Turniej algorytmów sztucznej inteligencji dla gry Shobu

Niniejsze repozytorium pozwala rozegrać pojedynek(ki) pomiędzy algorytmami sztucznej inteligencji dla gry **Shobu**.

Zasady gry Shobu

Przyjęte zostają reguły gry opisane na stronach:

- https://rules.dized.com/shobu/rule/gameplay/playing-the-game;
- https://364df235-af4b-4f4a-919f-d6c5b42b7d49.filesusr.com/ugd/693f33_ac912a3c391e4644a4d62a60ee2de749.pdf.

Dodatkowe założenia określają: maksymalny czas na wykonanie ruchu to **5 sekund**, limit ruchów wynosi **2000** (po tej liczbie ruchów gra jest uznawana za remis), gra kończy się również gdy gracz nie ma dostępnych ruchów.

Dodawanie swojego algorytmu

By utworzyć swojego *gracza* należy stworzyć plik w folderze `participants` zawierający 5 metod:

```
/*
  Autorzy.
  Opis algorytmu.
*/
module.exports = function () {
    return {
        init() {},
        startNextGame(color) { /* ... */ },
        getNextMove() { /* ... */ },
        applyOpponentMove(opponentMove) { /* ... */ },
        updateWithResult(winnerColor) { /* ... */ }
    }
};
```

Metoda `init` wywoływana jest **jednorazowo** przed turniejem i służy do inicjacji *gracza*.

Metoda `startNextGame` wywoływana jest przez każdą grą i na jej wejście podawany jest kolor gracza, gdzie `0` oznacza gracza czarnego a `1` gracza białego.

Pozycje kamieni na planszach do gry kodowane są za pomocą pozycji bitów (tj. od prawej do lewej, gdzie najmłodszy bit ma pozycje 0), gdzie koordynat `0x0` (lewy górny róg planszy) ma pozycję `0`:

```
(0,0)
   \
    0  1  2  3  x  x
    6  7  8  9  x  x
    12 13 14 15 x  x
    18 19 20 21 x  x
    x  x  x  x  x  x
    x  x
```

```
2^31                                                          2^0
x x x x x x x x x 21 20 19 18 x x 15 14 13 12 x x 9 8 7 6 x x 3 2 1 0
```

Przykładowo stan początkowy gry:

```
             WHITE's HomeBoards
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
--+---+---+---+---+--    --+---+---+---+---+--
0 | W | W | W | W | 0    0 | W | W | W | W | 0
1 | _ | _ | _ | _ | 1    1 | _ | _ | _ | _ | 1
2 | _ | _ | _ | _ | 2    2 | _ | _ | _ | _ | 2
3 | B | B | B | B | 3    3 | B | B | B | B | 3
--+---+---+---+---+--    --+---+---+---+---+--
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
X-------DARK---------ROPE--------LIGHT-------X
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
```

```
--+---+---+---+---+--    --+---+---+---+---+--
0 | W | W | W | W | 0    0 | W | W | W | W | 0
1 | _ | _ | _ | _ | 1    1 | _ | _ | _ | _ | 1
2 | _ | _ | _ | _ | 2    2 | _ | _ | _ | _ | 2
3 | B | B | B | B | 3    3 | B | B | B | B | 3
--+---+---+---+---+--    --+---+---+---+---+--
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
             BLACKS's HomeBoards
```

jest zakodowany jako:

```
[
    // BLACK's HomeBoards
    [
        // DARK
        [
            // BLACK's Stones
            0b00000000000111100000000000000000,
            // WHITE's Stones
            0b00000000000000000000000000001111
        ],
        // LIGHT
        [
            // BLACK's Stones
            0b00000000000111100000000000000000,
            // WHITE's Stones
            0b00000000000000000000000000001111
        ],
    ],
    // WHITE's HomeBoards
    [
        // DARK
        [
            // BLACK's Stones
            0b00000000000111100000000000000000,
            // WHITE's Stones
            0b00000000000000000000000000001111
        ],
        // LIGHT
        [
            // BLACK's Stones
            0b00000000000111100000000000000000,
            // WHITE's Stones
            0b00000000000000000000000000001111
        ],
    ]
]
```

Następnie w każdej turze gracz jest *proszony* o ruch poprzez metodę `getNextMove`, która musi zwrócić tablicę zawierającą koordynaty ruchu w postaci
`[kolorGracza, kolorPlanszyPasywnegoRuchu, kierunek, l.ruchów, pozycjaPasywnegoRuchu, kolorGraczaDlaAgresywnegoRuchu, pozycjaAgresywnegoRuchu]`.

W API gracz czarny zawsze oznaczony jest przez `0`, a biały przez `1`. Analogicznie, ciemna plansza oznaczona jest przez `0` a jasna przez `1`.

Przykład. Dla następującej (hipotetycznej) sytuacji w grze:

```
             WHITE's HomeBoards
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
--+---+---+---+---+--    --+---+---+---+---+--
0 | _ | _ | _ | _ | 0    0 | W | _ | _ | _ | 0
1 | _ | _ | _ | _ | 1    1 | _ | _ | _ | _ | 1
2 | _ | _ | _ | _ | 2    2 | _ | _ | _ | _ | 2
3 | _ | _ | _ | _ | 3    3 | _ | _ | _ | _ | 3
--+---+---+---+---+--    --+---+---+---+---+--
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
X-------DARK---------ROPE--------LIGHT------X
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
--+---+---+---+---+--    --+---+---+---+---+--
0 | _ | _ | W | B | 0    0 | _ | _ | _ | _ | 0
```

```
1 | _ | _ | _ | _ | 1    1 | _ | _ | _ | _ | 1
2 | _ | _ | _ | _ | 2    2 | _ | _ | _ | _ | 2
3 | _ | _ | _ | _ | 3    3 | _ | _ | _ | _ | 3
--+---+---+---+---+--    --+---+---+---+---+--
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
             BLACKS's HomeBoards
```

Wykonano ruch `[1,1,1,1,0,0,2]`, co tłumaczone jest na:

- `1` ruch wykonuje biały gracz,
- `1` na jasnej planszy po swojej strony (to wynika z zasad),
- w kierunku `1` co oznacza w prawo,
- o `1` pole,
- z punktu o pozycji `0`,
- agresywny ruch jest po stronie przeciwnika `0`,
- z pozycji `2`.

Wynik ruchu to:

```
             WHITE's HomeBoards
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
--+---+---+---+---+--    --+---+---+---+---+--
0 | _ | _ | _ | _ | 0    0 | _ | W | _ | _ | 0
1 | _ | _ | _ | _ | 1    1 | _ | _ | _ | _ | 1
2 | _ | _ | _ | _ | 2    2 | _ | _ | _ | _ | 2
3 | _ | _ | _ | _ | 3    3 | _ | _ | _ | _ | 3
--+---+---+---+---+--    --+---+---+---+---+--
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
X-------DARK---------ROPE--------LIGHT-------X
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
--+---+---+---+---+--    --+---+---+---+---+--
0 | _ | _ | _ | W | 0    0 | _ | _ | _ | _ | 0
1 | _ | _ | _ | _ | 1    1 | _ | _ | _ | _ | 1
2 | _ | _ | _ | _ | 2    2 | _ | _ | _ | _ | 2
3 | _ | _ | _ | _ | 3    3 | _ | _ | _ | _ | 3
--+---+---+---+---+--    --+---+---+---+---+--
  | 0 | 1 | 2 | 3 |        | 0 | 1 | 2 | 3 |
             BLACKS's HomeBoards
```

Po ruchu przeciwnika gracz otrzymuje informacje z koordynatami ruchu w postaci `[kolorGracza, kolorPlanszyPasywnegoRuchu, kierunek, l.ruchów, pozycjaPasywnegoRuchu, kolorGraczaDlaAgresywnegoRuchu, pozycjaAgresywnegoRuchu]` poprzez metodę `applyOpponentMove`.

Po zakończonej grze *gracz* otrzymuje informację o zwycięscy przez metodę `updateWithResult`, na wejście której podany jest kolor zwycięzcy (lub `-1` w przypadku remisu).

Nazwa pliku jest automatycznie nazwą gracza, proszę o jej ograniczenie do 12 znaków.

Turniej

Turniej rozgrywany jest w systemie każdy z każdym, pojedynczy mecz składa się z 100 gier (każdy program gra 50 jako czarny i 50 gier jako biały). Za wygraną w meczu gracz otrzymuje 1 punkt, za remis 0.5 oraz 0 w przypadku przegranej.

Turniej zostanie rozegrany na następującym sprzęcie:

- Intel Xeon W-2135 @ 3.70Ghz (6 rdzeni),
- 32 GB RAM,
- Ubuntu 20.04.1 LTS,
- wersja node.js-a podam w najbliższej przyszłości.

Inne

Komenda `npm install` instaluje niezbędne biblioteki.

Komenda `node tournament.js` uruchamia zawody.
Komenda `node tournament.js test` uruchamia zawody, w których starują trzej losowi gracze.
Komenda `node tournament.js timetest {nazwa pliku gracza}` uruchamia zawody, w których staruje losowy i wskazany gracz w celu pomiaru

czasu.

W repozytorium znajduje się implementacja losowego gracza, która jest dobrym punktem startu.