



PROJEKTOWANIE ALGORYTMÓW I METODY SZTUCZNEJ INTELIGENCJI

Autor:

Łukasz Walczak, 259278

Kierunek i wydział:

Automatyka i Robotyka, W12N

Termin zajęć:

Wtorek 13.15

Prowadzący:

Dr hab. inż. Andrzej Rusiecki

Projekt 3 - Kółko i krzyżyk

Spis treści

1	Cel ćwiczenia	1
2	Budowa gry	1
3	Algorytm minimax z alpha-betą cięć	1
4	Przebieg gry	2
5	Wnioski	4
6	Bibliografia	4

Link do repozytorium

<https://github.com/CrassusXY/TicTacToe>

1 Cel ćwiczenia

Należało zaimplementować grę w kółko i krzyżyk z wykorzystaniem algorytmu MinMax z alfa-beta cięciami. Gracz powinien posiadać możliwość definiowania rozmiaru pola (kwadratowego) wraz z ilością znaków w rzędzie.

2 Budowa gry

Program składa się z dwóch klas głównych:

- Klasa `tictactoe` - zawierająca całą mechanikę gry wraz z algorytmem minimax. Przechowuje ona informację o figurze gracza i komputera oraz planszę z grą. Posiada funkcje takie jak: inicjalizacja planszy pustymi polami, wyświetlenie planszy, settery, gettery oraz funkcję do wyznaczania optymalnego ruchu.
- Klasa `game` - jest to tak naprawdę klasa której celem jest obsługa całej mechaniki gry. Składa się ona z jednego obiektu typu `tictactoe` oraz z aktualnego wyniku gry.

3 Algorytm minimax z alfa-beta cięć

Głównym elementem całego programu jest algorytm minimax, który odpowiada za cały proces podejmowania decyzji przez komputer. Polega on na stawianiu i analizowaniu kolejnych ruchów. Analiza polega na maksymalizowaniu ruchów komputera, przy jednoczesnym minimalizowaniu opcji na ruch człowieka. Przy ocenie ruchu brana pod uwagę jest także aktualna głębokość w drzewie algorytmu, w taki sposób, żeby najlepiej oceniana była wygrana w najbliższym ruchu.

```
function minimax(isMax, depth) is
  if depth = maxdepth or node is a terminal node then
    return the heuristic value of node
  if maximizingPlayer then
    value :=
    for each child of node do
      value := max(value, minimax(!isMax, depth+1))
    return value
  else (* minimizing player *)
    value := +
    for each child of node do
      value := min(value, minimax(!isMax, depth+1))
    return value
```

Usprawnieniem tego algorytmu jest użycie alfa-beta cięć. Polega on na redukcji węzłów, które muszą być przeanalizowane. Warunkiem stopu jest znalezienie przynajmniej jednego rozwiązania czyniącego badany ruch gorszym od poprzedniego (założenie, że przeciwnik zawsze wykonuje najlepsze możliwe ruchy). Analizowanie takich gałęzi nie ma sensu, ponieważ nie przyniosą one optymalnego ruchu w końcowym rozrachunku. Ta technika pozwala zaoszczędzić bardzo dużo czasu, przez co możliwe jest użycie znacznie większej głębokości.

```
function minimax(isMax, depth) is
  if depth = maxdepth or node is a terminal node then
    return the heuristic value of node
  if maximizingPlayer then
    value :=
    for each child of node do
      value := max(value, minimax(!isMax, depth+1))
    return value
  else (* minimizing player *)
    value := +
```

```

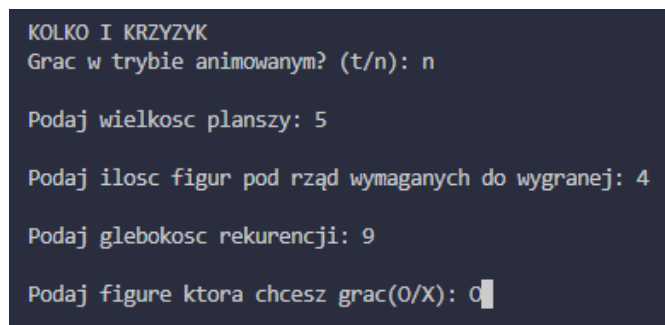
    for each child of node do
        value := min(value, minimax(!isMax, depth+1))
    return value

function minimax(isMax, depth, , ,)
if depth = 0 or node is a terminal node then
    return the heuristic value of node
if maximizingPlayer then
    value :=
    for each child of node do
        value := max(value, minimax(!isMax, depth+1, , ,))
        := max(, value)
    if value then
        break (* cutoff *)
    return value
else
    value := +
    for each child of node do
        value := min(value, minimax(!isMax, depth+1, , ,))
        := min(, value)
    if value then
        break (* cutoff *)
    return value

```

4 Przebieg gry

Gra zaczyna się od pobrania od użytkownika ustawień gry w jakich chce grać oraz jaką figurą chce być (założeniem jest, że grę zawsze zaczyna kółko, więc gracz w ten sposób wybiera czy chce zaczynać).



```

KOLKO I KRZYZYK
Grac w trybie animowanym? (t/n): n

Podaj wielkosc planszy: 5

Podaj ilosc figur pod rząd wymaganych do wygranej: 4

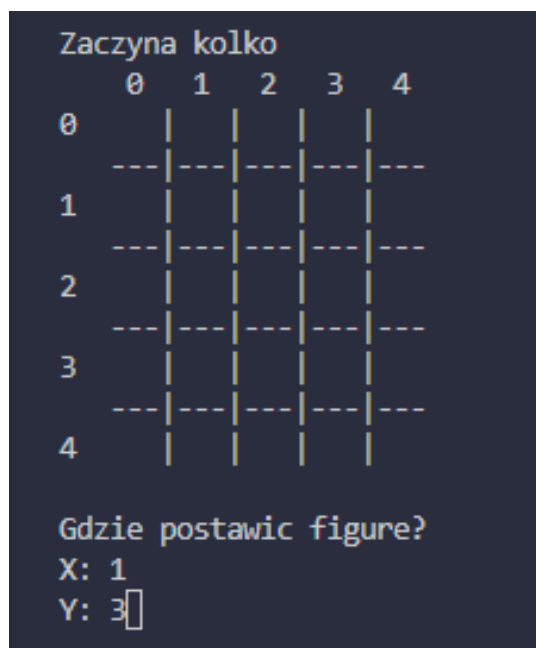
Podaj glebokosc rekurencji: 9

Podaj figure ktora chcesz grac(0/X): 0

```

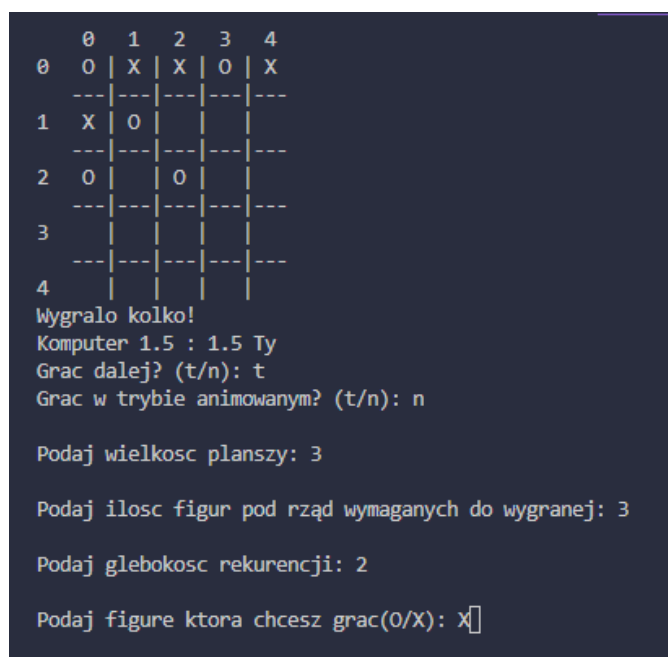
Rysunek 1: Uruchomienie gry

Po tym etapie wyświetla się plansza z grą na której na zmianę użytkownik jest pytany o ruch oraz komputer wykonuje swój ruch.



Rysunek 2: Przebieg gry

Gdy gra dobiega końca, użytkownik widzi wynik gry oraz jest pytany, czy chce grać dalej.



Rysunek 3: Zakończenie danej rozgrywki i ponowne jej uruchomienie

5 Wnioski

- Pisanie kodu zacząłem od stworzenia samej logiki gry, przy tym etapie nie było większych problemów. Najcięższe było napisanie optymalnego sprawdzania stanu planszy, ponieważ w dużej mierze to od niego zależy jak długo komputer szuka optymalnych ruchów. Jednym z usprawnień było pomijanie analizowania pól, w których nie ma szansy zaczynać się wygrywający szereg. Drugim usprawnieniem było pominięcie sytuacji, gdy nie padła jeszcze liczba ruchów dająca komukolwiek wygraną.
- W drugiej kolejności zająłem się napisaniem algorytmu minimax, który też nie był problematyczny biorąc pod uwagę ilość materiałów wyjaśniających go. Ponownie problemem było zoptymalizowanie go, ponieważ przy planszy 4x4, liczenie pierwszych ruchów trwało zdecydowanie zbyt długo. Pomogło wprowadzenie maksymalnej głębokości, jednak równocześnie zmniejszyło to dokładność ruchów komputera.
- W tym momencie zacząłem dodawać alpha-betę cięcie, która dała zaskakująco dobre rezultaty. Sprawdziłem sobie przy użyciu licznika, ile razy drzewo algorytmu minimax, dochodzi do stanu kończącego grę. W przypadku braku optymalizacji było to 255168 razy. Natomiast po dodaniu alpha-bety, liczba ta spadła do 8453. Początkowo ta wartość wydawało mi się, że jest to aż za dużo, jednak sprawdziłem i dokładność ruchów nie spadła.
- Dodatkową opcją sprawdzania algorytmu jest uruchamianie programu w trybie animacji. Polega on na wyświetlaniu na tablicy kolejnych ruchów wykonywanych przez komputer podczas analizy minimax. Dla początkowych ruchów jest to raczej zbędny bajer, jednak przy ruchach bliżej końca pokazuje nam on, że algorytm działa dokładnie z oczekiwaniami.

6 Bibliografia

- Wikipedia Minimaxa
<https://en.wikipedia.org/wiki/Minimax>
- Wikipedia Alpha-beta pruningu
https://en.wikipedia.org/wiki/Alpha-beta_pruning
- Analiza różnych możliwości gry przy planszy 3x3
<http://www.se16.info/hgb/tictactoe.htm>
- PDF z dobrą analizą alpha-beta cięcie
<https://www.whitman.edu/documents/Academics/Mathematics/2019/Felstiner-Guichard.pdf>
- Seria artykułów na geeksforgeeks
<https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/>