Czy komputer może myśleć jak człowiek?

Potrafimy udowodnić, że pewnych rzeczy komputer nie może zrobić. Czy którąś z nich potrafi zrobić człowiek?

— są podejrzenia, że pewne ludzkie rozumowania matematyczne nie redukują się do reguł obliczeniowych.

(patrz: Roger Penrose *Cienie umysłu*, Zysk i S-ka, Poznań 2000)

Twierdzenie (Kurt Gödel, 1930):
Istnieją takie własności liczb naturalnych, które są prawdziwe, ale których prawdziwości nie da się udowodnić formalnie.

Dowód: (uwierzcie!).

Dowód jakiejś własności przeprowadzony przez program komputerowy jest zawsze formalny — zachodzi odpowiedniość 1-1 między dowodami formalnymi a algorytmami (uwierzcie!). A więc istnieją prawdziwe własności liczb naturalnych, których prawdziwości komputery nie są w stanie dowieść.

Czy człowiek-matematyk jest równie ograniczony?

Czy komputer może myśleć jak człowiek?

Czy człowiek-matematyk jest równie ograniczony?
czyli
Czy człowiek-matematyk może coś zrobić oprócz przeprowadzenia *formalnego* (czyli algorytmicznego) dowodu?

A — algorytm obejmujący *wszystkie* możliwości matematyków. Dla danej tezy C algorytm A może albo

- zakończyć działanie z sukcesem; i wtedy C jest dowiedziona; albo
- nie zakończyć działania; i wtedy nie możemy wiedzieć.

$C_1, C_2, C_3, \ldots$ — ciąg *wszystkich* algorytmów z jednym argumentem (dają się tak ustawić — uwierzcie!). Algorytm C_q wczytuje jedną liczbę naturalną n i albo kończy działanie albo się zapętla.

A wczytuje dwie liczby q i n i albo

- kończy działanie z sukcesem; wtedy C_q zapętla się na n; albo
- nie kończy działania; wtedy nie możemy wiedzieć.

Czy komputer może myśleć jak człowiek?

A wczytuje dwie liczby q i n i albo

- kończy działanie z sukcesem; wtedy C_q zapętla się na n; albo
- nie kończy działania; wtedy nie możemy wiedzieć.

Więc:
Dla dow. q i n, jeśli A kończy pracę na q i n, to C_q nie kończy pracy na n.

Weźmy $q = n$:
Dla dow. n, jeśli A kończy pracę na n i n, to C_n nie kończy pracy na n.

Algorytm wczytujący n i zachowujący się jak A na n i n musi występować w ciągu $C_1, C_2, C_3, \ldots$; niech to będzie C_k. Wtedy:
Dla dow. n, jeśli C_k kończy pracę na n, to C_n nie kończy pracy na n.

Weźmy $n = k$: Jeśli C_k kończy pracę na k, to C_k nie kończy pracy na k.
Więc: C_k nie kończy pracy na k.
Więc: A nie kończy pracy na k i k.
Więc: Nie możemy wiedzieć, czy C_k kończy pracę na k.

Dowiedliśmy, że to jest prawda
i że nie możemy tego wiedzieć w sposób czysto algorytmiczny.

Czy komputer może myśleć jak człowiek?

Więc: C_k nie kończy pracy na k.
Więc: Nie możemy wiedzieć, czy C_k kończy pracę na k.

Dowiedliśmy, że to jest prawda
i że nie możemy tego wiedzieć w sposób czysto algorytmiczny.

A więc potrafimy stwierdzić prawdziwość czegoś inaczej niż w sposób algorytmiczny.

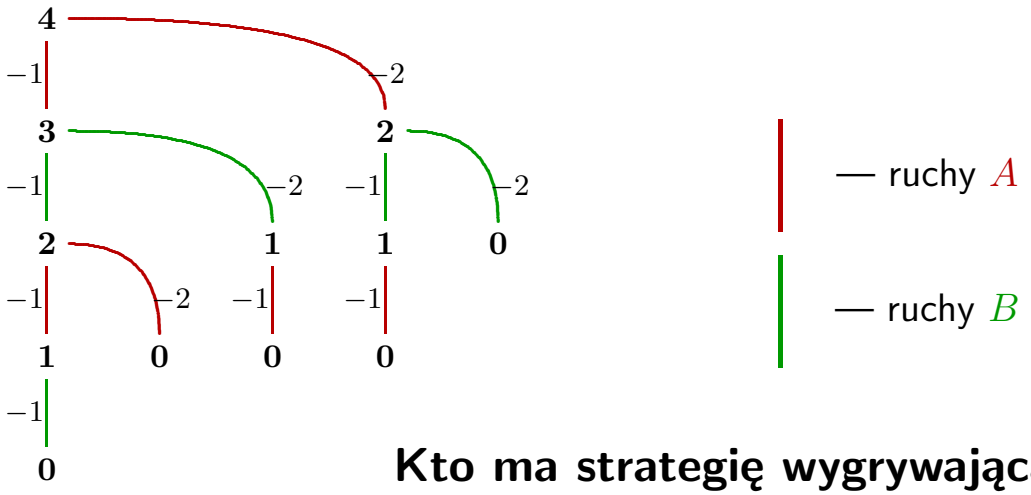
Tego nie może potrafić komputer.

Minimax

Minimax — metoda poszukiwania optymalnej strategii w *drzewie gry*:

Przykład:

W pudełku jest n zapałek. Gracze A i B kolejno wykonują ruchy; pojedynczy ruch polega na zabraniu z pudełka albo jednej albo dwóch zapałek. Kto zabierze ostatnią zapałkę, ten przegrywa.



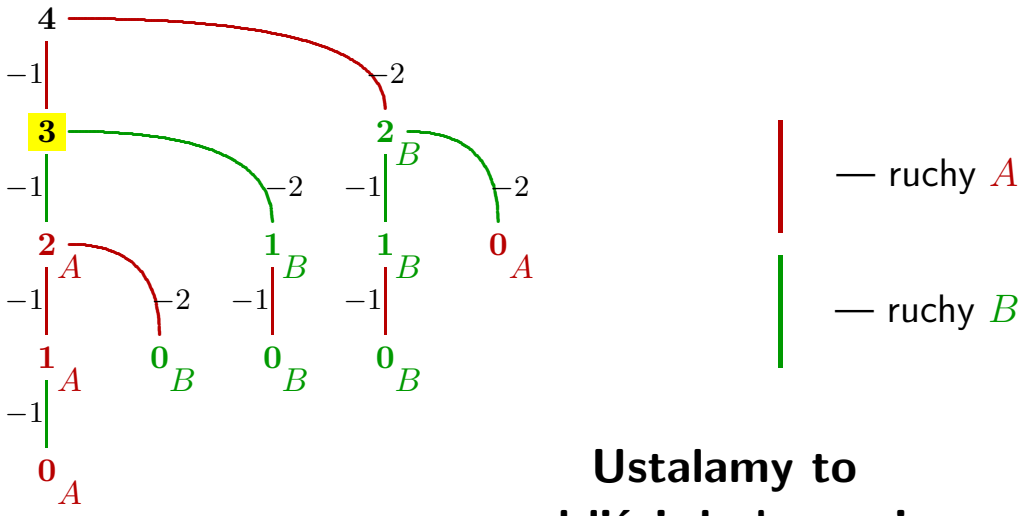
Kto ma strategię wygrywającą w którym węźle drzewa?

Minimax

Minimax — metoda poszukiwania optymalnej strategii w *drzewie gry*:

Przykład:

W pudełku jest n zapałek. Gracze A i B kolejno wykonują ruchy; pojedynczy ruch polega na zabraniu z pudełka albo jednej albo dwóch zapałek. Kto zabierze ostatnią zapałkę, ten przegrywa.



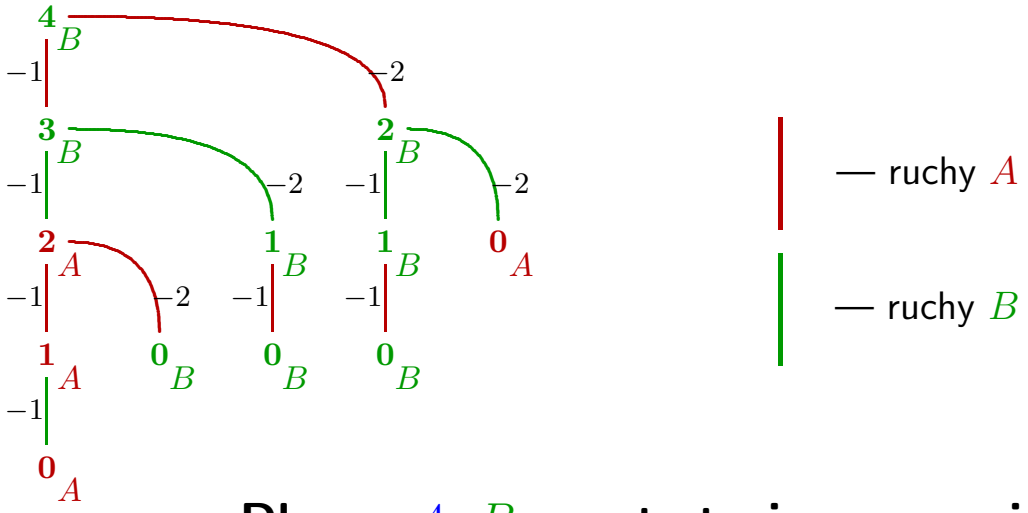
Ustalamy to od liści do korzenia

Minimax

Minimax — metoda poszukiwania optymalnej strategii w *drzewie gry*:

Przykład:

W pudełku jest n zapałek. Gracze A i B kolejno wykonują ruchy; pojedynczy ruch polega na zabraniu z pudełka albo jednej albo dwóch zapałek. Kto zabierze ostatnią zapałkę, ten przegrywa.



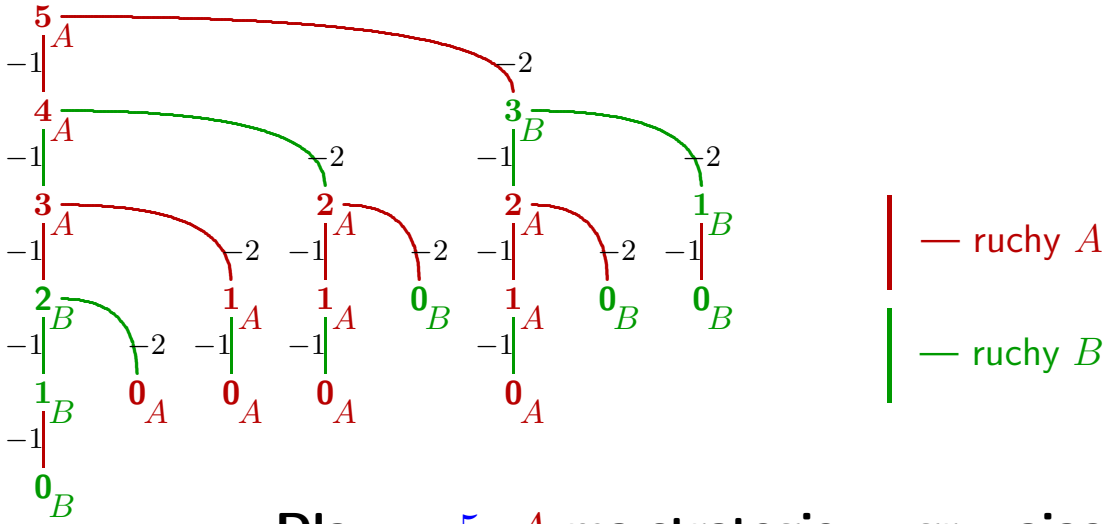
Dla $n = 4$: B ma strategię wygrywającą

Minimax

Minimax — metoda poszukiwania optymalnej strategii w *drzewie gry*:

Przykład:

W pudełku jest n zapałek. Gracze A i B kolejno wykonują ruchy; pojedynczy ruch polega na zabraniu z pudełka albo jednej albo dwóch zapałek. Kto zabierze ostatnią zapałkę, ten przegrywa.



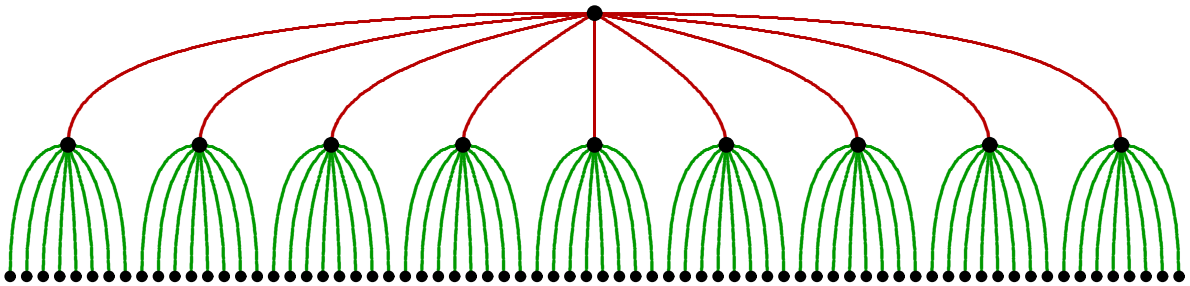
Dla $n = 5$: A ma strategię wygrywającą

Minimax

Czy tą samą metodę można zastosować do bardziej skomplikowanych gier?

- **Problem z wielkością drzewa gry**. Początkowo próbowałem zilustrować metodę minimaksu grą w kółko i krzyżyk:

  Pierwszy gracz może wykonać 9 różnych ruchów; drugi gracz 8 ruchów; itp. . . .



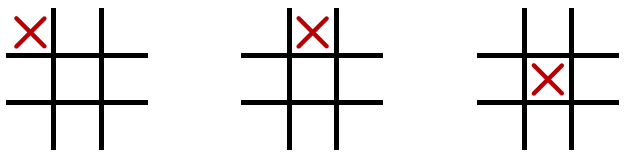
. . . i do końca jeszcze daleko . . .

Minimax

Czy tą samą metodę można zastosować do bardziej skomplikowanych gier?

- **Problem z wielkością drzewa gry**. Początkowo próbowałem zilustrować metodę minimaksu grą w kółko i krzyżyk — ale drzewo było zbyt duże na slajd. **Minimax analizuje od liści ku korzeniowi**; a więc dla rozpoczęcia analizy trzeba dysponować *całym* drzewem. Dla bardziej złożonych gier to może być nierealne.

- Istnieją **metody zmniejszenia drzewa gry**:
  - **ścisłe**, specyficzne dla danej gry; na przykład dziewięć różnych możliwych pierwszych ruchów w grze w kółko i krzyżyk to tylko symetryczne warianty trzech ruchów:



Minimax

Czy tą samą metodę można zastosować do bardziej skomplikowanych gier?

- **Problem z wielkością drzewa gry**.
- **Metody zmniejszenia drzewa gry**:
  - **ścisłe**, specyficzne dla danej gry;
  - **heurystyczne** — dalekie od ścisłości zasady pochodzące z praktyki działające *na ogół* nieźle (choć nie idealnie)

Metody **heurystyczne** stosujemy często na codzień:

„jeśli boli cię gardło, to ubierz się ciepło, zjedz gripex
i popij gorącym mlekiem z miodem i czosnkiem"

Ta metoda działa na ogół dobrze; ale zawodzi jeśli

- ból gardła jest skutkiem podrażnienia mechanicznego,
- ból gardła jest objawem cięższej choroby,
- masz spotkać się z szefem alergicznie reagującym na zapach czosnku,
- itp.

Minimax

Czy tą samą metodę można zastosować do bardziej skomplikowanych gier?

- **Problem z wielkością drzewa gry**.
- **Metody zmniejszenia drzewa gry**:
  - **ścisłe**, specyficzne dla danej gry;
  - **heurystyczne** — dalekie od ścisłości zasady pochodzące z praktyki działające *na ogół* nieźle (choć nie idealnie)

Metody **heurystyczne** stosujemy często na codzień w sytuacjach **braku informacji** lub **trudnego dostępu do informacji**.

W grach stosujemy metody heurystyczne, np.

„materialna przewaga opłaca się"
„lepiej jest zająć centrum planszy"
„damę brać pod siebie" (impasy w brydżu)

do oceny sytuacji lub dla odcięcia mało użytecznych możliwości.

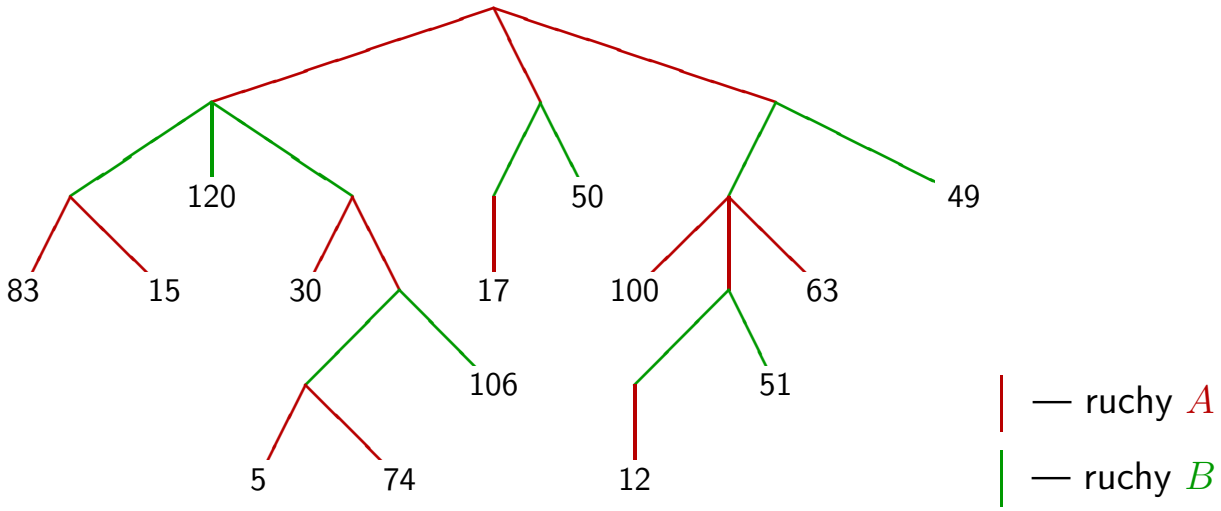
Minimax

Czy tą samą metodę można zastosować do bardziej skomplikowanych gier?

- **Problem z wielkością drzewa gry**.
- **Metody zmniejszenia drzewa gry**:
  - **ścisłe**, specyficzne dla danej gry;
  - **heurystyczne** — dalekie od ścisłości zasady pochodzące z praktyki działające *na ogół* nieźle (choć nie idealnie)
  - **odcinanie gałęzi drzewa** w oparciu o heurystyczną ocenę wartości sytuacji (patrz dalej)

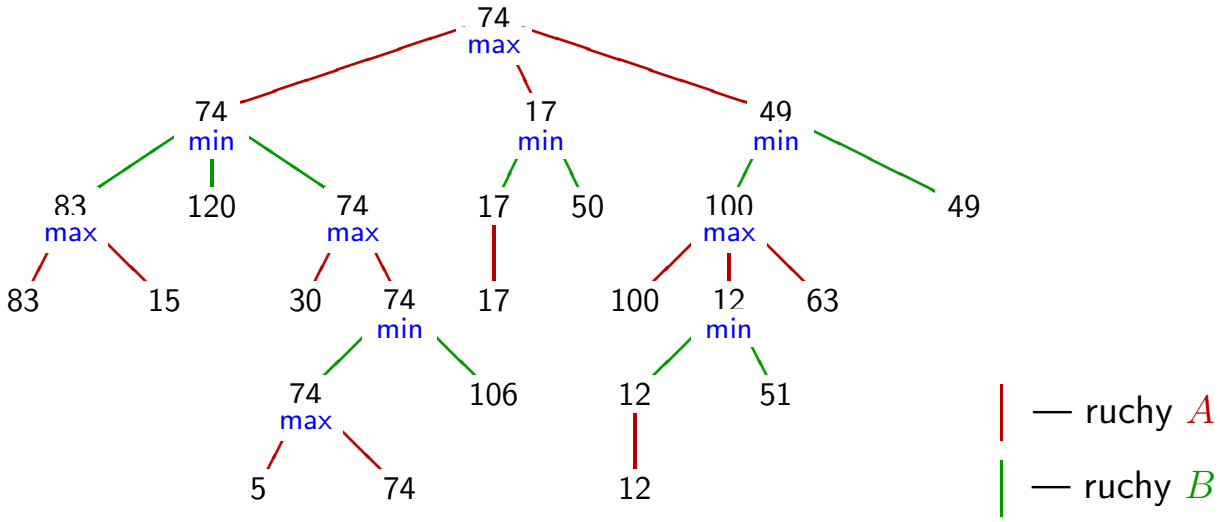
Minimax

Załóżmy, że mamy fukcję w oceniającą wartość sytuacji na planszy z punktu widzenia gracza A: jeśli $w(p_1) > w(p_2)$, to sytuacja p_1 jest dla A *korzystniejsza* od sytuacji p_2 (więc mniej korzystna dla B). W drzewie gry wpisujemy wartości funkcji w dla różnych sytuacji zaczynając od liści:



Minimax

Załóżmy, że mamy fukcję w oceniającą wartość sytuacji na planszy z punktu widzenia gracza A: jeśli $w(p_1) > w(p_2)$, to sytuacja p_1 jest dla A *korzystniejsza* od sytuacji p_2 (więc mniej korzystna dla B). W drzewie gry wpisujemy wartości funkcji w dla różnych sytuacji zaczynając od liści:



Strategia dla A: zawsze wybierać ruch *maksymalizujący* wartość w.
Strategia dla B: zawsze wybierać ruch *minimalizujący* wartość w.