Politechnika
Wrocławska

Projektowanie algorytmów i metod sztucznej inteligencji

Projekt trzeci, Zadanie na 5, kółko i krzyżyk

Data: 05.06.2023 Poniedziałek 13:15

Kamil Winnicki(263434)

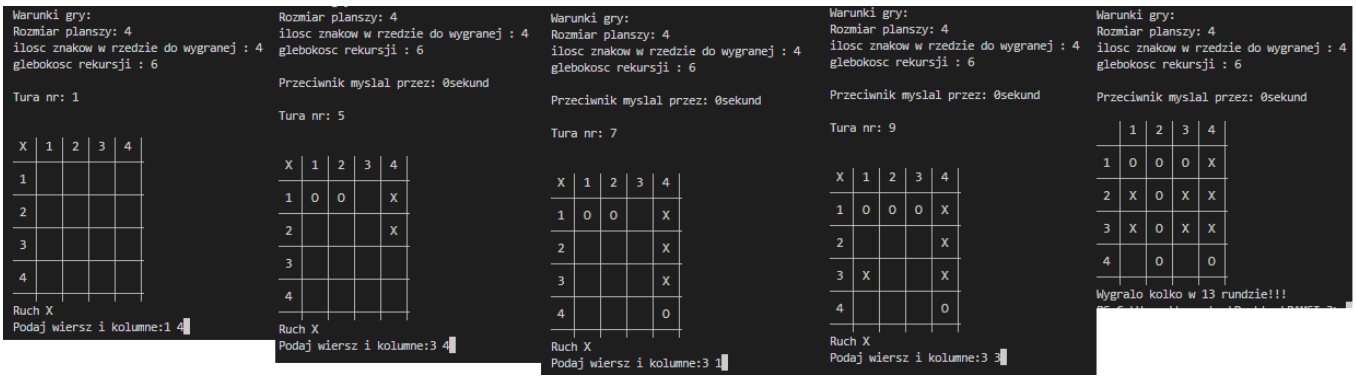
---

1 Wstep

Założeniem zadania na 5.0 była implementacja gry kółko i krzyżyk, w których ruchy przeciwnika maja być ustalane przy pomocy algorytmu MinMax, wraz z cieciami alfa-beta. Gracz powinien mieć możliwość ustalania rozmiaru planszy oraz ilości znaków w rzedzie do wygranej.

2 Opis działania programu

1. Program rozpoczyna od pobrania od użytkownika trzech wartości: rozmiaru planszy, ilości znaków w rzedzie potrzebnych do wygranej oraz głębokości rekursji.
2. Nastepnie sprawdza, czy podane wartości sa poprawne, czyli spełniaja wymagane warunki, np aby rozmiar planszy był wiekszy od 2 lub ilość znaków w rzedzie potrzebnych do wygranej była mniejsza lub równa wielkości planszy.
3. Losowo wybierany jest gracz rozpoczynajacy gre.
4. Program wykonuje petle, w której wyświetlana jest aktualna plansza gry oraz gracze na przemian wykonuja swoje ruchy, gdzie komputer to "O"(kółko) a gracz to "X"(krzyżyk).
5. Jeśli ruch wykonuje komputer (kółko), program automatycznie wybiera najlepszy możliwy ruch na podstawie algorytmu min-max z optymalizacja alfa-beta ciecia.
6. Jeśli ruch wykonuje człowiek (krzyżyk), program prosi o podanie współrzednych wybranego miejsca na planszy.
7. Po zakończeniu gry, program wyświetla plansze oraz informuje o wyniku rozgrywki: wygranej krzyżyka, wygranej kółka lub remisie.



Rysunek 1: Przykładowy przebieg rozgywki

3 Opis zastosowanych algorytmów sztucznej inteligencji

3.1 Algorytm MinMax

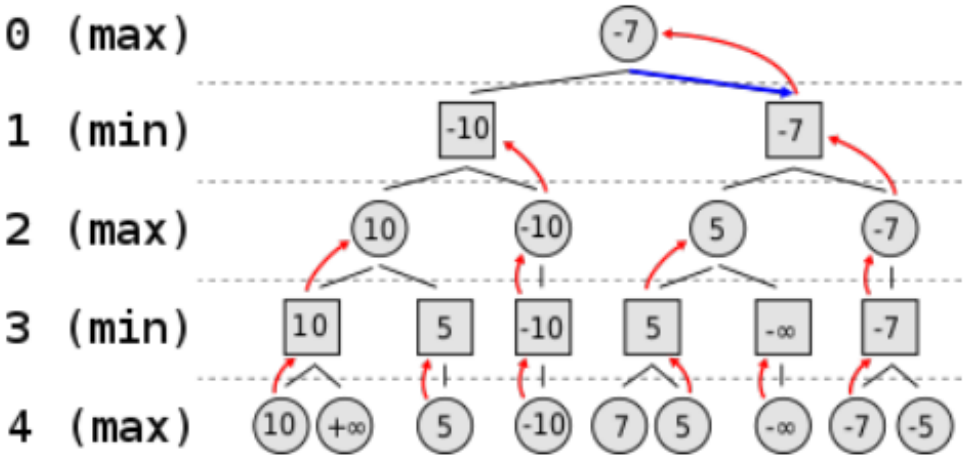
Algorytm Minimax jest używany w grach np. w takich jak kółko i krzyżyk, do podejmowania optymalnych decyzji. Działa na zasadzie przegladania drzewa możliwych ruchów w grze, aby przewidzieć najlepszy ruch dla danego gracza.

Algorytm zakłada, że gracze sa inteligentni i daża do maksymalizacji (gracz maksymalny-"X") lub minimalizacji (gracz minimalny -"O") wyniku.

Działanie algorytmu opiera sie na rekurencji. Algorytm przeglada drzewo gry, rozważajac wszystkie możliwe ruchy na każdym poziomie. Jeśli osiagnieto warunek zakończenia gry lub maksymalna głebokość rekurencji, algorytm zwraca wartość oceny planszy, która jest obliczana za pomoca funkcji "evaluateBoard()".

W przeciwnym razie, algorytm przechodzi przez wszystkie możliwe ruchy danego gracza. Dla każdego ruchu, algorytm rekurencyjnie wywołuje sam siebie, zmniejszajac głebokość o 1 i zmieniajac gracza. Wynik rekurencyjnego wywołania jest ocena stanu gry dla danego ruchu.

Gracz maksymalny wybiera ruch z najwyższa ocena, dażac do maksymalizacji wyniku, podczas gdy gracz minimalny wybiera ruch z najniższa ocena, dażac do minimalizacji wyniku. Algorytm kończy działanie, zwracajac najlepsza ocene dla danego gracza, na podstawie przeszukanych ruchów.

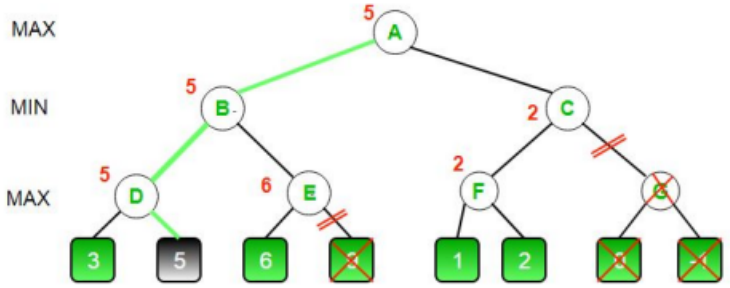


Rysunek 2: Przykład działania algorytmu minMax

3.2 Alfa- Beta ciecia

Podstawowym założeniem optymalizacji Alfa-Beta cieć jest to, że jeśli znaleziono już ruch, który prowadzi do lepszej wartości dla gracza minimalizujacego niż poprzedni najlepszy ruch gracza maksymalizujacego, to nie ma sensu kontynuować przeszukiwania innych gałezi tego drzewa, ponieważ gracz maksymalizujacy i tak nie wybierze tych ruchów. Podobnie, jeśli znaleziono już ruch, który prowadzi do lepszej wartości dla gracza maksymalizujacego niż poprzedni najlepszy ruch gracza minimalizujacego, to nie ma sensu kontynuować przeszukiwania innych gałezi, ponieważ gracz minimalizujacy i tak nie wybierze tych ruchów.

Algorytm Minimax z optymalizacja Alfa-Beta cieć wprowadza dwa parametry: alfa i beta. Parametr alfa reprezentuje najlepsza wartość dla gracza maksymalizujacego, a beta reprezentuje najlepsza wartość dla gracza minimalizujacego. Dzieki optymalizacji Alfa-Beta cieć, algorytm Minimax może znacznie ograniczyć liczbe przeszukiwanych wezłów, co prowadzi do znacznego przyspieszenia działania algorytmu.



Rysunek 3: Przykład działania algorytmu minMax z cieciami alfa beta

4 Testy wydajnościowe

W celu sprawdzenia wydajności programu wykonano testy wpływu głebokości rekurencji na czas wykonywania ruchu przez przeciwnika Testy wykonano dla planszy 7x7 oraz 4 znaków w rzedzie potrzebnych do wygranej. Wyniki przedstawiono w poniższej tabeli:

	czas ruchu przeciwnika[s]						
Poziom rekurencji	***ruch 1***	***ruch 2***	***ruch 3***	***ruch 4***	***ruch 5***	***ruch 6***	***ruch 7***
2	**0,0008**	**0,0016**	**0,0016**	**0,0014**	**0,0015**	**0,0015**	**0,0015**
4	**0,081**	**0,084**	**0,119**	**0,082**	**0,074**	**0,052**	**0,045**
5	**1,22**	**1,15**	**1,04**	**1,48**	**1,32**	**0,98**	**0,37**
6	**6,33**	**1,37**	**1,18**	**2,27**	**2,59**	**1,63**	**1,08**
7	**71**	**101**	**47**	**18**	**31**	**17**	**19**

5 Wnioski

- Głebokość rekurencji wpływa na trudność rozgrywki ponieważ przeciwnik potrafi przewidzieć wiecej ruchów w przód przez co potrafi zasymulować nasz ruch. Duża głebokość sprawia jednak że algorytm bardziej obciaża komputer przez co przeciwnik "długo myśli"nad ruchem co sprawia że rozgrywka jest nieprzyjemna.
- Dodanie alfa-beta cieć jest dużym usprawnieniem działania programu.
- Im mniejsza plansza tym przeciwnik radzi sobie lepiej - trudniej z nim wygrać.

6 bibliografia

https://www.geeksforgeeks.org/finding-optimal-move-in-tic-tac-toe-using-minimax-algorithm-in-game-theory/

https://www.neverstopbuilding.com/blog/minimax