Adam Kubiak

249480

Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 4

29.05.2020

Grupa: wtorek 15:15 – 16:55

1. Wstęp

Tematem projektu było stworzenie wybranej gry przy zastosowaniu algorytmu min-max w celu stworzenia sztucznej inteligencji przeciwko której można by grać. W moim przypadku wybrałem grę kółko i krzyżyk. Gracz otrzymał możliwość wyboru wielkości planszy do gry oraz ilości znaków w rzędzie potrzebnych do wygrania oraz kto zaczyna rozgrywkę.

2. Opis gry

Gra została stworzona za pomocą następujących klas:

• Game

• Board

• Player

• AI

Poszczególne klasy przechowują w sobie informacje najważniejsze dla nich. Pole gry przechowywane jest strukturze danych złożonej z vectorów, „tablicy" dwu wymiarowej typu char w klasie Board. Możliwymi wartościami jest 'X' oraz '0', ' ' oznacza puste pole. Klasa Player przechowuje informacje o tym jaki znak obiekt tejże klasy przechowuje. Znajduje się w niej także funkcja move(Board &board) dzięki której możemy przypisać znak do pola planszy wybranego przez gracza bez obawy że gracz źle wprowadzi współrzędne dzięki znajdującym się w tej funkcji zabezpieczeniom. Klasa AI dziedziczy od klasy Player w niej również znajduję się funkcja move(Board &board) która ma w sobie algorytm min-max. Klasa Game odpowiada za cały przebieg gry. Cała rozgrywka jest prezentowana w terminalu.

Gra odbywa się według następującego schematu:

1. Gracz wybiera tryb rozgrywki, ma wybór pomiędzy graniem przeciwko drugiemu graczowi lub AI. Jeśli gracz wybrał jako przeciwnika AI, może on także zdecydować kto zacznie rozgrywkę. Gracz musi także podać wymiary dla planszy oraz liczbę znaków w rzędzie gwarantującą zwycięstwo.

2. Program po wyświetleniu planszy czeka aż gracz poda współrzędne dla swojego ruchu, jeśli tura należy do AI, natychmiastowo zastaje wykonany ruch. Następuje ewaluacja czy gracz lub AI nie wygrał lub czy gra nie zakończyła się remisem.

3. Jeśli gra toczy się dalej, następuje powrót do punktu 2.

4. W razie końca gry wyświetla się odpowiednia informacja kto jest zwycięzcą i program wraca do menu głównego

Zastosowany przeze mnie algorytm min-max został poddany kilku modyfikacjom w celu zwiększenia wydajności. Po pierwsze, została dodana implementacja algorytmu cięć alfa-beta. Oznacza to iż algorytm nie będzie szukał najlepszego rozwiązania w gałęzi która jest gorsza od aktualnego rozwiązania. Po drugie została ograniczona maksymalna głębokość rekurencji do wartości:

depth == 5

Modyfikacje te sprawiły iż gra nawet na większej planszy może odbywać się bez zbędnych przerw. Interfejs programu wygląda następująco:

Menu:

```
Wybierz tryb gry:
1. Tryb jednoosobowy
2. Tryb dla dwoch graczy
1
Podaj wymiar tablicy: 3
Liczba potrzebnych znakow w rzedzie do wygranej: 3
0 - zaczyna AI
1- zaczyna gracz
```

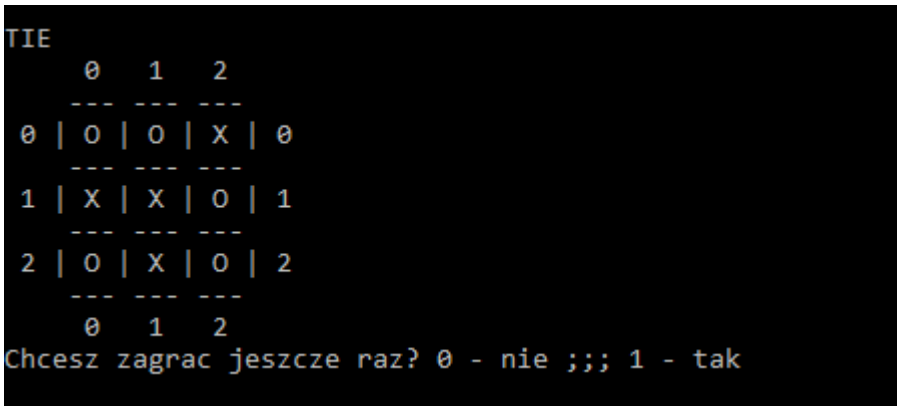
Rozgrywka podczas gdy zaczyna AI:

```
TIC TAC TOE!!!!
     0   1   2
    --- --- ---
 0 | O |   |   | 0
    --- --- ---
 1 |   |   |   | 1
    --- --- ---
 2 |   |   |   | 2
    --- --- ---
     0   1   2
Ruch gracza X
Numer wiersza:
```

Środek rozgrywki:

```
TIC TAC TOE!!!!
     0   1   2
    --- --- ---
 0 | O | O | X | 0
    --- --- ---
 1 |   | X |   | 1
    --- --- ---
 2 | O |   |   | 2
    --- --- ---
     0   1   2
Ruch gracza X
Numer wiersza:
```

Po zakończonej rozgrywce wyświetla się komunikat kto wygrał lub o remisie:



Sprawdzanie czy gra została zakończona odbywa się poprzez sprawdzenie wszystkich linii oraz rzędów a następnie poszczególnych skosów na których możliwe byłoby wygranie gry.

3. Opis algorytmu SI

Zastosowany przeze mnie algorytm cięć alfa-beta (pochodna min-max),opiera się na przeszukaniu całej planszy w poszukiwaniu najlepszego możliwego rezultatu dla szukającego, równocześnie szukając takiego rozwiązania które będzie najmniej korzystne dla strony przeciwnej. Dzieląc ten algorytm na punkty prezentuje się on następująco:

1. Wstaw swój znak w pierwsze wolne miejsce na planszy i znajdź dla niego wynik algorytmu min-max. Powtórz dla każdego wolnego miejsca.

a. Jeśli wygrywasz zwróć 10 – głębokość rekurencji

b. Inaczej jeśli przegrywasz zwróć -10 + głębokość rekurencji

c. Inaczej jeśli plansza jest pełna lub osiągnięto maksymalną głębokość zwróć 0

d. **Jeśli ruch MAX:**

i. Wstaw swój znak w pierwsze wolne miejsce i znajdź dla niego wynik algorytmu min-max

e. **Jeśli ruch MIN:**

i. Wstaw przeciwny znak w pierwsze wolne miejsce i znajdź dla niego wynik algorytmu min-max

2. Zwrócone wyniki porównaj w poszukiwaniu wartości maksymalnej i ten ruch wykonaj.

Dodane przeze mnie modyfikacje zwracające wartość przy wygranej lub przegranej pozwalają na szybszą wygraną lub grę do samego końca, w przypadku gdy nie ma możliwości wygrania. Modyfikacje wprowadzone przez algorytm cięć alfa-beta polegają na przypisaniu do wartości alfa wartości zwróconej przez algorytm min-max dla gracza MAX, oraz do wartości beta wartości zwróconej przez algorytm min-max dla gracza MIN. Podczas działania programu porównywane są wartości alfa i beta. Jeśli spełniony jest warunek alfa >= beta pętla kończy się i algorytm zwraca wartość najlepszego znalezionego rozwiązania do tej pory.

4. Wnioski

Napisana gra jest niemożliwa do wygrania w przypadku ilości znaków potrzebnych w linii równej wielkości planszy. W takim wypadku możliwy jest jedynie remis. Szansą na wygranie gracza jest stworzenie planszy o rozmiarze większym niż potrzebna ilość znaków i jednoczesnym rozpoczęciu gry. W przypadku dużych plansz, algorytm potrzebuje trochę czasu na znalezienie najlepszego możliwego ruchu. Wynika to z natury rekurencyjności algorytmu. Algorytm w podstawowej wersji potrzebował zbyt wiele czasu dla planszy 4x4 gdy nie był ograniczony ani przez cięcia alfa-beta ani przez głębokość rekurencji.

5. **Bibliografia**

- https://en.wikipedia.org/wiki/Minimax
- https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning