Zadanie 3 (WSI)

praca domowa 28.11.2020

Piotr Obst 304090 (grupa 103)

1. Treść zadania

Temat 3.1

Napisz program / skrypt, który buduje drzewo gry dla gry kółko i krzyżyk na planszy 3x3 (zasady: https://pl.wikipedia.org/wiki/K%C3%B3%C5%82ko_i_krzy%C5%BCyk), a następnie gra sam ze sobą. Zaimplementuj trzy sposoby prowadzenia gry:

1. W sposób losowy
2. Wykorzystując przegląd wyczerpujący
3. Wykorzystując algorytm minimax z wybraną przez siebie funkcją heurystyczną i różną maksymalną liczbą analizowanych ruchów w przód

Porównaj rozkłady wyników dla dużej liczby rozgrywek zawodników a vs. b, a vs. c oraz b vs. c.
Przeanalizuj czasy wykonania poszczególnych algorytmów.

2. Sposób uruchomienia

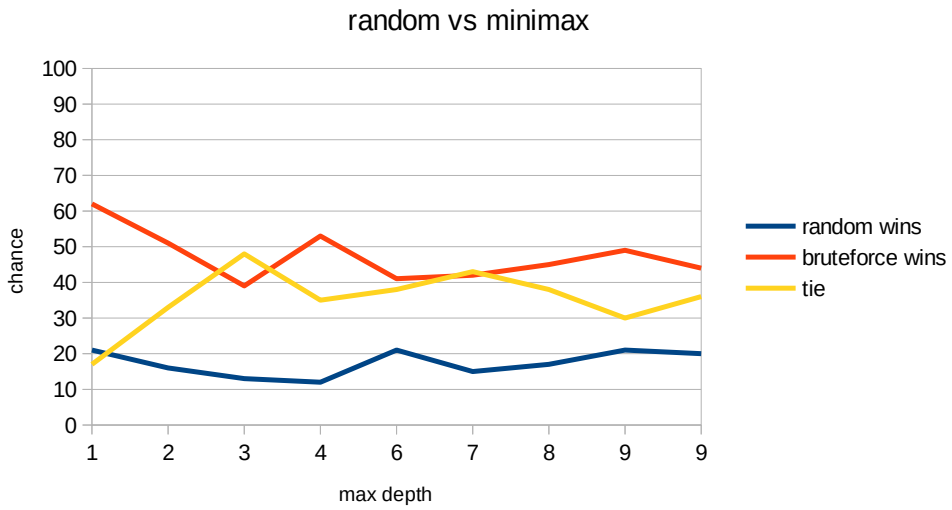
- Należy mieć zainstalowanego Pythona 3,
- Aby uruchomić program, należy wpisać w konsoli „python main.py”. Na niektórych maszynach należy użyć komendy python3 zamiast python.

3. Przykładowe wyjście programu

- przegląd wyczerpujący oznaczony jest jako bruteforce
- max depth oznacza maksymalną liczbę analizowanych ruchów w przód

```
Time for bruteforce initialization: 6.795612600
random wins: 0 bruteforce wins: 93 ties: 7 average game time: random - 0.000015245, bruteforce - 0.000018663
random wins: 21 minimax wins: 62 ties: 17 average game time: random - 0.000030494, minimax - 0.002455875 (max depth = 1)
random wins: 16 minimax wins: 51 ties: 33 average game time: random - 0.000042470, minimax - 0.013932245 (max depth = 2)
random wins: 13 minimax wins: 39 ties: 48 average game time: random - 0.000039716, minimax - 0.051704668 (max depth = 3)
random wins: 12 minimax wins: 53 ties: 35 average game time: random - 0.000040570, minimax - 0.186602202 (max depth = 4)
random wins: 21 minimax wins: 41 ties: 38 average game time: random - 0.000041390, minimax - 0.479255731 (max depth = 5)
random wins: 15 minimax wins: 42 ties: 43 average game time: random - 0.000040295, minimax - 0.950843058 (max depth = 6)
random wins: 17 minimax wins: 45 ties: 38 average game time: random - 0.000039862, minimax - 1.150219139 (max depth = 7)
random wins: 21 minimax wins: 49 ties: 30 average game time: random - 0.000039581, minimax - 1.177282545 (max depth = 8)
random wins: 20 minimax wins: 44 ties: 36 average game time: random - 0.000039076, minimax - 1.163541116 (max depth = 9)
bruteforce wins: 1 minimax wins: 0 ties: 0 average game time: bruteforce - 0.000037500, minimax - 0.002611000 (max depth = 1)
bruteforce wins: 1 minimax wins: 0 ties: 0 average game time: bruteforce - 0.000034000, minimax - 0.011579300 (max depth = 2)
bruteforce wins: 0 minimax wins: 0 ties: 1 average game time: bruteforce - 0.000056000, minimax - 0.062160000 (max depth = 3)
bruteforce wins: 0 minimax wins: 0 ties: 1 average game time: bruteforce - 0.000048200, minimax - 0.193452500 (max depth = 4)
bruteforce wins: 1 minimax wins: 0 ties: 0 average game time: bruteforce - 0.000034400, minimax - 0.478938000 (max depth = 5)
bruteforce wins: 1 minimax wins: 0 ties: 0 average game time: bruteforce - 0.000037200, minimax - 0.931781600 (max depth = 6)
bruteforce wins: 1 minimax wins: 0 ties: 0 average game time: bruteforce - 0.000034400, minimax - 1.056052700 (max depth = 7)
bruteforce wins: 1 minimax wins: 0 ties: 0 average game time: bruteforce - 0.000036900, minimax - 1.103954200 (max depth = 8)
bruteforce wins: 1 minimax wins: 0 ties: 0 average game time: bruteforce - 0.000037600, minimax - 1.106840200 (max depth = 9)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 0.003699300, bruteforce - 0.000038900 (max depth = 1)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 0.018871800, bruteforce - 0.000032200 (max depth = 2)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 0.090917800, bruteforce - 0.000035800 (max depth = 3)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 0.460152100, bruteforce - 0.000040600 (max depth = 4)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 1.722620300, bruteforce - 0.000047900 (max depth = 5)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 4.917757400, bruteforce - 0.000033800 (max depth = 6)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 9.584452500, bruteforce - 0.000033300 (max depth = 7)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 12.715893200, bruteforce - 0.000034500 (max depth = 8)
minimax wins: 0 bruteforce wins: 0 ties: 1 average game time: minimax - 12.945538500, bruteforce - 0.000040600 (max depth = 9)
```

4. **Rozkład wyników**
   - losowy vs bruteforce: wygrane losowego – 0%, wygrane bruteforce – 93%, remisy – 7%
   - losowy vs minimax

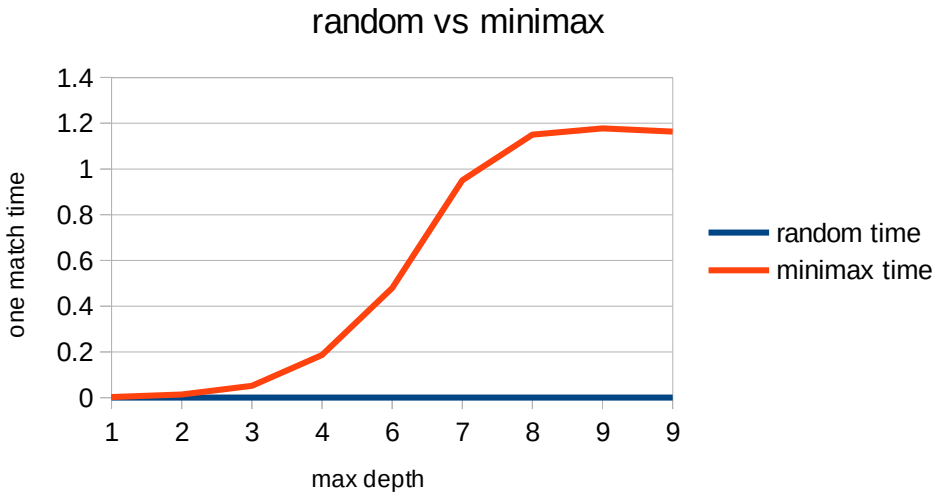


   - bruteforce vs minimax (zaczyna bruteforce)
     Dla max_depth = 3 oraz 4 jest remis, a dla pozostałych bruteforce wygrywa.
   - minimax vs bruteforce (zaczyna minimax)
     Zawsze remis.

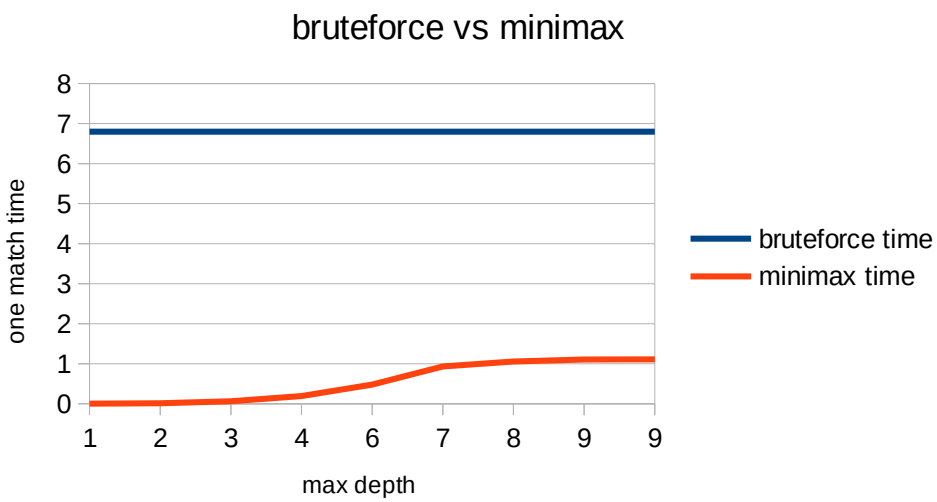
5. **Porównanie czasów wykonania**

   (Uwaga, do każdej gry algorytmu bruteforce doliczam czas obliczania drzewa gry, ale jest ono zawsze takie samo, i dzięki jego zapisywaniu w pliku można zaoszczędzić sporo czasu.)

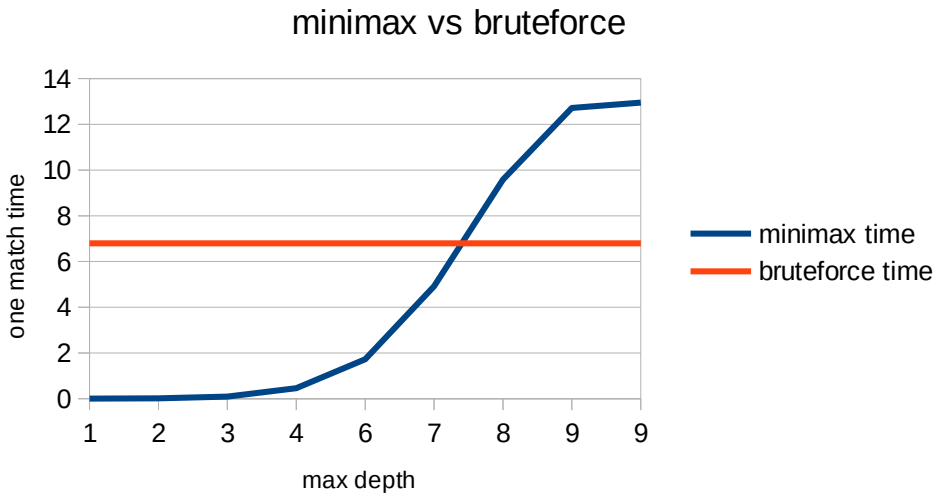
   - losowy vs bruteforce: losowy – 0.000015245s, bruteforce – 6.795631263
   - losowy vs minimax



- bruteforce vs minimax (zaczyna bruteforce)



- minimax vs bruteforce (zaczyna minimax)



6. Dodatkowe informacje

- Kod pisałem w oparciu o standard Pep8 oraz starałem się, aby był jak najbardziej uniwersalny i stabilny,
- Zgodnie z ustaleniami, starałem się pisać kod samodokumentujący się oraz to zadanie będzie jeszcze omawiane na konsultacjach, więc raczej nie ma potrzeby opisywania kodu w tym raporcie.