Dokumentacja projektu z przedmiotu język przetwarzania symbolicznego

Temat projektu : Kropki z min-max w języku Scala

Autorzy : Ewa Kowalska, Dominik Orliński

# Uruchomienie

Projekt należy uruchomić w odpowiednim środowisku, kompilującym język Scala

(np. IntelliJ IDEA z wtyczką Scala).

# Zasady gry

* W grze gracze na zmianę stawiają na kwadratowej planszy kropki (u nas 'o' oraz 'x')
* Celem gry jest zdobycie większej liczby punktów niż przeciwnik
* Punkty zdobywa się poprzez otoczenie kropki/kropek przeciwnika swoimi kropkami
* Otoczone kropki (przeciwnika) wraz z kropkami otaczającymi (danego gracza) tworzą bazę (danego gracza)
* Nie można otoczyć, kropki która tworzy bazę
* Za każdą otoczoną kropkę gracze dostają 1 pkt
* Gra kończy się gdy na planszy nie można już postawić kropki
* Wygrywa gracz w większą liczbą punktów

# Sposób gry

* Gracz rozpoczyna rozgrywkę podając rozmiar planszy, na której chce się zagrać.
* Następnie na zmianę ruch podejmuje gracz i komputerowy bot.
* Gracz określa punkt w którym chce postawić kropkę (podając wiersz i kolumnę)
* Numery kolumn i wierszy są indeksowane od 1 do n (rozmiar planszy)
* Jeśli gracz ruszy się w niedozwolony punkt (np. zajęty albo poza planszą) musi ponownie wybrać ruch
* Wybrany punkt zostanie zaznaczony na planszy literą 'o'
* Następnie ruch wykonuje bot
* Wybrany przez bota ruch zostaje zaznaczony na planszy literą 'x'
* Jeśli ruch gracza lub bota utworzył na planszy bazę zostanie to zaznaczone
* Gra toczy się dopóki na planszy nie można już postawić kropki

# Omówienie algorytmu bota

Bot korzysta z algorytmu min-max z przycinaniem alfa-beta. Określamy głębokość do której bot będzie sprawdzał ruchy (D). Bot rozpoczyna swoje rozważania od sprawdzenia gdzie może postawić kropkę. Dla każdej z tych możliwości tworzy planszę - jest to głębokość 1. Z kolei dla każdej z tych wygenerowanych plansz bot rozważa każdy możliwy ruch gracza (swojego przeciwnika) - jest to głębokość 2. Proces powtarza się do głębokości D. Na głębokości D obliczana jest różnica punktów, które zgromadził bot i gracz (nazwijmy to wynikiem). Ta wartość jest zwracana przez algorytm min-max. Wybierany jest ten ruch, który doprowadził do planszy z największym wynikiem (czyli planszy, na której bot uzyskał największą przewagę). Przycinanie alfa-beta zapewnia odrzucenie gałęzi, które nie poprawią wyniku.

# Spostrzeżenia

Algorytm min-max nie jest algorytmem, który przynosi zadowalający efekt. Po pierwsze w grze w kropki liczba możliwych ruchów jest bardzo duża. Dla małych planszy np. 10 na 10 jest ich na początku rozgrywki prawie setka. Jeśli bot ma rozważać ruchy do głębokości n na planszy o boku k to liczba możliwych plansz które musi rozważyć na początku rozgrywki wynosi bez mała . Dodatkowo ocenianie ruchu (wyniku na planszy) też trwa stosunkowo długo, ponieważ postawienie jeden kropki może zmienić całkowicie plansze (np. domknąć jakąś dużą bazę), więc za każdym razem (dla każdej planszy) trzeba dla każdej kropki wywołać algorytm, który weryfikuje czy kropka jest otoczona czy nie (w naszym przypadku jest to algorytm na wzór algorytmu full-fill). To wszystko sprawia, że bot nawet na średnich planszach (5 do 10) dla D w granicach 4-5 decyzję podejmuje długo. Ponadto bot gra strategią, polegającą na jak najdłuższym utrzymywaniu stanu remisu. Bot nie podejmuje „walki”. Bot zakłada, że zarówno on jak i jego przeciwnik grają optymalnie, a to znaczy, że przeciwnik się nie pomyli. Gra w kropki ma tę wadę, że można w niej permanentnie utrzymywać remis – podążać za taką strategią kropka za kropkę tworzyć np. rząd kropek. Dlatego też algorytm ma do wybrania spośród dużej liczby możliwych ruchów, ale każdy z nich przynosi taki sam efekt. Aby algorytm min-max działał lepiej potrzebna byłaby modyfikacja zasad gry np. pozwolenie na ruch tylko w takie miejsce, które sąsiaduje z już zajętym przez przeciwnika polem (to w oczywisty sposób zmusza do „walki”).