1. Wstęp

Cel zadania:
Celem laboratorium jest zapoznanie się z algorytmem alfa-beta przycinania, stosowanym w grach dwuosobowych, na przykładzie gry w monety. W omawianej grze gracze na przemian pobierają 1, 2 lub 3 monety ze stosu. Gracz który pobierze ostatnią monetę przegrywa.

2. Opis algorytmu

Algorytm alfa-beta przycinania służy do ograniczenia liczby przeszukiwanych węzłów w drzewie gry – dzięki niemu nie sprawdzamy gałęzi, które nie mają wpływu na końcowy wynik. W naszym przypadku:

- **Węzły MAX (gracz A)** wybierają ruch, który maksymalizuje wartość funkcji oceny.
- **Węzły MIN (gracz B)** wybierają ruch, który minimalizuje wartość funkcji oceny.
- Parametr **α (alpha)** jest aktualizowany tylko w węzłach MAX, a **β (beta)** tylko w węzłach MIN.
- Gdy spełniony jest warunek $\alpha \geq \beta$, dalsze przeszukiwanie danego węzła zostaje przerwane – następuje przycięcie.

W grze w monety przyjęto następujące wartości końcowe:

- **Wygrana (win):** +100
- **Przegrana (loss):** -100

3. Omówienie implementacji

Globalne stałe i zmienne:

- win = +100 oraz loss = -100 – definiują końcowe oceny gry.
- node_count – liczy, ile węzłów zostało wygenerowanych podczas przeszukiwania.
- node_order – przechowuje tekstową wizualizację drzewa (kolejność odwiedzanych węzłów i zmiany parametrów).
- prune_events – zapisuje momenty, w których nastąpiło przycięcie gałęzi.

Funkcja log_node(label, depth, alpha, beta):
Ta funkcja dodaje do listy node_order informację o bieżącym węźle (np. nazwę, stan, aktualne wartości α i β). Wcięcia odpowiadają głębokości, dzięki czemu drzewo jest czytelnie przedstawione.

Funkcje alpha_beta_max i alpha_beta_min:

- Sprawdzają, czy osiągnięto stan terminalny (stan równy 1 – oznacza, że gracz musi pobrać ostatnią monetę i przegrywa).

- Dla każdego możliwego ruchu (pobranie 1, 2 lub 3 monet) obliczają wartość nowego stanu.
- Aktualizują parametry α lub β, logują zmiany, a jeśli warunek przycięcia (α ≥ β) zostanie spełniony, przerywają dalsze przeszukiwanie.
- Na końcu zwracają ocenę dla danego węzła.

Funkcja main():

Funkcja main() resetuje globalne zmienne, ustala stan początkowy (w naszym przypadku 5 monet) oraz początkowe wartości α i β. Następnie, dla każdego możliwego ruchu gracza A w korzeniu drzewa, wywołuje rekurencyjnie procedury, zbiera oceny i wyznacza najlepszy ruch. Na końcu wyświetla:

- Wizualizację drzewa (lista node_order).
- Zdarzenia przycięcia (lista prune_events).
- Liczbę wygenerowanych węzłów.
- Oceny poszczególnych ruchów oraz uzasadnienie wyboru najlepszego.

4. Wizualizacja drzewa przeszukiwania i analiza wyników

Wyniki zapisane w pliku "wyniki.txt":

Interpretacja wyników:

- **Stan początkowy:**
  Gra zaczyna się od 5 monet.
- **Wizualizacja drzewa:**
  Drzewo przedstawiono tekstowo – w korzeniu rozpatrywane są trzy ruchy (pobranie 1, 2 lub 3 monet). Każdy ruch analizowany jest rekurencyjnie, a na kolejnych poziomach widoczne są zmiany stanu oraz aktualizacje wartości α i β. W niektórych węzłach, gdy spełniony jest warunek przycięcia, dalsze ruchy są pomijane.
- **Zmiany parametrów α i β oraz przycięcia:**
  W węzłach MAX aktualizowany jest parametr α, a w MIN – β. Przycięcia występują, gdy α jest większe lub równe β. Przykładowo, w węźle MIN (state=2) po ruchu 1, gdy wartość wynosi -100, następuje przycięcie, co widać w logu.
- **Oceny ruchów i wybór najlepszego:**
  Wszystkie ruchy w korzeniu dały wartość -100, co oznacza, że przy optymalnej grze przeciwnika gracz A nie ma gwarancji wygranej. Dlatego najlepszym ruchem, minimalizującym ryzyko porażki, jest pobranie 1 monety.
- **Liczba węzłów:**
  Algorytm wygenerował 7 węzłów, co potwierdza efektywność przycinania alfa-beta.