Skład zespołu:
Mateusz Winnicki

WSI ĆWICZENIE 3

Cel ćwiczenia

Tematem trzecich ćwiczeń są dwuosobowe gry deterministyczne. Zadaniem do wykonania jest implementacja algorytmu budującego drzewo gry Reversi, a następnie grającego z algorytmem wykonującym ruchy w sposób losowy.

Przebieg ćwiczenia

Zbadamy wyniki pojedynków komputera z samym sobą, w których wybiera on kolejne ruchy w sposób określony przez jeden z dwóch silników – losowy lub bazujący na algorytmie Minimax.

Przeprowadzimy porównanie wyników oraz czasu wyszukiwania ruchu przez algorytm Minimax w zależności od głębokości drzewa gry. Czynnikiem decydującym o wartości danego ruchu będzie liczba dysków przewagi danego gracza nad przeciwnikiem, po jego wykonaniu.

Aby uruchomić program należy to zrobić z parametrami określającymi: długość boku planszy, głębokość drzewa gry, rodzaj silnika sterującego graczem czarnym (gracz pierwszy) oraz rodzaj silnika sterującego graczem drugim – [rd | mm].
Dodanie do linii *-t* włączy opcję stopera, który będzie wyświetlał jaki ruch i po ilu sekundach wybrał dany silnik.
Dodanie do linii *-db* włączy opcję wyświetlania drzew gry budowanych przez algorytm Minimax.

Przykład:
> *py .\reversi.py -s 8 -d 3 -be mm -we rd -t -db*

Wyniki ćwiczenia i wnioski

Żeby zweryfikować, że niezależnie od koloru gracza, silnik Minimax ma większą szansę zwycięstwa uruchomimy go 5 razy na miejscu gracza czarnego i 5 razy na miejscu gracza białego i sprawdzimy, który silnik zwyciężył. Przeprowadzimy to badanie dla głębokości równej 3 i planszy o standardowych rozmiarach.

Numer gry	Kolor gracza Minimax	Zwycięzca (wynik)
1	Biały	Biały (44:20)
2	Biały	Biały (35:29)
3	Biały	Biały (43:20)
4	Biały	Biały (32:31)
5	Biały	Biały (38:25)
6	Czarny	Biały (35:29)
7	Czarny	Czarny (13:3)
8	Czarny	Czarny (39:23)
9	Czarny	Czarny (32:30)
10	Czarny	Czarny (40:18)

Otrzymane wyniki wskazują na zdecydowaną przewagę algorytmu Minimax nad losowym doborem ruchów. Widać również, że gra nie faworyzuje żadnego z graczy (pojedyncze zwycięstwo algorytmu losowego na pozycji gracza Białego możemy uznać za losowy łut szczęścia, patrząc na jej zdecydowanie liczniejsze porażki).

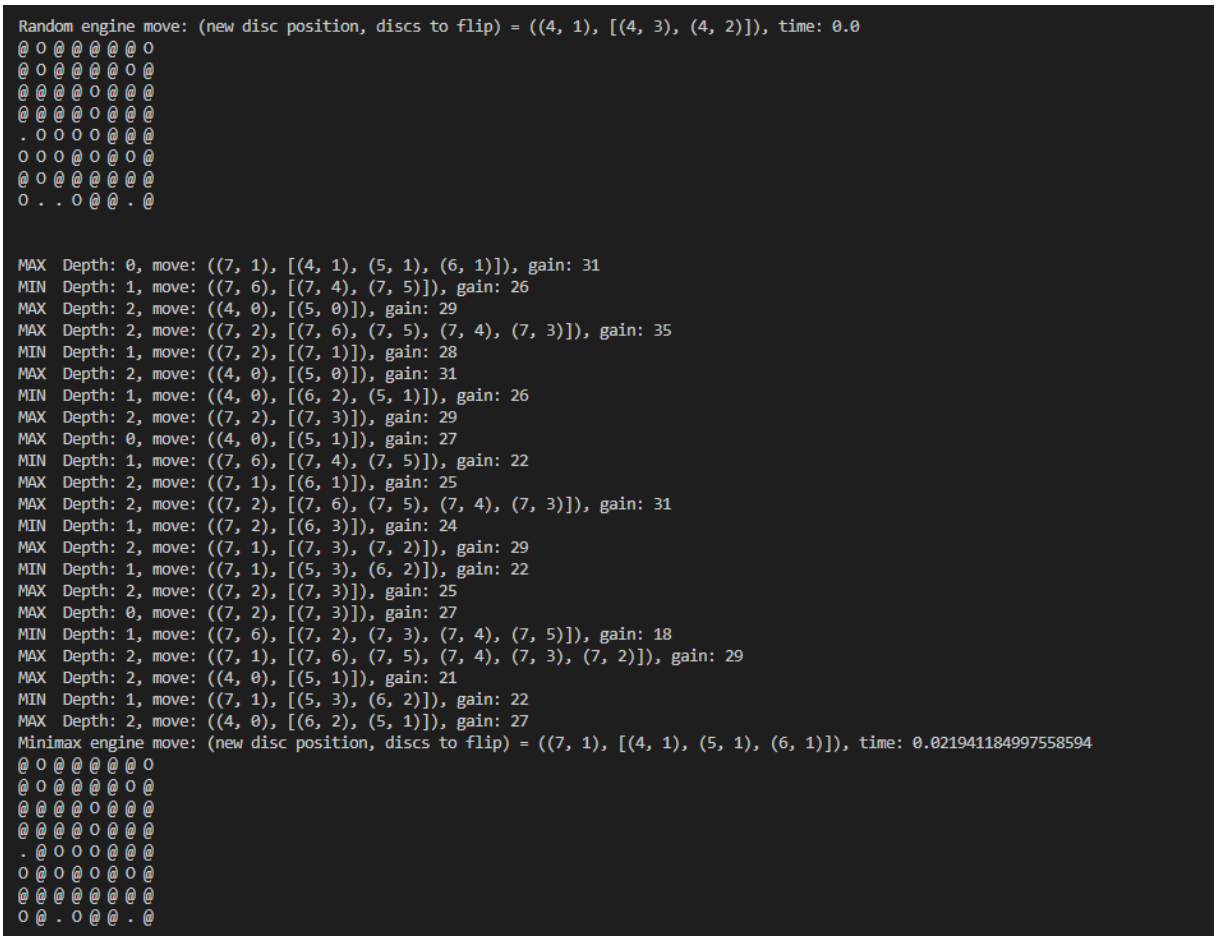
Zbadajmy wyniki przebiegu, w którym obaj gracze poruszają się w sposób nieprzypadkowy:

Wymiary planszy	8x8
Silnik gracza czarnego	Minimax
Silnik gracza białego	Minimax
Głębokość drzewa gry	3

Gra kończy się wynikiem 32:32 po wykonaniu 60 tur w trzech przebiegach pod rząd. Również po wyglądzie wzorów, w które układają się dyski graczy możemy zauważyć, że obydwie strony przyjmując tę samą taktykę nie są w stanie pokonać przeciwnika. Po powtórzeniu badania dla drzewa o głębokości 1 i 2 wyniki są takie same.

Przyjrzymy się teraz drzewu gry przy końcowych ruchach przebiegu o parametrach:

Wymiary planszy	8x8
Silnik gracza czarnego	Minimax
Silnik gracza białego	losowy
Głębokość drzewa gry	3



Rysunek 1 Drzewo gry przy wyborze przedostatniego ruchu dla gracza Minimax

Na powyższym rysunku widzimy stan wartości *gain* na kolejnych poziomach drzewa. W turze *MAX* algorytmowi zależy na jego zwiększeniu, a w turze *MIN* na jego zmniejszeniu.

Widzimy, że decydując się na ruch (7,1)[1] możemy zakończyć grę w najgorszym przypadku z wynikiem 29. Dla ruchu (4, 0) najgorszym możliwym wynikiem byłoby 25, a dla ruchu (7,2) – 21. Algorytm wybiera najlepszą z możliwości w bardzo krótkim czasie, ponieważ drzewo nie było zbyt skomplikowane.

[1] (7, 1) – 8. wiersz, 2. kolumna. Niestety logika tworzenia planszy jest odwrotna do układu kartezjańskiego, przez co szukamy współrzędnych dysków w inny sposób niż zwykle.

Następną rozgrywkę przeprowadzimy dla poniższych parametrów:

Wymiary planszy	8x8
Silnik gracza czarnego	Minimax
Silnik gracza białego	losowy
Głębokość drzewa gry	5

Sprawdźmy jak wzrost głębokości drzewa gry wpływa na czas wykonywania ruchu przez algorytm Minimax.

Po uruchomieniu programu z opcją stopera odkryjemy, że silnik losowy dokonuje wyborów w czasie zbliżonym do 0s, z kolei czas wyboru ruchu przez Minimax rośnie z każdą kolejną turą gry, by na koniec znowu zacząć spadać z racji coraz mniejszej liczby opcji, co jest jak najbardziej sensowne, ponieważ czas odkrycia najlepszego z ruchów bardzo szybko rośnie z liczbą możliwych do wykonania ruchów.

Tura gracza Minimax	**Czas wyboru najlepszego ruchu [s]**
1	0,30
2	0,93
3	1,72
4	3,42
5	18,81
6	26,34
7	21,81
8	68,26
9	38,54
10	111,04
11	158,30
12	272,28
…	…
24	14,11
25	8,74
26	1,73
27	0,58
28	0,11
29	0,02
30	0,002
SUMA	1873,82

Tablica wyników	
Wynik gracza losowego	21
Wynik gracza Minimax	43

Porównując średni czas wykonywania programu z pięciu pomiarów w zależności od maksymalnej głębokości drzewa gry otrzymujemy wyniki:

Maksymalna głębokość	**Średni czas trwania programu [s]**	**Odchylenie standardowe [s]**	**Minimalny czas [s]**	**Maksymalny czas [s]**	**Porażek Minimax**
1	0,41	0,05	0,33	0,48	1
2	1,51	0,75	0,21	2,45	0
3	7,08	2,96	1,37	7,35	1
4	177,94	58,13	81,38	231,37	0
5	1821,12	428,34	1211,24	2458,23	0

Zwiększanie parametru max_depth powyżej czterech wiąże się z bardzo dużym wzrostem czasu wykonywania programu. W przypadku starć z algorytmem losowym, niższe maksymalne głębokości algorytmu Minimax nie mają problemu ze znalezieniem zwycięskich strategii, jednak dla przeciwnika o większych możliwościach, niezbędne byłoby zastosowanie ulepszeń programu.

Mimo że niższe głębokości maksymalne są wystarczające, by pokonać algorytm losowy, możemy zauważyć większą stabilność wyników dla przebiegów programu z maksymalną głębokością równą 5. W żadnej z badanych prób, Minimax nie zakończył rozgrywki z wynikiem poniżej 40 dysków na planszy. Świadczy to o wzroście możliwości, który moglibyśmy wykorzystać z przeciwnikiem o lepszym kryterium ruchu niż losowe. W kontekście naszego ćwiczenia zwiększanie wartości max_depth jest niepotrzebnie dokładne, zważywszy na bardzo długi czas wykonywania.

Podsumowanie

Wyniki otrzymane w ćwiczeniu zdecydowanie świadczą o sensowności algorytmu Minimax, nawet jeśli jest to jego prostsza wersja, która nie wykorzystuje skomplikowanych ewaluacji heurystyki, wybierając kolejne ruchy. Poza modyfikacją funkcji wartościującej dany ruch, najbardziej oczywistym rozwinięciem badanego programu byłoby wykorzystanie omawianego na wykładzie i wspomnianego w treści zadania algorytmu alfa-beta. Zdecydowanie wpłynęłoby to na czas wykonywania programu, w szczególności dla głębokości max_depth >= 5, w których liczba możliwych sekwencji ruchów może rosnąć eksponencjalnie. Redukcja dużej liczby gałęzi na pewno nieopłacalnych, zdecydowanie usprawniłaby działanie programu.