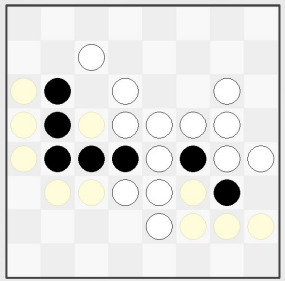
BOT ZA IGRU OTHELLO

Strahinja Sekulić SV34-2020 ▪ Nada Zarić SV63-2020

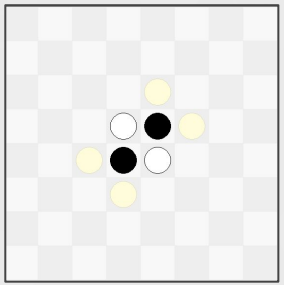
UVOD

Othello je strateška igra za dva igrača. Igra se na tabli dimenzija 8x8, sa dve vrste figura (crne i bele). Igrač koji na kraju igre ima više postavljenih figura je pobednik. Cilj projekta je napraviti bota koji će igrati optimalne poteze protiv čoveka ili drugog bota.



Slika 1 - Tabla za igru Othello

PRAVILA IGRE



Slika 2 - Početno stanje table

Na početku igre, postavljene su 4 figure (dve bele i dve crne) u sredinu table, raspoređenih u kvadratnom obliku sa naizmeničnim bojama. Igrači nasumično postavljaju svoje figure. Kada igrač postavi figuru na ploču, protivničke figure koje se nalaze između te nove figure i već postojeće figure trenutnog igrača menjaju boju i postaju figure trenutnog igrača. Moguće je pobediti na dva načina: ili će ostati samo jedna vrsta figura na tabli ili će pobediti igrač koji ima više postavljenih figura.

IMPLEMENTACIJA

Pre nego što krenemo da tražimo optimalan potez, potrebno je pronaći sve moguće poteze za trenutnog igrača. Moguće poteze označavamo sa kružićem svetlo žute boje (kao na slici 1 i slici 2). Pozicije ovih kružića se određuju tako što se iterira kroz sve ćelije u tabli. Za svaku ćeliju, od te pozicije iteriramo kroz sve moguće pravce (gore, dole, levo, desno i dijagonale). Ako na tom pravcu postoji sekvenca protivnikovih figura nakon koje se nalazi polje sa figurom igrača koji je na potezu, znači da je posmatrana ćelija legalan potez.

```python
class Agent(ABC):
    def __init__(self):
        self.turn = ''
        self.board = None
    def init(self, board: Board, turn: str):
        self.turn = turn
        self.board = board
    @abstractmethod
    def make_move(self):
        pass
```

Slika 3 - Snippet Agent abstraktne klase

Za implementaciju botova, koristili smo abstraktnu klasu Agent. Svaki bot implementira ovu abstraktnu klasu, a koristili smo sledeće algoritme:

- Random - nema argumenata
- MiniMax sa alpha-betha pruning-om - prima dubinu pretrage stabla
- Expectimax - prima dubinu pretrage stabla
- Monte Carlo Tree Search - prima broj simulacija koje obavlja u jednoj iteraciji i efekat pretraživanja (koliko dopuštamo da algoritam pretražuje nove čvorove)
- ProbCut - prima dubinu pretrage i probcut (u kom opsegu algoritam uzima u obzir posmatrani korak)

REZULTATI

Sve algoritme smo testirali tako što smo istoj osobi dali da igra igru protiv različitih agenata i sa različitim parametrima. Sve partije su odigrane pod istim uslovima - na istom hardverskom uređaju i sa istom veličinom table za igru (8x8 polja), gde je igru uvek započinjala test osoba. U tabeli su prikazani neki od rezultata (detaljinije u dokumentaciji)

Algoritam	Karakteristike	Tester	Bot	Nerešeno
Random	-	4	1	0
MiniMax	depth = 3	0	5	0
Expectimax	depth = 3	4	1	0
Monte Carlo TS	simulations = 500 exploration=1.5	3	1	2
ProbCut	depth = 3 probcut = 0.9	1	4	0

ZAKLjUČAK

U zavisnosti od implementiranog algoritma, kao i postavljenih parametara algoritma, imamo botove koji igraju na različitom nivou optimalnosti. Dalja unapređenja, kao što je bolja heuristika, mogu dodatno poboljšati preformanse botova u budućnosti.