# Sprawozdanie ćwiczenia nr 3 – Łukasz Szydlik

## Cel i opis eksperymentów

Celem ćwiczenia jest zaimplementowanie algorytmu minimax z obcinaniem $\alpha - \beta$ grający w kółko i krzyżyk. Następnie przeprowadzić symulacje gier próbując znaleźć jak najlepsze parametry algorytmu minimax oraz zbadać eksperymentalnie wpływ głębokości odcinania drzewa gry.

## Instrukcja programu

### Instalacja

1. Należy pobrać repozytorium

2. Wejść w folder lab3

3. Upewnić się, że posiadamy wymagane biblioteki:

‘pip install -r requirements.txt’

### Uruchomienie programu

W pliku config.json możemy ustawić interesujące parametry:

Typ gracza spośród - ”minimax”, ”random”, ”human”

"pruning_depth" - głębokość drzewa przeszukiwać dla gracza minimax, wartość wymagana przy wyborze gracza minimax

"gui": {true/false} – jeśli chcemy zagrać z interfejsem graficznym. W przypadku wartości false uruchomi się symulacja i nie jest możliwy wybór gracza ”human”.

"simulation_amount" – liczba wykonanych symulacji, w przypadku ustawienia gui na `false` wartość wymagana

Pamiętamy o tym aby program uruchamiać bezpośrednio z folderu lab3.

Program należy uruchomić za pomocą komendy:

`python main.py --config="config.json"`

Możemy wywołać main.py z dodatkowym parametrem:

--seed="{liczba_całkowita}" – ustawawi np.random.seed

Jeśli wartość gui jest ustawiona na true będziemy mogli rozegrać/przyjrzeć się rozgrywkę w kółko i krzyżyk, w przeciwnym wypadku wykona się symulacja a w konsoli pojawi się informacja ile razy poszczególny gracz wygrał oraz liczba remisów.

# Wyniki ćwiczenia

Po zaimplementowaniu algorytmu przeprowadzono następujące symulacje gier próbując znaleźć jak najlepsze parametry algorytmu minimax:

1. Gracz minimax vs gracz losowy
2. Gracz minimax vs gracz minimax

W tym celu gracz minimax podczas wyboru swojego ruchu najpierw generuje wszystkie następujące możliwości swojego ruchu. Następnie każda utworzona w ten sposób plansza jest przetwarzana przez algorytm alfa beta w celu ocenienia danego ruchu.

Przeprowadzono symulacje `gracz minimax vs gracz losowy` w celu oceny jakości algorytmu ze względu na głębokość poszukiwań oraz czy minimax był graczem rozpoczynającym.

| Rozpoczynający | Random |
|---|---|
| Głębokość | 8 |
| Wygrany | Ilość |
| Minimax | 94 |
| Random | 0 |
| Remis | 6 |
| Średni czas symulacji (s) | 0,49 |

| Rozpoczynający | Random |
|---|---|
| Głębokość | 5 |
| Wygrany | Ilość |
| Minimax | 89 |
| Random | 0 |
| Remis | 11 |
| Średni czas symulacji (s) | 0,25 |

| Rozpoczynający | Random |
|---|---|
| Głębokość | 2 |
| Wygrany | Ilość |
| Minimax | 87 |
| Random | 0 |
| Remis | 13 |
| Średni czas symulacji (s) | 0,15 |

| Rozpoczynający | Minimax |
|---|---|
| Głębokość | 8 |
| Wygrany | Ilość |
| Minimax | 99 |
| Random | 0 |
| Remis | 1 |
| Średni czas symulacji (s) | 0,78 |

| Rozpoczynający | Minimax |
|---|---|
| Głębokość | 5 |
| Wygrany | Ilość |
| Minimax | 97 |
| Random | 0 |
| Remis | 3 |
| Średni czas symulacji (s) | 0,39 |

| Rozpoczynający | Minimax |
|---|---|
| Głębokość | 2 |
| Wygrany | Ilość |
| Minimax | 96 |
| Random | 0 |
| Remis | 4 |
| Średni czas symulacji (s) | 0,15 |

| Rozpoczynający | Na przemian |
|---|---|
| Głębokość | 1 |
| Wygrany | Ilość |
| Minimax | 944 |
| Random | 0 |
| Remis | 56 |
| Średni czas symulacji (s) | 0,12 |

| Rozpoczynający | Na przemian |
|---|---|
| Głębokość | 0 |
| Wygrany | Ilość |
| Minimax | 947 |
| Random | 13 |
| Remis | 40 |
| Średni czas symulacji (s) | 0,11 |

Jak można zauważyć wraz ze wzrostem głębokości poszukiwań rósł czas pojedynczej symulacji natomiast wyniki były minimalnie lepsze. Zależność, że gracz minimax nie był graczem rozpoczynającym wpływało na lepsze wyniki gracza random.

Wyniki kolejnej symulacji `gracz minimax vs gracz minimax`:

| Rozpoczynający | Na przemian |
|---|---|
| Głębokość1 | 5 |
| Głębokość2 | 5 |
| Wygrany | Ilość |
| Minimax1 | 0 |
| Minimax2 | 0 |
| Remis | 100 |

| Rozpoczynający | Na przemian |
|---|---|
| Głębokość1 | 8 |
| Głębokość2 | 0 |
| Wygrany | Ilość |
| Minimax1 | 0 |
| Minimax2 | 0 |
| Remis | 100 |

Przeprowadzone eksperymenty ukazują, że gracz minimax nie wygra z graczem minimax bez względu na głębokość przeszukiwań.

# Wnioski

Jak mogliśmy zauważyć w wynikach eksperymentu głębokość przeszukiwań w takiej prostej grze jak kółko i krzyżyk nie pełni wielkiej roli w porównaniu ze skuteczną funkcją heurystyczną celu. Jak mogliśmy zauważyć dopiero głębokość równa 0, czyli ocena wyłącznie następnego ruchu pozwala graczowi random wygrać rozgrywkę. Natomiast ciekawą sytuacją jest że gracz minimax o głębokości 8 nie może wygrać z graczem minimax o głębokości 0. Jest to spowodowane tym, że algorytm minimax o głębokości 8 zakłada, że gracz minimax o głębokości 0 będzie grał optymalnie, co w każdej sytuacji prowadziłoby do remisu i decyduje się na pierwszy z listy ruchów o wartości 0 (remis), natomiast może to ten drugi ruch również o wartości 0 mógłby zapewnić wygraną w przypadku nie wybrania przez gracza minimax o głębokości 0 nieoptymalnej ścieżki o głębokości 8. Największy wpływ na to ma jednakowa funkcja heurystyczna. Jeśli dwaj gracze minimax mieliby różną tą funkcję to dla małych głębokości jeden z graczy miałby przewagę i mógłby wygrać.