Algorytmy Sortowania

Projekt 3
Projektowanie i Analiza Algorytmów

Danil Krassotov 282656

Wstęp

Celem niniejszego sprawozdania jest implementacja oraz analiza efektywności algorytmu sztucznej inteligencji w kontekście gry w warcaby. W ramach projektu zaimplementowano grę w warcaby, z wykorzystaniem algorytmy **Minimax** z przycinaniem *Alfa-Beta*.Dla algorytmu przeprowadzono pomiary, aby ocenić wydajność. Na podstawie zebranych danych zostaną wygenerowane wykresy i przedstawiona analiza wyników, co pozwoli na wyciągnięcie wniosków.

1 Podstawowa Logika Gry

Główna logika gry w warcaby została zbudowana w oparciu o kilka współpracujących ze sobą klas, które reprezentują podstawowe elementy gry oraz zarządzają jej przebiegiem. Moduły te są odpowiedzialne za stan planszy, definicję pionków i ruchów, a także za ogólne sterowanie rozgrywką.

1.1 Reprezentacja Elementów Gry

W celu modelowania stanu gry, zdefiniowano trzy podstawowe struktury: ‘Position‘, ‘Piece‘ i ‘Move‘.

Klasa ‘Position‘

Struktura ‘Position‘ jest używana do reprezentacji współrzędnych pola na planszy. Składa się z dwóch składowych: ‘row‘ (wiersz) i ‘col‘ (kolumna).

Klasa ‘Piece‘

Klasa ‘Piece‘ reprezentuje pojedynczy pionek na planszy. Przechowuje informacje o jego kolorze (biały lub czarny) oraz typie (pionek lub damki).

Klasa ‘Tile‘

Klasa ‘Tile‘ reprezentuje pojedyncze pole na planszy. Każde pole może zawierać pionek lub być puste. Klasa ta przechowuje również współrzędne pola (‘row‘, ‘col‘) oraz status podświetlenia (‘highlighted‘), używany głównie przez moduł wizualizacji.

Klasa ‘Move‘

Klasa ‘Move‘ enkapsuluje pojedynczy ruch w grze, zawierając informacje o pozycji początkowej (‘from‘), pozycji końcowej (‘to‘) oraz liście pozycji zbitych pionków (‘captured‘). Jest to kluczowe dla obsługi bić wielokrotnych w warcabach.

Klasa ‘Board‘

Klasa ‘Board‘ jest centralnym elementem logiki gry, odpowiadającym za utrzymywanie stanu planszy, inicjalizację, walidację ruchów oraz ich aplikowanie. Plansza jest reprezentowana jako dwuwymiarowa tablica obiektów ‘Tile‘.

Klasa ‘Game‘

Klasa ‘Game‘ odpowiada za przebiegi całej rozgrywki. Odpowiada za inicjalizację gry, zarządzanie kolejnością graczy oraz sprawdzanie warunków zakończenia gry.

2 Sztuczna Inteligencja (AI)

Moduł sztucznej inteligencji jest odpowiedzialny za podejmowanie decyzji o ruchach dla komputera. Wykorzystuje on algorytm przeszukiwania drzewa gry, wsparty funkcją oceniającą stan planszy, aby wybrać optymalny ruch.

2.1 Algorytm Minimax z przycinanie Alpha-Beta

Podstawą działania AI jest algorytm **Minimax** z optymalizacją w postaci **przycinania Alpha-Beta**. Minimax to algorytm rekurencyjny służący do podejmowania optymalnych decyzji. Działa poprzez przeszukiwanie drzewa gry, w którym węzły reprezentują stany planszy, a krawędzie – możliwe ruchy. Celem gracza maksymalizującego (AI) jest maksymalizacja wartości funkcji oceniającej, natomiast gracza minimalizującego (przeciwnik) – minimalizacja tej wartości.

Przycinanie Alpha-Beta to technika optymalizacyjna, która znacząco redukuje liczbę węzłów, które muszą zostać odwiedzone w drzewie przeszukiwania, bez wpływu na ostateczny wynik. Działa poprzez eliminowanie gałęzi, które na pewno nie prowadzą do optymalnego rozwiązania.

- **Alpha** (dla gracza maksymalizującego): Przechowuje najlepszą wartość znalezioną do tej pory dla gałęzi, która jest badana przez gracza maksymalizującego.
- **Beta** (dla gracza minimalizującego): Przechowuje najlepszą wartość znalezioną do tej pory dla gałęzi, która jest badana przez gracza minimalizującego.

Jeśli w trakcie przeszukiwania drzewa Alpha staje się większa lub równa Becie ($\alpha \geq \beta$), oznacza to, że bieżąca gałąź jest gorsza od już znalezionej ścieżki i można ją "odciąć", czyli przerwać dalsze przeszukiwanie.

2.2 Funkcja Oceniająca

Funkcja ‘evaluateBoard()‘ jest kluczowym elementem AI, ponieważ przypisuje liczbową wartość danemu stanowi planszy. Im wyższa wartość, tym lepsza jest pozycja dla gracza AI. Ocena opiera się na kilku czynnikach.

- **Wartość pionków**: Pionki - 100, damki - 300;
- **Mobilność**: Liczba dostępnych ruchów. Większa mobilność jest korzystna.
- **Postęp**: Nagradza pionki, które przesunęły się dalej w kierunku rzędu przeciwnika.
- **Kontrola centrum**: Preferuje pionki znajdujące się bliżej centrum planszy, które są strategicznie ważniejsze.
- **Kara za tchórzostwo**: Karze pionki znajdujące się na krawędziach planszy, ponieważ mają ograniczoną mobilność.
- **Bonus za ostatni rząd**: Nagradza pionki, które znajdują się w ostatnim rzędzie (gotowe do promocji na damki lub już jako damki).

2.3 Złożoność Obliczeniowa

W najgorszym przypadku, bez cięć, złożoność wynosi $O(b^d)$. Dzięki cięciom Alpha-Beta, średnia złożoność jest redukowana do $O(b^{d/2})$, co znacząco przyspiesza działanie algorytmu i pozwala na przeszukiwanie większych głębokości.

3 Analiza pomiarów

W celu oceny wydajności zaimplementowanego algorytmu sztucznej inteligencji (AI), przeprowadzono serię pomiarów czasu potrzebnego na znalezienie optymalnego ruchu. Pomiary zostały wykonane dla różnych głębokości przeszukiwania, od 1 do 6. Dla każdej głębokości przeprowadzono 5 testów, a wyniki przedstawiono jako średnią, minimalną i maksymalną wartość czasu działania.

3.1 Wyniki Pomiarów

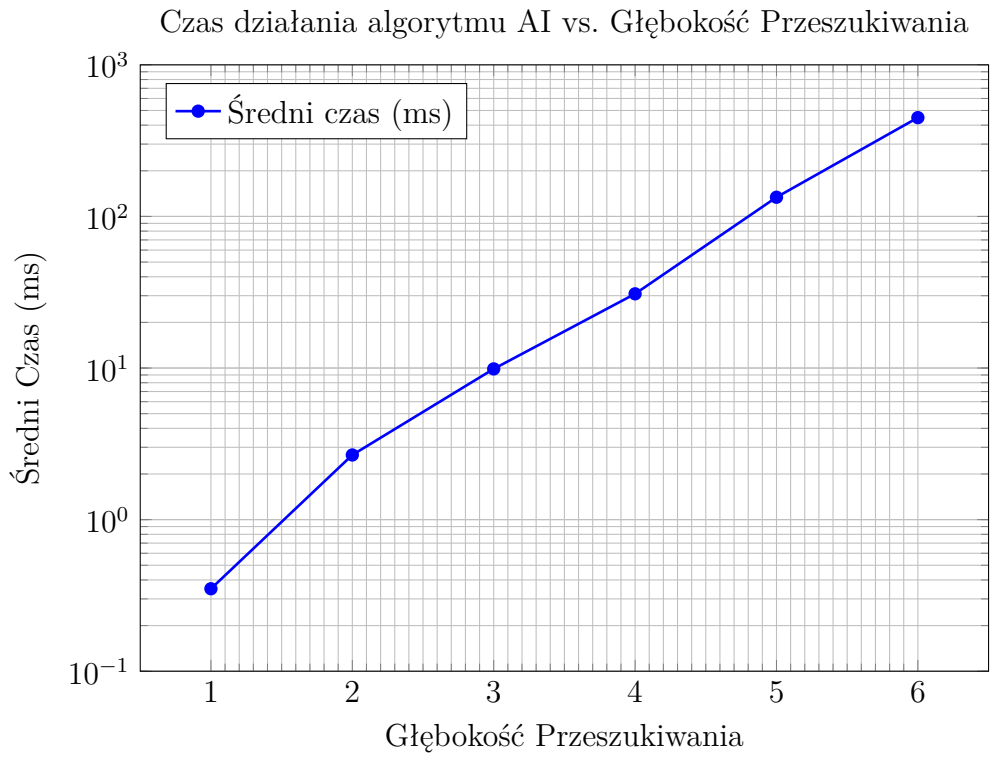
Poniższa tabela przedstawia zebrane dane dotyczące czasu działania AI w milisekundach (ms) w zależności od głębokości przeszukiwania.

Tabela 1: Wyniki pomiarów wydajności algorytmu AI

Głębokość	Średnia (ms)	Min (ms)	Max (ms)	Testów
1	0.35	0.35	0.35	5
2	2.67	2.38	2.93	5
3	9.87	9.64	9.98	5
4	30.87	30.22	32.52	5
5	133.84	132.62	135.59	5
6	448.41	441.23	459.29	5

3.2 Wykresy i Interpretacja

Poniższy wykres przedstawia średni czas działania algorytmu AI w funkcji głębokości przeszukiwania. Oś Y jest w skali logarytmicznej, co pozwala lepiej zaobserwować wykładniczy charakter wzrostu czasu.



Rysunek 1: Wykres czasu działania algorytmu AI w zależności od głębokości przeszukiwania (skala logarytmiczna na osi Y)

Analiza Złożoności Czasowej

Zgodnie z oczekiwaniami teoretycznymi dla algorytmu Minimax z cięciami Alpha-Beta, czas działania rośnie w sposób wykładniczy wraz ze wzrostem głębokości przeszukiwania. Na wykresie, mimo zastosowania skali logarytmicznej na osi Y, wyraźnie widać nieliniowy wzrost.

Przedstawione stosunki czasów pomiędzy kolejnymi głębokościami potwierdzają to:

- Stosunek czasu głębokości 2 do głębokości 1: 7.69x
- Stosunek czasu głębokości 3 do głębokości 2: 3.70x
- Stosunek czasu głębokości 4 do głębokości 3: 3.13x
- Stosunek czasu głębokości 5 do głębokości 4: 4.34x
- Stosunek czasu głębokości 6 do głębokości 5: 3.35x

Teoretyczna złożoność dla Alpha-Beta wynosi $O(b^{d/2})$, gdzie b to współczynnik rozgałęzienia, a d to głębokość. Obserwowane wyniki są zgodne z tym modelem. Dla głębszych

przeszukiwań (np. powyżej 6), czas działania szybko staje się nieakceptowalny dla rozgrywki w czasie rzeczywistym.

Pomiary potwierdzają, że algorytm AI, choć skuteczny w znajdowaniu dobrych ruchów, wymaga optymalizacji dla osiągnięcia większej głębokości przeszukiwania w rozsądnym czasie.

4 Wnioski

Kluczowym elementem projektu jest **moduł Sztucznej Inteligencji**, wykorzystujący algorytm **Minimax z cięciami Alpha-Beta**. Algorytm ten efektywnie przeszukuje drzewo gry, a zastosowanie cięć redukuje liczbę analizowanych stanów planszy, co jest kluczowe dla osiągnięcia sensownych czasów odpowiedzi. Funkcja oceniająca pozwala AI na podejmowanie racjonalnych i często optymalnych decyzji.

Analiza pomiarów wydajności potwierdziła teoretyczne przewidywania dotyczące złożoności obliczeniowej. Czas działania algorytmu AI rośnie wykładniczo wraz ze wzrostem głębokości przeszukiwania, co jest charakterystyczne dla problemów przeszukiwania drzewa gry. Chociaż cięcia Alpha-Beta znacząco zmniejszają ten wzrost, już na głębokości 6 czasy stają się znaczące (ponad 400 ms). Pokazuje to, że dla osiągnięcia większej mocy AI, konieczne są dalsze optymalizacje.

Podsumowując, projekt oferuje zarówno funkcjonalną grę w warcaby, jak i demonstrację działania algorytmów AI. Zaimplementowane podejście jest skalowalne i chętnie ulepszę projekt po sesji.