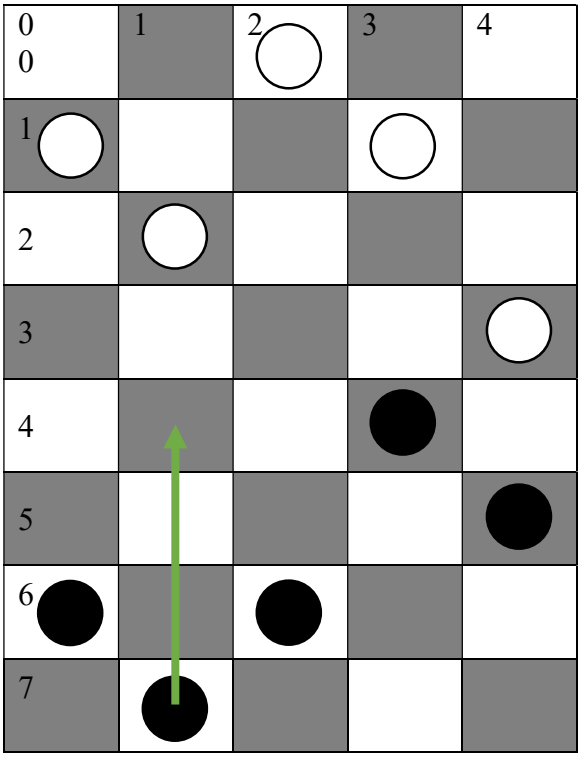
Projekt zespołowy 2

Chińskie szachy

Tym razem Wasze zadanie będzie trochę inne niż za pierwszym razem. Zacznijmy od wytłumaczenia zasad gry, z którą przyjdzie Wam się zmierzyć.

Zasady gry

Chińskie szachy (prawdziwa nazwa gry jest inna, lecz jej podanie znacznie ułatwiłoby rozwiązanie) to gra, która jest dość podobna do gry, która pojawiła się na naszych pierwszych zajęciach. Tym razem mamy jednak planszę $n \times m$ (dla ułatwienia przyjmijmy $3 \leq n, m \leq 20$), na której znajduje się $2n$ pionów (n w kolorze czarnym i n w kolorze białym). W danej kolumnie znajduje się dokładnie 1 pion czarny i 1 pion biały.



Ustawienie początkowe

Początkowo piony ustawione są symetrycznie na zerowym i ostatnim wierszy. Na wierszu zerowym są ustawione piony białe, a na ostatnim czarne.

Ruch

Tym razem w grze ruch będzie nieco inny niż możecie go pamiętać z początkowych zajęć. Podczas swojej tury można poruszyć jednego piona (swojego koloru) o dowolną dodatnią ilość pól, lecz **tylko do przodu** (patrząc od strony, do której aktualnie należy ruch). Na przykład na zaprezentowanym rysunku, zakładając, że ruch należy do czarnych, mogą oni wykonać ruch pionem na kolumnie 1 o 3 pola (ruch oznaczony zieloną strzałką). Możemy zatem jednoznacznie określić ruch przy pomocy 3 wartości: kolor, kolumna i liczba pól, o które ma się przesunąć pion (dla tego przykładu: czarny, 1, 3).

Warunki zwycięstwa

Gracz zwycięża, jeśli jego przeciwnik wykona niepoprawny ruch, toteż celem jest doprowadzić do tego, aby przeciwnik nie mógł wykonać żadnego poprawnego ruchu (tutaj już analogicznie do gry z 1. zajęć). W takiej sytuacji wszystkie piony znajdują się w odległości ruchu 1 – na sąsiadujących polach patrząc kolumnami).

Wymagania projektowe

Interfejs użytkownika

Program musi posiadać interfejs użytkownika:

1) Wybór opcji:
    a. Zmień rozmiar planszy
    b. Nowa gra 2-osobowa
    c. Nowa gra 1-osobowa
    d. Zakończ działanie programu.

Interfejs ma też komunikować o końcu gry, oraz o jej zwycięzcy.

Dobrze widziane dodatkowe funkcjonalności interfejsu, takie jak możliwość wprowadzenia imion graczy (i posługiwanie się nimi w komunikacji z użytkownikiem).

Dodatkowe punkty za stworzenie interfejsu graficznego. Sugerowane jest w takim wypadku użycie biblioteki pygame (https://www.pygame.org/docs/). Przykład prostego interfejsu do gry w klasyczne szachy (można się zainspirować): https://www.geeksforgeeks.org/create-a-chess-game-in-python/

Moduł przeprowadzania gry

1) W przypadku gry 2-osobowej gracze wykonują ruchy naprzemiennie. Plansza powinna być tu wyświetlana po każdym ruchu.
2) W przypadku gry 1-osobowej komputer musi wykonywać ruch jako jeden z graczy. Napisanie programu decyzyjnego jest jednym z następnych wymagań projektowych. Plansza powinna być wyświetlana tylko na początku i po ruchu komputera.
3) W obu przypadkach zaczynający gracz jest losowy.
4) Dodatkowo punktowana możliwość ustawienia gry AI vs AI

Program decyzyjny (AI)

1) Program ma wybierać ruch w zastanej pozycji. Istotne jest wymyślenie odpowiedniej strategii gry.
2) Dodatkowo punktowane jest implementacja wielu poziomów rozgrywki.
3) Możliwe są metody komputerowe znajdowania rozwiązania (analiza drzewa pozycji, przechowywanie danych w plikach, itd.). Macie gwarancję, że klasa programu decyzyjnego nie będzie tworzona po raz kolejny, więc wszystko co zapiszecie w tej klasie zostanie tak jak powinno być. Pamiętajcie aby, jeśli zmieniacie konstruktor klasy, wywołać najpierw konstruktor klasy nadrzędnej *super().__init__(...)*
4) Punkty za program AI są dodatkowo uzależnione od zestawienia z innymi programami w grupie (na podstawie 100 gier AI vs AI).

Ze względu na późniejszą automatyzację klasa decyzyjna **MUSI** dziedziczyć po klasie abstrakcyjnej *definitions.decider_base.DeciderBase* i implementować metodę *move()*.

Konstruktor klasy programu decyzyjnego **<u>musi przyjmować dokładnie 2 argumenty</u>**:

`board` – jest odniesieniem do planszy. Korzystamy z mutowalności, więc jej stan będzie się zmieniał poprzez działanie innych czynników. Klasa *Board* jest dokładnie opisana w pliku *definitons.board*.

`color` – jest kolorem, którym ma zagrać wasz program. *Pawn.Color.WHITE* lub *Pawn.Color.BLACK*.

Wymagania stałe *(ta lista wymagań jest z grubsza taka sama dla wszystkich projektów):*

1) **Plan realizacji projektu** – rozpisanie kroków do wykonania, ich terminów, podział obowiązków między członków zespołu, założenie buforu czasowego
2) **Dokumentacja techniczna** – na dokumentację składają się 2 istotne elementy:

   Przy okazji następnych projektów nauczymy się również programów do tworzenia automatycznej dokumentacji projektowej.

   a. <u>Komentarze w kodzie programu</u> – dobrze okomentowany, spójnie napisany kod sam w sobie dość dobrze spełnia postulat samoobjaśnialności. Warto pochylić się nad programami do automatyzacji dokumentacji (np. Sphinx) – można zdobyć za to dodatkowe punkty, a ułatwia to również stworzenie dokumentacji zewnętrznej.
   b. <u>Dokumentacja zewnętrzna</u> – dokumentacja w formie pliku (może być pdf), zawierająca opis każdej funkcji programu, przedstawienie działania programu, rozpisanie przykładowej gry. W związku z oceną naszego projektu obowiązkowe jest dodanie objaśnienia kto był odpowiedzialny za kolejne elementy projektu. Wypisane osoby mogą zostać poproszone o przedstawienie swojej części.
3) **Prezentacja** – projekt będzie przedstawiany przed całą grupą. Przygotowanie prezentacji (w dowolnej formie) jest również elementem projektu. Prezentacja ma trwać nie dłużej niż 15 min.
4) **Czytelność i segmentacja kodu** – kod powinien być podzielony na funkcje i pliki (dla przypomnienia – jeśli chcemy użyć funkcji napisanej w innym pliku to importujemy ją na zasadzie **`from`** `<nazwa_pliku>` **`import`** `<nazwa_funkcji1>, <nazwa_funkcji2>`).

Sposób oceny projektu

Za projekt otrzymuje się 3 oceny cząstkowe według poniższej punktacji:

1) ***(25 pkt)*** **Dokumentacja techniczna projektu i czytelność kodu:**
    a. *(5 pkt)* Plan realizacji projektu
    b. *(5 pkt)* Dobrze opisana dokumentacja zewnętrzna
    c. *(5 pkt)* Komentarze w kodzie
    d. *(10 pkt)* za czytelność kodu, a w tym odpowiednia segmentacja projektu
    w razie wątpliwości zachęcam do korzystania ze standardu PEP 8. Link: https://www.python.org/dev/peps/pep-0008/

2) ***(75 pkt)*** **Ocena całokształtu rozwiązania:**
    a. *(15 pkt)* interfejs użytkownika
    b. *(25 pkt)* za moduł gry 2-osobowej
    c. *(35 pkt)* za własny program AI, z czego:
        i. *(10 pkt)* za moduł gry jednoosobowy
        ii. *(25 pkt)* za program decyzyjny:
            1. 10 pkt za program, który nie popełnia nieprawidłowych ruchów
            2. do 20 pkt na podstawie skuteczności algorytmu względem innych algorytmów zaproponowanych przez nauczyciela.

Możliwe jest otrzymanie dodatkowych punktów za implementację dodatkowych funkcjonalności, a co za tym idzie w teorii możliwe jest zdobycie powyżej 100% za projekt.

Uwaga! Przypominam, że konieczne jest zabezpieczenie programu przed działaniem niepoprawnych danych.