Teoria gier i dwuosobowe gry z doskonałą informacją

dr inż. Piotr Beling

Uniwersytet Łódzki

2021 (ostatnie poprawki: 2022)

http://pbeling.w8.pl

Podstawowe pojęcia teorii gier

- **teoria gier** – dział matematyki badający optymalne zachowania w przypadku konfliktu interesów;
- **gra** – sytuacja konfliktu lub jej matematyczny model;
- **gracz** – uczestnik gry;
- strategie graczy wpływają na przebieg gry (lub prawdopodobieństwa poszczególnych przebiegów), aż do jej końca;
- po zakończeniu gry każdy gracz otrzymuje **wypłatę**, która jest miarą jego zwycięstwa.

Podstawowe pojęcia teorii gier

- **teoria gier** – dział matematyki badający optymalne zachowania w przypadku konfliktu interesów;
- **gra** – sytuacja konfliktu lub jej matematyczny model;
- **gracz** – uczestnik gry;
- strategie graczy wpływają na przebieg gry (lub prawdopodobieństwa poszczególnych przebiegów), aż do jej końca;
- po zakończeniu gry każdy gracz otrzymuje **wypłatę**, która jest miarą jego zwycięstwa.

Podstawowe pojęcia teorii gier

- **teoria gier** – dział matematyki badający optymalne zachowania w przypadku konfliktu interesów;
- **gra** – sytuacja konfliktu lub jej matematyczny model;
- **gracz** – uczestnik gry;
- strategie graczy wpływają na przebieg gry (lub prawdopodobieństwa poszczególnych przebiegów), aż do jej końca;
- po zakończeniu gry każdy gracz otrzymuje **wypłatę**, która jest miarą jego zwycięstwa.

Podstawowe pojęcia teorii gier

- **teoria gier** – dział matematyki badający optymalne zachowania w przypadku konfliktu interesów;
- **gra** – sytuacja konfliktu lub jej matematyczny model;
- **gracz** – uczestnik gry;
- strategie graczy wpływają na przebieg gry (lub prawdopodobieństwa poszczególnych przebiegów), aż do jej końca;
- po zakończeniu gry każdy gracz otrzymuje **wypłatę**, która jest miarą jego zwycięstwa.

Podstawowe pojęcia teorii gier

- **teoria gier** – dział matematyki badający optymalne zachowania w przypadku konfliktu interesów;
- **gra** – sytuacja konfliktu lub jej matematyczny model;
- **gracz** – uczestnik gry;
- strategie graczy wpływają na przebieg gry (lub prawdopodobieństwa poszczególnych przebiegów), aż do jej końca;
- po zakończeniu gry każdy gracz otrzymuje **wypłatę**, która jest miarą jego zwycięstwa.

Strategia i równowaga Nasha

- **Strategia czysta** gracza to przyporządkowanie jego decyzji do każdej pozycji gry w której może się on znaleźć.
- **Strategia mieszana** (lub **zrandomizowana**) gracza to rozkład pokazujący prawdopodobieństwa użycia przez niego poszczególnych, dostępnych mu strategii czystych.
- **Równowaga Nasha** to taki wybór strategii poszczególnych graczy, że jeśli dokładnie jeden, dowolny gracz zmieni przypisaną mu strategię, to jego oczekiwana wypłata nie wzrośnie.
- Słynne **twierdzenie Nasha** mówi, że dla każdej, skończonej gry istnieje co najmniej jedna równowaga w strategiach mieszanych.

Strategia i równowaga Nasha

- **Strategia czysta** gracza to przyporządkowanie jego decyzji do każdej pozycji gry w której może się on znaleźć.
- **Strategia mieszana** (lub **zrandomizowana**) gracza to rozkład pokazujący prawdopodobieństwa użycia przez niego poszczególnych, dostępnych mu strategii czystych.
- **Równowaga Nasha** to taki wybór strategii poszczególnych graczy, że jeśli dokładnie jeden, dowolny gracz zmieni przypisaną mu strategię, to jego oczekiwana wypłata nie wzrośnie.
- Słynne **twierdzenie Nasha** mówi, że dla każdej, skończonej gry istnieje co najmniej jedna równowaga w strategiach mieszanych.

Strategia i równowaga Nasha

- **Strategia czysta** gracza to przyporządkowanie jego decyzji do każdej pozycji gry w której może się on znaleźć.
- **Strategia mieszana** (lub **zrandomizowana**) gracza to rozkład pokazujący prawdopodobieństwa użycia przez niego poszczególnych, dostępnych mu strategii czystych.
- **Równowaga Nasha** to taki wybór strategii poszczególnych graczy, że jeśli dokładnie jeden, dowolny gracz zmieni przypisaną mu strategię, to jego oczekiwana wypłata nie wzrośnie.
- Słynne **twierdzenie Nasha** mówi, że dla każdej, skończonej gry istnieje co najmniej jedna równowaga w strategiach mieszanych.

Strategia i równowaga Nasha

- **Strategia czysta** gracza to przyporządkowanie jego decyzji do każdej pozycji gry w której może się on znaleźć.
- **Strategia mieszana** (lub **zrandomizowana**) gracza to rozkład pokazujący prawdopodobieństwa użycia przez niego poszczególnych, dostępnych mu strategii czystych.
- **Równowaga Nasha** to taki wybór strategii poszczególnych graczy, że jeśli dokładnie jeden, dowolny gracz zmieni przypisaną mu strategię, to jego oczekiwana wypłata nie wzrośnie.
- Słynne **twierdzenie Nasha** mówi, że dla każdej, skończonej gry istnieje co najmniej jedna równowaga w strategiach mieszanych.

Gry dzielimy według różnych kryteriów, m.in. na:

- **dwuosobowe**
- **wieloosobowe** (powyżej dwóch graczy)

pojedynczy gracz może modelować drużynę rzeczywistych graczy, np. parę w brydżu

- **o sumie stałej** (lub **zerowej**)

suma wypłat wszystkich graczy jest stała, niezależna od przebiegu gry; gdy wynosi 0, mówimy o grach o sumie zerowej; ponieważ grę o sumie stałej można łatwo sprowadzić do gry o sumie zerowej, to pojęcia te często są utożsamiane

- **o sumie zmiennej**

- **deterministyczne**

bez czynników losowych

- **niedeterministyczne (stochastyczne)**

np. z rzutem kostką, rozdawaniem kart

- **z doskonałą informacją**

wszyscy gracze posiadają doskonałą informację, tj. dokładnie obserwują i zapamiętują wszystkie zagrania, więc w każdej chwili w pełni znają stan gry

- **bez doskonałej informacji**

np. karciane, w których rozdawanie kart nie jest w pełni zaobserwowane przez graczy; nie znają oni pełnego stanu gry, bo np. widzą tylko swoje karty

Gry dzielimy według różnych kryteriów, m.in. na:

- **dwuosobowe**
- **wieloosobowe** (powyżej dwóch graczy)

pojedynczy gracz może modelować drużynę rzeczywistych graczy, np. parę w brydżu

- **o sumie stałej** (lub **zerowej**)
- **o sumie zmiennej**

suma wypłat wszystkich graczy jest stała, niezależna od przebiegu gry; gdy wynosi 0, mówimy o grach o sumie zerowej; ponieważ grę o sumie stałej można łatwo sprowadzić do gry o sumie zerowej, to pojęcia te często są utożsamiane

- **deterministyczne**

bez czynników losowych

- **niedeterministyczne (stochastyczne)**

np. z rzutem kostką, rozdawaniem kart

- **z doskonałą informacją**

wszyscy gracze posiadają doskonałą informację, tj. dokładnie obserwują i zapamiętują wszystkie zagrania, więc w każdej chwili w pełni znają stan gry

- **bez doskonałej informacji**

np. karciane, w których rozdawanie kart nie jest w pełni zaobserwowane przez graczy; nie znają oni pełnego stanu gry, bo np. widzą tylko swoje karty

Gry dzielimy według różnych kryteriów, m.in. na:

- **dwuosobowe**
- **wieloosobowe** (powyżej dwóch graczy)

pojedynczy gracz może modelować drużynę rzeczywistych graczy, np. parę w brydżu

- **o sumie stałej** (lub **zerowej**)

suma wypłat wszystkich graczy jest stała, niezależna od przebiegu gry; gdy wynosi 0, mówimy o grach o sumie zerowej; ponieważ grę o sumie stałej można łatwo sprowadzić do gry o sumie zerowej, to pojęcia te często są utożsamiane

- **o sumie zmiennej**

- **deterministyczne**

bez czynników losowych

- **niedeterministyczne (stochastyczne)**

np. z rzutem kostką, rozdawaniem kart

- **z doskonałą informacją**

wszyscy gracze posiadają doskonałą informację, tj. dokładnie obserwują i zapamiętują wszystkie zagrania, więc w każdej chwili w pełni znają stan gry

- **bez doskonałej informacji**

np. karciane, w których rozdawanie kart nie jest w pełni zaobserwowane przez graczy; nie znają oni pełnego stanu gry, bo np. widzą tylko swoje karty

Gry dzielimy według różnych kryteriów, m.in. na:

- **dwuosobowe**
- **wieloosobowe** (powyżej dwóch graczy)

pojedynczy gracz może modelować drużynę rzeczywistych graczy, np. parę w brydżu

- **o sumie stałej** (lub **zerowej**)
- **o sumie zmiennej**

suma wypłat wszystkich graczy jest stała, niezależna od przebiegu gry; gdy wynosi 0, mówimy o grach o sumie zerowej; ponieważ grę o sumie stałej można łatwo sprowadzić do gry o sumie zerowej, to pojęcia te często są utożsamiane

- **deterministyczne**

bez czynników losowych

- **niedeterministyczne (stochastyczne)**

np. z rzutem kostką, rozdawaniem kart

- **z doskonałą informacją**

wszyscy gracze posiadają doskonałą informację, tj. dokładnie obserwują i zapamiętują wszystkie zagrania, więc w każdej chwili w pełni znają stan gry

- **bez doskonałej informacji**

np. karciane, w których rozdawanie kart nie jest w pełni zaobserwowane przez graczy; nie znają oni pełnego stanu gry, bo np. widzą tylko swoje karty

Cechy typowych modeli dla wybranych, popularnych gier logicznych

Modele	Deterministyczne	Niedeterministyczne
Z doskonałą informacją	szachy, warcaby, go, czwórki, kółko i krzyżyk, gomoku	backgammon, monopoly, chińczyk
Bez doskonałej informacji	statki, zgadywanie w której ręce?	brydż, poker, skat, scrabble

Obszar zainteresowań

Dalej będziemy zajmowali się grami:

- dwuosobowymi,
- o stałej sumie wypłat,
- deterministycznymi,
- skończonymi,
- z doskonałą informacją.

Przykłady:

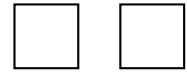
szachy, warcaby, kółko i krzyżyk, GO

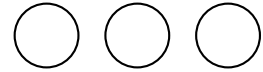
Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \mathrm{ev})$, taka że:

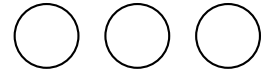
- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników; $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$; tj. brak cykli powracających do pozycji, które już były;
- $\mathrm{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\mathrm{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, Max, dla gry zakończonej w s; wypłata drugiego gracza, Min, to $1 - \mathrm{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\mathrm{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\mathrm{ev}(s)$, jest typu Max lub Min).

Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \mathrm{ev})$, taka że:

- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników; $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$; tj. brak cykli powracających do pozycji, które już były;
- $\mathrm{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\mathrm{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, Max, dla gry zakończonej w s; wypłata drugiego gracza, Min, to $1 - \mathrm{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\mathrm{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\mathrm{ev}(s)$, jest typu Max lub Min).

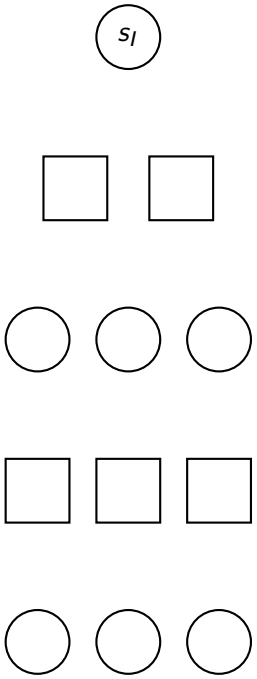






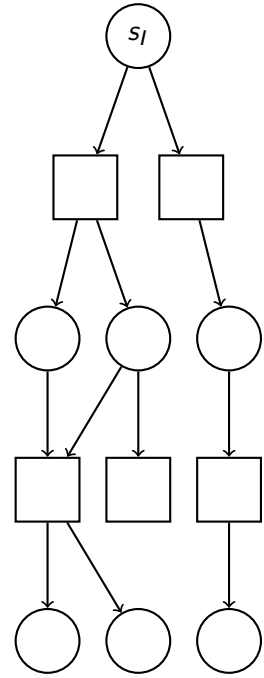
Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \mathrm{ev})$, taka że:

- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników;
  $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$;
  tj. brak cykli powracających do pozycji, które już były;
- $\mathrm{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\mathrm{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, Max, dla gry zakończonej w s; wypłata drugiego gracza, Min, to $1 - \mathrm{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\mathrm{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\mathrm{ev}(s)$, jest typu Max lub Min).



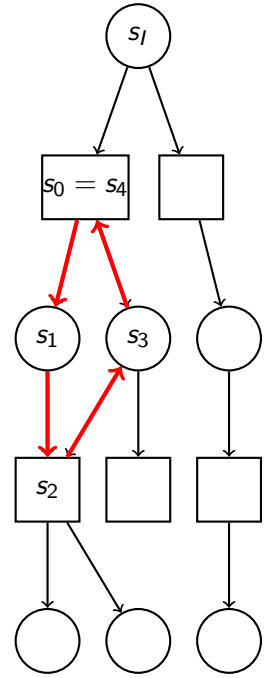
Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \text{ev})$, taka że:

- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników;
  $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$;
  tj. brak cykli powracających do pozycji, które już były;
- $\text{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\text{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, Max, dla gry zakończonej w s; wypłata drugiego gracza, Min, to $1 - \text{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\text{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\text{ev}(s)$, jest typu Max lub Min).



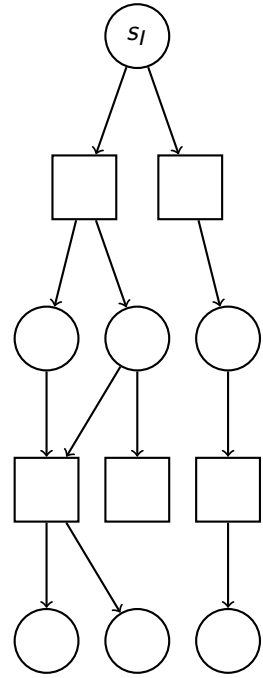
Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \mathrm{ev})$, taka że:

- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników;
  $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$;
  tj. brak cykli powracających do pozycji, które już były;
- $\mathrm{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\mathrm{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, *Max*, dla gry zakończonej w s; wypłata drugiego gracza, *Min*, to $1 - \mathrm{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\mathrm{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\mathrm{ev}(s)$, jest typu *Max* lub *Min*).



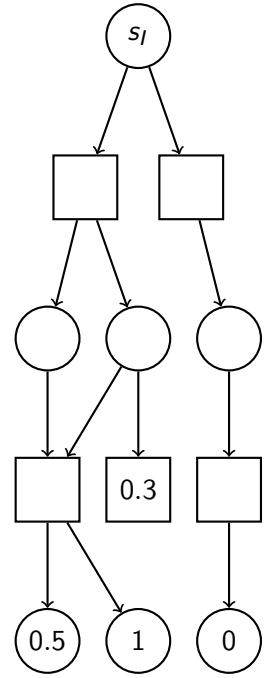
Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \text{ev})$, taka że:

- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników;
  $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$;
  tj. brak cykli powracających do pozycji, które już były;
- $\text{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\text{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, *Max*, dla gry zakończonej w s; wypłata drugiego gracza, *Min*, to $1 - \text{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\text{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\text{ev}(s)$, jest typu *Max* lub *Min*).



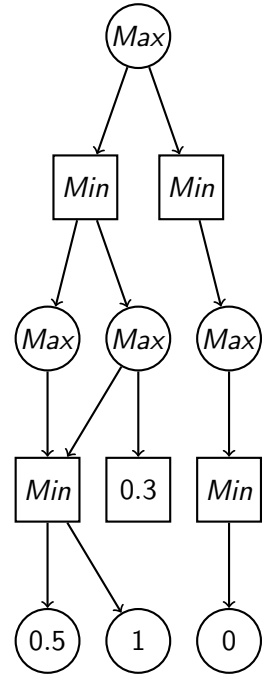
Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \mathrm{ev})$, taka że:

- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników;
  $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$;
  tj. brak cykli powracających do pozycji, które już były;
- $\mathrm{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\mathrm{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, *Max*, dla gry zakończonej w s; wypłata drugiego gracza, *Min*, to $1 - \mathrm{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\mathrm{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\mathrm{ev}(s)$, jest typu *Max* lub *Min*).



Gra dwuosobowa, skończona, deterministyczna, z doskonałą informacją, o sumie stałej (dalej zwana grą) to czwórka $(\mathbb{S}, s_I, N, \text{ev})$, taka że:

- $\mathbb{S}$ to skończony zbiór pozycji (stanów) gry;
- $s_I \in \mathbb{S}$ to pozycja początkowa, od której gra się zaczyna;
- $N : \mathbb{S} \to 2^{\mathbb{S}}$ to funkcja następników;
  $N(s)$ to zbiór następników $s \in \mathbb{S}$, tj. pozycji osiągalnych z s poprzez wykonanie jednego ruchu;
- nie istnieje ciąg pozycji $s_1, \ldots, s_n$, gdzie $n > 1$ i $s_k \in N(s_{k-1})$ dla $k = 2, \ldots, n$, taki że $s_1 = s_n$;
  tj. brak cykli powracających do pozycji, które już były;
- $\text{ev} : \mathbb{S} \to \{Max, Min\} \cup [0, 1]$ to funkcja wypłaty, taka że:
- gdy $N(s) = \emptyset$ (s jest końcowy/terminalny) to $\text{ev}(s) \in [0, 1]$ i oznacza wypłatę pierwszego gracza, *Max*, dla gry zakończonej w s; wypłata drugiego gracza, *Min*, to $1 - \text{ev}(s)$, zaś suma wypłat to 1;
- gdy $N(s) \neq \emptyset$ (s jest decyzyjny) to $\text{ev}(s) \in \{Max, Min\}$ i wskazuje kto idzie w s (o s mówimy też że, zależnie od wartości $\text{ev}(s)$, jest typu *Max* lub *Min*).



Uwagi do definicji gry

- $\mathbb{S}$ z N opisują acykliczny graf skierowany: $\mathbb{S}$ – wierzchołki; N – sąsiadów.
- Przedział wypłat $[0, 1]$ należy traktować symbolicznie, często używany jest inny.
- Z założenia o braku cykli i skończoności $\mathbb{S}$ wynika skończoność każdej rozgrywki i, następnie, warunek stopu wielu algorytmów, w tym omawianego dalej Min-Max.
- Założenie o braku cykli nie wyklucza modelowania rzeczywistych gier w których rozgrywka może przebiegać wielokrotnie przez taką samą pozycję.
- Powtórzenie rzeczywistej pozycji można zamodelować stanem terminalnym o wypłacie, zazwyczaj odpowiadającej remisowi, na którą gracze zgodzą się by uniknąć nieskończonej rozgrywki.
- Taką ideę realizują zresztą zasady wielu gier, np. zgodnie z regułami Międzynarodowej Federacji Szachowej (FIDE), gracz może zdecydować o zakończeniu gry remisem, gdy identyczna pozycja pojawi się na szachownicy przynajmniej po raz trzeci.

Uwagi do definicji gry

- $\mathbb{S}$ z N opisują acykliczny graf skierowany: $\mathbb{S}$ – wierzchołki; N – sąsiadów.
- Przedział wypłat $[0, 1]$ należy traktować symbolicznie, często używany jest inny.
- Z założenia o braku cykli i skończoności $\mathbb{S}$ wynika skończoność każdej rozgrywki i, następnie, warunek stopu wielu algorytmów, w tym omawianego dalej Min-Max.
- Założenie o braku cykli nie wyklucza modelowania rzeczywistych gier w których rozgrywka może przebiegać wielokrotnie przez taką samą pozycję.
- Powtórzenie rzeczywistej pozycji można zamodelować stanem terminalnym o wypłacie, zazwyczaj odpowiadającej remisowi, na którą gracze zgodzą się by uniknąć nieskończonej rozgrywki.
- Taką ideę realizują zresztą zasady wielu gier, np. zgodnie z regułami Międzynarodowej Federacji Szachowej (FIDE), gracz może zdecydować o zakończeniu gry remisem, gdy identyczna pozycja pojawi się na szachownicy przynajmniej po raz trzeci.

Uwagi do definicji gry

- $\mathbb{S}$ z N opisują acykliczny graf skierowany: $\mathbb{S}$ – wierzchołki; N – sąsiadów.
- Przedział wypłat $[0, 1]$ należy traktować symbolicznie, często używany jest inny.
- Z założenia o braku cykli i skończoności $\mathbb{S}$ wynika skończoność każdej rozgrywki i, następnie, warunek stopu wielu algorytmów, w tym omawianego dalej Min-Max.
- Założenie o braku cykli nie wyklucza modelowania rzeczywistych gier w których rozgrywka może przebiegać wielokrotnie przez taką samą pozycję.
- Powtórzenie rzeczywistej pozycji można zamodelować stanem terminalnym o wypłacie, zazwyczaj odpowiadającej remisowi, na którą gracze zgodzą się by uniknąć nieskończonej rozgrywki.
- Taką ideę realizują zresztą zasady wielu gier, np. zgodnie z regułami Międzynarodowej Federacji Szachowej (FIDE), gracz może zdecydować o zakończeniu gry remisem, gdy identyczna pozycja pojawi się na szachownicy przynajmniej po raz trzeci.

Uwagi do definicji gry

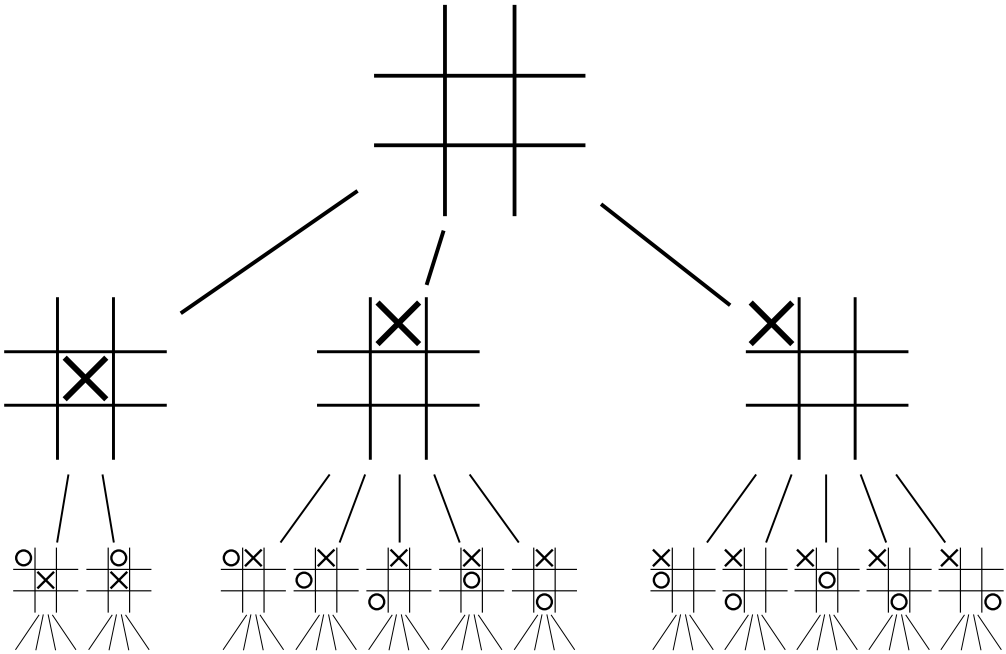
- $\mathbb{S}$ z N opisują acykliczny graf skierowany: $\mathbb{S}$ – wierzchołki; N – sąsiadów.
- Przedział wypłat $[0, 1]$ należy traktować symbolicznie, często używany jest inny.
- Z założenia o braku cykli i skończoności $\mathbb{S}$ wynika skończoność każdej rozgrywki i, następnie, warunek stopu wielu algorytmów, w tym omawianego dalej Min-Max.
- Założenie o braku cykli nie wyklucza modelowania rzeczywistych gier w których rozgrywka może przebiegać wielokrotnie przez taką samą pozycję.
- Powtórzenie rzeczywistej pozycji można zamodelować stanem terminalnym o wypłacie, zazwyczaj odpowiadającej remisowi, na którą gracze zgodzą się by uniknąć nieskończonej rozgrywki.
- Taką ideę realizują zresztą zasady wielu gier, np. zgodnie z regułami Międzynarodowej Federacji Szachowej (FIDE), gracz może zdecydować o zakończeniu gry remisem, gdy identyczna pozycja pojawi się na szachownicy przynajmniej po raz trzeci.

Uwagi do definicji gry

- $\mathbb{S}$ z N opisują acykliczny graf skierowany: $\mathbb{S}$ – wierzchołki; N – sąsiadów.
- Przedział wypłat $[0, 1]$ należy traktować symbolicznie, często używany jest inny.
- Z założenia o braku cykli i skończoności $\mathbb{S}$ wynika skończoność każdej rozgrywki i, następnie, warunek stopu wielu algorytmów, w tym omawianego dalej Min-Max.
- Założenie o braku cykli nie wyklucza modelowania rzeczywistych gier w których rozgrywka może przebiegać wielokrotnie przez taką samą pozycję.
- Powtórzenie rzeczywistej pozycji można zamodelować stanem terminalnym o wypłacie, zazwyczaj odpowiadającej remisowi, na którą gracze zgodzą się by uniknąć nieskończonej rozgrywki.
- Taką ideę realizują zresztą zasady wielu gier, np. zgodnie z regułami Międzynarodowej Federacji Szachowej (FIDE), gracz może zdecydować o zakończeniu gry remisem, gdy identyczna pozycja pojawi się na szachownicy przynajmniej po raz trzeci.

Uwagi do definicji gry

- $\mathbb{S}$ z N opisują acykliczny graf skierowany: $\mathbb{S}$ – wierzchołki; N – sąsiadów.
- Przedział wypłat $[0, 1]$ należy traktować symbolicznie, często używany jest inny.
- Z założenia o braku cykli i skończoności $\mathbb{S}$ wynika skończoność każdej rozgrywki i, następnie, warunek stopu wielu algorytmów, w tym omawianego dalej Min-Max.
- Założenie o braku cykli nie wyklucza modelowania rzeczywistych gier w których rozgrywka może przebiegać wielokrotnie przez taką samą pozycję.
- Powtórzenie rzeczywistej pozycji można zamodelować stanem terminalnym o wypłacie, zazwyczaj odpowiadającej remisowi, na którą gracze zgodzą się by uniknąć nieskończonej rozgrywki.
- Taką ideę realizują zresztą zasady wielu gier, np. zgodnie z regułami Międzynarodowej Federacji Szachowej (FIDE), gracz może zdecydować o zakończeniu gry remisem, gdy identyczna pozycja pojawi się na szachownicy przynajmniej po raz trzeci.



wierzchołki = pozycje gry krawędzie = możliwe ruchy

Definicja strategii

Niech:

- $G = (\mathbb{S}, s_I, N, \text{ev})$ – gra,
- $g \in \{Max, Min\}$ – gracz gry G,
- $D = \{s \in \mathbb{S}; \text{ev}(s) = g\}$ – zbiór pozycji decyzyjnych gracza g.

Wtedy:

- **Czysta strategia** gracza g w grze G to funkcja $\sigma : D \to \mathbb{S}$, taka że $\sigma(s) \in N(s)$ dla każdego $s \in D$.
  Dla każdej pozycji $s \in D$, $\sigma(s)$ wskazuje ruch, który wykona gracz g.
- **Behawioralna strategia** gracza g w grze G to funkcja, która każdej pozycji $s \in D$ przyporządkowuje rozkład prawdopodobieństwa na zbiorze $N(s)$.
  Wskazuje ona prawdopodobieństwa wykonania poszczególnych ruchów.

Definicja strategii

Niech:

- $G = (\mathbb{S}, s_I, N, \text{ev})$ – gra,
- $g \in \{Max, Min\}$ – gracz gry G,
- $D = \{s \in \mathbb{S}; \text{ev}(s) = g\}$ – zbiór pozycji decyzyjnych gracza g.

Wtedy:

- **Czysta strategia** gracza g w grze G to funkcja $\sigma : D \to \mathbb{S}$, taka że $\sigma(s) \in N(s)$ dla każdego $s \in D$.
  Dla każdej pozycji $s \in D$, $\sigma(s)$ wskazuje ruch, który wykona gracz g.
- **Behawioralna strategia** gracza g w grze G to funkcja, która każdej pozycji $s \in D$ przyporządkowuje rozkład prawdopodobieństwa na zbiorze $N(s)$.
  Wskazuje ona prawdopodobieństwa wykonania poszczególnych ruchów.

Definicja strategii

Niech:

- $G = (\mathbb{S}, s_I, N, \text{ev})$ – gra,
- $g \in \{Max, Min\}$ – gracz gry G,
- $D = \{s \in \mathbb{S}; \text{ev}(s) = g\}$ – zbiór pozycji decyzyjnych gracza g.

Wtedy:

- **Czysta strategia** gracza g w grze G to funkcja $\sigma : D \to \mathbb{S}$, taka że $\sigma(s) \in N(s)$ dla każdego $s \in D$.
  Dla każdej pozycji $s \in D$, $\sigma(s)$ wskazuje ruch, który wykona gracz g.
- **Behawioralna strategia** gracza g w grze G to funkcja, która każdej pozycji $s \in D$ przyporządkowuje rozkład prawdopodobieństwa na zbiorze $N(s)$.
  Wskazuje ona prawdopodobieństwa wykonania poszczególnych ruchów.

Definicja strategii

Niech:

- $G = (\mathbb{S}, s_I, N, \text{ev})$ – gra,
- $g \in \{Max, Min\}$ – gracz gry G,
- $D = \{s \in \mathbb{S}; \text{ev}(s) = g\}$ – zbiór pozycji decyzyjnych gracza g.

Wtedy:

- **Czysta strategia** gracza g w grze G to funkcja $\sigma : D \to \mathbb{S}$, taka że $\sigma(s) \in N(s)$ dla każdego $s \in D$.
  Dla każdej pozycji $s \in D$, $\sigma(s)$ wskazuje ruch, który wykona gracz g.
- **Behawioralna strategia** gracza g w grze G to funkcja, która każdej pozycji $s \in D$ przyporządkowuje rozkład prawdopodobieństwa na zbiorze $N(s)$.
  Wskazuje ona prawdopodobieństwa wykonania poszczególnych ruchów.

Definicja strategii

Niech:

- $G = (\mathbb{S}, s_I, N, \text{ev})$ – gra,
- $g \in \{Max, Min\}$ – gracz gry G,
- $D = \{s \in \mathbb{S}; \text{ev}(s) = g\}$ – zbiór pozycji decyzyjnych gracza g.

Wtedy:

- **Czysta strategia** gracza g w grze G to funkcja $\sigma : D \to \mathbb{S}$, taka że $\sigma(s) \in N(s)$ dla każdego $s \in D$.
  Dla każdej pozycji $s \in D$, $\sigma(s)$ wskazuje ruch, który wykona gracz g.
- **Behawioralna strategia** gracza g w grze G to funkcja, która każdej pozycji $s \in D$ przyporządkowuje rozkład prawdopodobieństwa na zbiorze $N(s)$.
  Wskazuje ona prawdopodobieństwa wykonania poszczególnych ruchów.

Funkcja MinMax i optymalna strategia

- Zdefiniujmy funkcję $\mathrm{MinMax} : \mathbb{S} \to [0,1]$:

$$\mathrm{MinMax}(s) = \begin{cases} \mathrm{ev}(s) & \text{gdy } \mathrm{ev}(s) \in [0,1] \\ \max_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Max} \\ \min_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Min} \end{cases}.$$

- $\mathrm{MinMax}(s)$ – wypłata jaką uzyska gracz *Max* po zakończeniu gry, przy założeniu, że gra rozpocznie się w s i że obaj jej uczestnicy będą grali optymalnie.
- Optymalna strategia (i równowaga Nasha): w każdej sytuacji $s \in \mathbb{S}$ wybrać ruch $n \in N(s)$ taki, że $\mathrm{MinMax}(n) = \mathrm{MinMax}(s)$.
- $\mathrm{MinMax}(s_I)$ – wartość (minimaksowa) gry (s_I – stan początkowy gry).

Funkcja MinMax i optymalna strategia

- Zdefiniujmy funkcję $\mathrm{MinMax} : \mathbb{S} \to [0,1]$:

$$\mathrm{MinMax}(s) = \begin{cases} \mathrm{ev}(s) & \text{gdy } \mathrm{ev}(s) \in [0,1] \\ \max_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Max} \\ \min_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Min} \end{cases}.$$

- $\mathrm{MinMax}(s)$ – wypłata jaką uzyska gracz *Max* po zakończeniu gry, przy założeniu, że gra rozpocznie się w s i że obaj jej uczestnicy będą grali optymalnie.
- Optymalna strategia (i równowaga Nasha): w każdej sytuacji $s \in \mathbb{S}$ wybrać ruch $n \in N(s)$ taki, że $\mathrm{MinMax}(n) = \mathrm{MinMax}(s)$.
- $\mathrm{MinMax}(s_I)$ – wartość (minimaksowa) gry (s_I – stan początkowy gry).

Funkcja MinMax i optymalna strategia

- Zdefiniujmy funkcję $\mathrm{MinMax} : \mathbb{S} \to [0,1]$:

$$\mathrm{MinMax}(s) = \begin{cases} \mathrm{ev}(s) & \text{gdy } \mathrm{ev}(s) \in [0,1] \\ \max_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Max} \\ \min_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Min} \end{cases}.$$

- $\mathrm{MinMax}(s)$ – wypłata jaką uzyska gracz *Max* po zakończeniu gry, przy założeniu, że gra rozpocznie się w s i że obaj jej uczestnicy będą grali optymalnie.
- Optymalna strategia (i równowaga Nasha): w każdej sytuacji $s \in \mathbb{S}$ wybrać ruch $n \in N(s)$ taki, że $\mathrm{MinMax}(n) = \mathrm{MinMax}(s)$.
- $\mathrm{MinMax}(s_I)$ – wartość (minimaksowa) gry (s_I – stan początkowy gry).

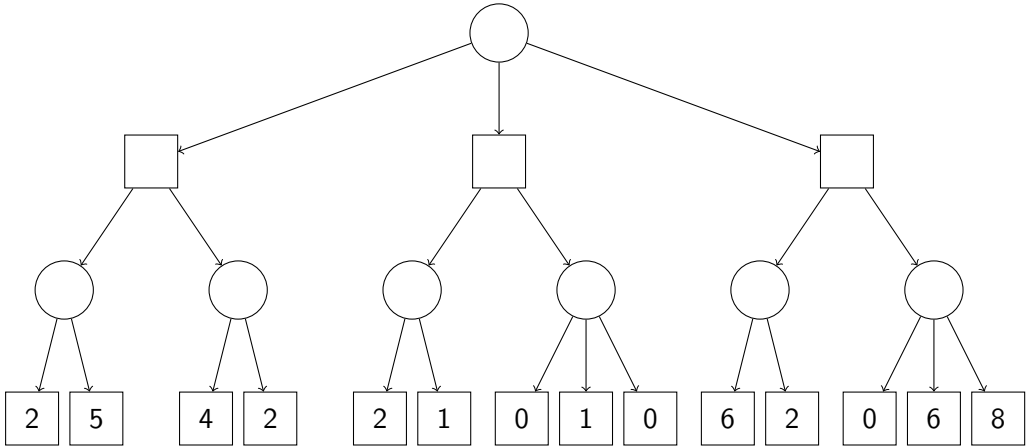
Funkcja MinMax i optymalna strategia

- Zdefiniujmy funkcję $\mathrm{MinMax} : \mathbb{S} \to [0,1]$:

$$\mathrm{MinMax}(s) = \begin{cases} \mathrm{ev}(s) & \text{gdy } \mathrm{ev}(s) \in [0,1] \\ \max_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Max} \\ \min_{n \in N(s)} \mathrm{MinMax}(n) & \text{gdy } \mathrm{ev}(s) = \mathit{Min} \end{cases}.$$

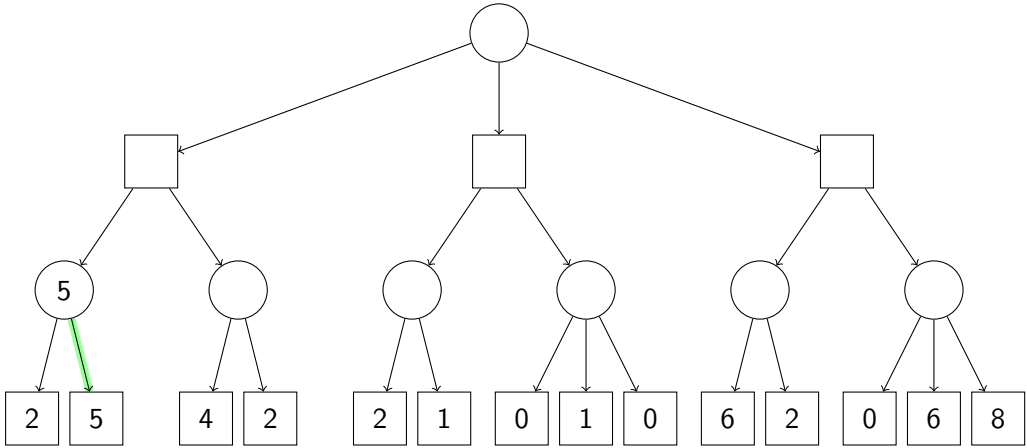
- $\mathrm{MinMax}(s)$ – wypłata jaką uzyska gracz *Max* po zakończeniu gry, przy założeniu, że gra rozpocznie się w s i że obaj jej uczestnicy będą grali optymalnie.
- Optymalna strategia (i równowaga Nasha): w każdej sytuacji $s \in \mathbb{S}$ wybrać ruch $n \in N(s)$ taki, że $\mathrm{MinMax}(n) = \mathrm{MinMax}(s)$.
- $\mathrm{MinMax}(s_I)$ – wartość (minimaksowa) gry (s_I – stan początkowy gry).

Który ruch wykonać? – wartości minimaksowe pozycji



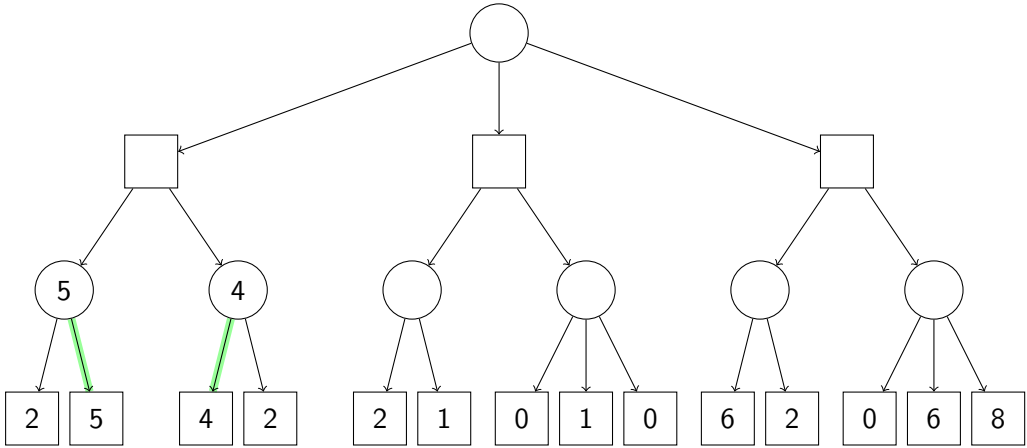
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



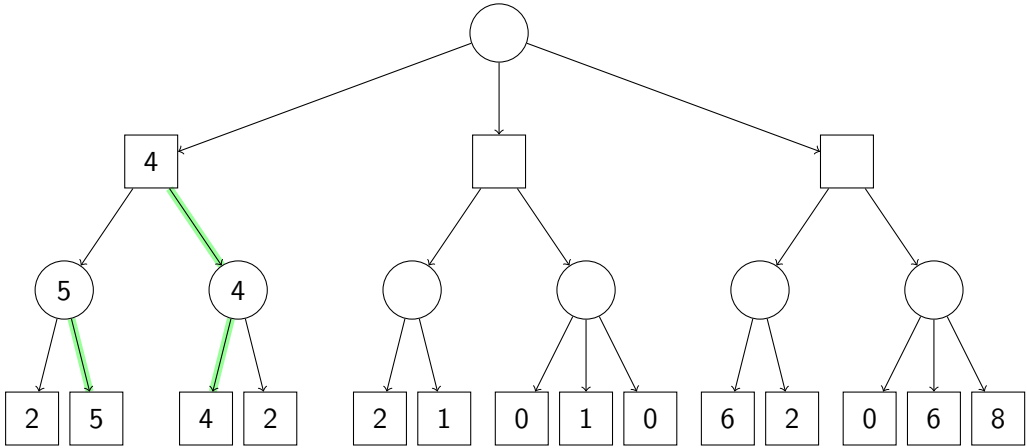
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



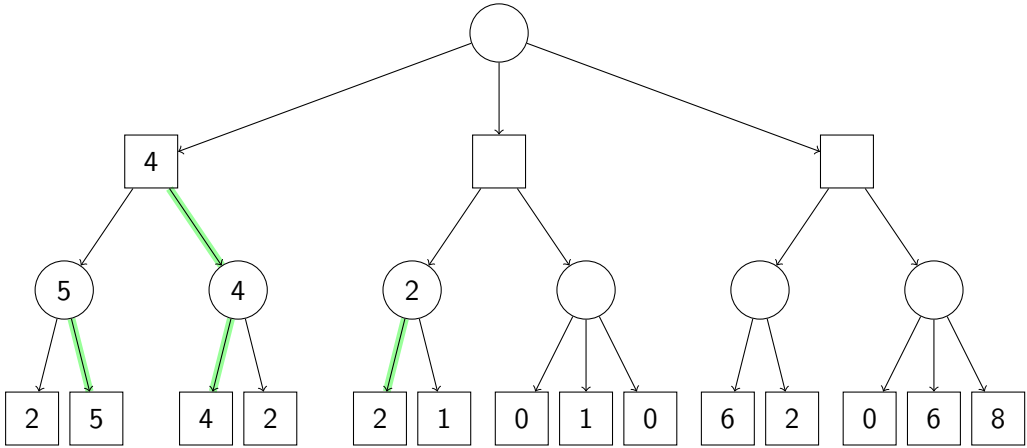
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



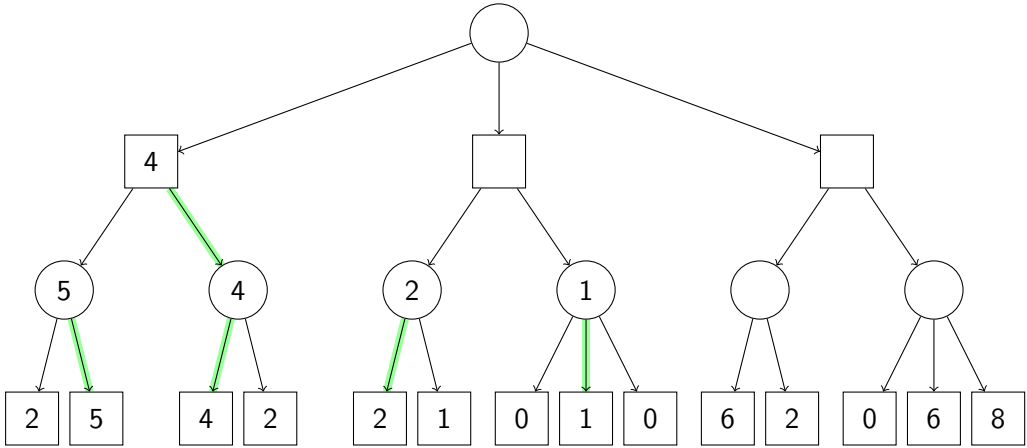
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



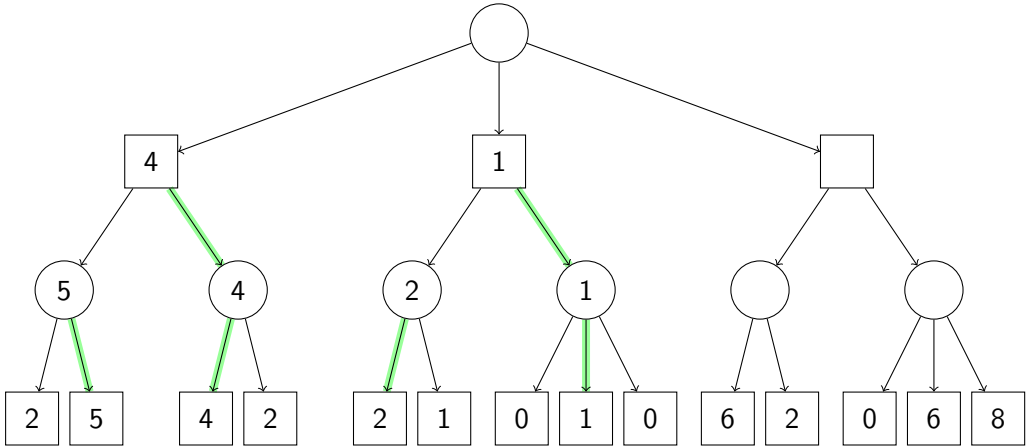
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



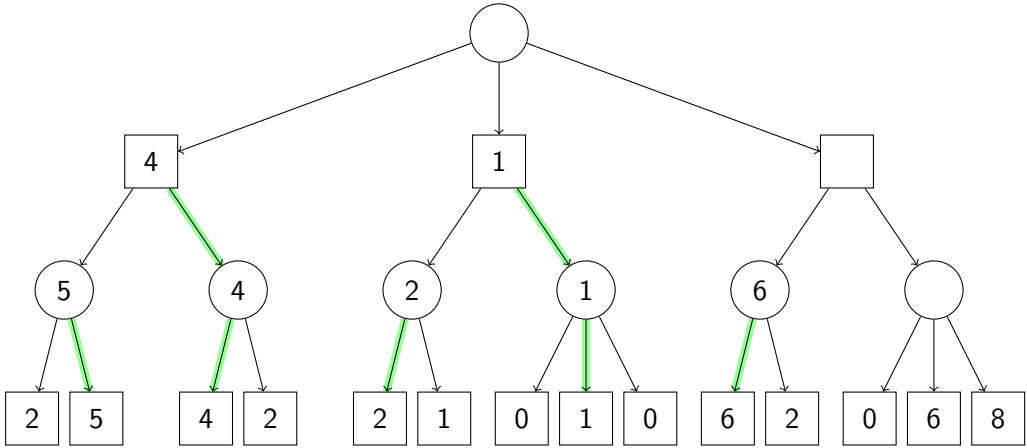
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



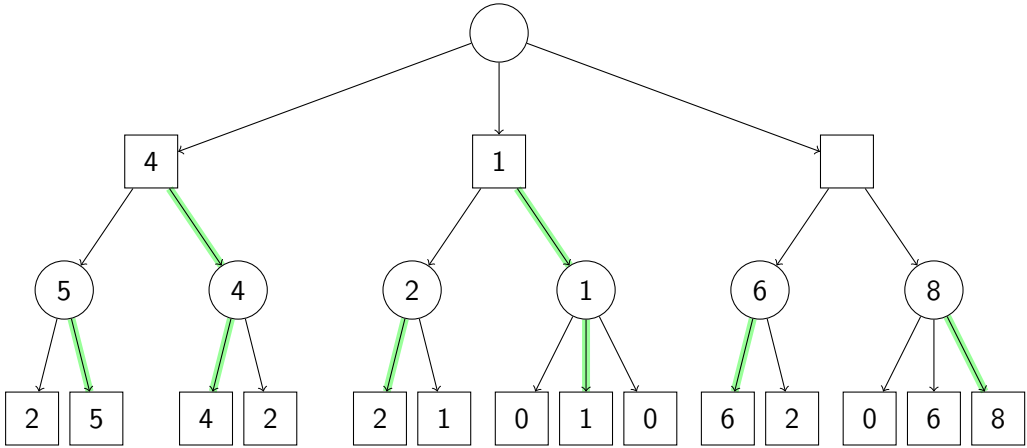
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



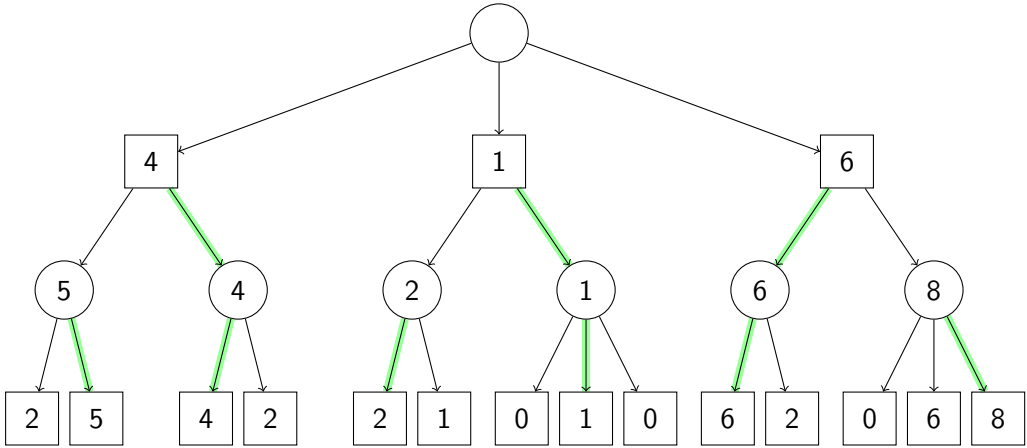
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



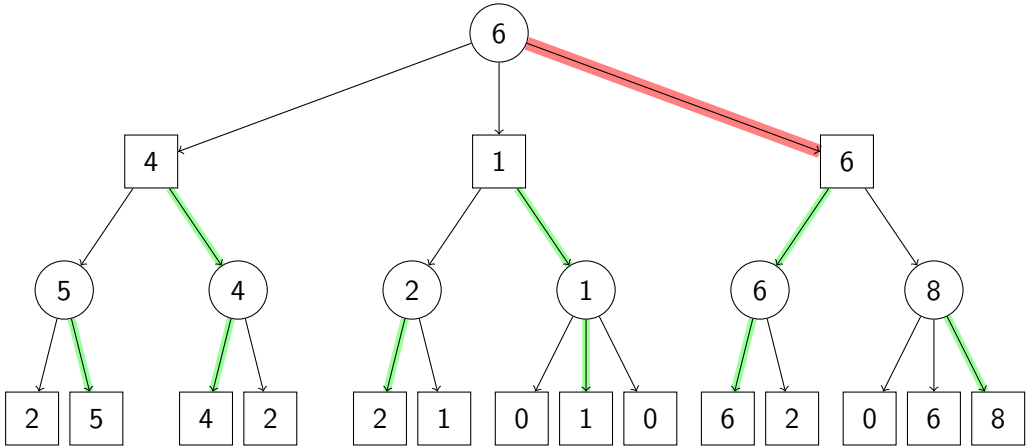
◯ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



◯ – ruch gracza *Max* □ – ruch gracza *Min*

Pseudokod algorytmu obliczającego wartość minimaksową

Funkcja obliczająca wartość minimaksową pozycji s:

```
1  fun MinMax(s):
2      // ev(s) ∈ [0,1] ⟹ MinMax(s) = ev(s)
3      if ev(s) ∈ [0,1]: return ev(s)
4      // ev(s) = Max ⟹ MinMax(s) = max_{n∈N(s)} MinMax(n)
5      if ev(s) = Max:
6          res ← 0
7          foreach m ∈ N(s):
8              res ← max(res, MinMax(m))
9      else: // ev(s) = Min ⟹ MinMax(s) = min_{n∈N(s)} MinMax(n)
10         res ← 1
11         foreach m ∈ N(s):
12             res ← min(res, MinMax(m))
13     return res
```

Pseudokod algorytmu wyznaczającego optymalny ruch

Funkcja wyznaczająca za pomocą algorytmu Min-Max optymalny ruch $n \in N(s)$, tj. spełniający $\text{MinMax}(n) = \text{MinMax}(s)$:

```
1  fun BestMoveByMinMax(s):
2      best_move ← None
3      if ev(s) = Max:
4          foreach m ∈ N(s):
5              v ← MinMax(m)
6              if best_move = None or v > best_score:
7                  best_move ← m
8                  best_score ← v
9      else: // ev(s) = Min
10         foreach m ∈ N(s):
11             v ← MinMax(m)
12             if best_move = None or v < best_score:
13                 best_move ← m
14                 best_score ← v
15     return best_move
```

Szacowana złożoność wybranych gier

gra	logarytm dziesiętny ze złożoności	
	przestrzeni stanów[1]	drzewa gry[2]
Kółko i krzyżyk (3x3)	3	5
Czwórki (Connect Four, 7x6)	13	21
Rozgrywka w widne w brydżu (średnio)	17	29
Warcaby amerykańskie	18	31
Othello (8x8)	28	58
Szachy	47	123
Go (19x19)	171	360

Źródła: [Allis 1994], [Herik 2002], [Beling doktorat]

W przypadku wielu gier, czas pracy algorytmu MinMax jest nieakceptowalny.

[1] liczba różnych pozycji osiągalnych z pozycji początkowej

[2] liczba liści w drzewie minimaksowym o głębokości ograniczonej do minimum niezbędnego do ustalenia wartości pozycji początkowej

Szacowana złożoność wybranych gier

gra	logarytm dziesiętny ze złożoności	
	przestrzeni stanów[1]	drzewa gry[2]
Kółko i krzyżyk (3x3)	3	5
Czwórki (Connect Four, 7x6)	13	21
Rozgrywka w widne w brydżu (średnio)	17	29
Warcaby amerykańskie	18	31
Othello (8x8)	28	58
Szachy	47	123
Go (19x19)	171	360

Źródła: [Allis 1994], [Herik 2002], [Beling doktorat]

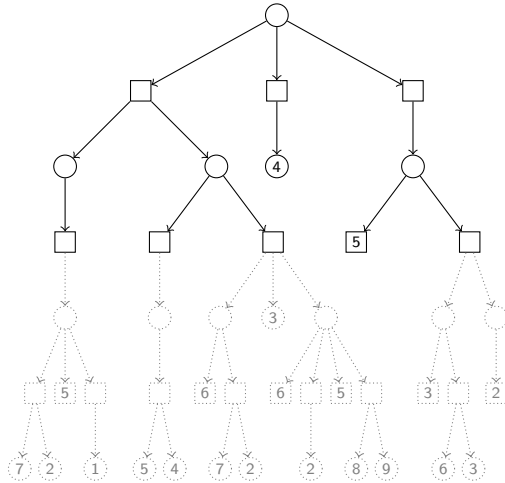
W przypadku wielu gier, czas pracy algorytmu MinMax jest nieakceptowalny.

[1] liczba różnych pozycji osiągalnych z pozycji początkowej

[2] liczba liści w drzewie minimaksowym o głębokości ograniczonej do minimum niezbędnego do ustalenia wartości pozycji początkowej

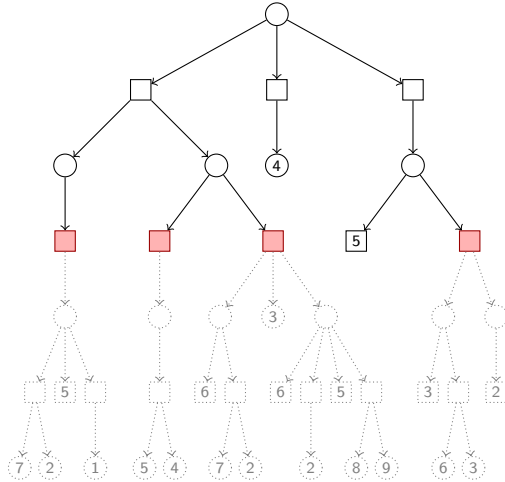
Ograniczenie głębokości poszukiwań

- drzewo poszukiwań ograniczone do zadanej głębokości
- po jej osiągnięciu wartość pozycji jest szacowana heurystycznie (przy wykorzystaniu wiedzy dziedzinowej)
- siła gry rośnie wraz z trafnością tego szacowania oraz głębokością poszukiwań



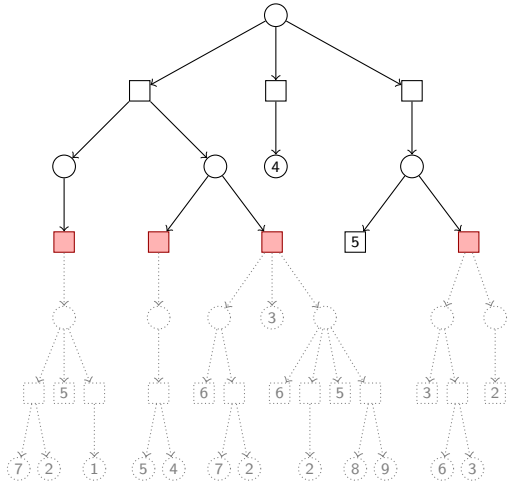
Ograniczenie głębokości poszukiwań

- drzewo poszukiwań ograniczone do zadanej głębokości
- po jej osiągnięciu wartość pozycji jest szacowana heurystycznie (przy wykorzystaniu wiedzy dziedzinowej)
- siła gry rośnie wraz z trafnością tego szacowania oraz głębokością poszukiwań



Ograniczenie głębokości poszukiwań

- drzewo poszukiwań ograniczone do zadanej głębokości
- po jej osiągnięciu wartość pozycji jest szacowana heurystycznie (przy wykorzystaniu wiedzy dziedzinowej)
- siła gry rośnie wraz z trafnością tego szacowania oraz głębokością poszukiwań



Pseudokod algorytmu Min-Max z ograniczoną głębokością poszukiwań

Funkcja obliczająca przybliżoną wartości minimaksowej pozycji s.

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0,1]: return ev(s)
3   if d = 0: return MinMaxApprox(s)
4   if ev(s) = Max:
5     res ← 0
6     foreach m ∈ N(s):
7       res ← max(res, MinMaxDepth(m, d-1))
8   else: // ev(s) = Min
9     res ← 1
10    foreach m ∈ N(s):
11      res ← min(res, MinMaxDepth(m, d-1))
12  return res
```

- Parametr d określa maksymalną głębokość poszukiwań,
- gdy zostanie ona osiągnięta, zwracana jest wartość MinMaxApprox(s) – *statyczna / heurystyczna* ocena s.
- MinMaxApprox : $\mathbb{S} \to [0,1]$ nazywana jest *funkcją oceniająca* lub *heurystyczną*,
- aproksymuje ona MinMax, tj. MinMaxAprox(s) $\approx$ MinMax(s),
- powinna robić to szybko (często korzystając z wiedzy dziedzinowej o grze), bez analizowania następników s.

Pseudokod algorytmu Min-Max z ograniczoną głębokością poszukiwań

Funkcja obliczająca przybliżoną wartości minimaksowej pozycji s.

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0,1]: return ev(s)
3   if d = 0: return MinMaxApprox(s)
4   if ev(s) = Max:
5     res ← 0
6     foreach m ∈ N(s):
7       res ← max(res, MinMaxDepth(m, d-1))
8   else: // ev(s) = Min
9     res ← 1
10    foreach m ∈ N(s):
11      res ← min(res, MinMaxDepth(m, d-1))
12  return res
```

- Parametr d określa maksymalną głębokość poszukiwań,
- gdy zostanie ona osiągnięta, zwracana jest wartość MinMaxApprox(s) – *statyczna / heurystyczna* ocena s.
- MinMaxApprox : $\mathbb{S} \to [0,1]$ nazywana jest *funkcją oceniająca* lub *heurystyczną*,
- aproksymuje ona MinMax, tj. MinMaxAprox(s) $\approx$ MinMax(s),
- powinna robić to szybko (często korzystając z wiedzy dziedzinowej o grze), bez analizowania następników s.

Pseudokod algorytmu Min-Max z ograniczoną głębokością poszukiwań

Funkcja obliczająca przybliżoną wartości minimaksowej pozycji s.

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0,1]: return ev(s)
3   if d = 0: return MinMaxApprox(s)
4   if ev(s) = Max:
5     res ← 0
6     foreach m ∈ N(s):
7       res ← max(res, MinMaxDepth(m, d-1))
8   else: // ev(s) = Min
9     res ← 1
10    foreach m ∈ N(s):
11      res ← min(res, MinMaxDepth(m, d-1))
12  return res
```

- Parametr d określa maksymalną głębokość poszukiwań,
- gdy zostanie ona osiągnięta, zwracana jest wartość MinMaxApprox(s) – *statyczna / heurystyczna* ocena s.
- MinMaxApprox : $\mathbb{S} \to [0,1]$ nazywana jest *funkcją oceniająca* lub *heurystyczną*,
- aproksymuje ona MinMax, tj. MinMaxApprox(s) $\approx$ MinMax(s),
- powinna robić to szybko (często korzystając z wiedzy dziedzinowej o grze), bez analizowania następników s.

Pseudokod algorytmu Min-Max z ograniczoną głębokością poszukiwań

Funkcja obliczająca przybliżoną wartości minimaksowej pozycji s.

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0,1]: return ev(s)
3   if d = 0: return MinMaxApprox(s)
4   if ev(s) = Max:
5     res ← 0
6     foreach m ∈ N(s):
7       res ← max(res, MinMaxDepth(m, d-1))
8   else: // ev(s) = Min
9     res ← 1
10    foreach m ∈ N(s):
11      res ← min(res, MinMaxDepth(m, d-1))
12  return res
```

- Parametr d określa maksymalną głębokość poszukiwań,
- gdy zostanie ona osiągnięta, zwracana jest wartość MinMaxApprox(s) – *statyczna / heurystyczna* ocena s.
- MinMaxApprox : $\mathbb{S} \to [0,1]$ nazywana jest *funkcją oceniająca* lub *heurystyczną*,
- aproksymuje ona MinMax, tj. MinMaxAprox(s) $\approx$ MinMax(s),
- powinna robić to szybko (często korzystając z wiedzy dziedzinowej o grze), bez analizowania następników s.

Pseudokod algorytmu Min-Max z ograniczoną głębokością poszukiwań

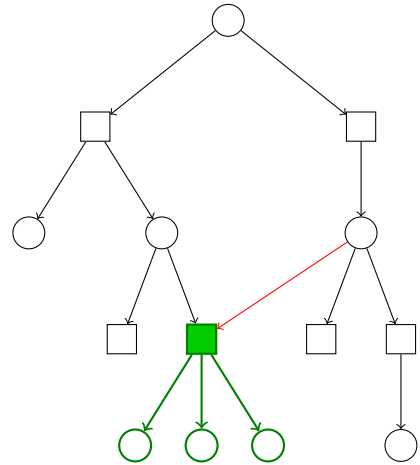
Funkcja obliczająca przybliżoną wartości minimaksowej pozycji s.

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0,1]: return ev(s)
3   if d = 0: return MinMaxApprox(s)
4   if ev(s) = Max:
5     res ← 0
6     foreach m ∈ N(s):
7       res ← max(res, MinMaxDepth(m, d-1))
8   else: // ev(s) = Min
9     res ← 1
10    foreach m ∈ N(s):
11      res ← min(res, MinMaxDepth(m, d-1))
12  return res
```

- Parametr d określa maksymalną głębokość poszukiwań,
- gdy zostanie ona osiągnięta, zwracana jest wartość MinMaxApprox(s) – *statyczna / heurystyczna* ocena s.
- MinMaxApprox : $\mathbb{S} \to [0,1]$ nazywana jest *funkcją oceniająca* lub *heurystyczną*,
- aproksymuje ona MinMax, tj. MinMaxAprox(s) $\approx$ MinMax(s),
- powinna robić to szybko (często korzystając z wiedzy dziedzinowej o grze), bez analizowania następników s.

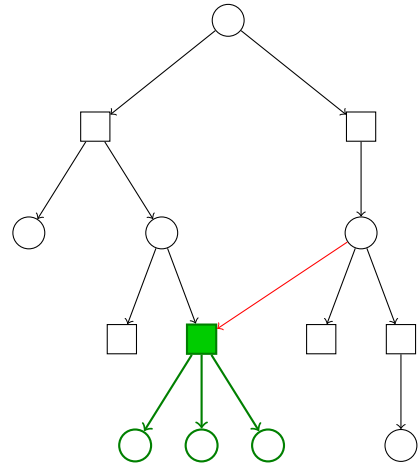
Tablica Transpozycji

- cel: uniknie redundantnych obliczeń;
- **transpozycja** – różne sekwencje ruchów prowadzą do tej samej pozycji;
- **tablica transpozycji** zawiera wyniki uzyskane dla dotychczas zbadanych pozycji.



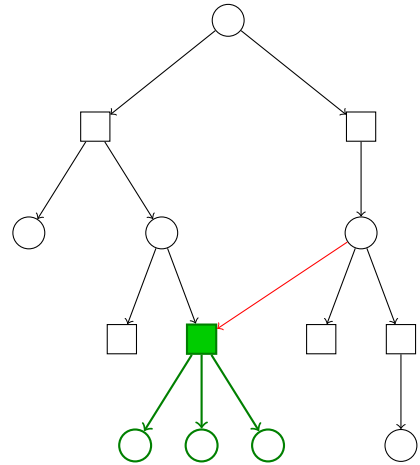
Tablica Transpozycji

- cel: uniknie redundantnych obliczeń;
- **transpozycja** – różne sekwencje ruchów prowadzą do tej samej pozycji;
- **tablica transpozycji** zawiera wyniki uzyskane dla dotychczas zbadanych pozycji.



Tablica Transpozycji

- cel: uniknie redundantnych obliczeń;
- **transpozycja** – różne sekwencje ruchów prowadzą do tej samej pozycji;
- **tablica transpozycji** zawiera wyniki uzyskane dla dotychczas zbadanych pozycji.



Min-Max z tablicą transpozycji – pseudokod

Funkcja obliczająca wartość minimaksową pozycji s:

```
1  fun MinMax(s):
2      if ev(s) ∈ [0, 1]: return ev(s)
3      if s ∈ TT: return TT[s]
4      if ev(s) = Max:
5          res ← 0
6          foreach m ∈ N(s):
7              res ← max(res, MinMax(m))
8      else: // ev(s) = Min
9          res ← 1
10         foreach m ∈ N(s):
11             res ← min(res, MinMax(m))
12     TT[s] ← res
13     return res
```

- Tablica transpozycji TT to tablica asocjacyjna (mapa) pamiętająca pozycje gry (klucze) i ich wartość minimaksowe,
- zazwyczaj implementowana jako tablica haszująca.
- Zaraz po obliczeniu wartość pozycji, jest ona zapisywana w TT.
- Próba użycia wartości z TT odbywa się przed rekurencyjną analizą ruchów.

Min-Max z tablicą transpozycji – pseudokod

Funkcja obliczająca wartość minimaksową pozycji s:

```
1  fun MinMax(s):
2      if ev(s) ∈ [0, 1]: return ev(s)
3      if s ∈ TT: return TT[s]
4      if ev(s) = Max:
5          res ← 0
6          foreach m ∈ N(s):
7              res ← max(res, MinMax(m))
8      else: // ev(s) = Min
9          res ← 1
10         foreach m ∈ N(s):
11             res ← min(res, MinMax(m))
12     TT[s] ← res
13     return res
```

- Tablica transpozycji TT to tablica asocjacyjna (mapa) pamiętająca pozycje gry (klucze) i ich wartość minimaksowe,
- zazwyczaj implementowana jako tablica haszująca.
- Zaraz po obliczeniu wartość pozycji, jest ona zapisywana w TT.
- Próba użycia wartości z TT odbywa się przed rekurencyjną analizą ruchów.

Min-Max z tablicą transpozycji – pseudokod

Funkcja obliczająca wartość minimaksową pozycji s:

```
1  fun MinMax(s):
2      if ev(s) ∈ [0,1]: return ev(s)
3      if s ∈ TT: return TT[s]
4      if ev(s) = Max:
5          res ← 0
6          foreach m ∈ N(s):
7              res ← max(res, MinMax(m))
8      else: // ev(s) = Min
9          res ← 1
10         foreach m ∈ N(s):
11             res ← min(res, MinMax(m))
12     TT[s] ← res
13     return res
```

- Tablica transpozycji TT to tablica asocjacyjna (mapa) pamiętająca pozycje gry (klucze) i ich wartość minimaksowe,
- zazwyczaj implementowana jako tablica haszująca.
- Zaraz po obliczeniu wartość pozycji, jest ona zapisywana w TT.
- Próba użycia wartości z TT odbywa się przed rekurencyjną analizą ruchów.

Min-Max z tablicą transpozycji – pseudokod

Funkcja obliczająca wartość minimaksową pozycji s:

```
1  fun MinMax(s):
2      if ev(s) ∈ [0,1]: return ev(s)
3      if s ∈ TT: return TT[s]
4      if ev(s) = Max:
5          res ← 0
6          foreach m ∈ N(s):
7              res ← max(res, MinMax(m))
8      else: // ev(s) = Min
9          res ← 1
10         foreach m ∈ N(s):
11             res ← min(res, MinMax(m))
12     TT[s] ← res
13     return res
```

- Tablica transpozycji TT to tablica asocjacyjna (mapa) pamiętająca pozycje gry (klucze) i ich wartość minimaksowe,
- zazwyczaj implementowana jako tablica haszująca.
- Zaraz po obliczeniu wartość pozycji, jest ona zapisywana w TT.
- Próba użycia wartości z TT odbywa się przed rekurencyjną analizą ruchów.

Min-Max z ograniczoną głębokością i tablicą transpozycji – pseudokod

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0, 1]: return ev(s)
3   if s ∈ TT:
4     v, ttd = TT[s]
5     if ttd ⩾ d: return v
6   if d = 0: MinMaxApprox(s)
7   if ev(s) = Max:
8     res ← 0
9     foreach m ∈ N(s):
10      res ← max(res, MinMaxDepth(m, d−1))
11  else: // ev(s) = Min
12    res ← 1
13    foreach m ∈ N(s):
14      res ← min(res, MinMaxDepth(m, d−1))
15  TT[s] ← res, d
16  return res
```

Dokładność wyniku rośnie wraz z głębokością przeszukiwania d.

By nie użyć wyniku z TT o zbyt małej dokładności należy:

- oprócz wartości pozycji, zapisać także głębokość obliczeń d;
- upewnić się że głębokość dla wartości odczytanej z TT wynosi co najmniej d.

Min-Max z ograniczoną głębokością i tablicą transpozycji – pseudokod

```
1  fun MinMaxDepth(s, d):
2    if ev(s) ∈ [0, 1]: return ev(s)
3    if s ∈ TT:
4      v, ttd = TT[s]
5      if ttd ⩾ d: return v
6    if d = 0: MinMaxApprox(s)
7    if ev(s) = Max:
8      res ← 0
9      foreach m ∈ N(s):
10       res ← max(res, MinMaxDepth(m, d−1))
11   else: // ev(s) = Min
12     res ← 1
13     foreach m ∈ N(s):
14       res ← min(res, MinMaxDepth(m, d−1))
15   TT[s] ← res, d
16   return res
```

Dokładność wyniku rośnie wraz z głębokością przeszukiwania d.

By nie użyć wyniku z TT o zbyt małej dokładności należy:

- oprócz wartości pozycji, zapisać także głębokość obliczeń d;
- upewnić się że głębokość dla wartości odczytanej z TT wynosi co najmniej d.

Min-Max z ograniczoną głębokością i tablicą transpozycji – pseudokod

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0, 1]: return ev(s)
3   if s ∈ TT:
4     v, ttd = TT[s]
5     if ttd ⩾ d: return v
6   if d = 0: MinMaxApprox(s)
7   if ev(s) = Max:
8     res ← 0
9     foreach m ∈ N(s):
10      res ← max(res, MinMaxDepth(m, d−1))
11  else: // ev(s) = Min
12    res ← 1
13    foreach m ∈ N(s):
14      res ← min(res, MinMaxDepth(m, d−1))
15  TT[s] ← res, d
16  return res
```

Dokładność wyniku rośnie wraz z głębokością przeszukiwania d.

By nie użyć wyniku z TT o zbyt małej dokładności należy:

- oprócz wartości pozycji, zapisać także głębokość obliczeń d;
- upewnić się że głębokość dla wartości odczytanej z TT wynosi co najmniej d.

Min-Max z ograniczoną głębokością i tablicą transpozycji – pseudokod

```
1  fun MinMaxDepth(s, d):
2   if ev(s) ∈ [0, 1]: return ev(s)
3   if s ∈ TT:
4     v, ttd = TT[s]
5     if ttd ⩾ d: return v
6   if d = 0: MinMaxApprox(s)
7   if ev(s) = Max:
8     res ← 0
9     foreach m ∈ N(s):
10      res ← max(res, MinMaxDepth(m, d−1))
11  else: // ev(s) = Min
12    res ← 1
13    foreach m ∈ N(s):
14      res ← min(res, MinMaxDepth(m, d−1))
15  TT[s] ← res, d
16  return res
```

Dokładność wyniku rośnie wraz z głębokością przeszukiwania d.

By nie użyć wyniku z TT o zbyt małej dokładności należy:

- oprócz wartości pozycji, zapisać także głębokość obliczeń d;
- upewnić się że głębokość dla wartości odczytanej z TT wynosi co najmniej d.

Bibliografia

- **Louis V. Allis** *Searching for Solutions in Games and Artificial Intelligence*, rozprawa doktorska, 1994
- **H. Jaap van den Herik, et al.** *Games solved: Now and in the future*, Artificial Intelligence 134/2002
- **P. Beling** *Szybkie algorytmy decyzyjne w grach z doskonałą informacją i ich wykorzystanie w grach jej pozbawionych*, rozprawa doktorska, Uniwersytet Łódzki, 2016
- **E. Rasmussen** *Games and Information - An Introduction to Game Theory*, Oxford: Blackwell Publishers, 1989

Dziękuję za uwagę!