Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Kółko i Krzyżyk

Autor: Dariusz Litwiński
Prowadzący: Mgr Inż. Marek Marków
Rok akademicki: 2018/2019
Kierunek: Informatyka
Rodzaj studiów: SSI
Semestr: 1
Termin laboratorium: czwartek, 8:30 - 10:00
Sekcja: 10
Termin oddania sprawozdania: 2018-12-29

1.Treść zadania

Napisać program realizujący grę w Kółko i Krzyżyk, wraz ze sztuczną inteligencją odpowiadającą za ruchy komputera. Zanim rozgrywka się rozpocznie, losowana jest kolejność kolejek. Po każdej kolejce komputer sprawdza, czy rozgrywka powinna się zakończyć. Po zakończeniu gry, komputer podaje ile razy grę wygrał komputer a ile razy gracz. Program uruchamiany jest z linii poleceń wraz z parametrem odpowiadającym za wielkość planszy: 3 dla planszy 3x3, 5 dla planszy 5x5.

2.Analiza zadania

Zagadnienie przedstawia problem rozróżnienia i wybrania najlepszego ruchu dla komputera w danej sytuacji.

2.1 Struktury danych

W programie wykorzystano drzewo binarne do przechowywania stanów planszy dla danej wielkości planszy oraz listę jednokierunkową do przechowywania wskaźników na korzenie odpowiednich drzew. Zastosowanie drzewa binarnego jest możliwe dzięki przypisaniu wartości do danych pól na planszy (0- pole wolne, 1 – pole zajęte przez komputer, -1 – pole zajęte przez gracza), dzięki czemu uzyskujemy drzewo, w którym szybko jesteśmy w stanie odnaleźć dany stan gry.

2.2 Algorytmy

Program sortuje tablice stanów gry iterując po nich i w momencie, w którym natknie się na różne wartości decyduje, czy dana tablica powinna się znaleźć po lewej czy po prawej stronie drzewa binarnego. W momencie ruchu komputera symuluje się wszystkie możliwości, a następnie wybiera się tą możliwość, której węzeł ma przypisaną największą wartość. W przypadku, kiedy dwie możliwości są nieznane (nie ma ich w drzewie) lub mają taką samą wartość, losuje się, którą należy wybrać.

3.Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu parametr będący wielkością planszy, na której będzie odbywała się rozgrywka. W przypadku, kiedy podana liczba jest ujemna, parzysta lub przekazany parametr w ogóle nie jest liczbą program wypisze "Nieprawidlowy parametr". Program będzie poszukiwał plików o nazwie odpowiadającej parametrowi, który został podany, w przypadku, kiedy plik nie zostanie znaleziony, użytkownik zostanie o tym poinformowany.

4.Specyfikacja wewnętrzna

Program został podzielony na trzy pliki, jeden z deklaracjami i dwa z definicjami.

4.1 Ogólna struktura programu

W funkcji głównej po sprawdzeniu, czy parametr jest prawidłowy jest wykonywana funkcja singleGame(). Funkcja ta rysuje plansze o zadanej wielkości podanej parametrem za pomocą funkcji drawBoard(). Podczas

pierwszego uruchomienia funkcji drawBoard() inicjalizowana jest struktura Gamestate Game, w której to znajdują się m.in. tablice zawierające stan gry, historię ruchów, jakie zostały wykonane podczas pojedynczej gry oraz tablica która służy do rysowania planszy. Po narysowaniu planszy program próbuje pobrać z pliku informacje potrzebne do odtworzenia drzewa binarnego za pomocą funkcji readFile(). Następnie funkcja checkIfLoaded() sprawdza, czy drzewo binarne zostało poprawnie wczytane z pliku i jeśli nie, to samodzielnie je tworzy. Następnie zostaje losowane, kto ma rozpocząć rozgrywkę. Ruchy graczy odbywają się poprzez funkcje playerTurn() oraz CPUTurn(). Komputer wybiera pole, które ma wybrać za pomocą funkcji scoreCheck(), która to symuluje wszystkie możliwe ruchy w danym momencie i zwraca ten, który ma największą „wartość". Pola dążące do pozytywnych rezultatów (wygrana komputera lub remis) mają wartość powyżej 0, a przegrywające poniżej 0. Jeżeli dany stan gry jest nieznany, to zakładamy, że jego wartość wynosi 0. Po każdym ruchu gra sprawdza funkcją checkForEnd(), czy gra powinna się zakończyć. W przypadku, w którym gra się zakończyła, drzewo binarne zostaje zapisane do pliku w postaci wartości tablicy zawierającej stan gry wraz z jej wartością za pomocą funkcji writeFile(). Następnie funkcja scoreUpdate() odpowiednio zmienia wartości przypisane danym stanom planszy. Za pomocą tablicy moves odtwarzamy każdy stan gry, jaki pojawił się w trakcie rozgrywki i w przypadku pozytywnego rezultatu (wygrana, remis) wartość każdego stanu gry jest zwiększana o 1 w przypadku remisu i 3 w przypadku wygranej. W przypadku przegranej zmieniamy wartość jedynie ostatniego stanu gry (powodującego przegraną) oraz przedostatniego (zła decyzja umożliwiająca graczowi wygraną) na -1. Dzięki funkcji playerScoreUpdate() możemy wykorzystać strategie gracza do nauki naszej sztucznej inteligencji. Funkcja ta różni się od scoreUpdate() jedynie tym, że tutaj rezultatem pozytywnym jest wygrana gracza, a negatywnym wygrana komputera, oraz tym, że działa na tablicy zawierającej stany gry odwrotne do rzeczywistych (Pola zaznaczone przez gracza traktowane są jak te zaznaczone przez komputer i vice versa). Po tym użytkownik jest pytany, czy chce rewanżu, w przypadku, gdy chce, cała funkcja singleGame() jest powtarzana, w przeciwnym funkcja sessionSummary() wyświetla statystyki rozgrywki oraz usuwa wszelkie dynamicznie zaalokowane tablice/struktury.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załaczniku.

5.Testowanie

Program został przetestowany w wielu różnych scenariuszach, nieprawidłowe parametry, nieprawidłowe numery pól wybrane przez gracza, sytuacje, w których plik nie istnieje. Program zwróci błąd, kiedy wartości tablicy znajdujące się w pliku będą inne niż [-1,0,1].

6.Wnioski

O ile napisanie programu realizującego samą grę w kółko i krzyżyk było proste, tak zaimplementowanie sztucznej inteligencji okazało się nie lada wyzwaniem. Szczególnie trudne było zaimplementowanie tego algorytmu w taki sposób, aby komputer się nie „zrażał” do pewnych pól, pomimo, że nadal można je wybrać i wygrać/zremisować.