### Zadanie

Zaimplementować algorytm min-max z przycinaniem alfa-beta. Algorytm ten należy zastosować do gry w proste warcaby (checekers/draughts). Niech funkcja oceny planszy zwraca różnicę pomiędzy stanem planszy gracza a stanem przeciwnika. Za pion przyznajemy 1 punkt, za damkę 10 p.

Przygotowałem dla Państwa kod, który powinien ułatwić wykonanie zadania. Zamiast mojego kodu, osoby chętne mogą napisać własny kod pomocniczy. Nie można używać kodu z Internetu, czy bardziej ogólnie, kodu, którego nie jest się autorem.

Wiem co jest dostępne w Internecie, większość dostępnych implementacji ma cechy szczególne, po których łatwo je rozpoznać.

Zasady gry (w skrócie: wszyscy ruszają się po 1 polu. Pionki tylko w kierunku wroga, damki w dowolnym) z następującymi modyfikacjami:

- bicie nie jest wymagane
- dozwolone jest tylko pojedyncze bicie (bez serii).

### Pytania (odpowiedzi proszę umieścić w dokumencie tekstowym)

- Czy gracz sterowany przez AI zachowuje się rozsądnie z ludzkiego punktu widzenia? Jeśli nie to co jest nie tak?

Niech komputer gra z komputerem (bez wizualizacji), zmieniamy parametry jednego z oponentów, badamy jak zmiany te wpłyną na liczbę jego wygranych. Należy zbadać wpływ:

- Głębokości drzewa przeszukiwań
- Alternatywnych funkcji oceny stanu, np.:
  1. nagrody jak w wersji podstawowej + nagroda za stopień zwartości grupy (jak wszyscy blisko siebie to OK, no chyba, że da się coś zabrać przeciwnikowi)
  2. za każdy pion na własnej połowie planszy otrzymuje się 5 nagrody, na połowie przeciwnika 7, a za każdą damkę 10.
  3. za każdy nasz pion otrzymuje się nagrodę w wysokości: (5 + numer wiersza, na którym stoi pion) (im jest bliżej wroga tym lepiej), a za każdą damkę: 10.

## ZAŁOŻENIA

W mojej pracy analizowałem 2 boty grające ze sobą w warcaby (tylko pojedyncze bicie, bez przymusu), które wykorzystywały algorytm minmax z przycinaniem alfa-beta. Każdy z botów mógł używać odrębnej funkcji oceny.

Funkcje oceny które analizowałem:

- evaluate() – 1pkt za pion, 10pkt za damkę

- evaluate2() – 5pkt za pion na własnej połowie, 7pkt za pion na połowie przeciwnika, 10pkt za damkę

- evaluate3() – 1pkt za pion + numer wiersza w którym się znajduje (liczony od 0), 10pkt za damkę

Każda z funkcji zwracała dodanie punkty dla niebieskiego gracza, oraz ujemne dla białego.

Każda z funkcji zwracała (+-) 999pkt gdy odpowiedni gracz wygrał

## WYNIKI

| Biały | | Niebieski | | |
|---|---|---|---|---|
| Funkcja oceny | Głębokość | Funkcja oceny | Głębokość | Rezultat |
| evaluate | 5 | evaluate | 5 | Biały |
| | | | | |
| evaluate | 6 | evaluate | 5 | Biały |
| evaluate | 4 | evaluate | 5 | Niebieski |
| evaluate | 3 | evaluate | 5 | Biały |
| evaluate | 2 | evaluate | 5 | Remis |
| evaluate | 1 | evaluate | 5 | Niebieski |
| | | | | |
| evaluate | 5 | evaluate | 6 | Niebieski |
| evaluate | 5 | evaluate | 4 | Biały |
| evaluate | 5 | evaluate | 3 | Biały |
| evaluate | 5 | evaluate | 2 | Biały |
| evaluate | 5 | evaluate | 1 | Biały |
| | | | | |
| evaluate | 5 | evaluate2 | 5 | Remis |
| evaluate2 | 5 | evaluate | 5 | Biały |
| evaluate2 | 5 | evaluate2 | 5 | Remis |
| evaluate | 5 | evaluate3 | 5 | Biały |
| evaluate3 | 5 | evaluate | 5 | Remis |
| evaluate3 | 5 | evaluate3 | 5 | Remis |

## WNIOSKI

Ten format warcabów jest dosyć specyficzny, bowiem posiadanie damki oznacza, że nie można przegrać, o ile będzie się wykonywało poprawne ruchy. Powodowało to, że boty poruszały się w dosyć nienaturalny sposób, na przykład nie używały damki do atakowania przeciwnika, tylko wykorzystywały ją do pominięcia ruchu i zmuszenia przeciwnika do kolejnego posunięcia.

Zasady te spowodowały, że duża część partii skończyła się remisem. Wydaję mi się również, że taki format faworyzuje gracza białego, który zaczyna.

Największy wpływ na rezultat gry zdaję się mieć funkcja oceny. Co z tego, że przeszukamy drzewo na kilku więcej poziomach, jeśli nasz algorytm nie potrafi określić czy dany ruch jest dobry. Jest to szczególnie widoczne w tych rezultatach:

| | | | | |
|---|---|---|---|---|
| evaluate | 6 | evaluate | 5 | Biały |
| evaluate | 4 | evaluate | 5 | Niebieski |
| evaluate | 3 | evaluate | 5 | Biały |
| evaluate | 2 | evaluate | 5 | Remis |
| evaluate | 1 | evaluate | 5 | Niebieski |

Biały remisuje a nawet wygrywa z niebieskim, pomimo znacznie mniejszej głębokości przeszukiwania.

Głębokość przeszukiwania ma znaczenie dopiero, gdy dobierzemy odpowiednią funkcję oceny. Analizując rozgrywki zauważyłem, że zmniejszając głębokość dla niebieskiego gracza, gracz biały coraz szybciej (w mniejszej ilości ruchów) pokonuje gracza niebieskiego. W tym wypadku funkcja oceny zdaje się być dobrze dobrana dla gracza który zaczyna i wówczas różnica w głębokości przeszukiwań jest widoczna. Warto również zaznaczyć, że zwiększenie głębokości o 1 powodowało znaczne zwiększenie czasu działania algorytmu.