazaljke na satu: **11 ostrva**
- Suprotno od kazaljke na satu: **7 ostrva**
- Broj ostrva povezanih sa najkraćim obodnim putem:
- Min(9, 7, 11, 9, 11, 7) = **7**
- **Pobeda zelenog igrača**

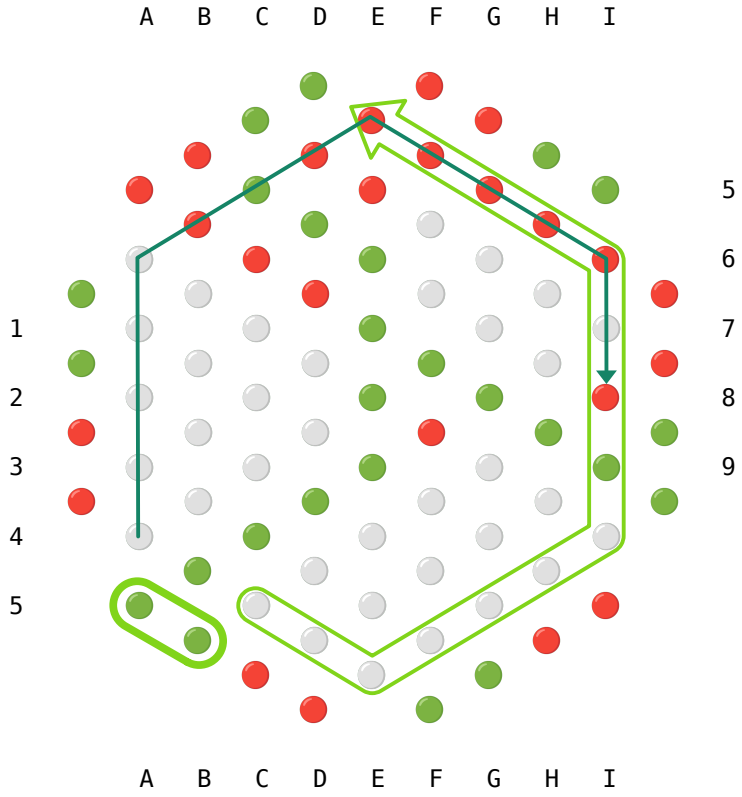
Zadatak II – Interfejs (krajnje stanje)



- Primer 2:
- Putevi od 1. ostrva:

- U smeru kazaljke na satu: **9 ostrva**
- Suprotno od kazaljke na satu: **9 ostrva**

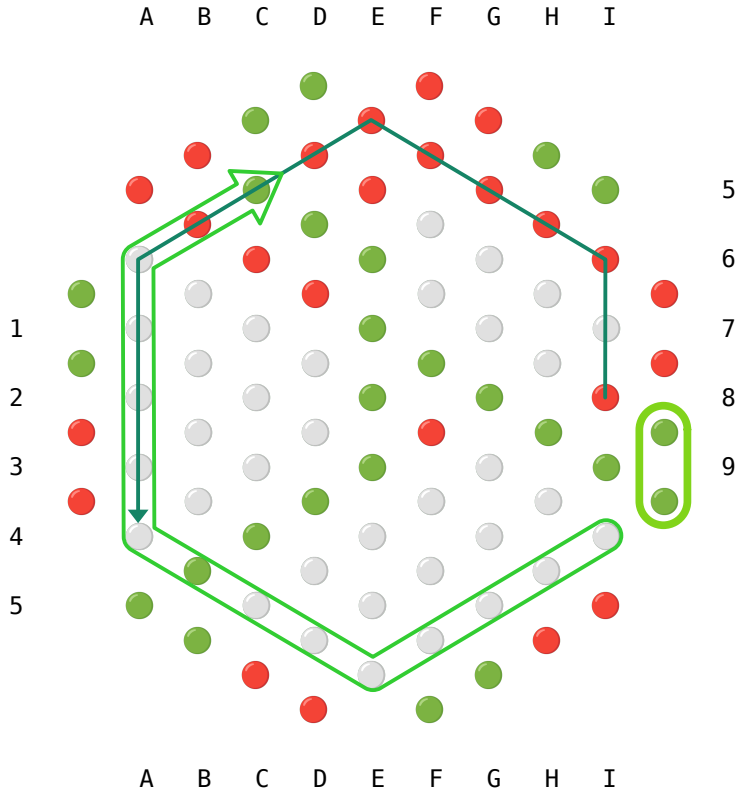
Zadatak II – Interfejs (krajnje stanje)



- Primer 2:
- Putevi od 2. ostrva:

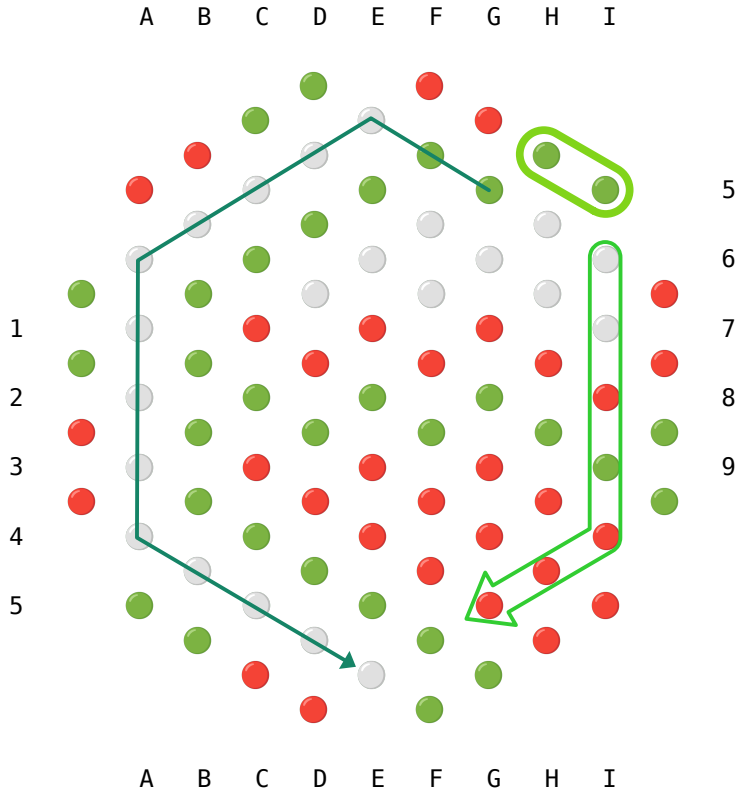
- U smeru kazaljke na satu: **9 ostrva**
- Suprotno od kazaljke na satu: **9 ostrva**

Zadatak II – Interfejs (krajnje stanje)



- Primer 2:
- Putevi od 3. ostrva:
- U smeru kazaljke na satu: **9 ostrva**
- Suprotno od kazaljke na satu: **9 ostrva**
- Broj ostrva povezanih sa najkraćim obodnim putem:
- Min(9, 9, 9, 9, 9, 9) = **9**
- **Pobeda zelenog igrača**

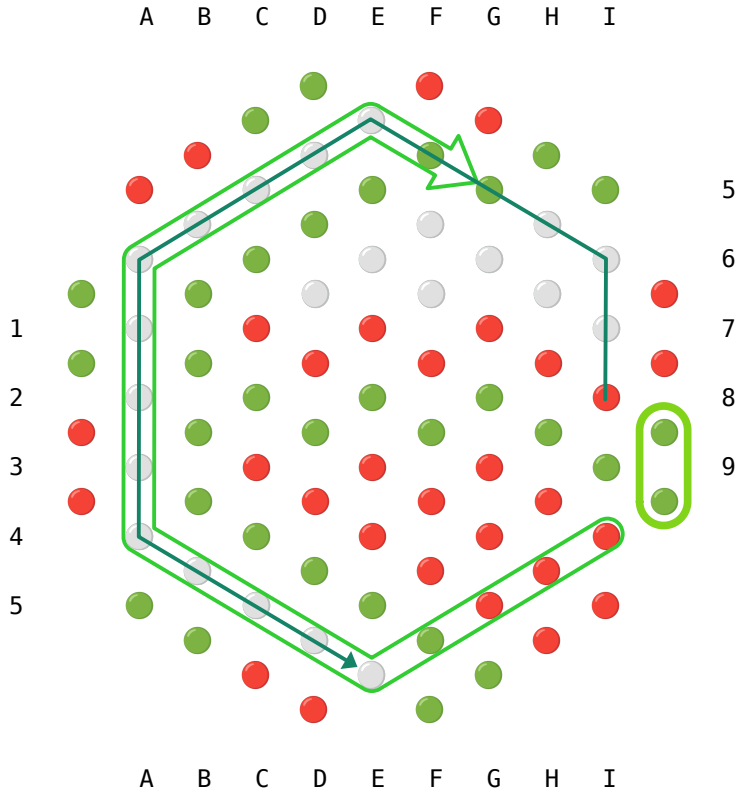
Zadatak II – Interfejs (krajnje stanje)



- Primer 3:
- Putevi od 1. ostrva:

- U smeru kazaljke na satu: **5 ostrva**
- Suprotno od kazaljke na satu: **9 ostrva**

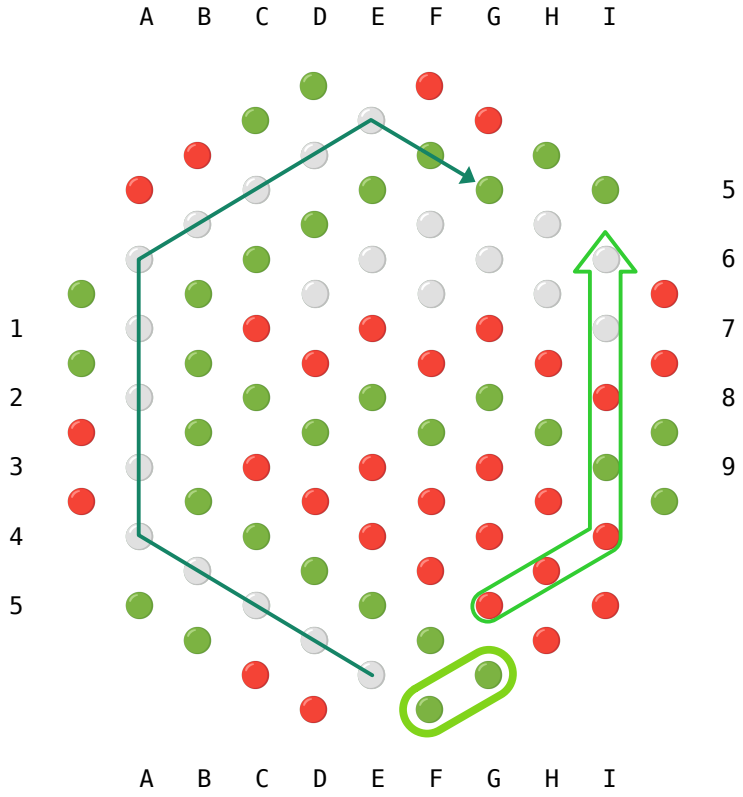
Zadatak II – Interfejs (krajnje stanje)



- Primer 3:
- Putevi od 2. ostrva:

- U smeru kazaljke na satu: **11 ostrva**
- Suprotno od kazaljke na satu: **11 ostrva**

Zadatak II – Interfejs (krajnje stanje)



- Primer 3:
- Putevi od 3. ostrva:
- U smeru kazaljke na satu: **9 ostrva**
- Suprotno od kazaljke na satu: **5 ostrva**
- Broj ostrva povezanih sa najkraćim obodnim putem:
- Min(5, 9, 11, 11, 9, 5) = **5**
- **Još uvek nema pobednika**

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
- Omogućiti izbor odigravanja partije između dva čoveka ili 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.