Mesterséges Intelligencia
GKNB_INTM002

ReversiAI

Hercsel Péter
CSLOP1

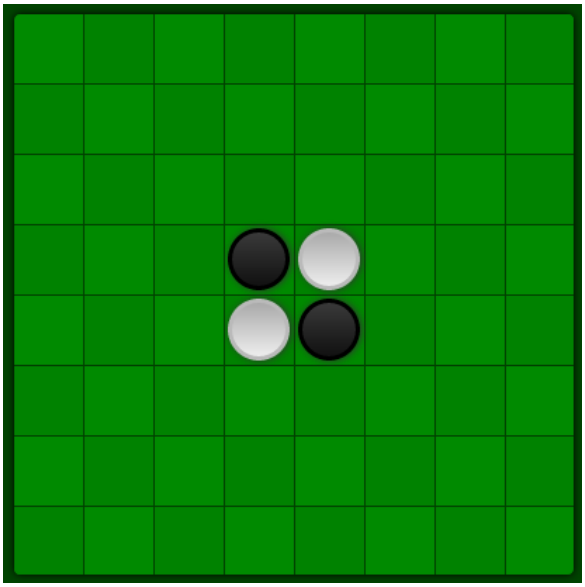
A Reversi (Fonákolós, Otthello) bemutatása

A Reversi (vagy másnéven Othello/Fonákolós) egy táblás stratégiai játék, amelyet két játékos játszik egy 8x8-as négyzetrácsos táblán. A játék célja az ellenfél korongjainak elfoglalása és megfordítása. A játékosok felváltva helyezik el a saját színű korongjaikat a táblán, az ellenfél korongjai mellé, oly módon, hogy az ellenfél korongok túloldalán egy saját színű korong van. Ezután a frissen lerakott korong és az összes saját korong között lévő ellenfél korongot megfordítják, hogy az adott játékos színére változzanak.

A Reversi a játékosok stratégiai gondolkodását és tervezését teszteli, mivel egy-egy lépés hosszú távú következményekkel járhat. A játékosoknak figyelembe kell venniük az ellenfél lehetséges lépéseit és előre kell tervezniük a saját lépéseiket annak érdekében, hogy maximalizálják a saját korongjaik számát a játék végére.

A mesterséges intelligencia alkalmazása a Reversi játékban lehetővé teszi számunkra, hogy fejlett algoritmusokkal és gépi tanulási módszerekkel megtervezzünk olyan programokat, amelyek versenyezhetnek vagy akár meg is előzhetik az emberi játékosokat. Ezek a programok képesek lehetnek az optimális stratégia megtalálására, és kihasználhatják a számítási erőt és az algoritmusok sebességét a játék során.

A következő fejezetekben részletesen bemutatom a Reversi játék szabályait, az alapvető algoritmusokat és módszereket, amelyeket a mesterséges intelligencia implementálásához használhatunk, valamint bemutatom az általam választott megközelítést és implementációt.



1. ábra: Reversi kezdő pozíció

Szabályok

A Reversi (vagy Othello) játék szabályai egyszerűek, de a stratégiai mélysége jelentős kihívást jelenthet. Íme a játék lépéseinek és szabályainak áttekintése:

Játéktábla:

- A Reversi játékot egy 8x8-as négyzetrácsos táblán játsszák.
- A játék elején négy korongot helyeznek el a tábla közepén: két fekete és két fehér korongot, úgy, hogy a fehér korongokat a felső bal és alsó jobb sarokban, a fekete korongokat pedig a felső jobb és alsó bal sarokban helyezik el.

Játékosok:

- Két játékos vesz részt a játékban, az egyik fekete, a másik fehér.
- A fekete játékos kezd.

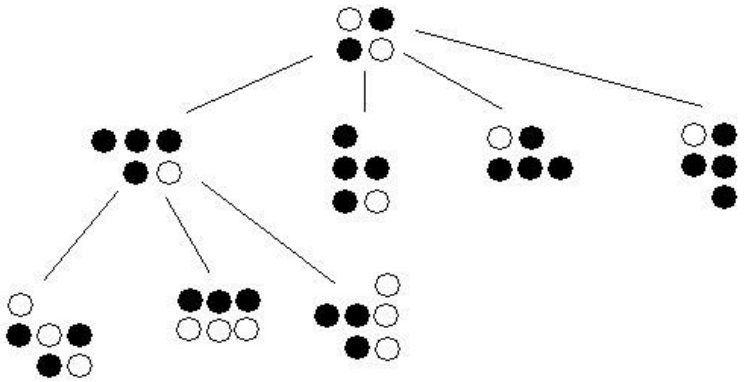
Lépések:

- A játékosok felváltva helyeznek el egy-egy korongot a táblán saját színükben.
- A korongot olyan irányba kell helyezni, hogy az egy vagy több ellenfél korongot közvetlenül kövesse, és az utolsó korong a sorban a saját színű korong legyen.

Szabályok:

- Egy lépés csak akkor érvényes, ha legalább egy ellenfél korongot átforgat a saját korongja.
- Ha egy játékos nem tud érvényes lépést tenni, akkor a másik játékos következik. Ha egyik játékos sem tud érvényes lépést tenni, a játék véget ér.
- A játék véget ér, amikor a tábla megtelik, vagy amikor mindkét játékos nem tud már érvényes lépést tenni.
- A játék végeztével a győztes az a játékos, akinek több korongja van a táblán.

A Reversi játék egyre bonyolultabbá válik a játék során, ahogy a korongok egyre több területet elfoglalnak és egyre kevesebb hely marad a lépések számára. A játék során stratégiailag fontos, hogy figyelemmel kísérjük az ellenfél lépéseit és előre tervezzük a saját lépéseinket annak érdekében, hogy maximalizáljuk a saját korongjaink számát a játék végére.



2. ábra: az első pár lépés

Módszerek

Aktuális állás kiértékelés:

Minden megoldásnak az alapja egy kiértékelési rendszer. Ebben a játékban lényegében egy ilyen rendszer van, ami az alapján működik, hogy a tábla hányad részét uraljuk. Ezen felül lehet súlyozni a bizonyos mezőket. Az én általam használ kiértékelés az alábbi kódrészlettel kerül kiértékelésre:

```
def updateScore(self):
    self.black = self.white = 0
    for row in self.board:
       for cell in row:
            if cell == "+":
                self.white += 1
            elif cell == "-":
                self.black += 1
    return self.white / (self.black + self.white)
```

Ezen felül még súlyoztam a sarkok pontszámát kettővel amikor az algoritmus eléri a legmélyebb pontját, mivel a sarkokba helyezett korongok a játék végéig ott maradnak.

```
if depth == 0:
    corners = "".join([self.game.board[i][j] for i in (0,7) for j in (0,7)])
    whiteInCorners = corners.count("+")
    blackInCorners = corners.count("-")
    return self.game.updateScore() + (2*whiteInCorners) - (2*blackInCorners)
```

Algoritmusok:

Megerősítéses Tanulás:

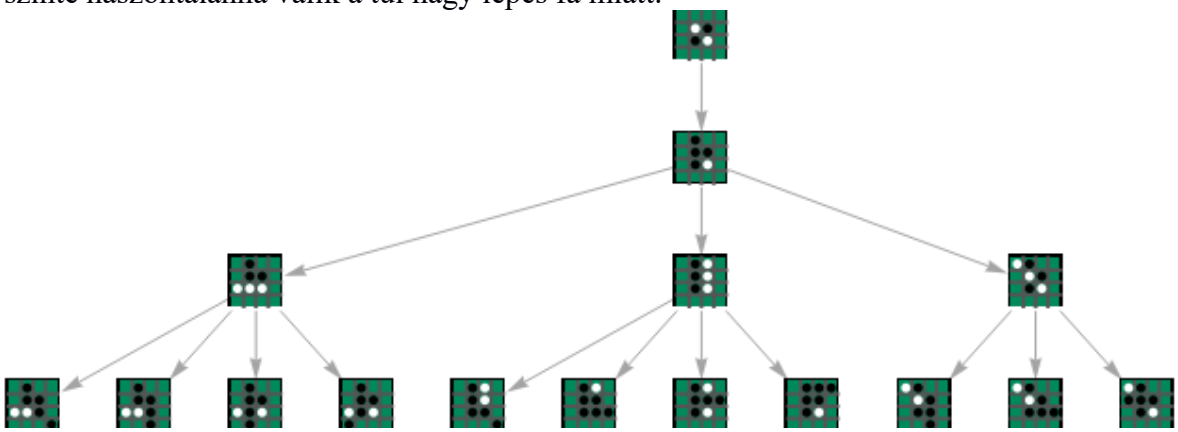
A megerősítéses tanulás olyan technika, amely lehetővé teszi a gép számára, hogy tapasztalatokból tanuljon és fejlessze a stratégiáját a játék során. A gép játszatja magát az ellenfelek ellen, majd a visszajelzéseket felhasználva frissíti a stratégiáját annak érdekében, hogy maximalizálja a nyerési esélyeit.

Ilyen visszajelzés lehet, hogyha a végső pozíciókból statisztikát vezetve megnéznénk, mely mezők hány százalékban voltak elfoglalva a nyertes által, így súlyozhatjuk a mezők fontosságát a játék során.

Én a végső algoritmusban ezt megtettem mikor díjaztam azokat a lépéseket, amikkel a sarkokat elfoglaljuk, de később megfigyeltem, hogy büntetést lehet kiszabni a sarkok mellett lévő mezőkre, mivel az esetek többségében ezek a lépések azzal járnak, hogy az ellenfél szerzi meg a sarkokat.

Brute force megoldás

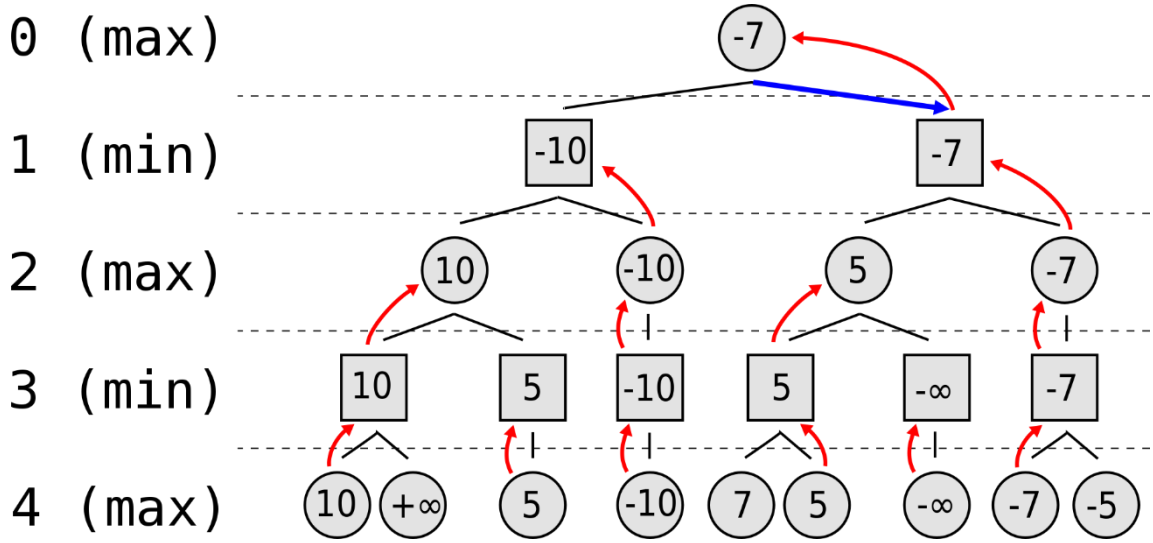
A Reversi brute force megközelítése esetén az összes lehetséges lépést végig próbáljuk, majd azokat értékeljük, hogy melyik lenne a legjobb döntés az aktuális játékállapotban. Mivel a Reversi játék egy viszonylag kis táblamérettel rendelkezik (például 4x4-es táblával is játszható) a brute force módszer még életképes lehet, viszont nagyobb táblaméretnél már szinte haszontalanná válik a túl nagy lépés fa miatt.



3. ábra: Brute-Force a 3. lépésig

Minimax algoritmus

A Minimax algoritmus egy olyan fa-kiterjesztéses kereső algoritmus, amely segít a játékfa kiértékelésében és a legjobb lépés megtalálásában. A módszer lényege, hogy a játékfa minden lehetséges állapotát végig vizsgálja egy adott mélységig, majd a játékosok felváltva hozzák meg a legjobb döntést. Ez a módszer hatékonyan alkalmazható Reversi játékban.

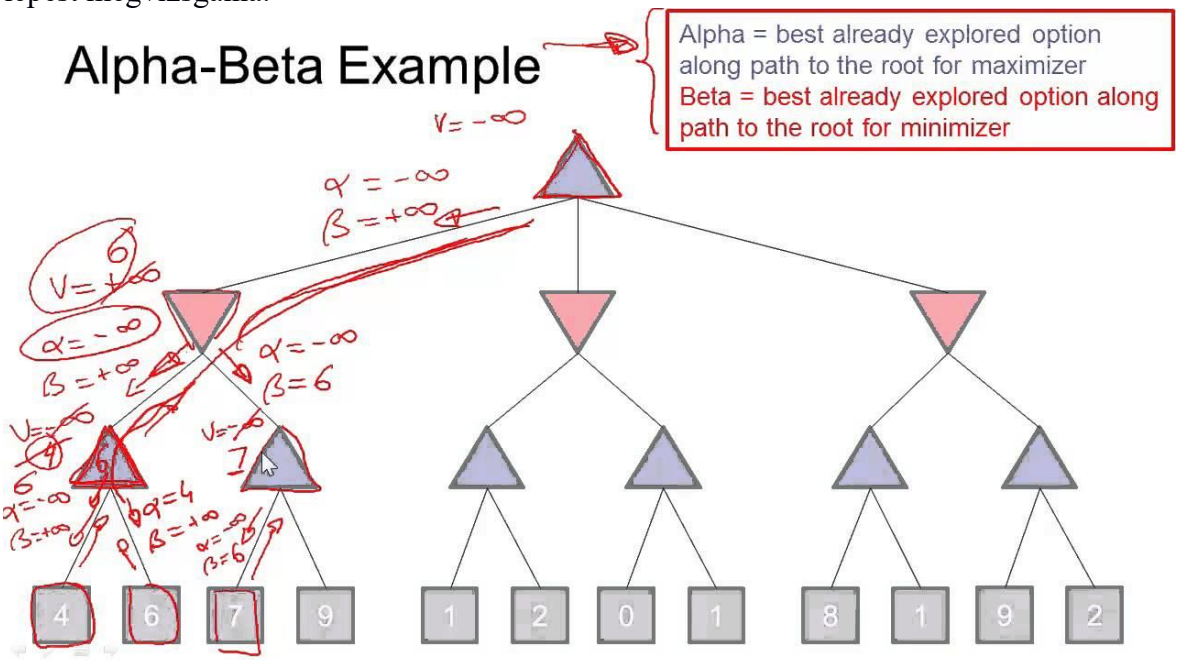


4. ábra: Minimax gráf

Alfa-Béta metszet

Az "alfa béta metszet" algoritmus egy olyan keresési algoritmus, amelyet a játékokban, például a sakkban vagy a reversiben, az optimális lépés megtalálására használnak azáltal, hogy hatékonyan vágja el a felesleges ágakat a keresési fában, így csökkentve a keresési időt. Ezáltal lehetővé teszi a számítógépek számára, hogy hatékonyan megtalálják az optimális lépéseket nagyobb játékfákon is.

Az algoritmus nevében az "alfa" és a "béta" értékek a keresési fák egyes csomópontjainak alsó és felső határait jelzik, amelyek segítenek abban, hogy csak azokat a lehetőségeket vizsgáljuk, amelyek valószínűleg javítják a jelenlegi legjobb megoldást. Ezáltal az algoritmus gyorsabban megtalálhatja az optimális lépést anélkül, hogy az összes lehetséges lépést megvizsgálná.

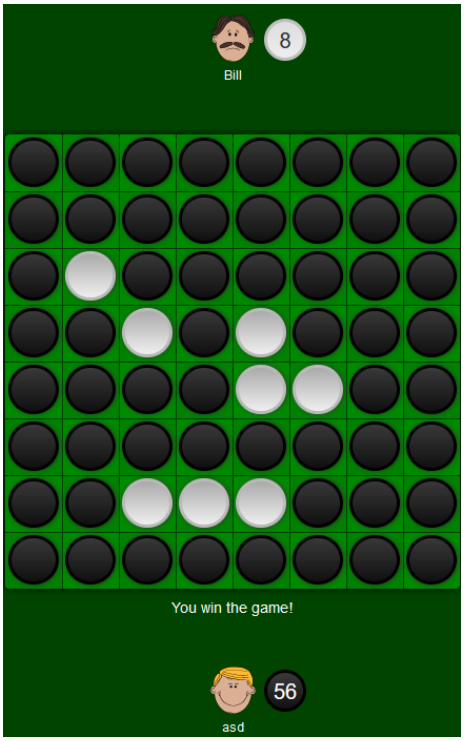


5. ábra: Alfa Béta metszet

A végső megoldásomban én is az Alfa Béta metszetet választottam, mivel ezt tartottam legmegfelelőbbnek a táblaméretet és a lehetséges lépések számát figyelembe véve. A megoldásom egy google colab oldalról származik, ahol amőba algoritmusokat mérkőztetnek meg egymással (Monte Carlo Tree Search vs Alpha-Beta pruning).
Az oldalról származó kódot rászabtam a Reversi játékra azáltal. Például a kiértékelésnél nem azt vettem figyelembe, hogy egy bizonyos lépés után a fa további részében, hányszor tett nyerő lépést az intelligencia, hanem gyakorlatilag kiértékeltem a már említett módon a csomópontokat.

Tesztelés

A tesztelést manuális módon végeztem egy online Reversi oldal ellen, ahol az oldal ellen játszottam a lépéseket, amit az elkészített algoritmusom kiszámolt. Az oldalnak 2 nehézségi fokozata van és a megfelelő beállításokkal (főként az 5-os mélységig történő számolással) nagyon simán megverte.



6. ábra: A smart opponent elleni végső pozíció

Lehetséges fejlesztések, javítások

- A kódomban natúr ptyhon listákat és tuple-t használtam.Ezek helyett lehetett volna NumPy tömböket használni, amik lényegesen gyorsítanának a program futásán. Továbba néhány hiba még maradt a kódban, mint például, ha a felhasználó érvénytelen lépést tesz a színek megfordulnak.
- Néha az AI nem jó lépésekből választ, így invalidnak érzékeli a lépést.
- Az AI néha nem hajtja végre az Alfa-Béta metszetet így a futás idő lényegesen megnő.
- A már említett teszt weboldallal automatikusan lehetne teszteltetni az AI-t
- GPU használat vagy multithreading
- Megnyitási könyv implementálás

Github

A programot és ezt a dokumentumot az alábbi github repositoryba töltöttem fel:

https://github.com/dontrajik/reversiAI

Irodalomjegyzék

- Google colab - Monte Carlo Tree Search
- https://cardgames.io/reversi/
- Alfa Beta pruning

Tartalom