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

-Struktura boardTree będąca węzłami drzewa binarnego, zawiera:

-wskaźnik na lewy węzeł potomny (left)

-wskaźnik na prawy węzeł potomny (right)

-tablicę AI służącą do zapisu stanu planszy w danym momencie gry (AI)

-wartość danego stanu (score)

Struktura listOfRoots będąca elementami listy zawierającej korzenie drzew binarnych, zawiera:

-wskaźnik na następny element listy (pNext)

-wskaźnik na korzeń drzewa binarnego (root)

-długość planszy, dla której dane drzewo jest wykorzystywane (span)

Struktura Game służąca do przechowywania informacji o grze, struktura zawiera:

-tablice T przechowującą znaki poszczególnych graczy, jeśli zajęte oraz numery pól, gdy wolne.

-tablicę AI służącą do zapisu poszczególnych stanów planszy (0 – pole wolne, -1 pole zajęte przez gracza, 1 pole zajęte przez komputer).

-tablicę moves zapisującą w odpowiedniej kolejności pola wybrane przez graczy.

-flagę initialized służącą do rozróżniania, czy gra powinna zostać zainicjalizowana funkcją initializeGamestate().

-zmienną span służącą do przechowywania długości planszy w obecnej rozgrywce

-znak CPU przechowujący znak odpowiadający komputerowi

-znak Player przechowujący znak odpowiadający graczowi

-zmienną Winner, która przyjmuje wartość 0 w momencie, gdy rozgrywka trwa, 1, gdy gracz wygra, 2, gdy komputer oraz 3, gdy rozgrywka zakończyła się remisem

-zmienną Turn_Counter, w której przechowywana jest ilość wykonanych ruchów w danej rozgrywce

-zmienną Who_First, która zawiera informacje o tym, kto wykonał pierwszy ruch w rozgrywce. 0 dla gracza, 1 dla komputera.

-zmienną CPU_Wins, w której przechowywana jest ilość zwycięstw komputera w danej sesji

-zmienną Player_Wins, w której przechowywana jest ilość zwycięstw gracza w danej sesji

-znak Rematch służący do wyboru, czy gracz chcę rozegrać jeszcze jedną grę, czy chce zakończyć sesje

-wskaźnik list_header na pierwszy element listy korzeni drzew

-wskaźnik current_root na korzeń drzewa binarnego

Funkcja compareBoards() porównująca dwie tablice stanów planszy:

-parametr Game - Struktura gry, z której funkcja pobiera wielkość planszy oraz obecny stan tablicy AI

-parametr board - Tablica AI znajdująca się w danym węźle drzewa binarnego

-zwraca wartość -1,1,0 w zależności, czy tablica obecnego stanu gry mniejsza, większa lub równa

Funkcja findRoot() znajdujaca korzen drzewa dla danej wielkosci planszy:

-parametr header - Pierwszy wezel listy korzeni drzewa

-parametr span - Dlugosc planszy

-zwraca wskaznik na wezel listy zawierajacy wskaznik na odpowiedni korzen drzewa

Funkcja addRoot() dodajaca wezel zawierajacy korzen drzewa dla odpowiedniej wielkosci planszy:

-parametr header - Pierwszy wezel listy korzeni drzewa

-parametr span - Dlugosc planszy

-zwraca wskaznik na wezel listy zawierajacy wskaznik na odpowiedni korzen drzewa

Funkcja deleteList() usuwajaca liste zawierajaca korzenie drzewa:

-parametr header - Pierwszy wezel listy korzeni drzewa

Funkcja find() szukająca danego stanu planszy w drzewie binarnym, wykorzystująca funkcje compareBoards():

-parametr root - Korzeń drzewa, z którego gra korzysta

-parametr Game - Struktura gry, którą funkcja przekazuje funkcji compareBoards

-zwraca wskaźnik na węzeł drzewa zawierający daną tablice

Funkcja addToTree() dodająca dany stan planszy do drzewa binarnego, wykorzystująca funkcje compareBoards():

-parametr root - Korzeń drzewa, z którego gra korzysta

-parametr Game - Struktura gry, którą funkcja przekazuje funkcji compareBoards

-zwraca wskaźnik na korzeń drzewa

Funkcja readFile() sczytująca drzewo binarne gry z odpowiedniego pliku:

-parametr root - wskaźnik, do którego funkcja ma dołączyć sczytane drzewo

-parametr Game - Struktura gry, z której funkcja pobiera wielkość planszy

-parametr f - Strumień wejściowy, za pomocą którego odczytujemy

-zwraca wskaźnik na pierwszy węzeł, który został dołączony

Funkcja writeFile() zapisująca drzewo binarne gry do odpowiedniego pliku

-parametr root - Korzeń zapisywanego drzewa

-parametr f - Strumień wyjściowy, za pomocą którego zapisujemy

Funkcja scoreCheck() sprawdzająca wartość możliwych ruchów w danym momencie:

-param Game - Struktura gry, z której funkcja pobiera tablice AI

-zwraca numer pola, który w danym momencie jest najlepszy lub 0, gdy funkcja nie znajdzie odpowiedniego pola

Funkcja invertAI() tworząca tablice, w której ruchy gracza traktowane są jak ruchy komputera i vice versa w celu wykorzystania ruchów gracza do nauki:

-parametr Game - Struktura gry, z której funkcja pobiera tablice AI, która jest bazą do stworzenia nowej tablicy

-zwraca wskaźnik na nowa tablice

Funkcja playerScoreUpdate() wykorzystujaca ruchy gracza do nauki sztucznej inteligencji:

-parametr Game - Struktura gry, z której funkcja pobiera tablice moves, oraz ilość ruchów, po którym rozgrywka się zakończyła

-parametr board - Tablica stworzona przez funkcje invertAI

Funkcja scoreUpdate() zmieniająca wartość stanów gry, które wystąpiły w danej rozgrywce zgodnie z wynikiem (Wygrana +3, Remis +1, Przegrana -1). W przypadku przegranej jest to jedynie ostatni i przedostatni:

-parametr Game - Struktura gry, z której funkcja pobiera tablice AI, tablice moves, oraz ilość ruchów, po którym rozgrywka się zakończyła

Funkcja convertParam() sprawdzajaca, czy podany parametr jest prawidlowy i gdy jest prawidlowy zamieniajaca go na int:

-parametr input - parametr podany z linii komend

-zwraca przetworzony parametr

Funkcja initializeGamestate() inicjująca rozgrywkę, pyta gracza o parametry gry i tworzy potrzebne tablice:

-parametr Game - Niezainicjowana struktura gry

Funkcja playerTurn(), podczas której odbywa się ruch gracza:

-parametr Game - Struktura gry, do której zapisany będzie ruch gracza

-zwraca strukturę gry zaktualizowaną o ruch gracza

Funkcja CPUTurn(), podczas której odbywa się ruch komputera:

-parametr Game - Struktura gry, do której zapisany będzie ruch komputera

-zwraca strukturę gry zaktualizowaną o ruch komputera

Funkcja checkForEnd(), która sprawdza, czy gra powinna się zakończyć, jeśli tak, to ustawia zmienną Winner w strukturze gry na odpowiednią wartość (3 dla remisu):

-parametr Game Struktura gry, do której będzie zapisane to, czy gra powinna się skończyć

Funkcja checkForWin(), która sprawdza, czy gra w danym momencie posiada zwycięzce, jeśli tak, to ustawia zmienną Winner w strukturze gry na odpowiednią wartość (1 dla Gracza, 2 dla Komputera):

-parametr Game - Struktura gry, do której będzie zapisane to, czy gra powinna się skończyć

-zwraca true, jeśli gra ma zwycięzcę, false, jeśli nie

Funkcja drawBoard(), która rysuje pole gry:

-parametr Game - Struktura gry, z której funkcja będzie pobierać to, jaki znak powinien się pojawić, na danym polu

Funkcja singleGame(), w której zawarte są wszystkie funkcje potrzebne do rozegrania pojedynczej gry

-parametr Game - Struktura gry, na której program operuje

-zwraca strukturę gry w takim stanie, który pozwala na ponowne rozegranie gry

Funkcja sessionSummary(), która wypisuje statystyki gracza i komputera, oraz czyści wszystko, co było dynamicznie zaalokowane:

-parametr Game - Struktura gry, z której program uzyskuje dostęp do tablic i struktur do usunięcia

## 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. Kolejnym wyzwaniem, jakie napotkałem było zaimplementowanie mechanizmu pozwalającemu na naukę od gracza, jednak po przezwyciężeniu tych wszystkich problemów mogę śmiało przyznać, że ten projekt był cennym doświadczeniem, które na pewno zaowocuje w przyszłości.