Sprawdzarka turniejowa

Uruchomienie

```
python3 local_ai_dueller_2023.py [--verbose 0|1|2] [--num_games N] GRA PROGRAM0 PROGRAM1
```

Uruchomi num_games rozgrywek między graczem w pliku PROGRAM0 a graczem PROGRAM1 (domyślnie 10 partii). Oba programy muszą być plikami wykonalnymi (na przykład skryptami bashowymi zawierającymi wiersz python3 kod_gracza.py). Gracze będą grali na zmianę. Opcja verbose wyświetli przebieg gier.

Uwaga: w razie problemów możesz spróbować podać pełną ścieżkę do plików

Uwaga 2: poleca się przeprowadzenie pierwszych rozgrywek przy użyciu agentów ze strony kursu (np. iago.py (aka reversi_random.py) vs. iago.py

Protokół komunikacji ze sprawdzaczką:

Program czyta ze standardowego wejścia i pisze na standardowe wyjście.
Sprawdzaczka może przysłać następujące komunikaty:

- UGO %f %f -- gracz otrzymujący ten komunikat program ma rozpocząć grę. Parametry podają czas na wykonanie ruchu i pozostały czas na grę.
- HEDID %f %f ... -- gracz otrzymujący ten komunikat ma zareagować na ruch przeciwnika. Pierwsze dwie liczby oznaczają czas na ruch i pozostały czas na grę, dalsza część linii zależy od gry i zawiera opis ruchu
- ONEMORE -- gracz ma się zresetować, zaczyna się nowa gra
- BYE -- gracz ma się wyłączyć

Gracze wysyłają sprawdzaczce następujące komunikaty:

- RDY -- określenie gotowości, wysyłane po starcie programu i po prośbie o reset.
- IDO ... -- odpowiedź na UGO i HEDID: ruch gracza, w formacie zależnym od gry.

Uwagi:

- Gracz wie, że zaczyna rozgrywkę jeśli po starcie dostaje komunikat UGO. Jeśli po starcie dostaje komunikat HEDID... oznacza to że jest drugim graczem.
- Czas inicjalizacji (i czas po resecie) jest limitowany z limitem 5s.
- Limity czasów dla ruchów są podawane przy każdym ruchu - liczy się czas mierzony przez sprawdzaczkę (a więc z narzutem na komunikację).

Gracz może wypisywać komunikaty do debugowania na stderr, kanał ten jest ignorowany przez sprawdzaczkę.

Przykładowa komunikacja:

```
P0 -> S: `RDY`
P1 -> S: `RDY`
S -> P0: `UGO 5.000000 60.000000`
P0 -> S: `IDO 3 2`
S -> P1: `HEDID 5.000000 60.000000 3 2`
P1 -> S: `IDO 2 2`
S -> P0: `HEDID 5.000000 59.970938 2 2`
P0 -> S: `IDO 5 4`
S -> P1: `HEDID 5.000000 59.962543 5 4`
P1 -> S: `IDO 4 2`
S -> P0: `HEDID 5.000000 59.938420 4 2`
P0 -> S: `IDO 2 3`
...
```

Protokoły dla poszczególnych gier ?

Reversi

Ruch zapisujemy jako parę liczb całkowitych przedzielonych spacją. Ruch pasujący jest parą -1 -1, dołożenia kamienia to dwie liczby z zakresu 0-7. Początkowe ustawienie planszy to

```
........
........
........
...#o...
...o#...
........
........
........
```

Grę rozpoczyna gracz o.

Dżungla

Ruch zapisujemy jako czwórkę liczb całkowitych przedzielonych spacją: XS YS XD YD gdzie XS, YS to współrzędne pola startowego, a XD, YD to współrzędne pola docelowego (współrzędne X zawierają się w [0, 6], a Y w [0, 8]). Ruch pasujący to -1 -1 -1 -1. Początkowe ustawienie planszy to:

```
L.....T
.D...C.
R.J.W.E
.......
.......
.......
e.w.j.r
.c...d.
t.....l
```

Partię rozpoczyna zawsze gracz grający małymi literami.

Szachy

Ruch zapisujemy zgodnie z UCI, formacie tekstowym obslugiwanym między innymi przez python-chess

Ostatnia modyfikacja: wtorek, 29 kwietnia 2025, 12:49

?