Izabella Cwojdzińska 249001
Czwartek 9:15 – 11:00
Prowadzący: Dr inż. Łukasz Jeleń

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 3: Gra

1. Wprowadzenie:
   Celem projektu było napisanie programu opartego na algorytmie Min-Max. Badany był on na podstawie gry zwanej kółko i krzyżyk, w której jest możliwość zmiany:
   - rozmiaru kwadratowego pola (do dyspozycji: 3x3 i większe)
   - ilości wygrywających znaków w rzędzie

2. Algorytm min-max z obcięciem alfa-beta:
   Jest to algorytm przeszukujący oraz redukujący liczbę węzłów, które muszą być rozwiązywane w drzewach przeszukujących przez algorytm min-max. Właśnie takie rozwiązanie wykorzystywane jest w grach z dwoma graczami (kółko i krzyżyk, szachy). Warunkiem stopu jest znalezienie co najmniej pojedynczego rozwiązania, które czyni obecnie badaną opcję ruchu gorszą od poprzednich opcji. Wybranie takiej opcji nie przyniosłoby korzyści graczowi ruszającemu się, dlatego też nie ma potrzeby przeszukiwać dalej gałęzi drzewa tej opcji. Dzięki tej technice możemy zaoszczędzić czas poszukiwania bez zmiany wyniku działania algorytmu.
   Obcięcia alfa – beta zostały wprowadzone, gdyż bez tego rozwiązania program potrzebował bardzo dużo czasu, aby obliczyć wszystkie możliwe rozwiązania dla rozmiaru planszy większych niż 3. Dodatkowo osobna funkcja, która zwraca głębokość przeszukiwania w drzewie algorytmu też miała duży wpływ na optymalizację działania algorytmu oraz skrócenie czasu obliczeniowego.

3. Budowa programu
   Najważniejsze moduły aplikacji:
   - Board.h – Tutaj jest definicja klasy Board, wraz z najpotrzebniejszymi metodami wykorzystywanymi do badania przebiegu gry takie jak: check_who_win(), is_movie_left().
   - AI.h – Zawiera :
     - ➔ funkcję algorytmu – minimax(…), która znajduje najlepszy możliwy ruch – best_move(…)
     - ➔ funkcję do dynamicznego ustawiania głębokości przeszukiwania - depth_2(…),
     - ➔ funkcję zwracającą indeksy ruchu komputera do wstawienia na planszę gry – c_put(…).
   - Player.h – Zawiera funkcje potrzebne do wstawienia ruchu pobranego od gracza.

- ➔ enter_move (…) – funkcja rekurencyjna, wywołująca sama siebie, gdy gracz popełni jakiś błąd np. gdy poda indeks większy niż rozmiar planszy, lub gdy wybierze już zajęte miejsce.
- ➔ put(…) – funkcja, która wstawia na planszę znak gracza. Umożliwia nam grać w 2 osoby.

- Move.h – Przechowuje numer kolumny oraz wiersza, aby wstawić ruch. Zawiera przeciążone operatory przesunięć bitowych.
- Interface_handling.h – Zawiera pomniejszone funkcje, dzięki którym łatwiej jest użytkownikowi komunikować się z programem.
- main.cpp – Znajduje się tutaj funkcja driver(…), która bada czas wykonania pierwszego ruchu komputera. Pierwszy przypadek to taki gdy zaczyna rozgrywkę gracz, a drugi gdy rozpoczyna komputer. Dodatkowo w tym pliku jest pętla, która odpowiada za daną rozgrywkę

4. Wyniki:

Rozmiar	Komputer ma pierwszy ruch [ms]	Komputer ma drugi ruch [ms]
3x3	197,05	21,17
4x4	28500,37	15316,11
5x5	56446,97	46358,94
6x6	20617,31	17587,25
7x7	2424,71	2207,97
8x8	8141,06	7457,29
9x9	22877,38	22613,09
10x10	643,59	596,65
15x15	11725,74	11684,73

Analiza głębokości przeszukiwania na podstawie rozmiaru planszy:	
Rozmiar: 3	Głębokość: 9
Rozmiar: <4;7)	Głębokość: 9 – rozmiar + 1
Rozmiar: <7;10)	Głębokość: 3
Rozmiary: 10 i większe	Głębokość: 2

Dzięki takiemu rozwiązaniu, czas oczekiwania aż komputer wykona ruch jest dużo mniejszy. W każdym wypadku komputer stara się blokować ruch wygrywający gracza. Jednakże, im większy rozmiar planszy, tym ocena najlepszego ruchu komputera jest niższa. To samo tyczy się zmiany ilości znaków w rzędzie potrzebnych do wygrania.

5. Wnioski:
   - Stosując obcięcia alfa – beta dla rozmiarów większych od 3, znacząco zmniejszamy ilość czasu potrzebnego komputerowi, w celu przeszukania wszystkich możliwości.
   - Wprowadzenie dynamicznej oceny głębokości przeszukiwania rozwiązało problem dla rozmiarów większych od 4.

- Program sprawdza czy indeksy nie wyszły poza rozmiar oraz sprawdza czy w danym miejscu nie stoi już jakiś znak. Jednak gdy wpiszemy coś innego niż liczba, to program się zatrzyma.
- Zastosowany algorytm bardzo dobrze sobie radzi na mniejszych rozmiarach. Jednakże wszystko jest podyktowane przez zastosowanie różnej głębokości w celu zmniejszenia czasu obliczeniowego.
- Dla mniejszych rozmiarów komputer jest praktycznie nie do pokonania.

6. Bibliografia:

    - GeeksForGeeks, *Minimax Algorithm in Game Theory - Set 4 (Alfa-Beta Prunning)*. Pobrano z: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-findingoptimal-move/
    - GeeksForGeeks, *Minimax Algorithm in Game Theory - Set 3 (Finding optimal move)*. Pobrano z: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-findingoptimal-move/
    - Wikipedia, *Algorytm Alfa-Beta*. Pobrano z: https://pl.wikipedia.org/wiki/Algorytm_alfa-beta