Základy umělé inteligence

Hry v extenzivní formě, Algoritmus Minimax

Ing. Tomáš Řehořek

Katedra teoretické informatiky (KTI), Fakulta informačních technologií (FIT)
České vysoké učení technické v Praze (ČVUT)

BI-ZUM, LS 2016/17, 9. předn.

https://edux.fit.cvut.cz/courses/BI-ZUM/

Minulá přednáška

Definovali jsme dvě základní oblasti teorie her:

- hry v **normální formě**
  - hra vyjádřena herní maticí,
- hry v **extenzivní formě**
  - hra vyjádřena herním stromem.

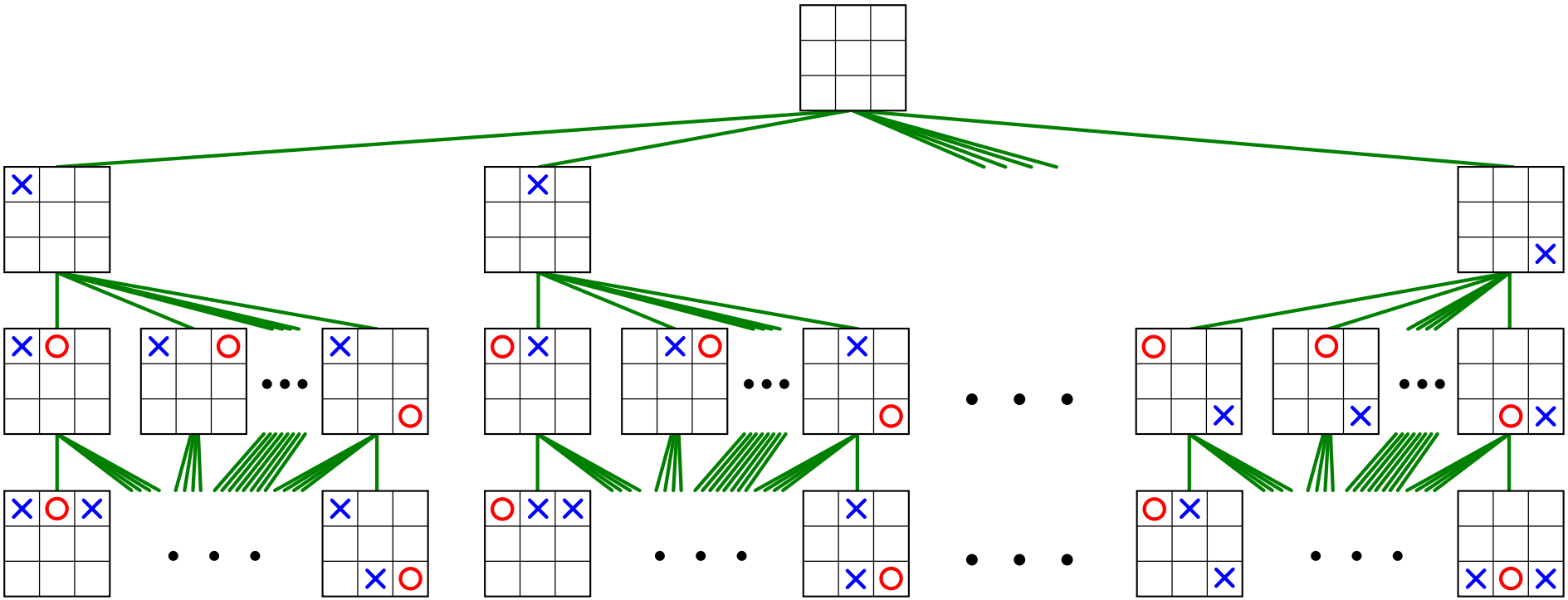
Rozbrali jsme hry v **normální formě**:

- kanonické hry v NF,
- analýza akčních profilů
  - Paretova optima,
  - Nashova equilibria.

Dnešní přednáška: **Hry v extenzivní formě**

- algoritmy prohledávání herního stromu

Co jsou hry v extenzivní formě?



Příklady

Příklady her, které lze přirozeně reprezentovat v extenzivní formě:

- Piškvorky,
- Šachy,
- Dáma,
- Reversi,
- Go,
- ...

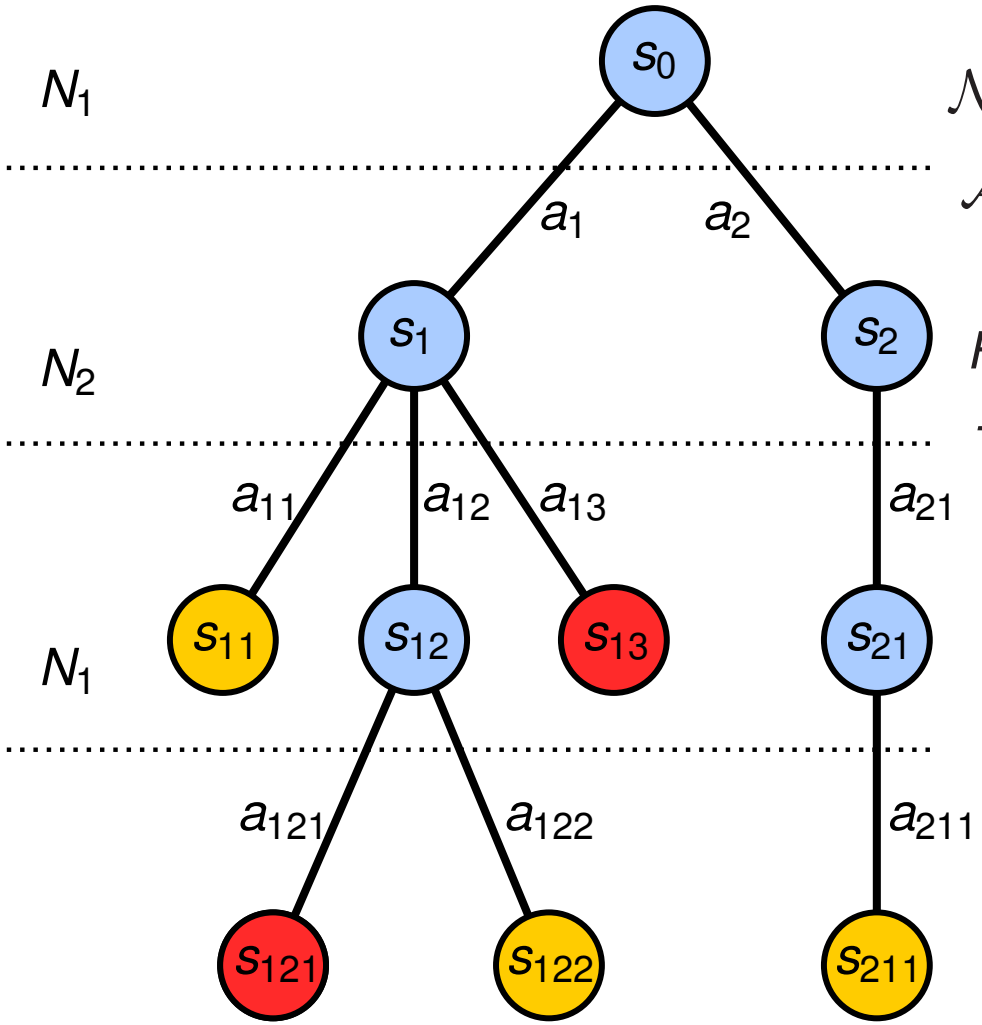
Hry v extenzivní formě

Definice: Konečná hra v extenzivní formě

Konečná hra v extenzivní formě pro n hráčů je osmice $(\mathcal{N}, \mathcal{A}, H, T, \chi, \rho, \sigma, u)$

- $\mathcal{N} = \{N_1, \ldots, N_n\}$ je konečná množina **hráčů**,
- $\mathcal{A}$ je množina **akcí**,
- H je množina **rozhodovacích uzlů**,
- $\chi\colon H \to 2^{\mathcal{A}}$ popisuje, které akce jsou v daném uzlu k dispozici,
- $\rho\colon H \to \mathcal{N}$ říká, který hráč je v daném uzlu na tahu,
- T je množina terminálních uzlů, $T \cap H = \{\}$,
- σ je prostá parciální **funkce následnictví** $\sigma\colon H \times \mathcal{A} \to H \cup T$
  - $\forall h_1, h_2 \in H \, \forall a_1, a_2 \in \mathcal{A}\colon \sigma(h_1, a_1) = \sigma(h_2, a_2) \Rightarrow (h_1 = h_2 \wedge a_1 = a_2)$,
  - díky tomu, že je σ prostá, má graf rozhodovacích uzlů a terminálů tvar **stromu**, který označujeme jako **herní strom**
- $u = (u_1, \ldots, u_n)$, kde $u_i\colon T \to \mathbb{R}$ vyjadřuje utilitu hráče N_i v jednotlivých terminálních uzlech (listech stromu)

Příklad: Triviální hra v extenzivní formě



rozhodovací uzel
vítězství N_1
vítězství N_2

$$\mathcal{N} = \{N_1, N_2\}$$

$$\mathcal{A} = \{a_1, a_2, a_{11}, a_{12}, a_{13}, a_{21}, a_{121}, a_{122}, a_{211}\}$$

$$H = \{s_0, s_1, s_2, s_{12}, s_{21}\}$$

$$T = \{s_{11}, s_{13}, s_{121}, s_{122}, s_{211}\}$$

$$\chi: s_0 \mapsto \{a_1, a_2\}, s_1 \mapsto \{a_{11}, a_{12}, a_{13}\}, \ldots, s_{21} \mapsto \{a_{211}\}$$

$$\rho: s_0, s_{12}, s_{21} \mapsto N_1, s_1, s_2 \mapsto N_2$$

$$\sigma: (s_0, a_1) \mapsto s_1, (s_0, a_2) \mapsto s_2, (s_1, a_{11}) \mapsto s_{11}, (s_1, a_{12}) \mapsto s_{12}, \ldots, (s_{21}, a_{211}) \mapsto s_{211}$$

$$u_1: s_{11}, s_{122}, s_{211} \mapsto 1, s_{121}, s_{13} \mapsto -1$$

$$u_2: s_{11}, s_{122}, s_{211} \mapsto -1, s_{121}, s_{13} \mapsto 1$$

Dvouhráčové zero-sum hry

- předchozí definice uvažuje obecný počet hráčů,
- prominentní postavení však mají **dvouhráčové**, **zero-sum** hry:
  - již známe z her v normální formě

Definice: Dvouhráčová zero-sum hra

Dvouhráčová zero-sum hra v extenzivní formě je hra v extenzivní formě $(\mathcal{N}, \mathcal{A}, H, T, \chi, \rho, \sigma, u)$, kde:

1. $|\mathcal{N}| = 2$,
2. $u = (u_1, u_2)$,
3. $\forall t \in T\colon u_1(t) + u_2(t) = 0$.

Motivace: šachy, dáma, piškvorky, ...

- v terminálním uzlu t nastává vždy jeden z následujících stavů:
  - hráč N_1 vyhrává, hráč N_2 prohrává $\rightsquigarrow u_1(t) = 1, u_2(t) = -1$,
  - hráč N_1 prohrává, hráč N_2 vyhrává $\rightsquigarrow u_1(t) = -1, u_2(t) = 1$,
  - remíza $\rightsquigarrow u_1(t) = 0, u_2(t) = 0$.

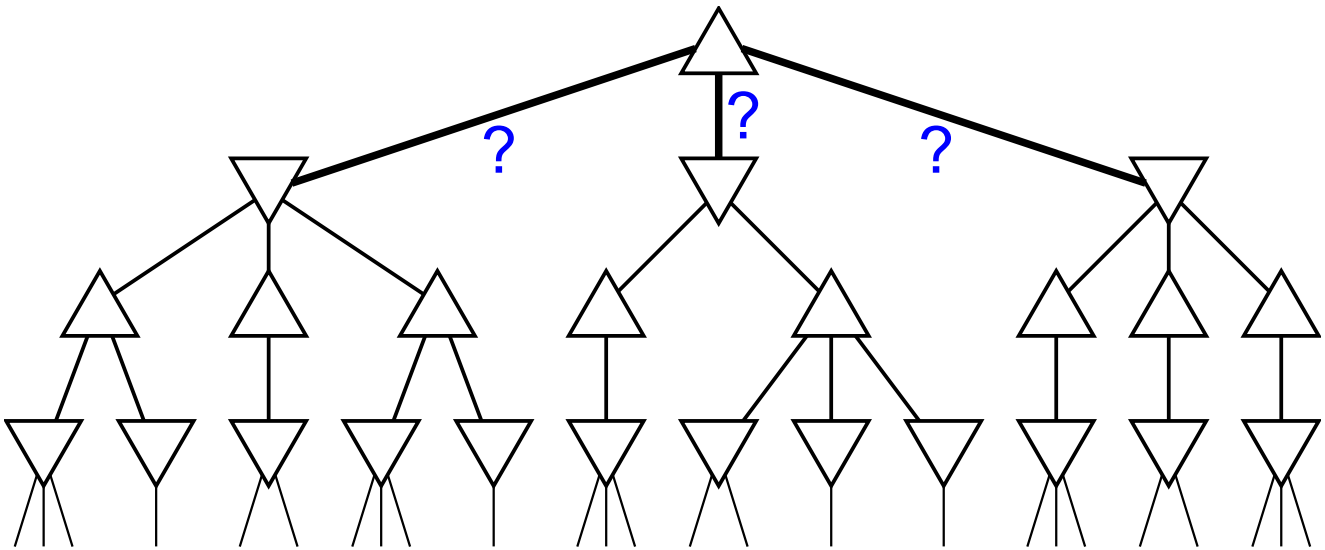
Velikost herních stromů v praxi

Hra v extenzivní formě indukuje herní strom, typicky sestávající z obrovského množství uzlů:

- **Piškvorky** 3×3 **(Tic-Tac-Toe)**
  - dětská hra, triviální pro dospělého člověka,
  - 5478 platných konfigurací,
  - 255168 možných průběhů,
- **Dáma (Checkers)**
  - $\approx 10^{20}$ platných konfigurací,
  - $\approx 10^{40}$ možných průběhů,
- **Šachy (Chess)**
  - $\approx 10^{45}$ platných konfigurací,
  - $\approx 10^{123}$ možných průběhů,
    - ★ jeden z prvních odhadů je znám jako Shannonovo číslo: 10^{120}

Hráči *MIN* a *MAX*

- uvažujeme-li zero-sum dvouhráčovou hru, lze utilitní funkce obou hráčů nahradit jedinou funkcí $u\colon T \to \mathbb{R}$,
- hráče pak můžeme intuitivně pojmenovat:
  - ***MAX*** – uzly v herním stromu kreslíme symbolem $\triangle$
    - ★ hráč, který je na tahu v kořeni herního stromu,
    - ★ jeho snahou je u **maximalizovat**,
  - ***MIN*** – uzly v herním stormu kreslíme symbolem $\triangledown$
    - ★ hráč, na něhož nahlížíme jako na soupeře,
    - ★ jeho snahou je u **minimalizovat**,



Perfektní hra (Perfect play)

Pokud máme k dispozici kompletní herní strom, můžeme tahy rozdělit na:

- **bezchybné**
  - hráč *MAX* ve stavu s zahrál akci a^* maximalizující zaručenou minimální hodnotu u v terminálu podstromu (při pesimistickém očekávání reakce hráče *MIN*)
  - **příklad**: hráč může hru vyhrát a skutečně zvolil akci, která k výhře směřuje.
- **chybné**
  - hráč zahrál suboptimální akci vzhledem k pesimitistickému odhadu reakce soupeře, jeho pozice se zhoršila,
  - **příklad**: hráč mohl hru zaručeně vyhrát, ovšem zvolil akci, která soupeři umožňuje prohru zvrátit,

Řekneme, že hráč hraje **perfektní hru**, pokud nedělá chyby, tj. pokud **v každém tahu volí bezchybnou akci**.

Perfektní hra (Perfect play)

≫ *The unlimited intellect assumed in the theory of games, on the other hand, never makes a mistake and a smallest winning advantage is as good as mate in one. A game between two such mental giants, Mr. A and Mr. B, would proceed as follows. They sit down at the chessboard, draw the colours, and then survey the pieces for a moment. Then either*

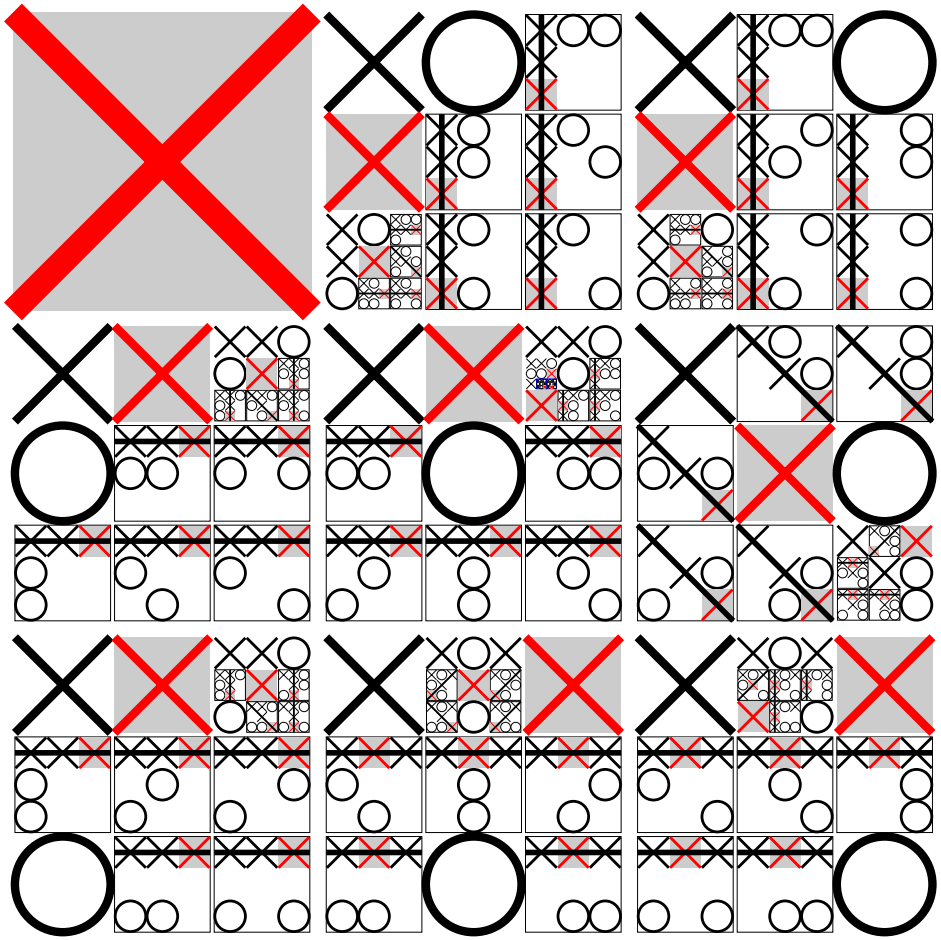
(1) Mr. A says, "I resign" or

(2) Mr. B says, "I resign" or

(3) Mr. A says, "I offer a draw," and Mr. B replies, "I accept." ≪

Claude E. Shannon, 1950

Ilustrace: Perfektní hra pro hráče × v Tic-Tac-Toe



(Převzato z Wikipedie)

Perfektní hra v praxi

- většina her staví na tom, že je nesmírně obtížné hrát je perfektně,
  - kombinatorická exploze známá ze všech oblastí zájmu AI,
- není duševně ani výpočetně únosné zvážit všechny možné průběhy hry
  - hra šachy má 10^{45} platných konfigurací,
- praktické hraní komplikovaných her nezbytně využívá **aproximací** a **heuristik**
  - např. kvalitu dané konfigurace v šachách můžeme vyjádřit pomocí „materiálních“ indikátorů:
    - ★ počet pěšáků, střelců, koní, věží a dam na vlastní/soupeřově straně,
- to vede na **prohledávání herního stromu s omezenou hloubkou**
  - uzly expandujeme do hloubky maximálně d,
  - v hloubce d ohodnotíme uzly heuristickou **evaluační funkcí**,
  - heuristických hodnot v listech využijeme k odvození optimální akce v kořenu stromu

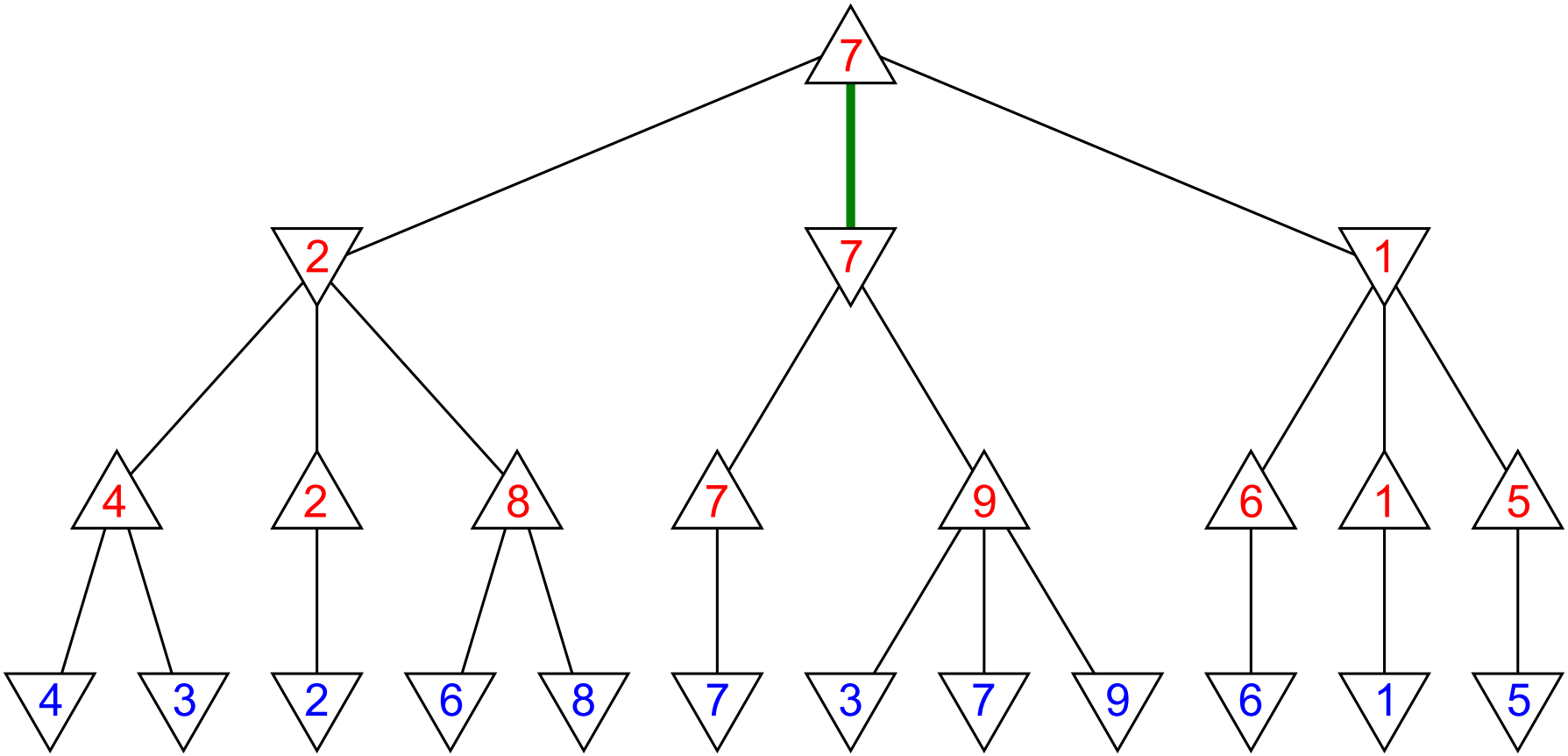
Algoritmus Minimax

- jednoduchý algoritmus, který umožňuje zvolit optimální akci v kořenu stromu vzhledem k ohodnoceným listům,
- předpokládá perfektní hru ze strany soupeře,

Základní princip algoritmu:

1. Průchodem do hloubky generuj kompletní herní strom (až do maximální hloubky d) z počátečního uzlu x_0 (tento uzel je typu *MAX*),
2. Při uzavírání uzlu x anotuj x hodnotou:
   - $eval[x] \leftarrow u(x)$, je-li x terminál,
     - ★ $u(x)$ může být skutečná (pokud je x skutečným terminálem) nebo heuristická (pokud byla expanze ukončena v hloubce d) evaluace uzlu,
   - $eval[x] \leftarrow \max_{a \in \chi(x)} eval[\sigma(x, a)]$, je-li x rozhodovací *MAX* uzel,
   - $eval[x] \leftarrow \min_{a \in \chi(x)} eval[\sigma(x, a)]$, je-li x rozhodovací *MIN* uzel,
3. Vrať akci $a \in \arg \max_{a \in \chi(x_0)} eval[\sigma(x_0, a)]$.

Algoritmus Minimax: Příklad

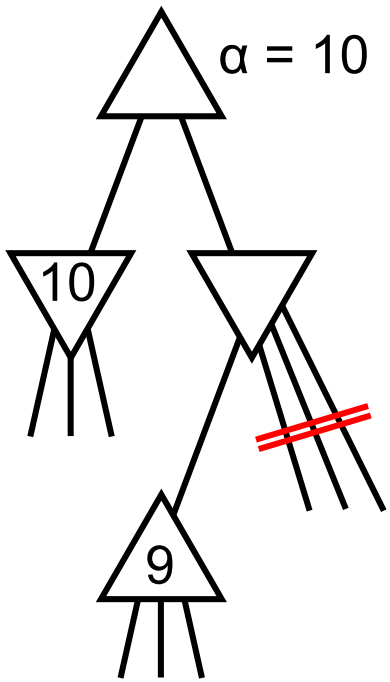


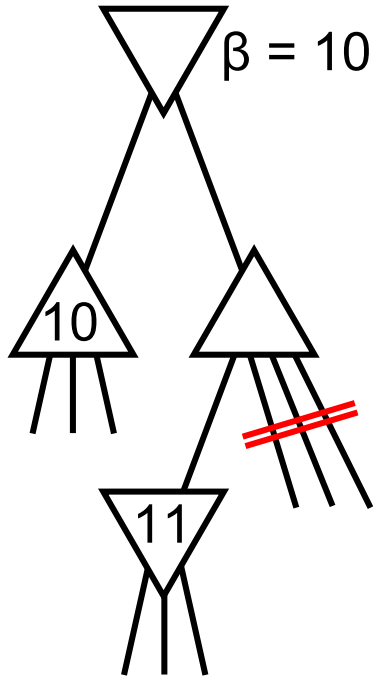
Algoritmus Minimax: Problémy

- prohledávání s omezenou hloubkou d je typicky výpočetně zvládnutelné jen pro malá d
  - průměrný větvicí faktor v šachách je 35,
  - i začátečníci dokáží porazit algoritmus hrubé síly,
  - v BI-ZUM tradiční indikátor úlohy pro AI
- základní variantu algoritmu Minimax je nezbytné **optimalizovat**
  - nejčastější vylepšení: **alfa-beta prořezávání**

Alfa-beta prořezávání

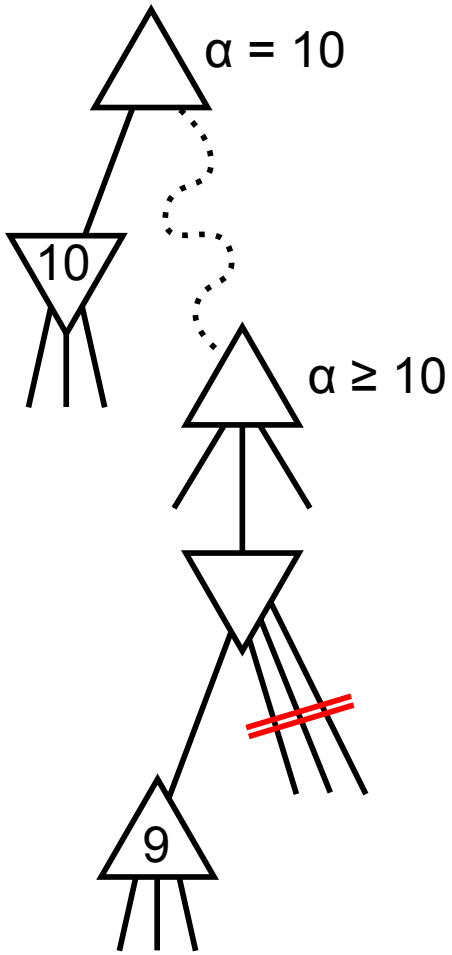
Základní princip

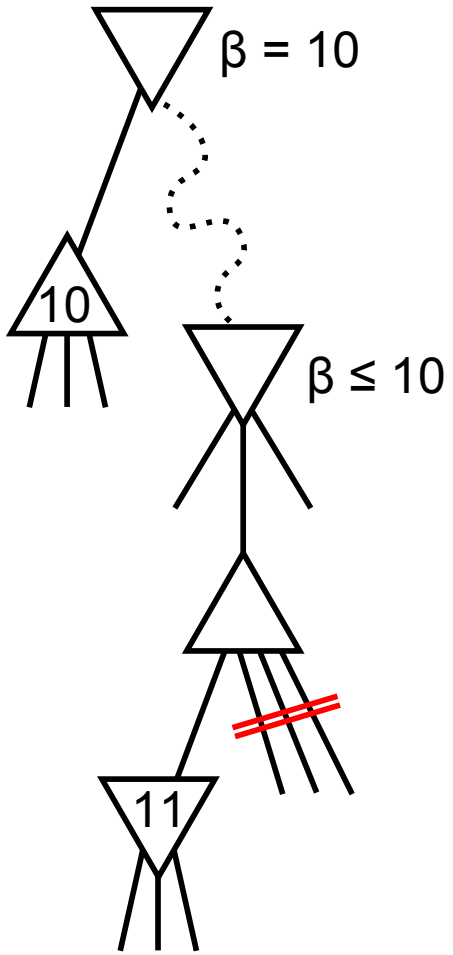




Alfa-beta prořezávání

Zobecněný princip





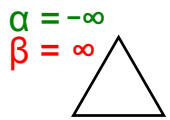
Alfa-beta prořezávání

Algoritmus v každém expandovaném uzlu uchovává dvě hodnoty:

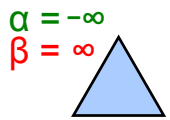
- α – největší utilita, kterou hráč *MIN* zaručeně nesníží při perfektní hře hráče *MAX*...
- β – nejmenší utilita, kterou hráč *MAX* zaručeně nezvýší při perfektní hře hráče *MIN*...

...někde na cestě od aktuálního uzlu směrem ke kořeni (včetně těchto uzlů)

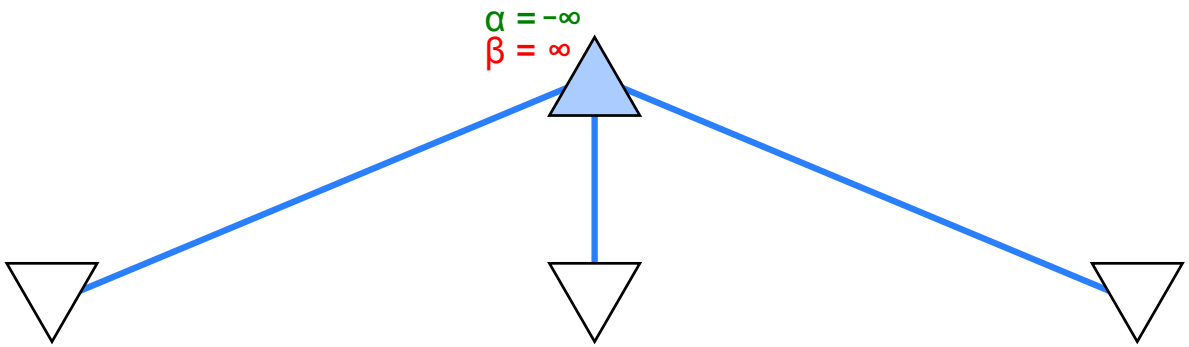
Minimax s alfa-beta prořezáváním: Příklad



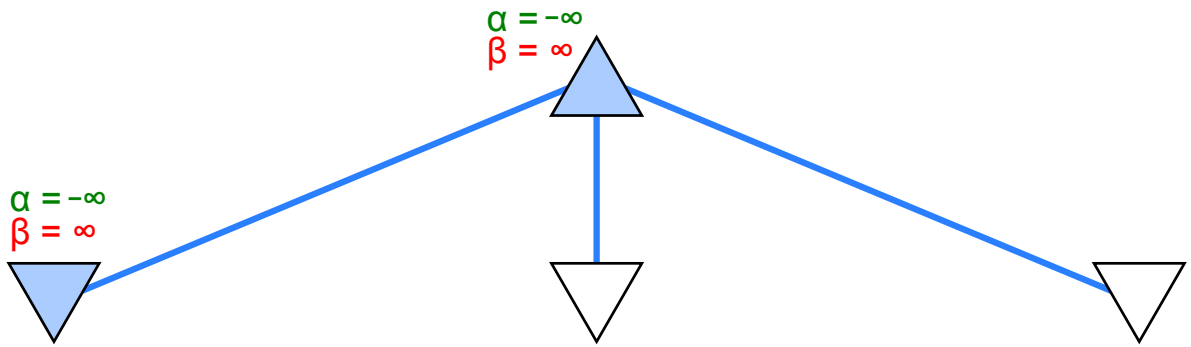
Minimax s alfa-beta prořezáváním: Příklad



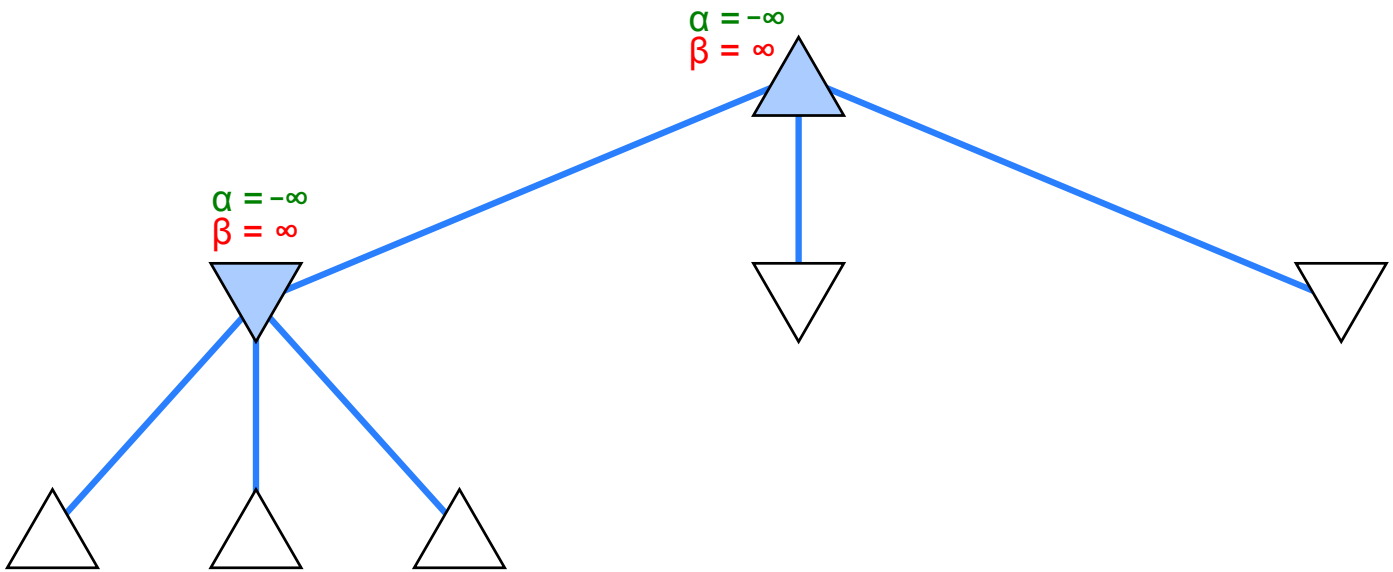
Minimax s alfa-beta prořezáváním: Příklad



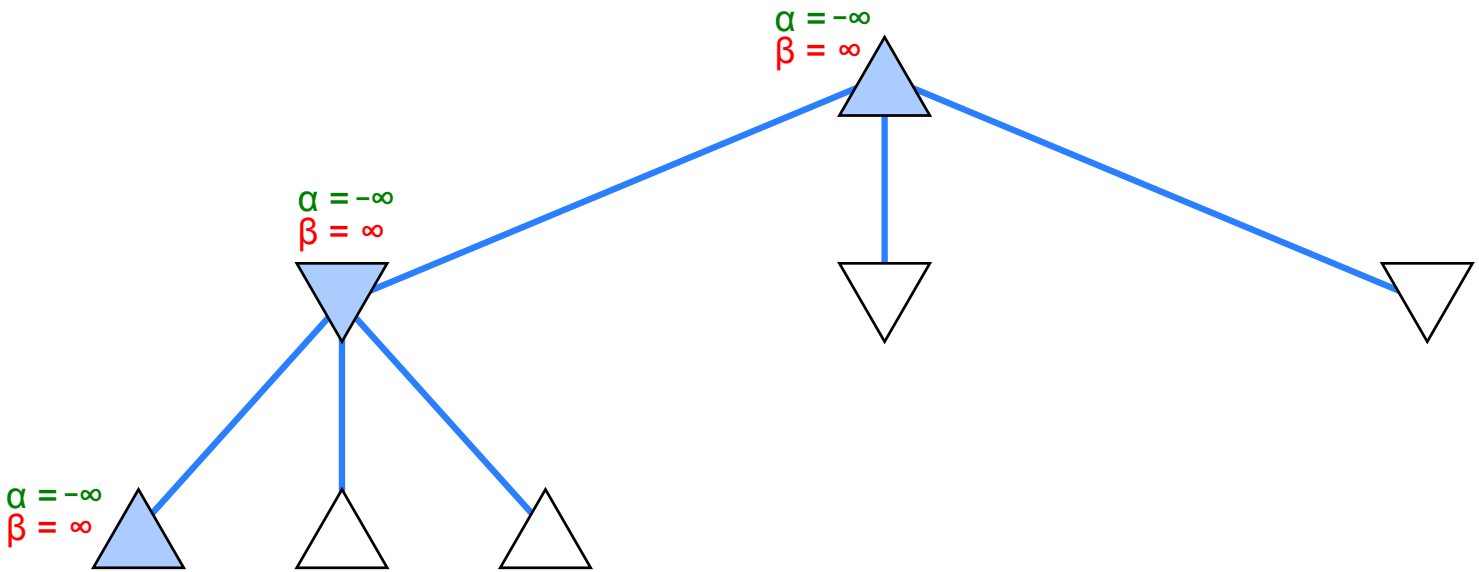
Minimax s alfa-beta prořezáváním: Příklad



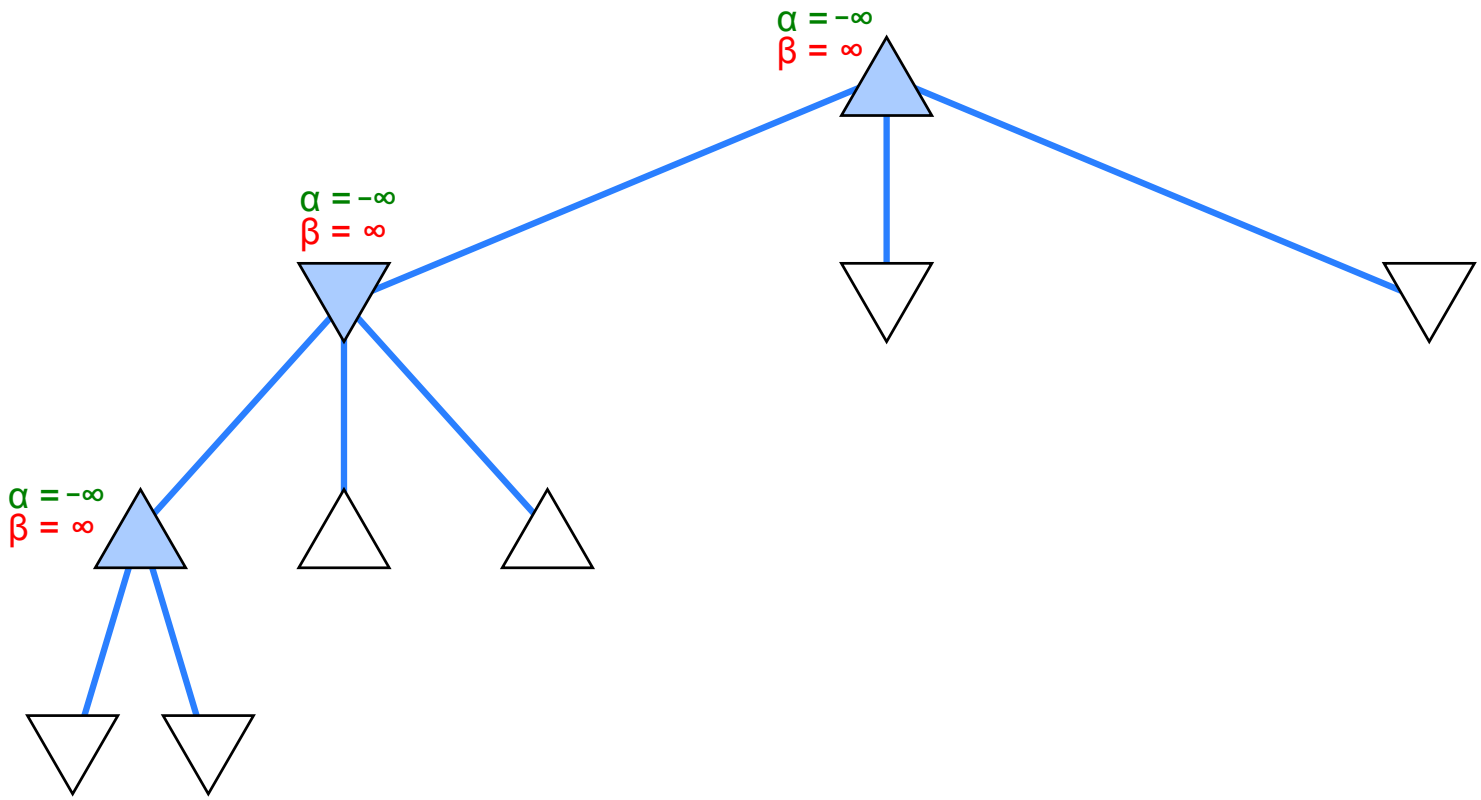
Minimax s alfa-beta prořezáváním: Příklad



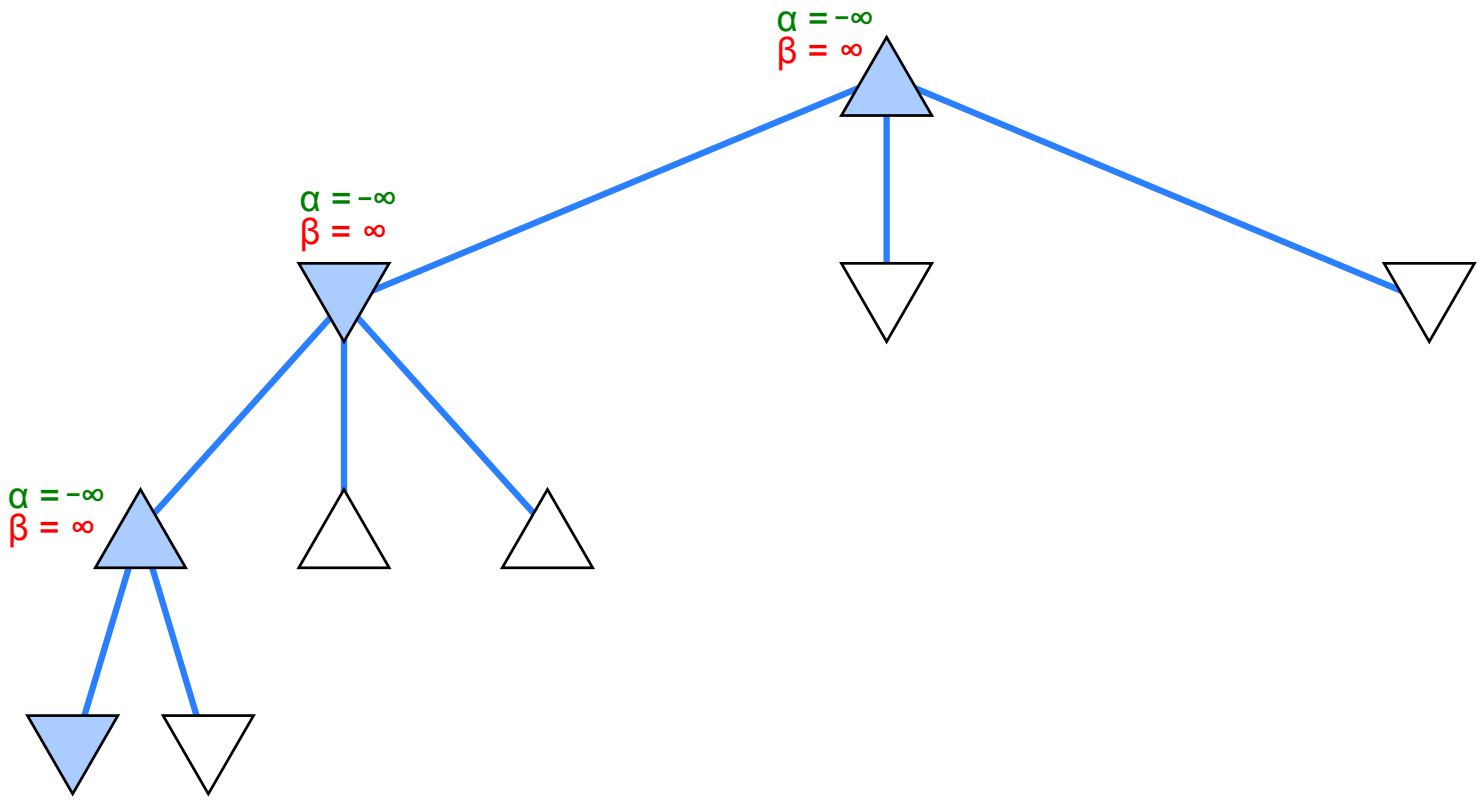
Minimax s alfa-beta prořezáváním: Příklad



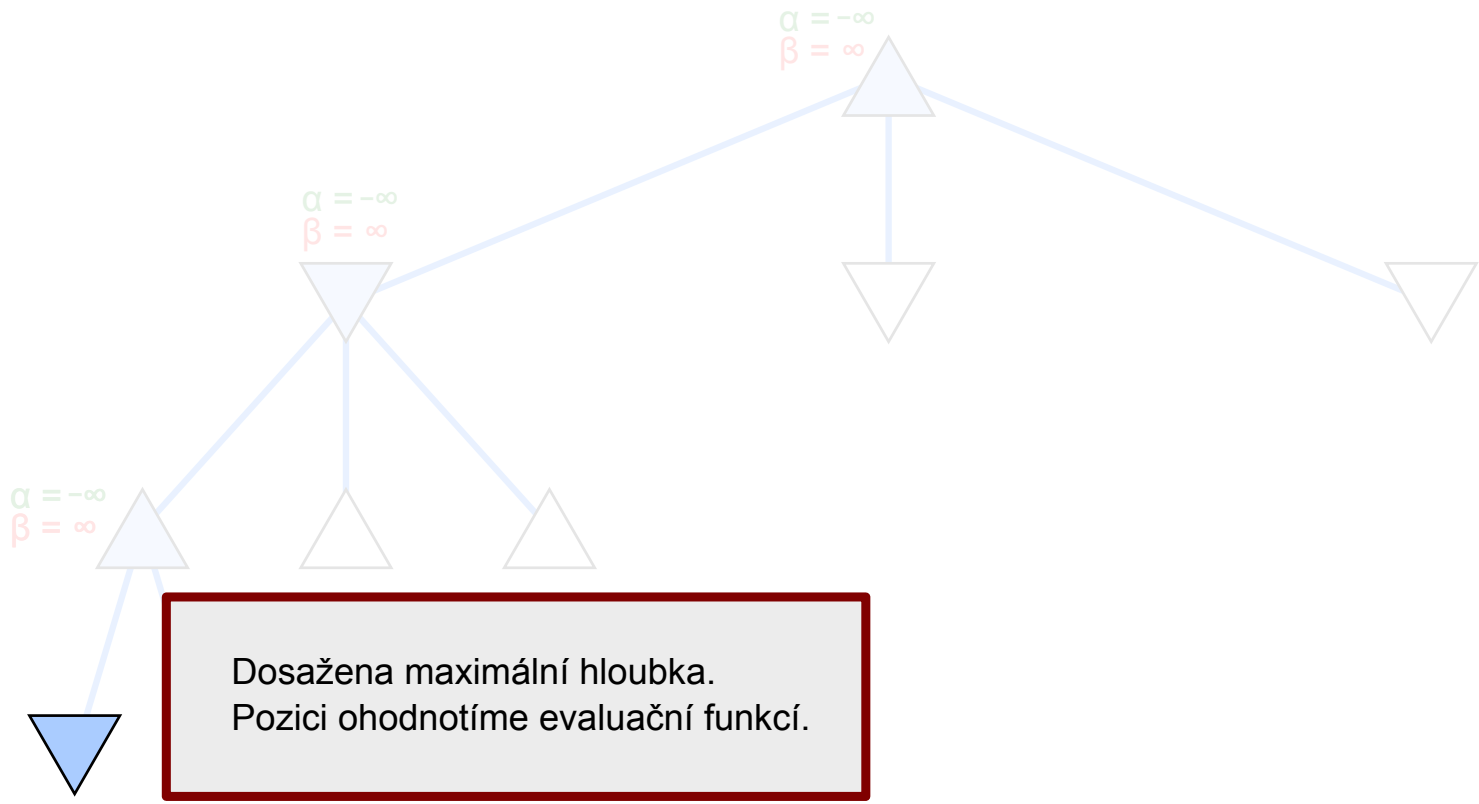
Minimax s alfa-beta prořezáváním: Příklad



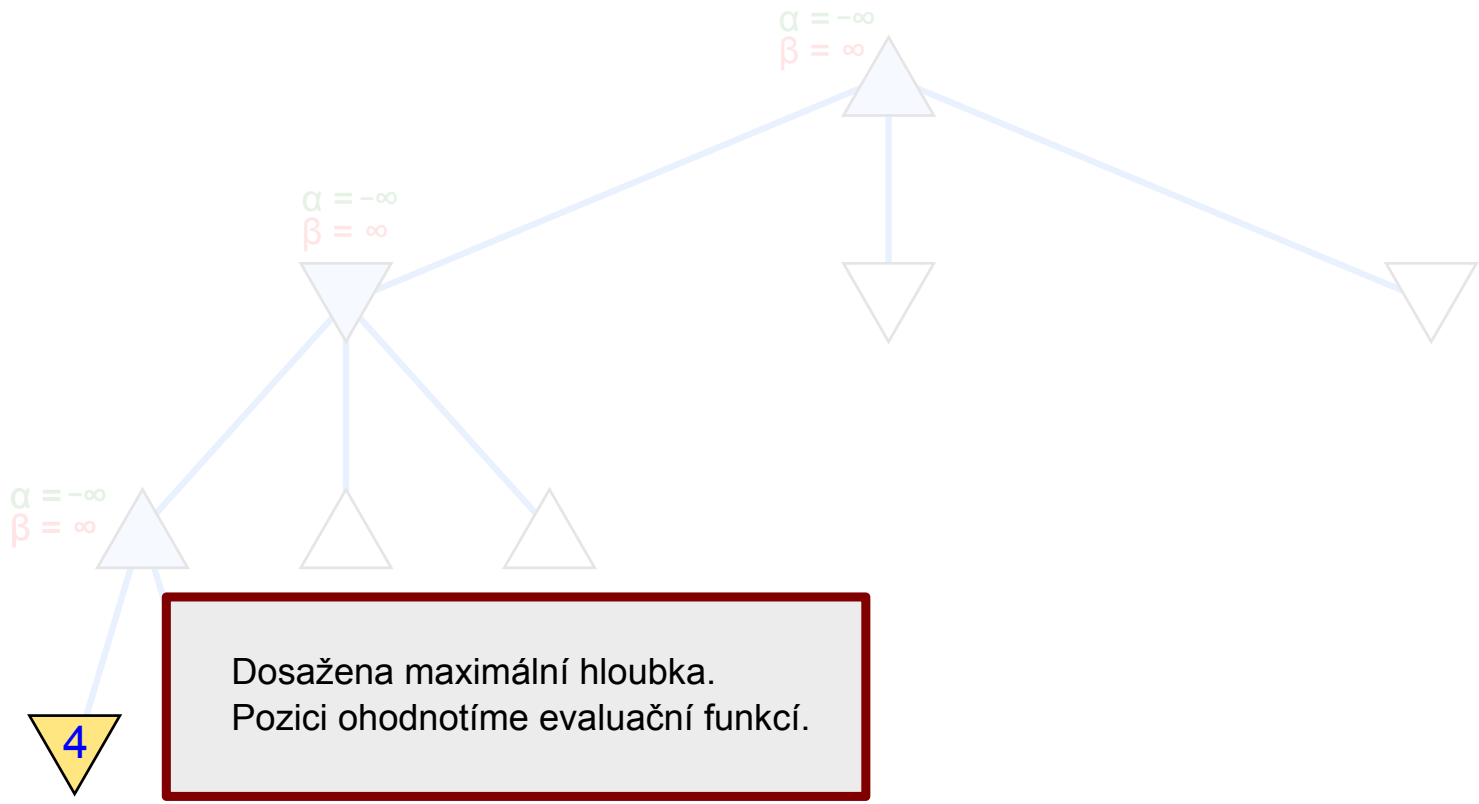
Minimax s alfa-beta prořezáváním: Příklad



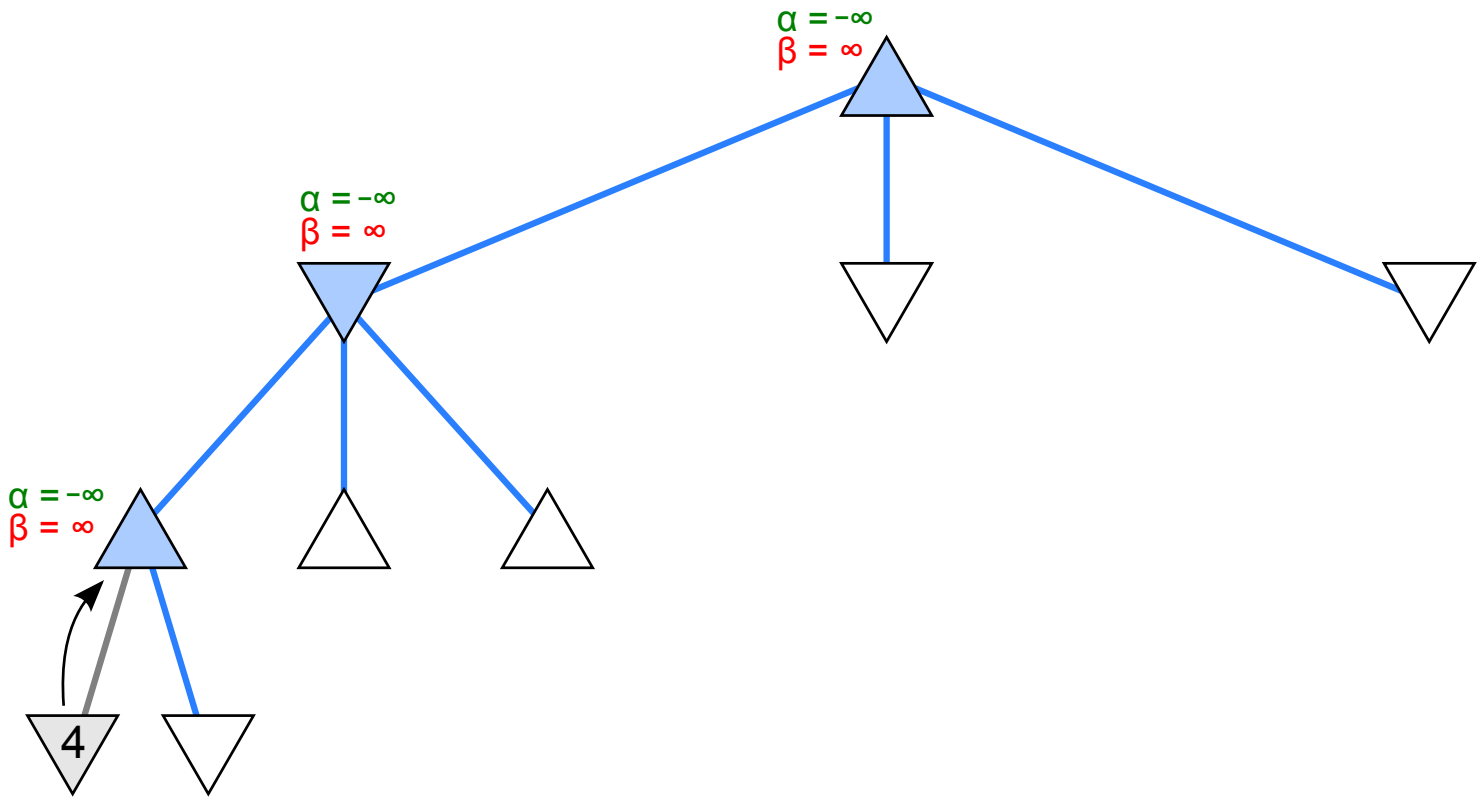
Minimax s alfa-beta prořezáváním: Příklad



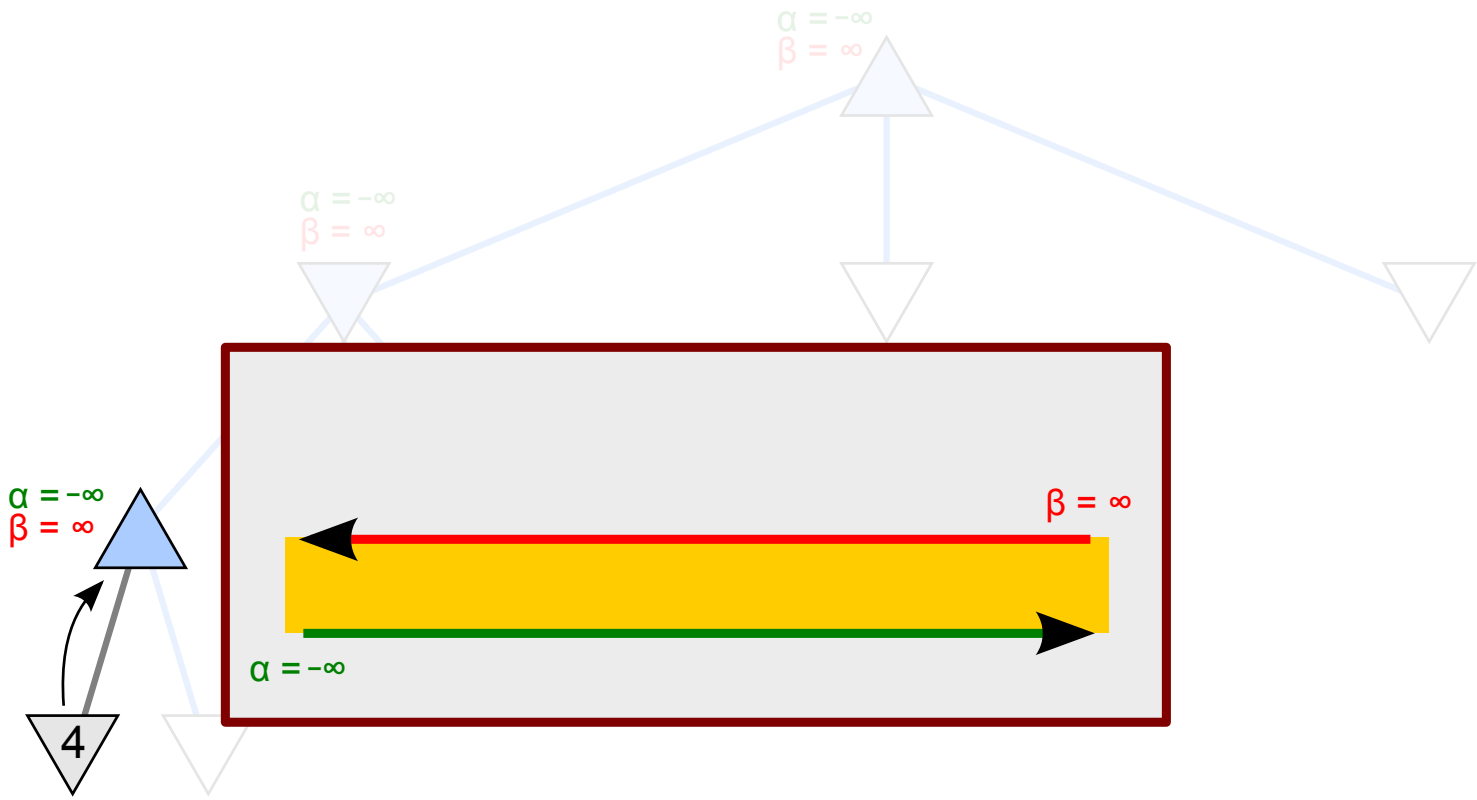
Minimax s alfa-beta prořezáváním: Příklad



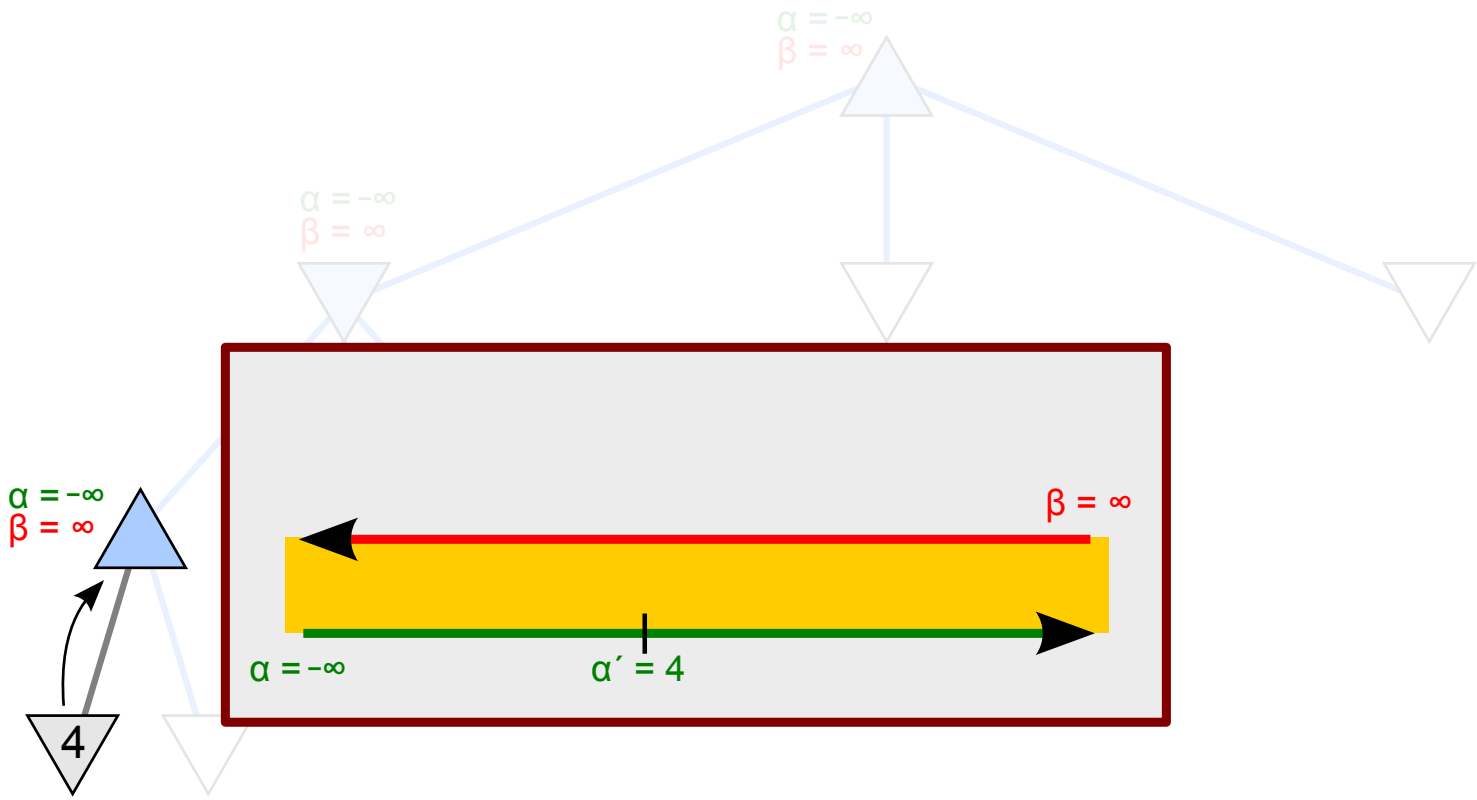
Minimax s alfa-beta prořezáváním: Příklad



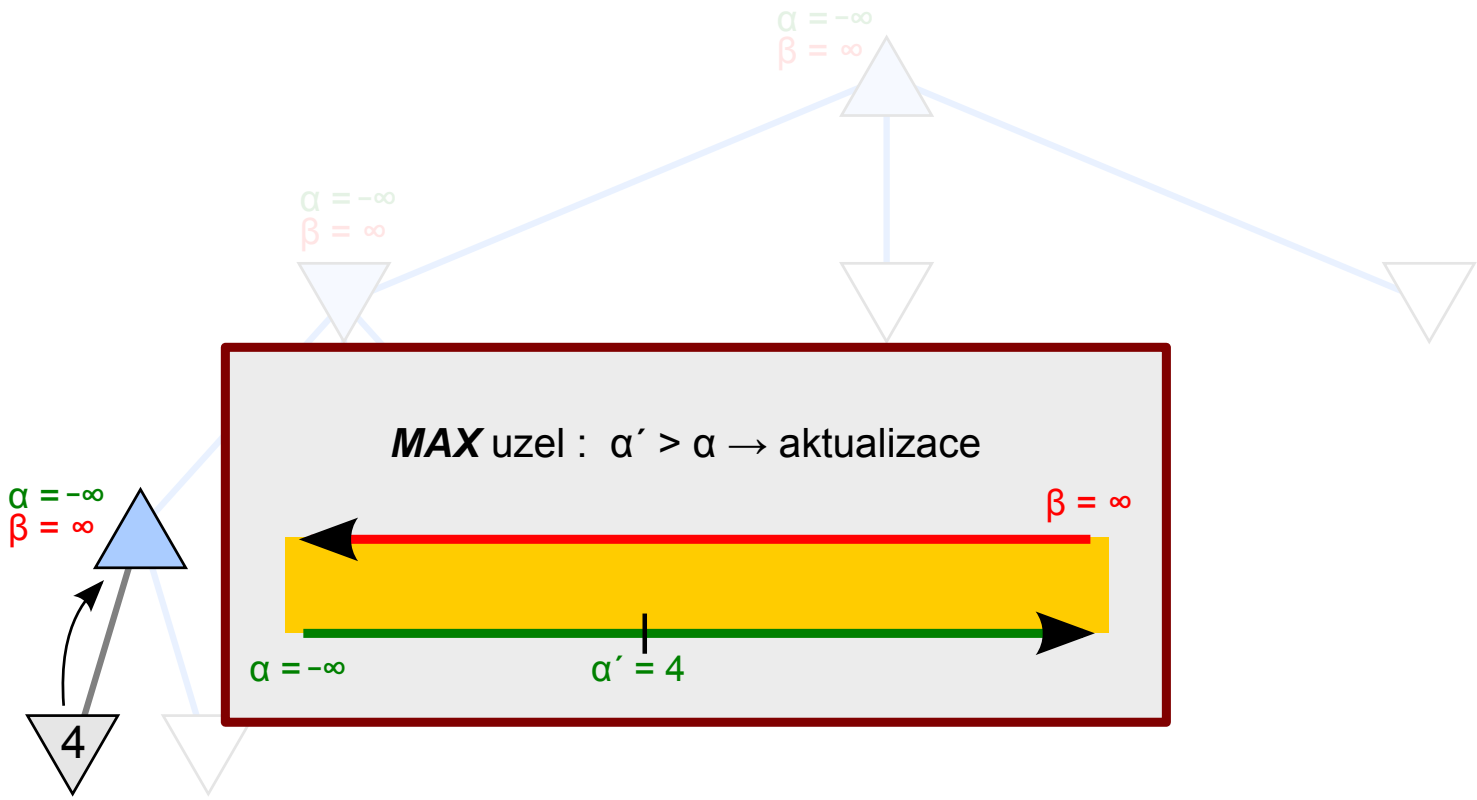
Minimax s alfa-beta prořezáváním: Příklad



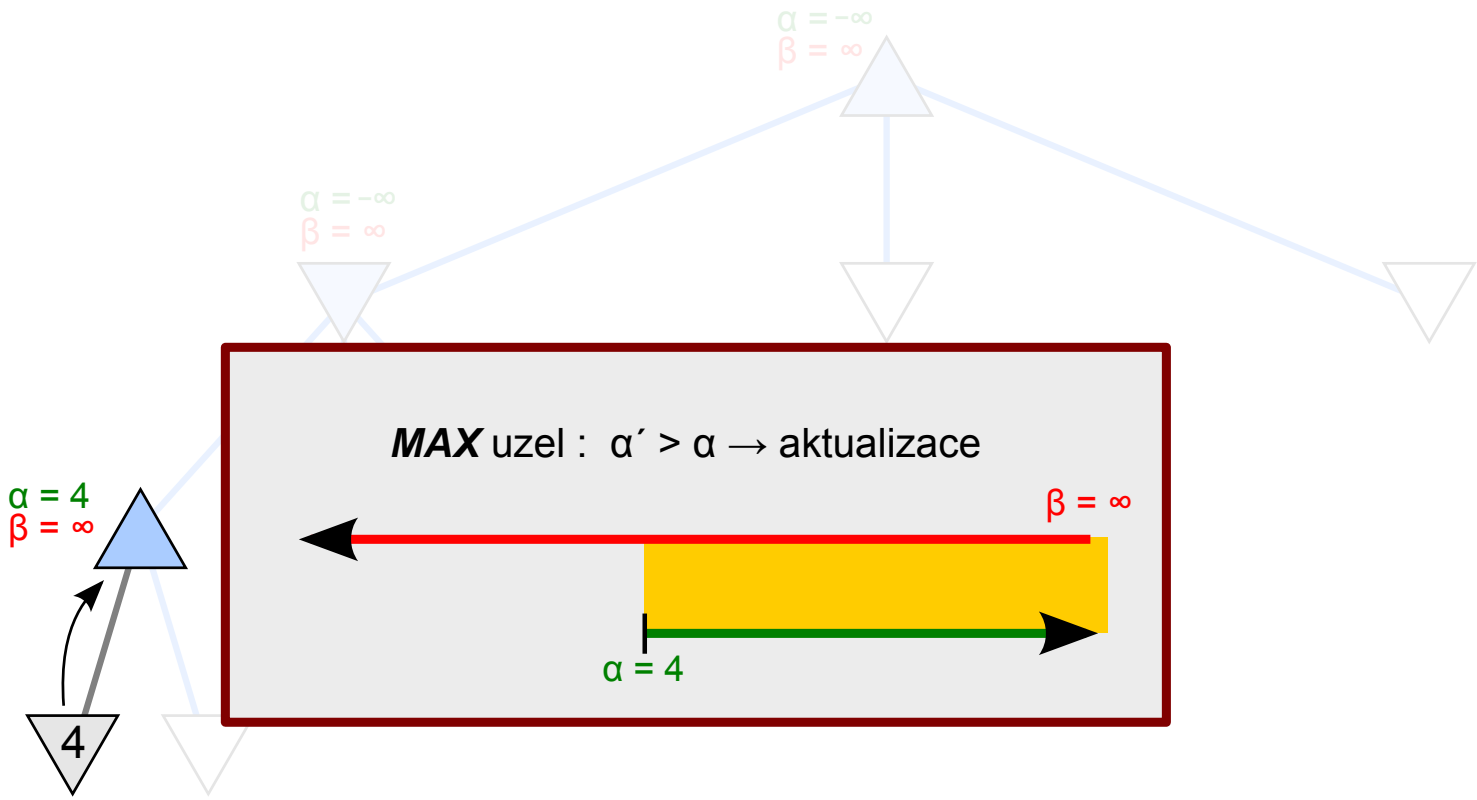
Minimax s alfa-beta prořezáváním: Příklad



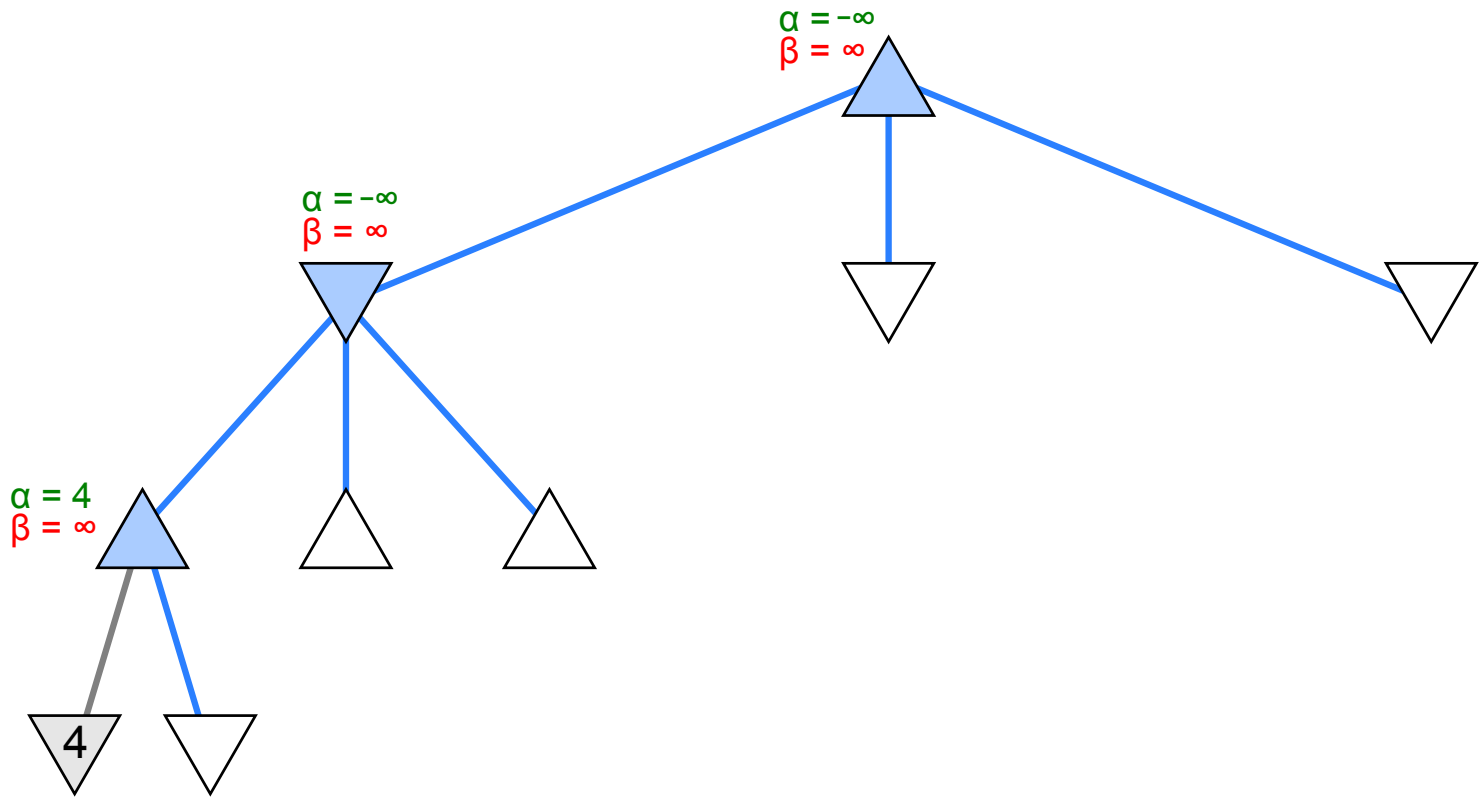
Minimax s alfa-beta prořezáváním: Příklad



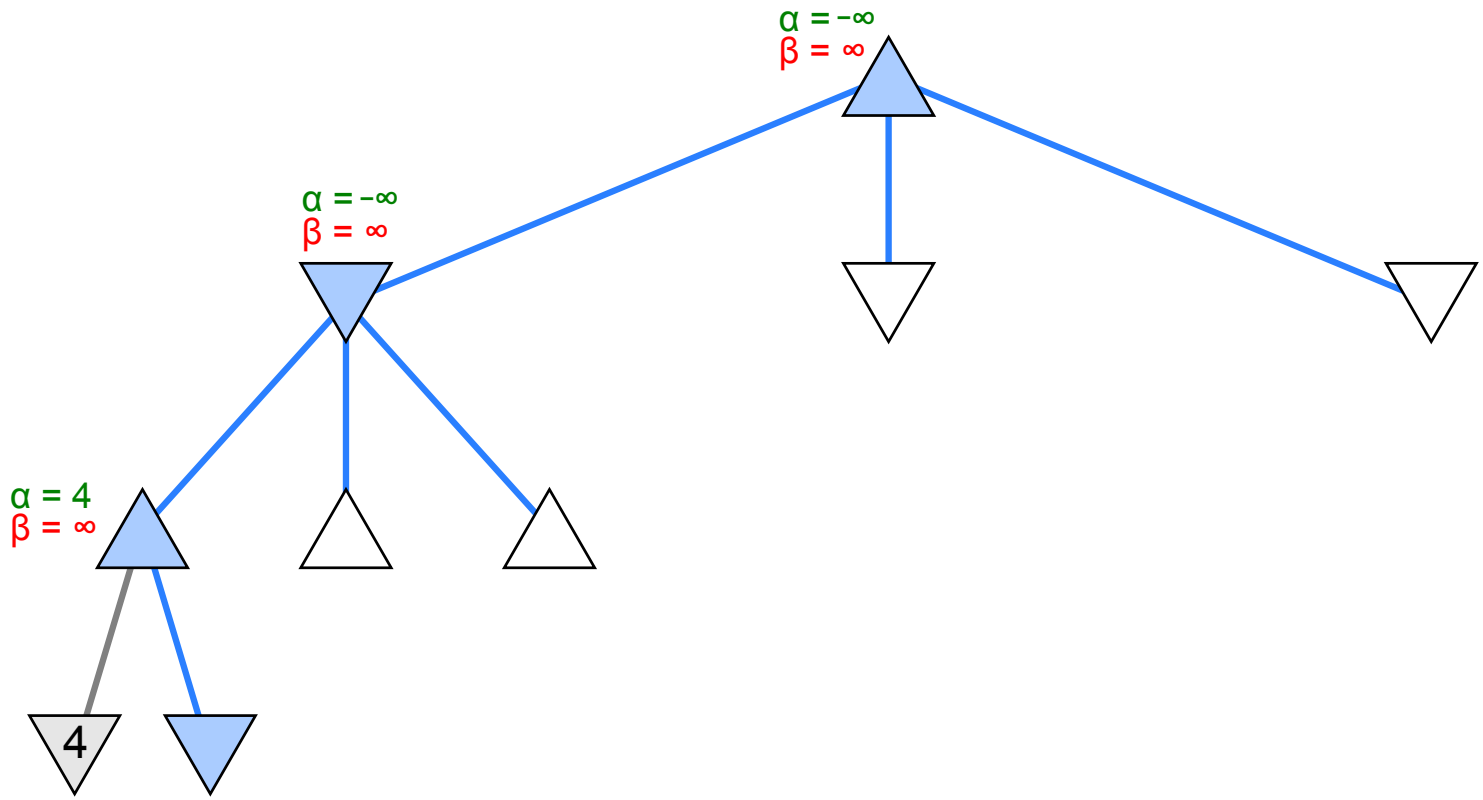
Minimax s alfa-beta prořezáváním: Příklad



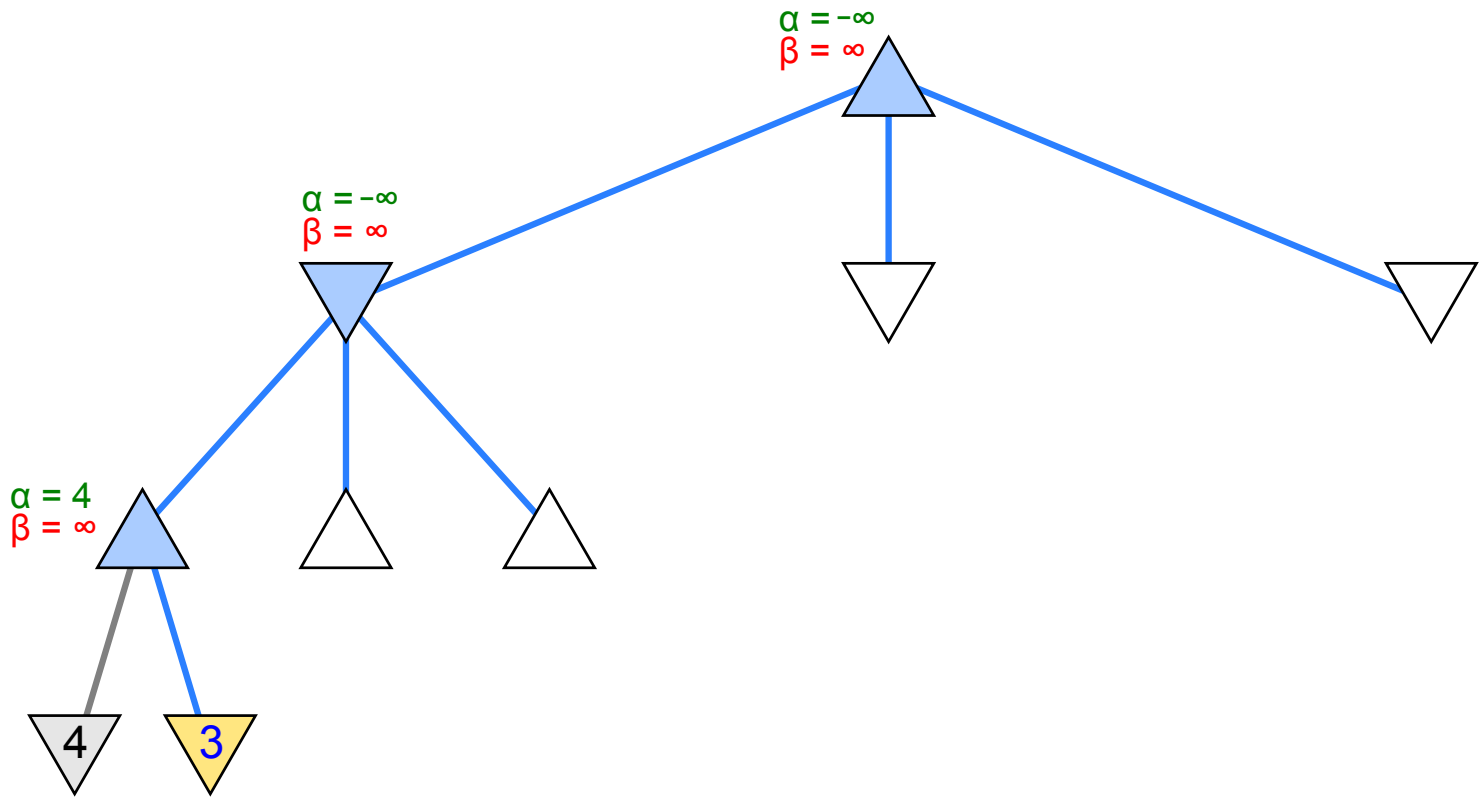
Minimax s alfa-beta prořezáváním: Příklad



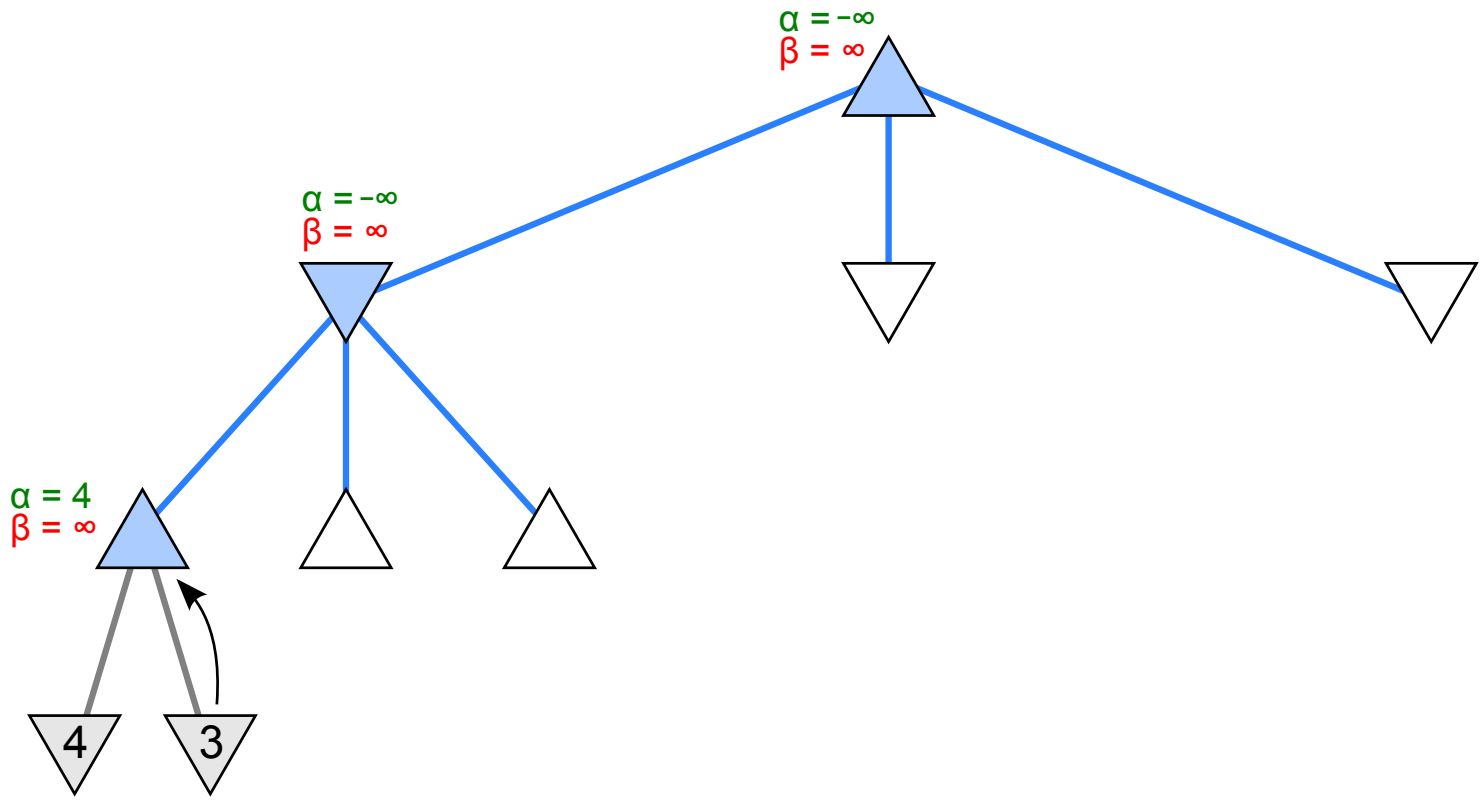
Minimax s alfa-beta prořezáváním: Příklad



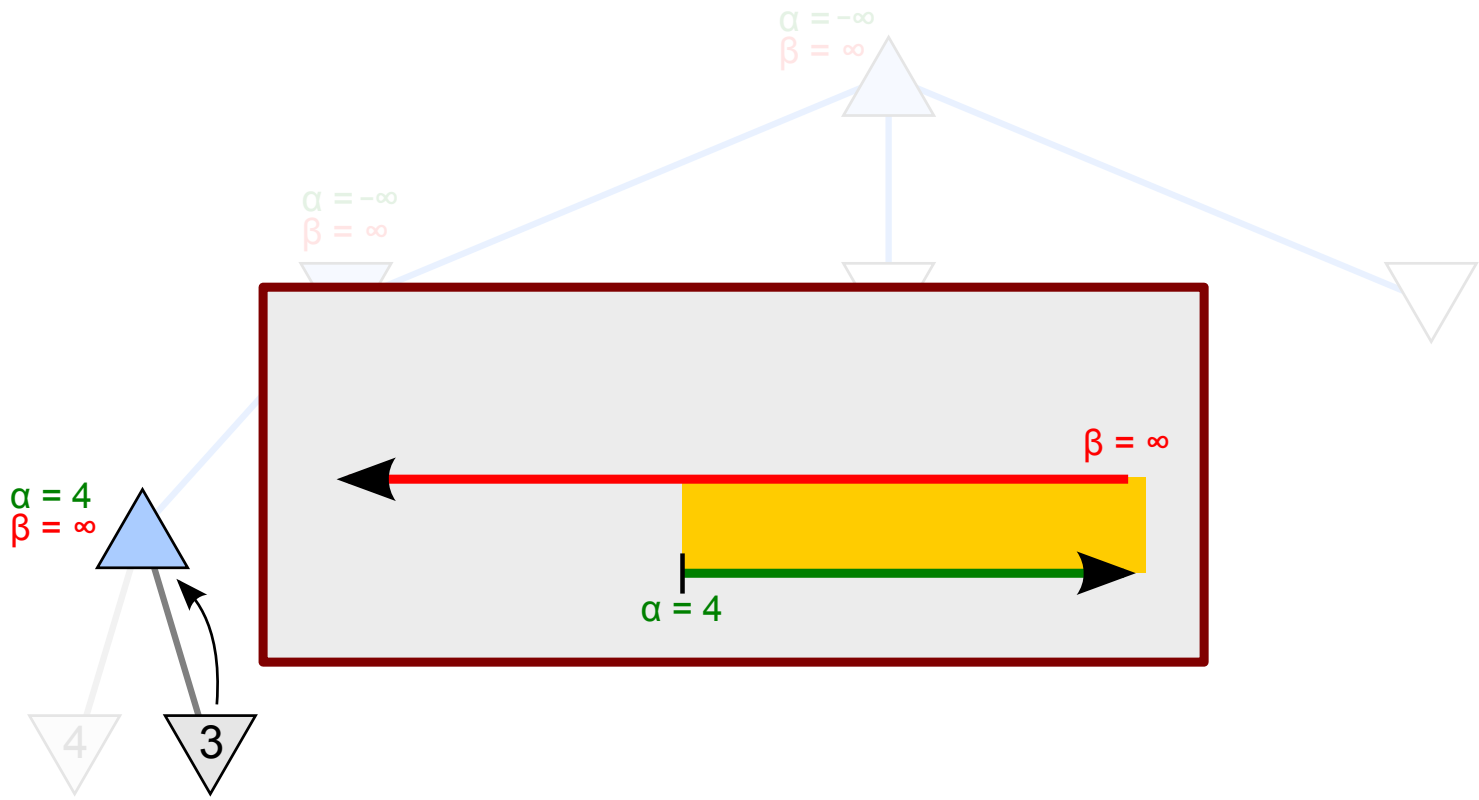
Minimax s alfa-beta prořezáváním: Příklad



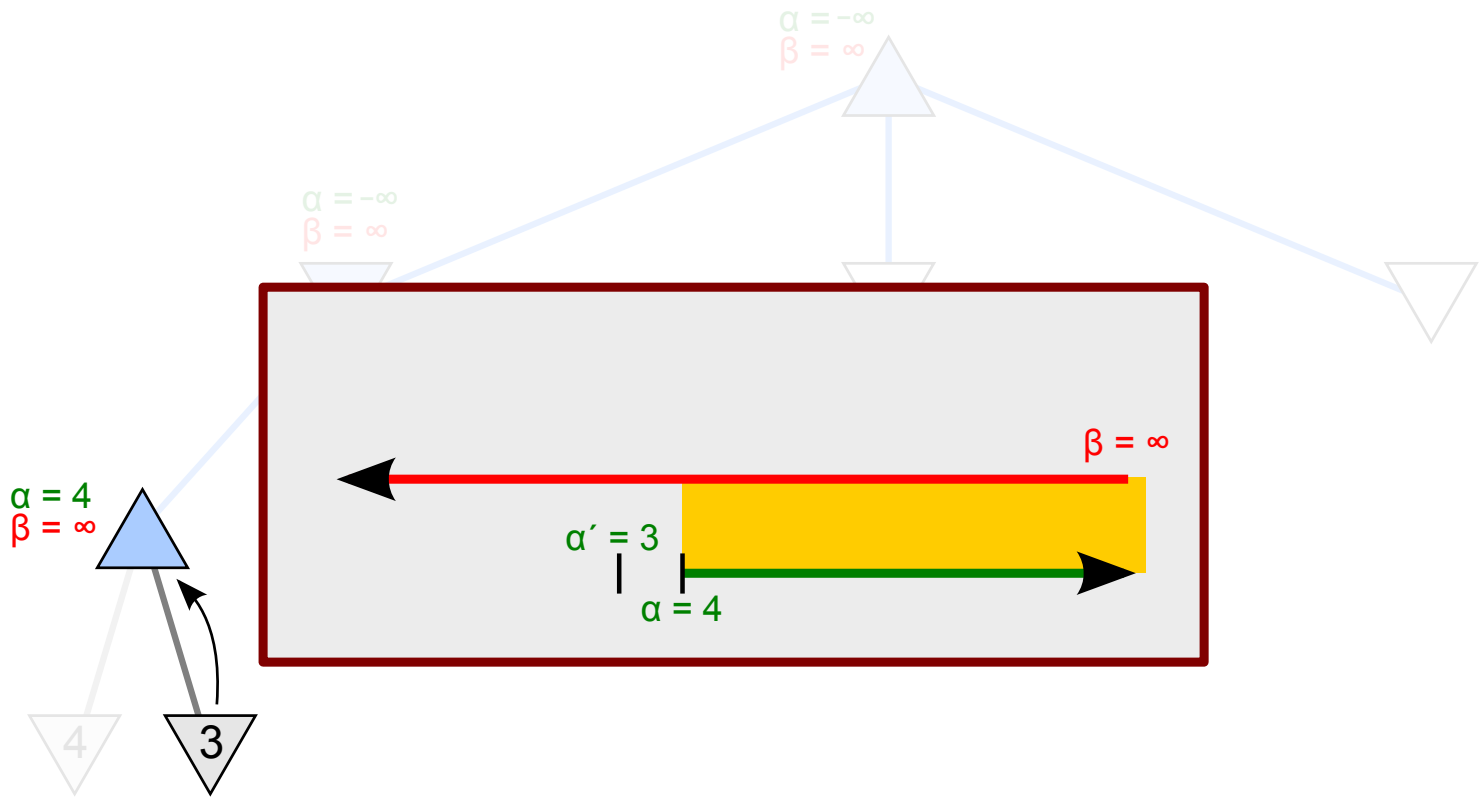
Minimax s alfa-beta prořezáváním: Příklad



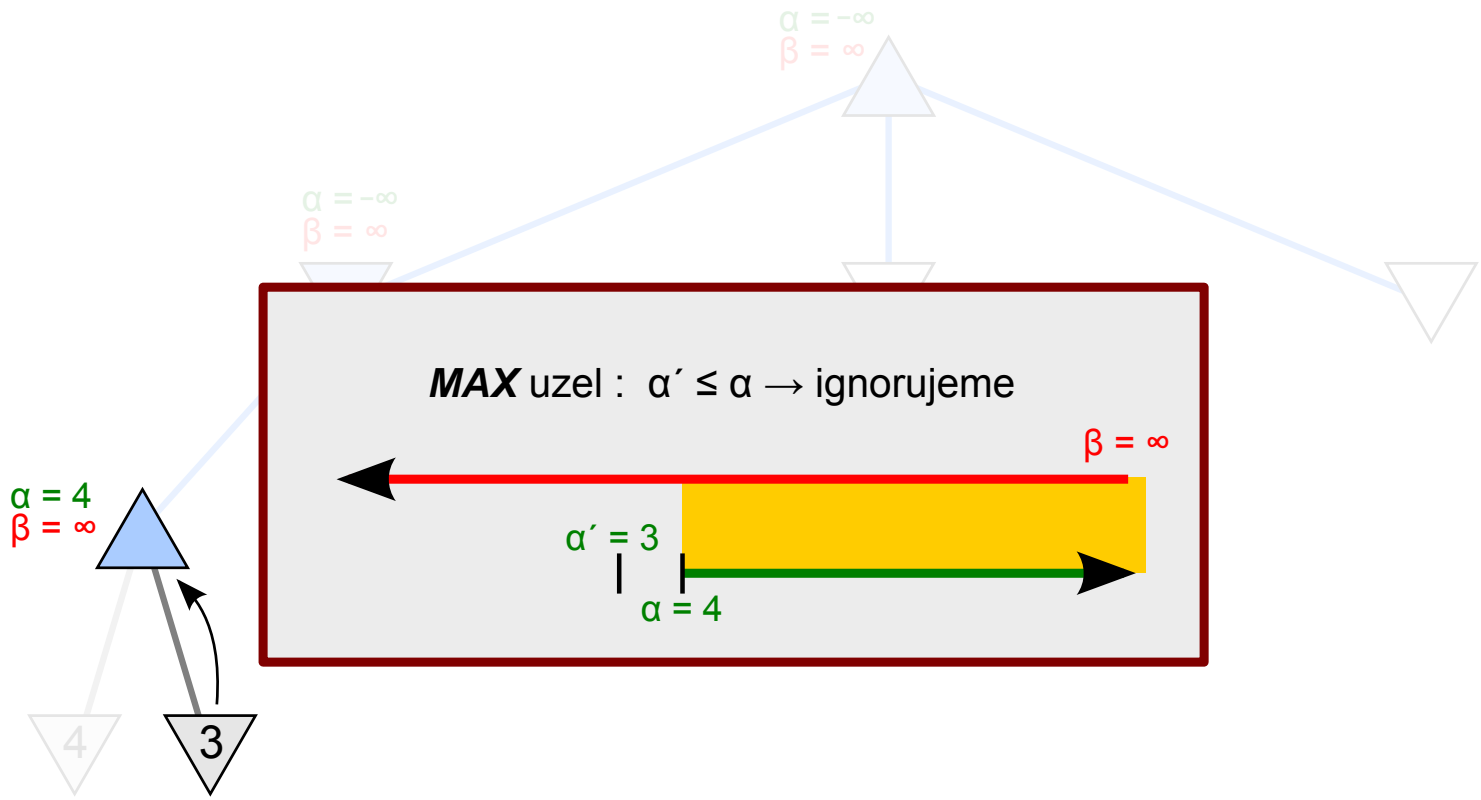
Minimax s alfa-beta prořezáváním: Příklad



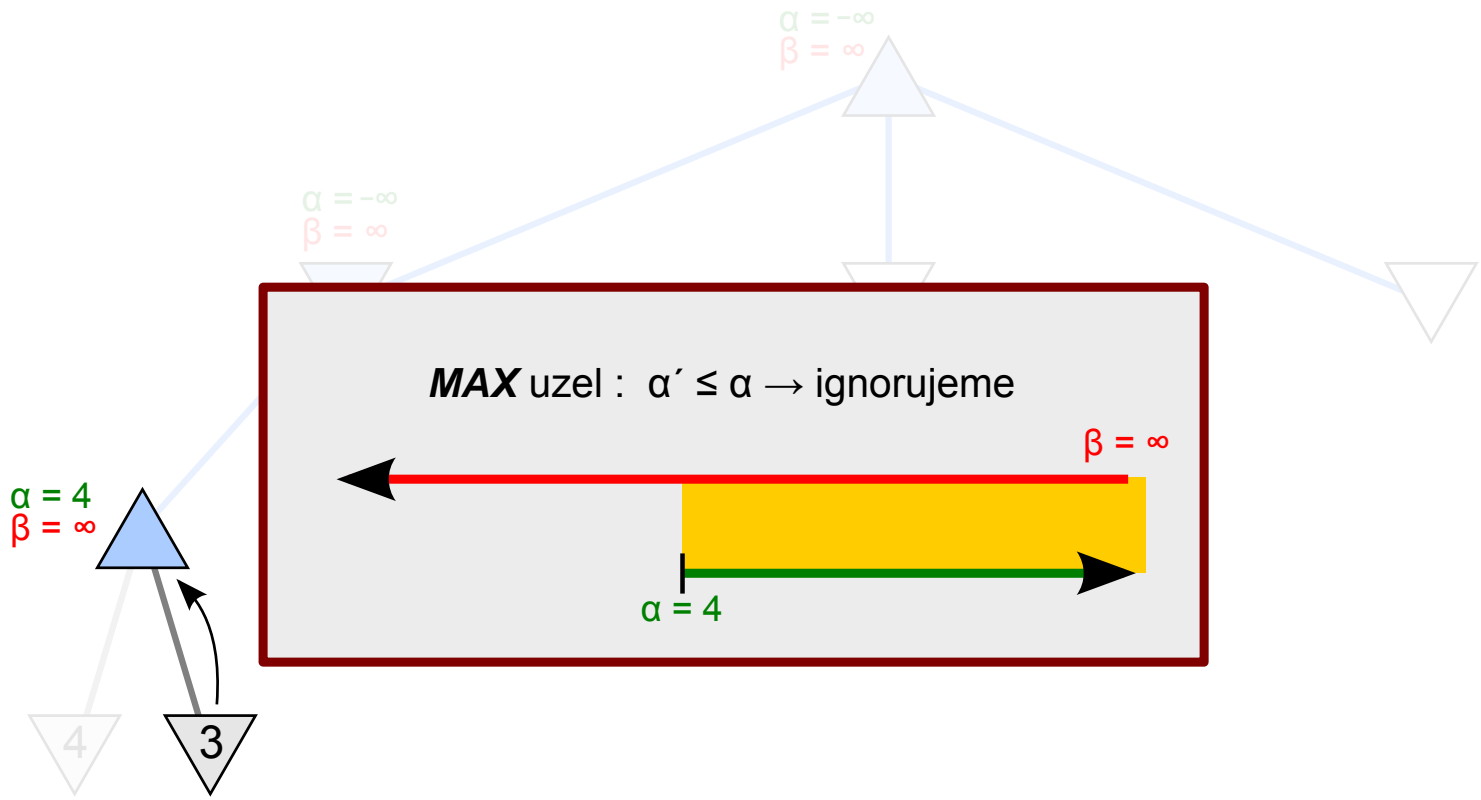
Minimax s alfa-beta prořezáváním: Příklad



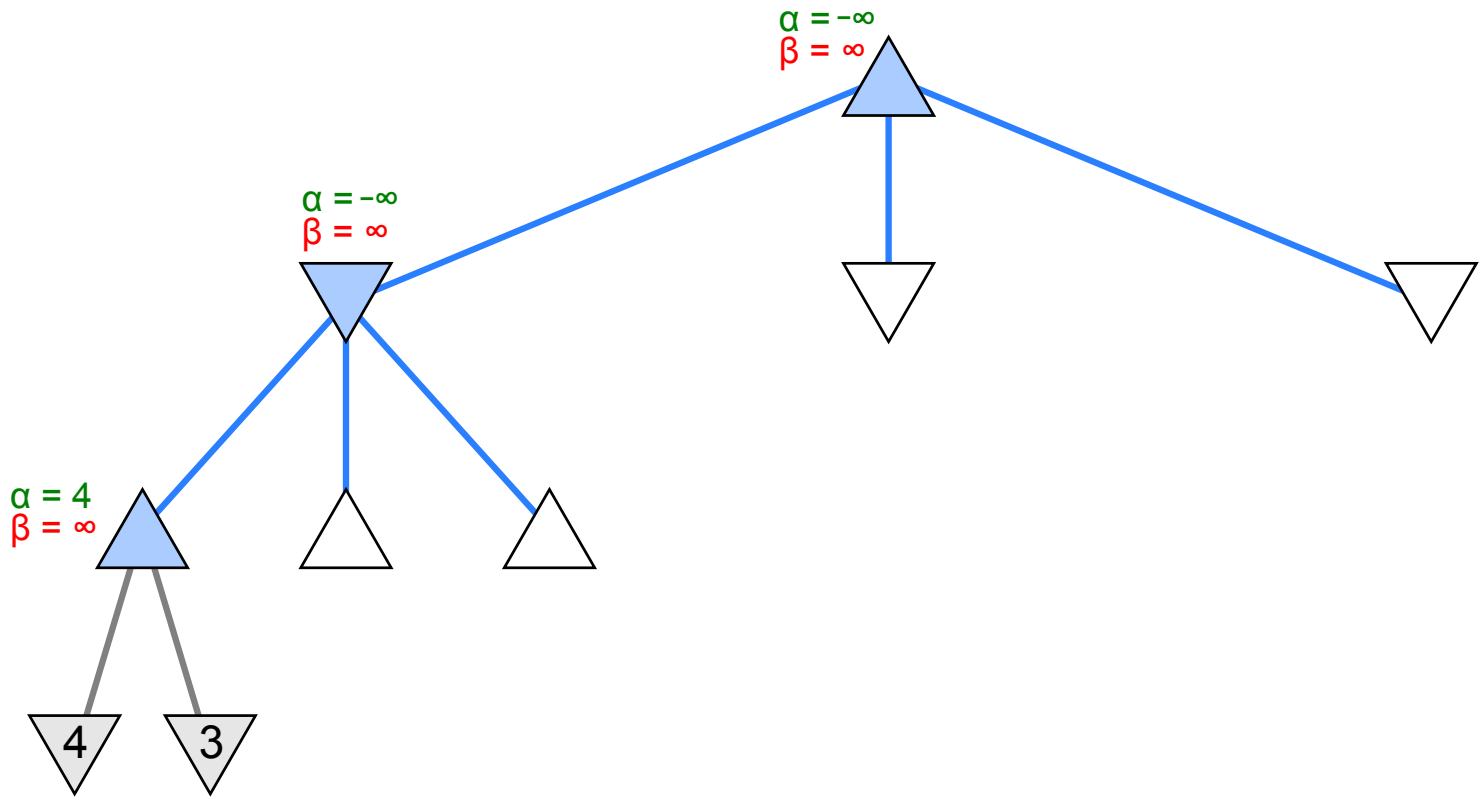
Minimax s alfa-beta prořezáváním: Příklad



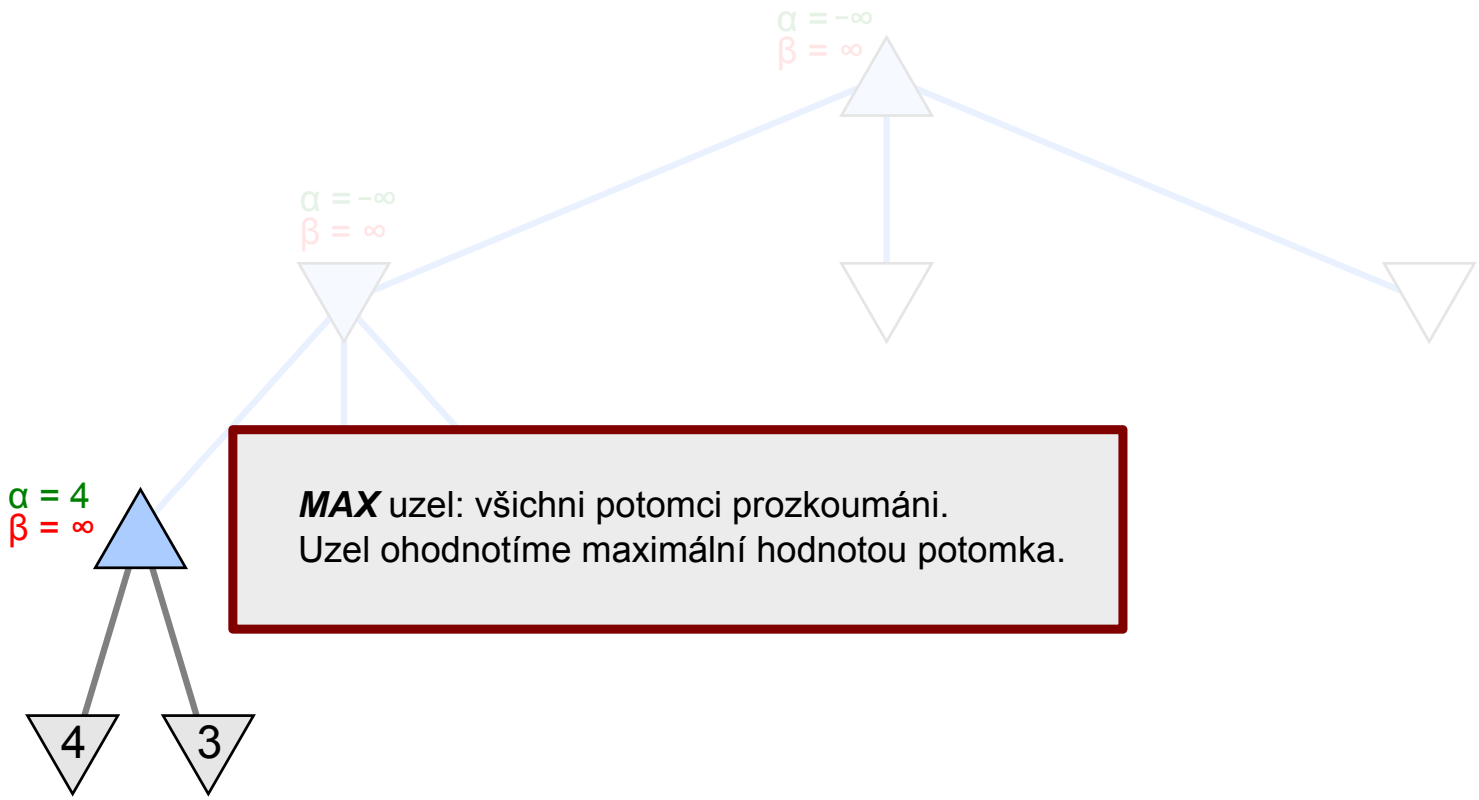
Minimax s alfa-beta prořezáváním: Příklad



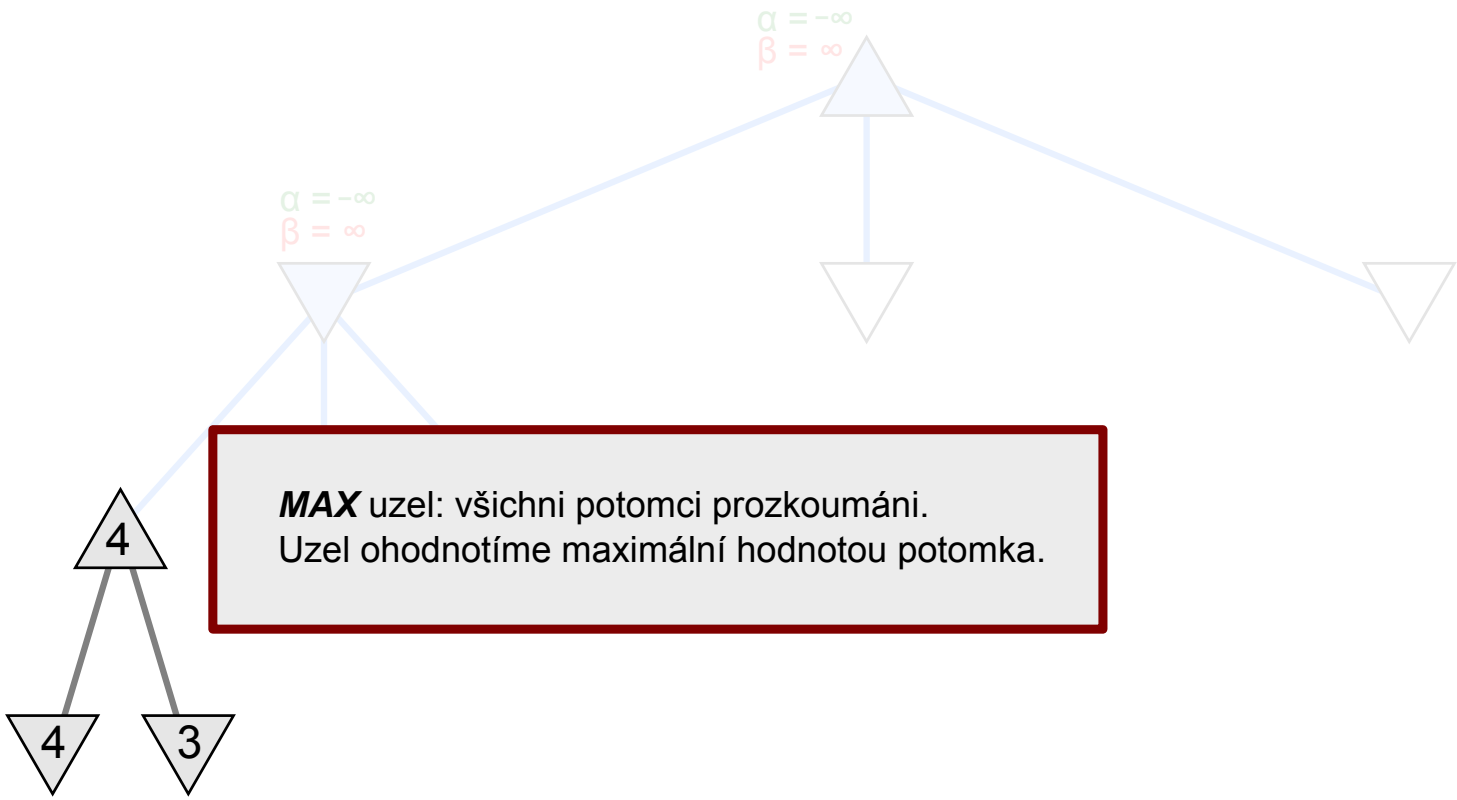
Minimax s alfa-beta prořezáváním: Příklad



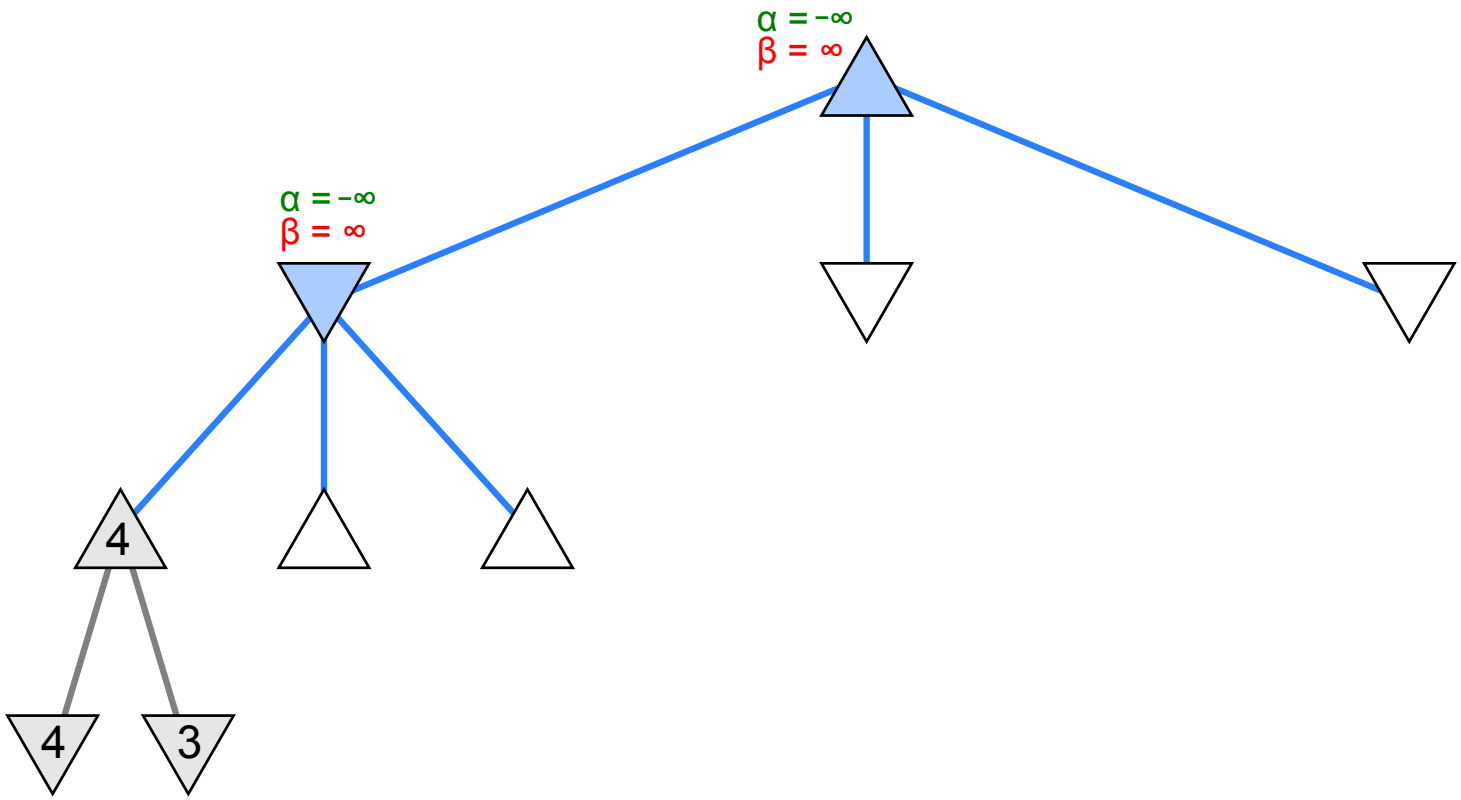
Minimax s alfa-beta prořezáváním: Příklad



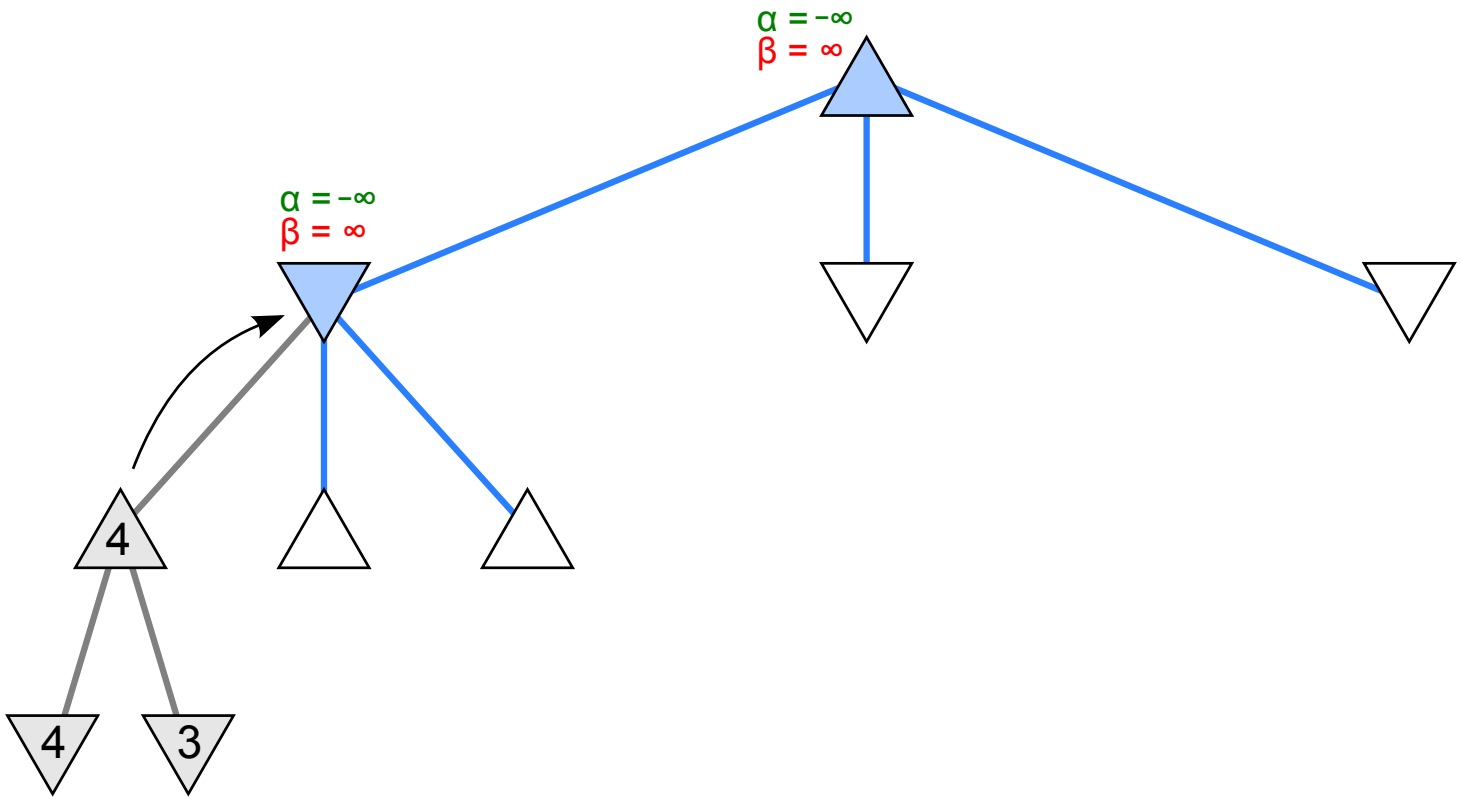
Minimax s alfa-beta prořezáváním: Příklad



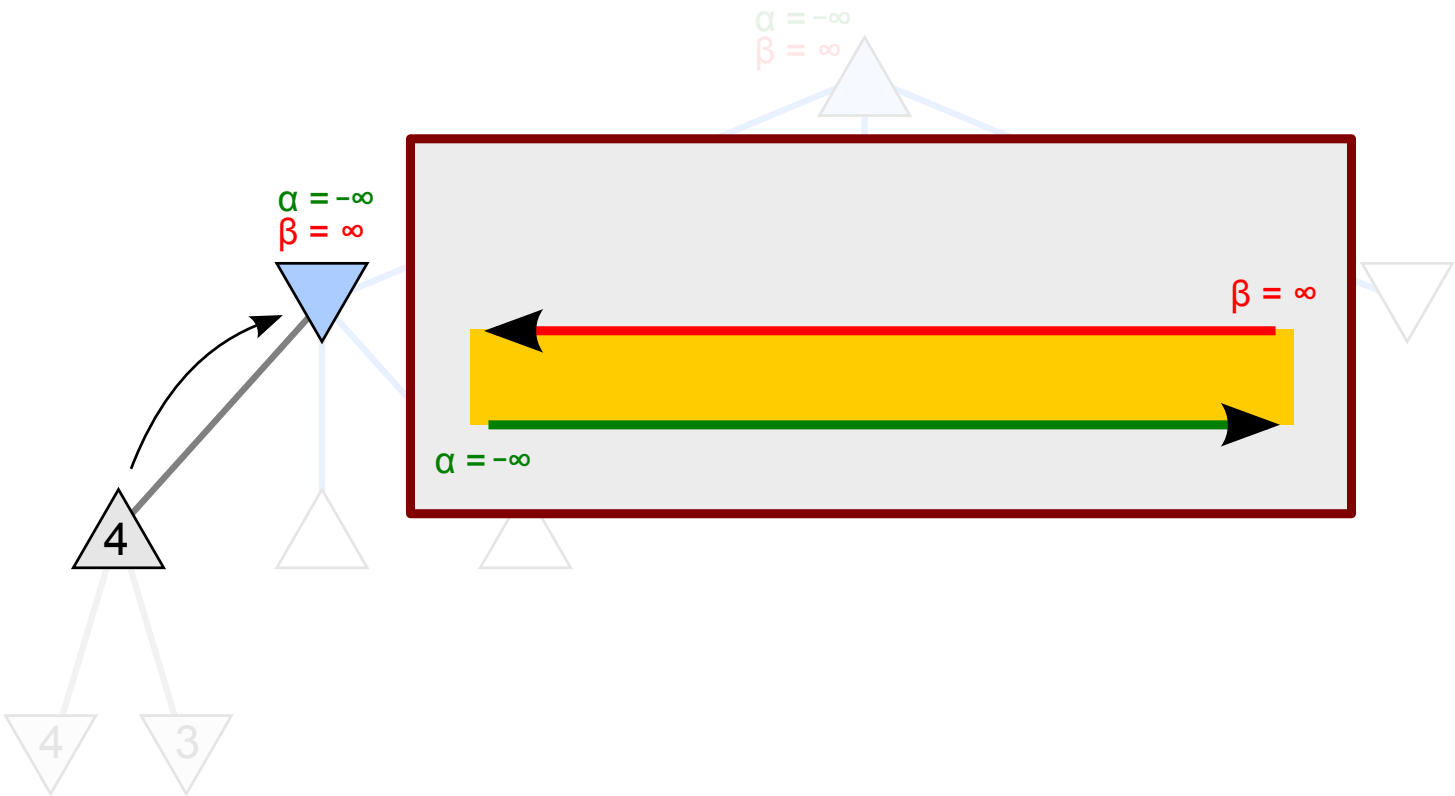
Minimax s alfa-beta prořezáváním: Příklad



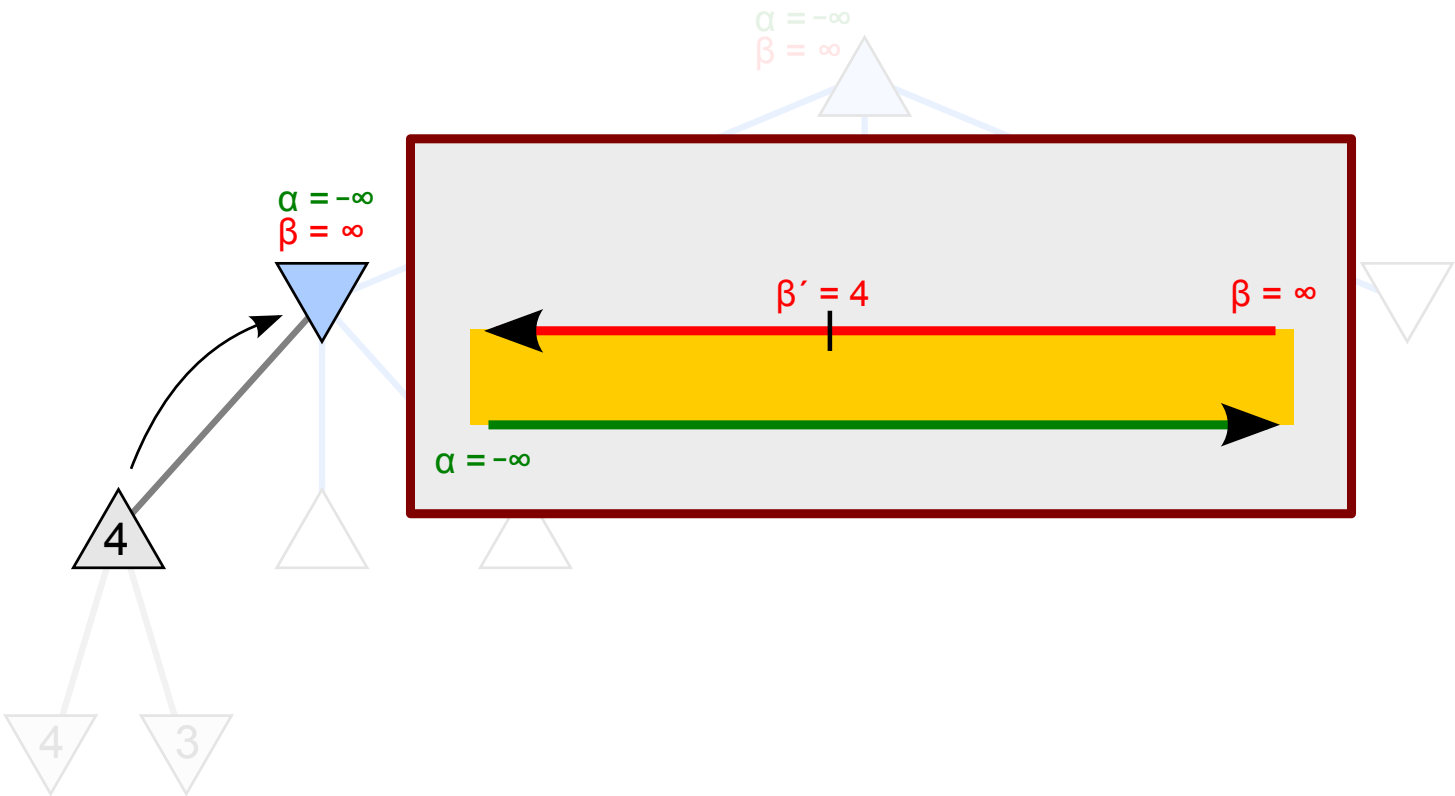
Minimax s alfa-beta prořezáváním: Příklad



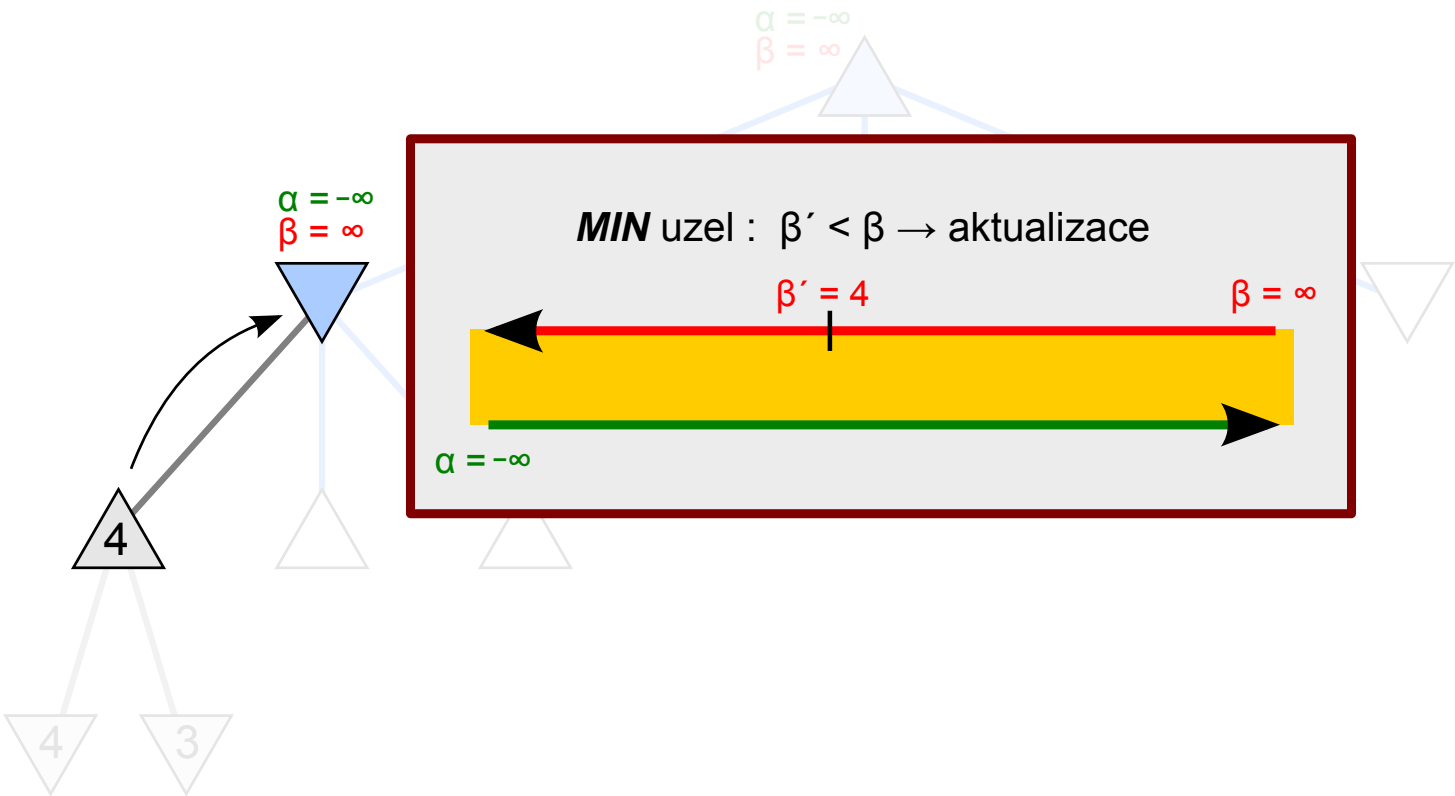
Minimax s alfa-beta prořezáváním: Příklad



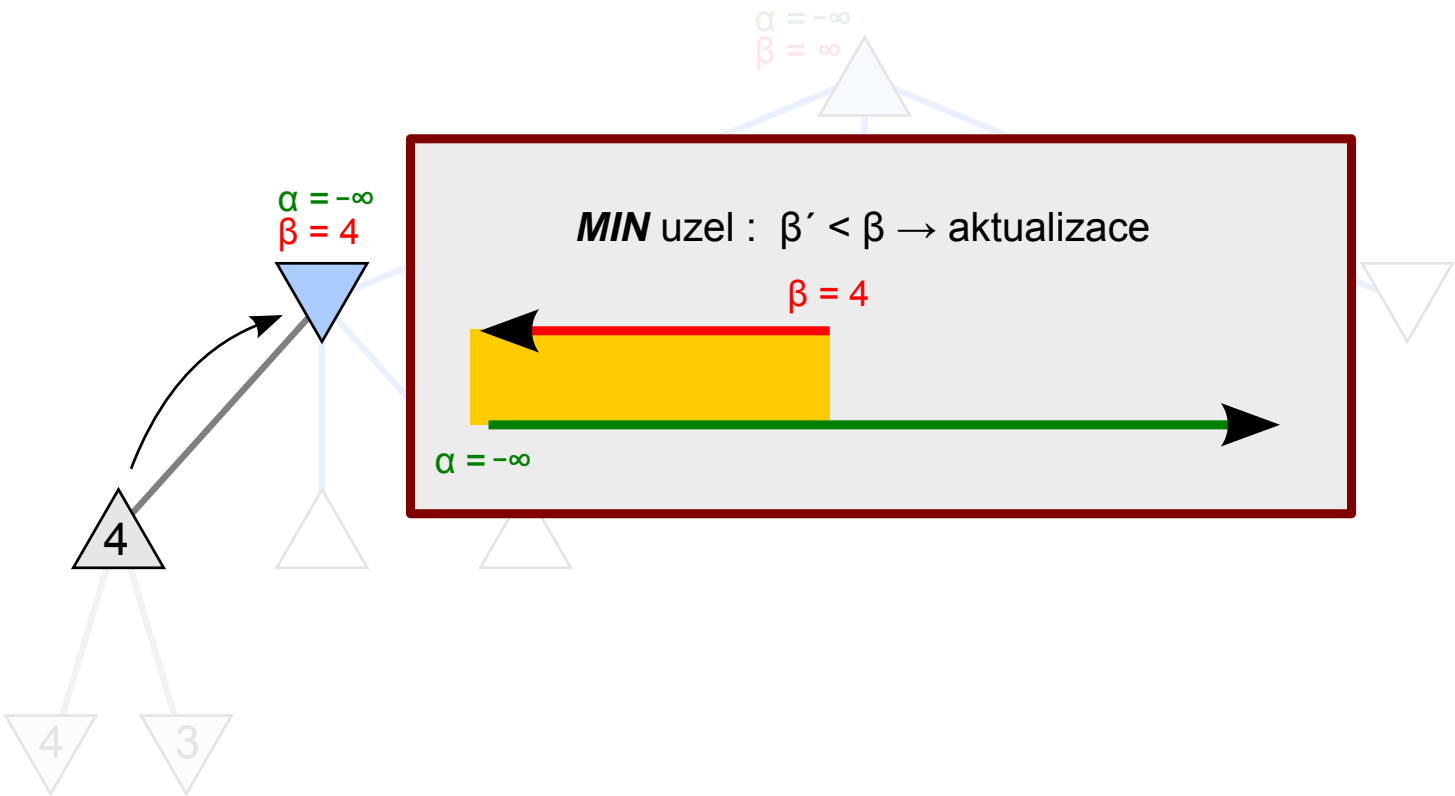
Minimax s alfa-beta prořezáváním: Příklad



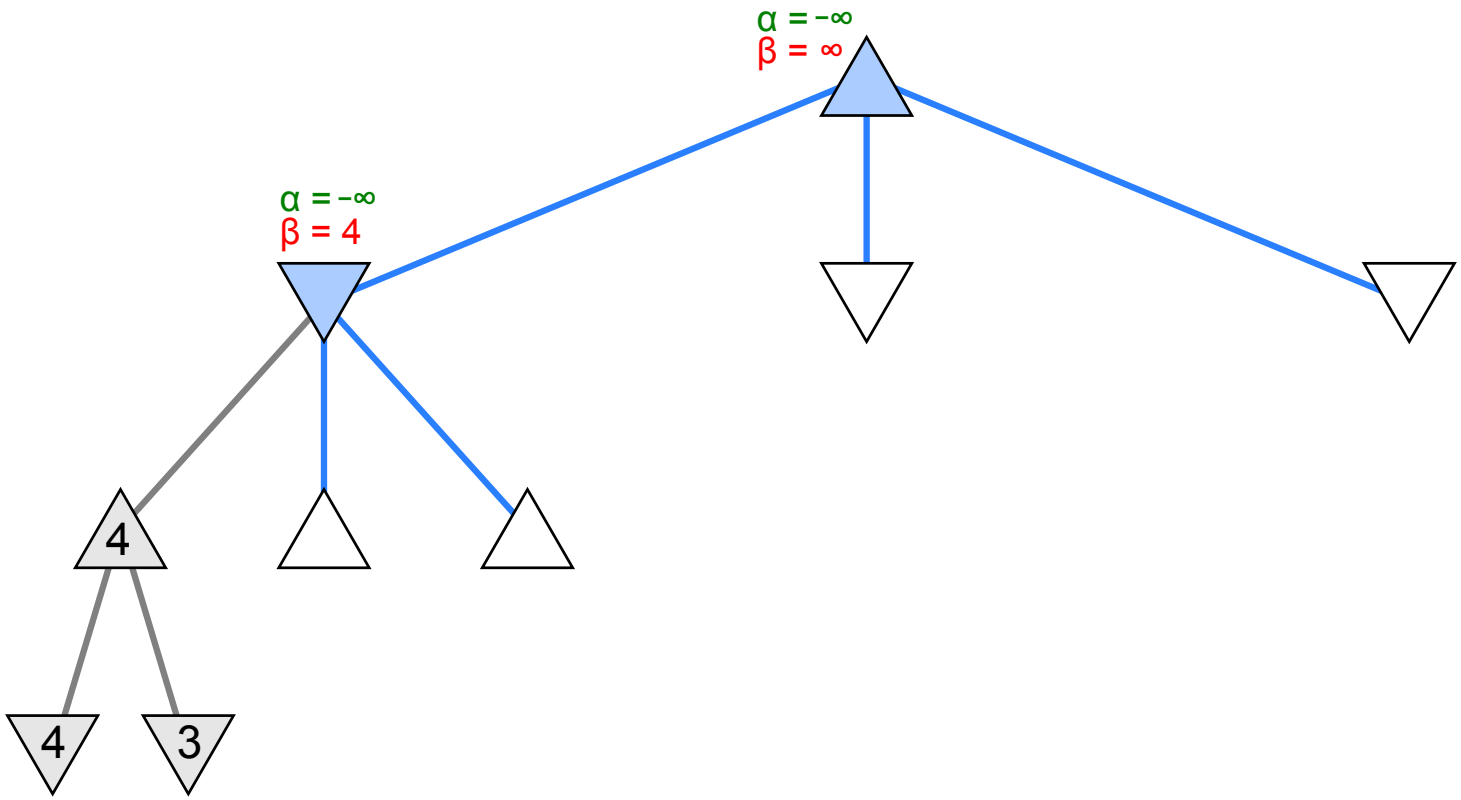
Minimax s alfa-beta prořezáváním: Příklad



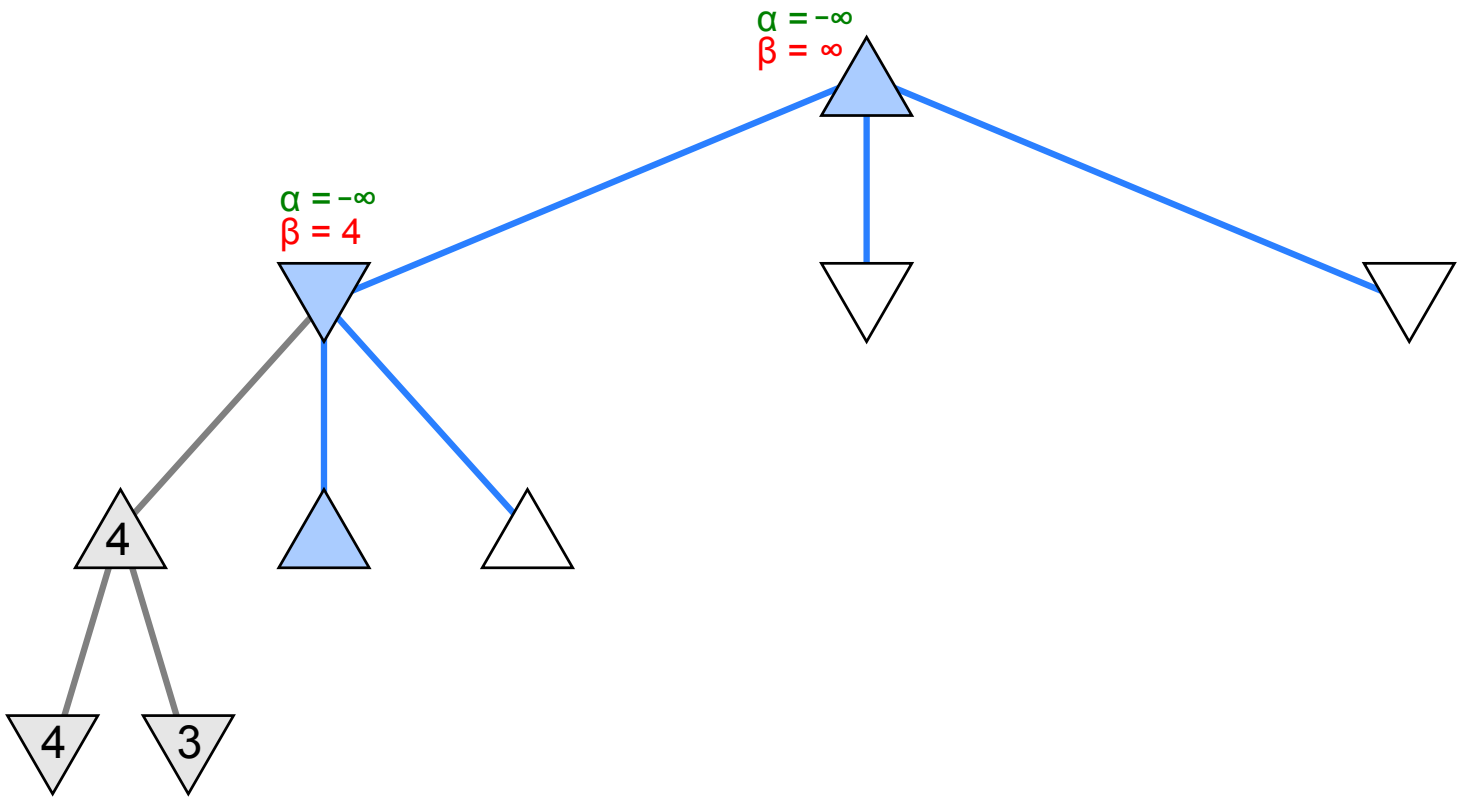
Minimax s alfa-beta prořezáváním: Příklad



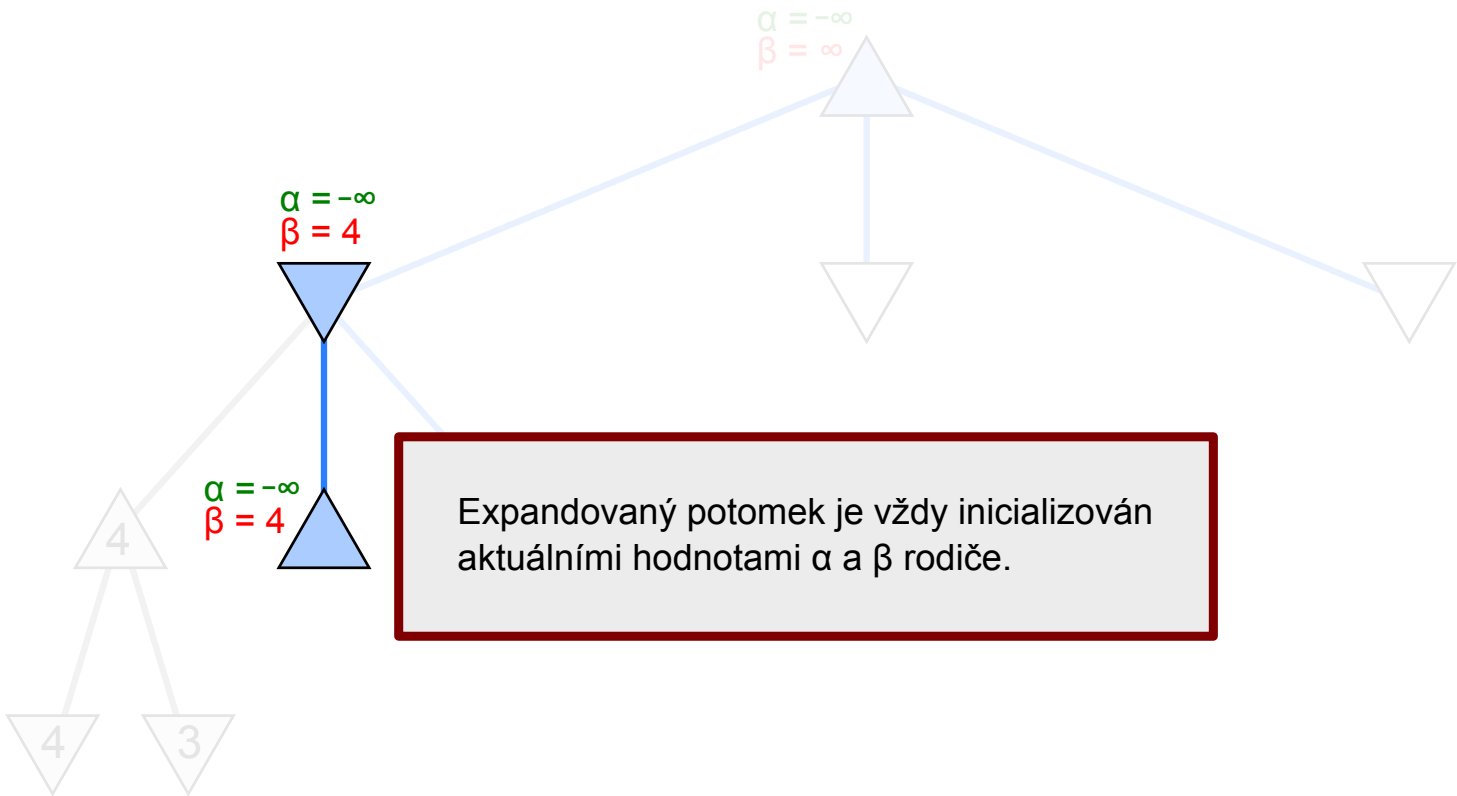
Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad

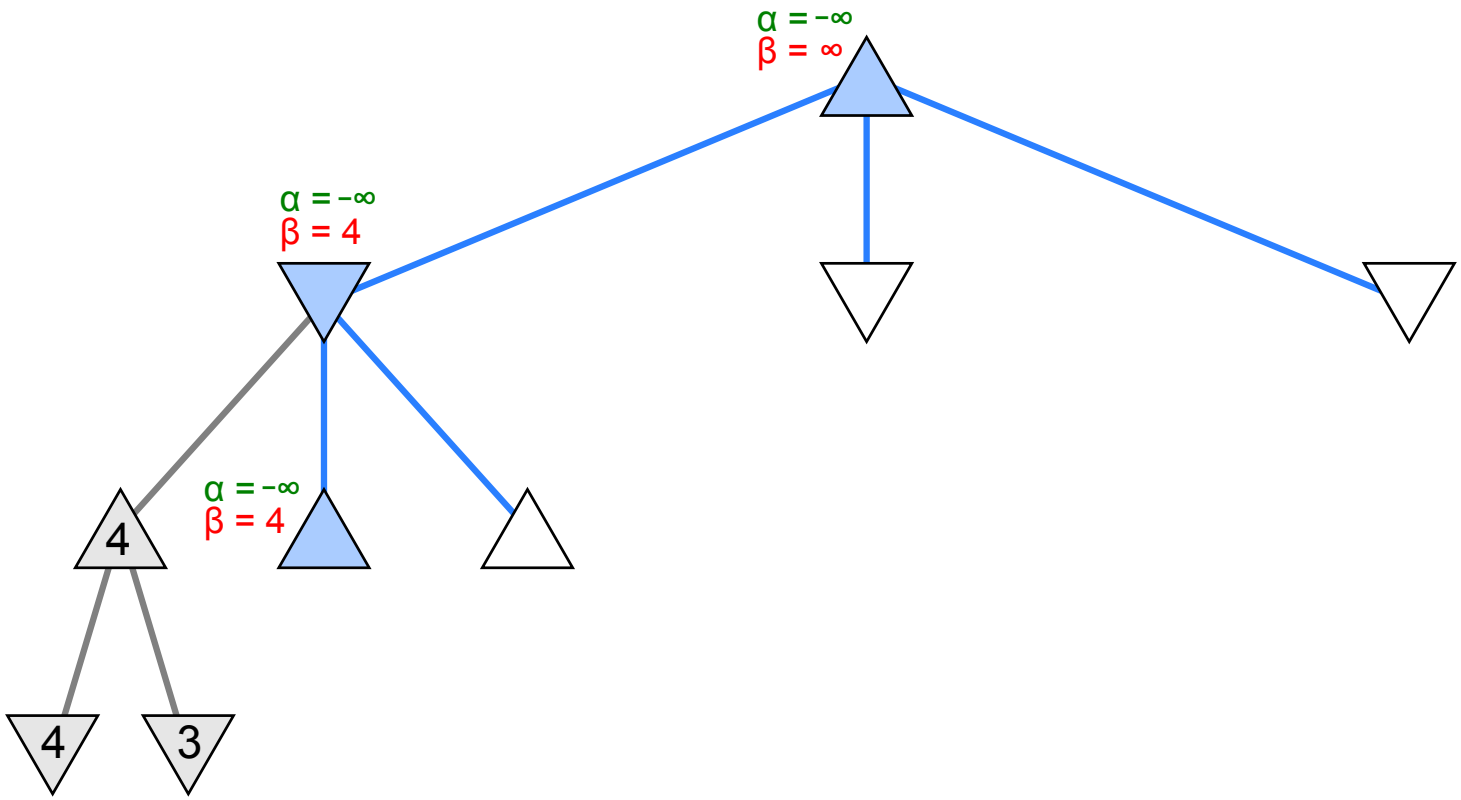


Minimax s alfa-beta prořezáváním: Příklad

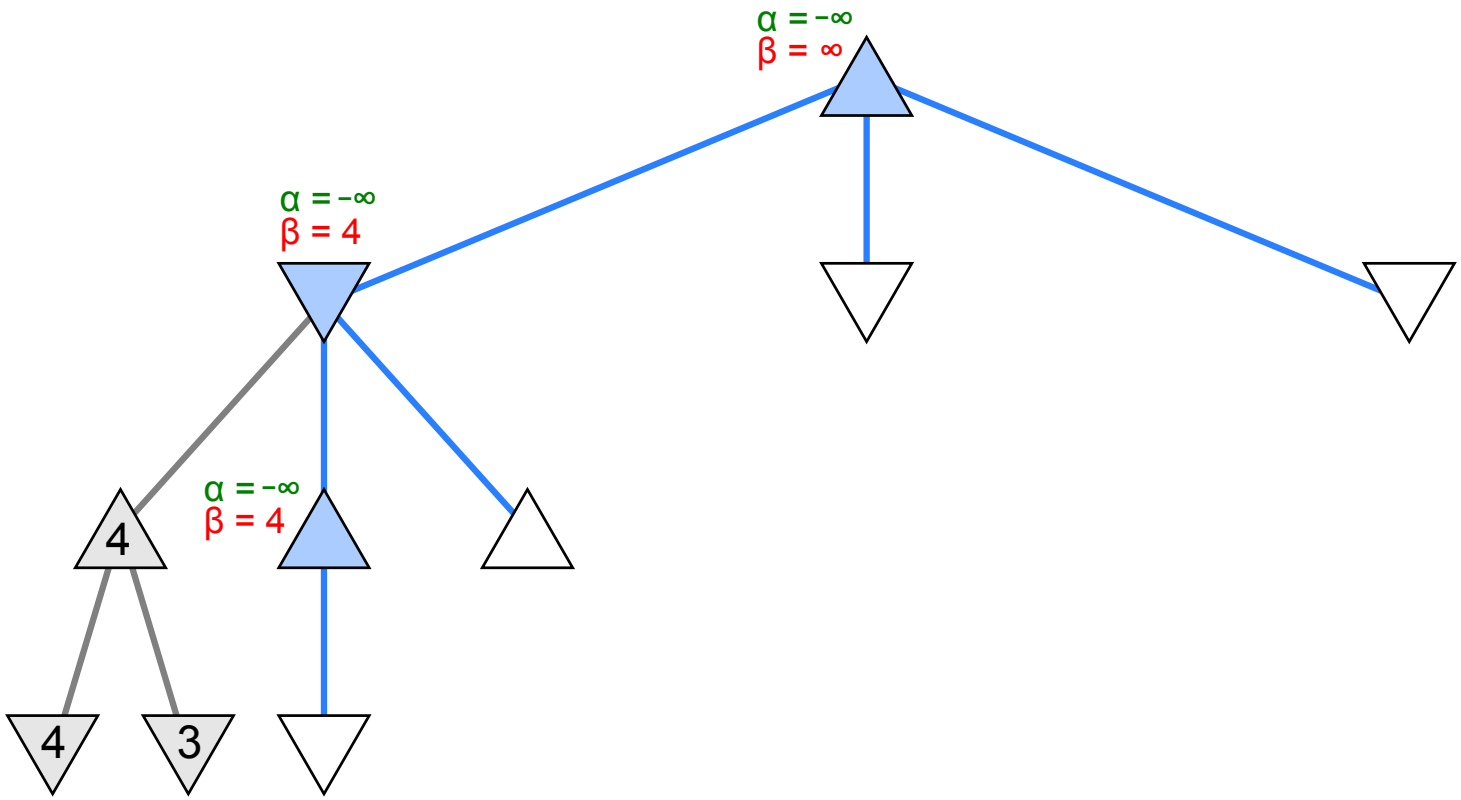


Expandovaný potomek je vždy inicializován aktuálními hodnotami α a β rodiče.

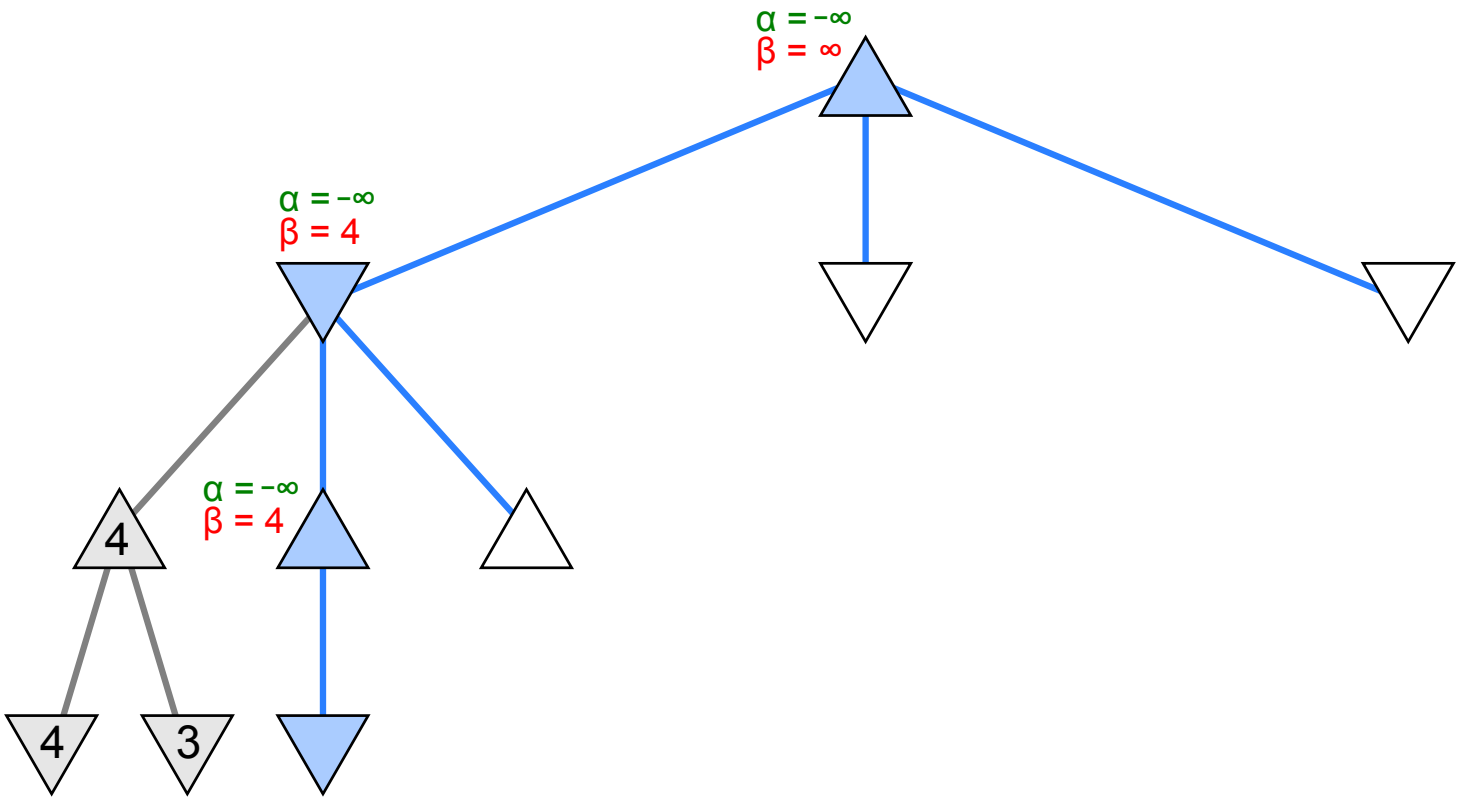
Minimax s alfa-beta prořezáváním: Příklad



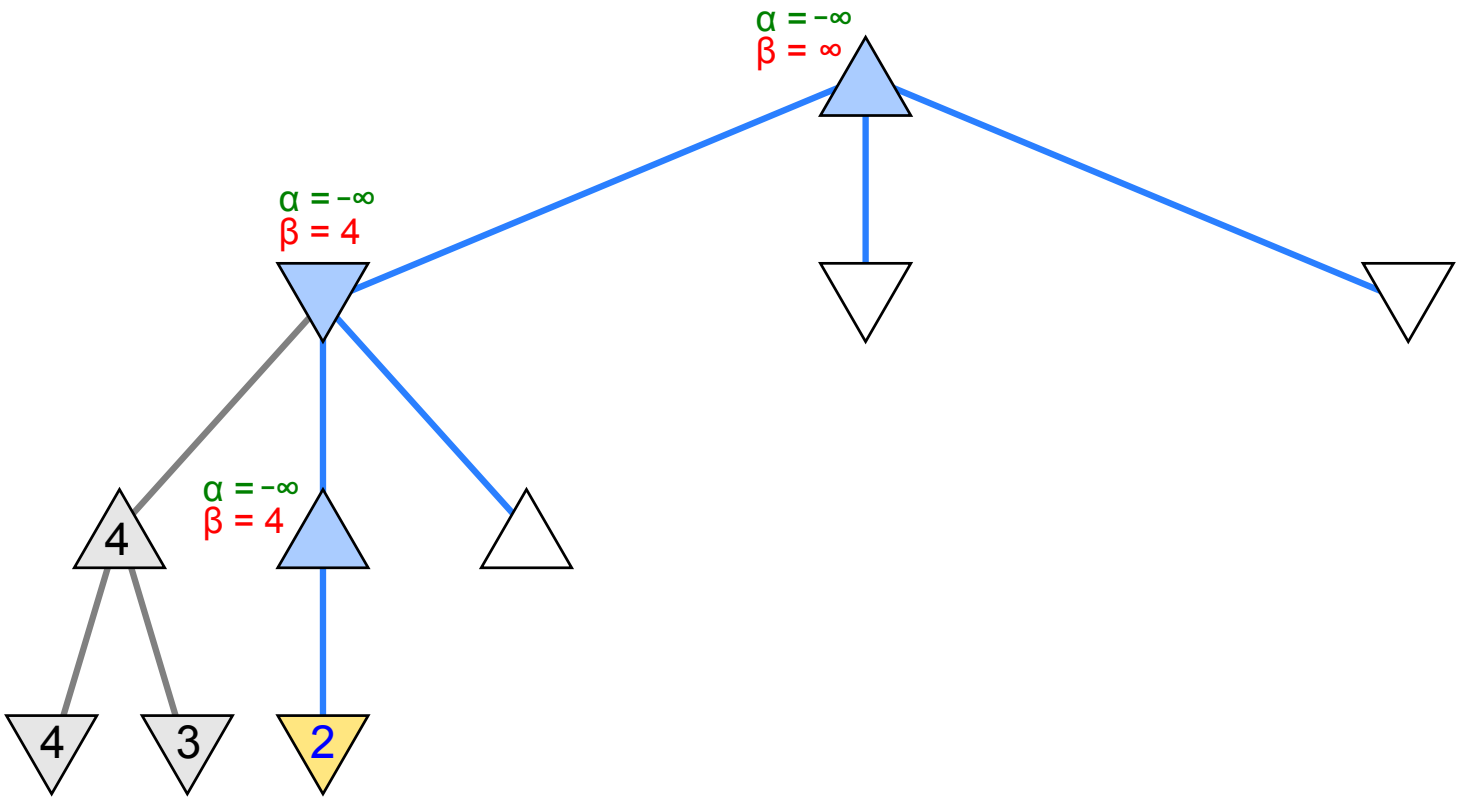
Minimax s alfa-beta prořezáváním: Příklad



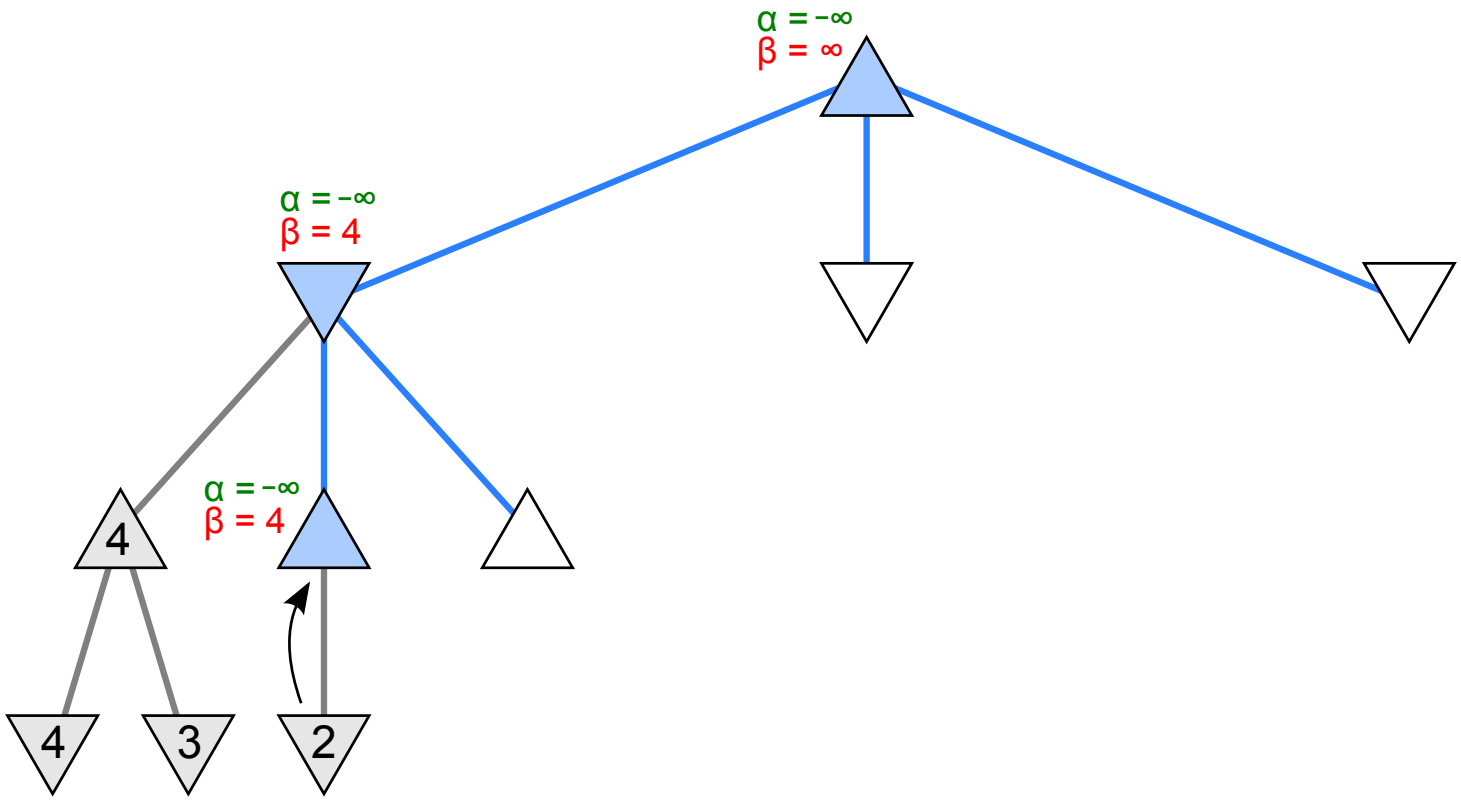
Minimax s alfa-beta prořezáváním: Příklad



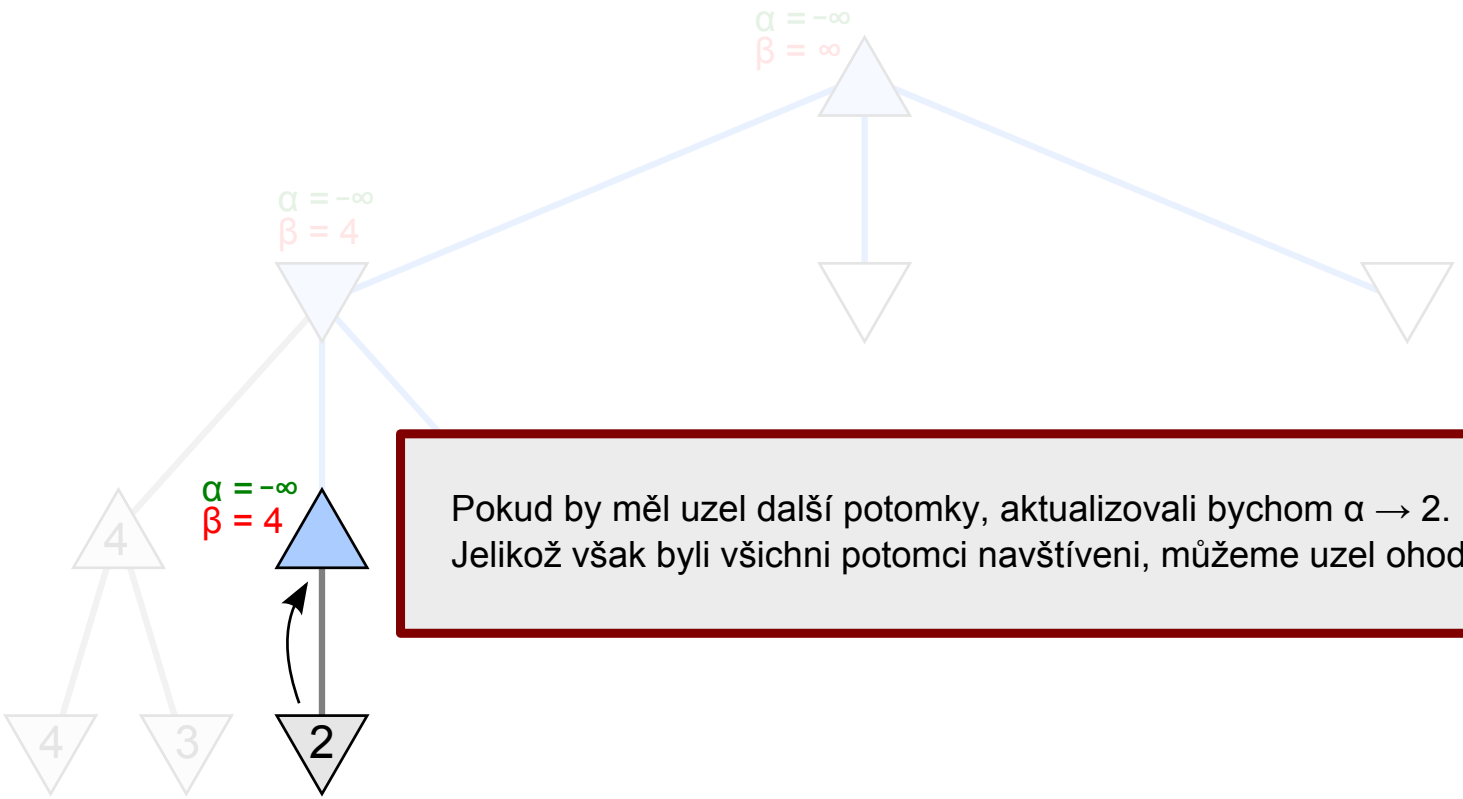
Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad

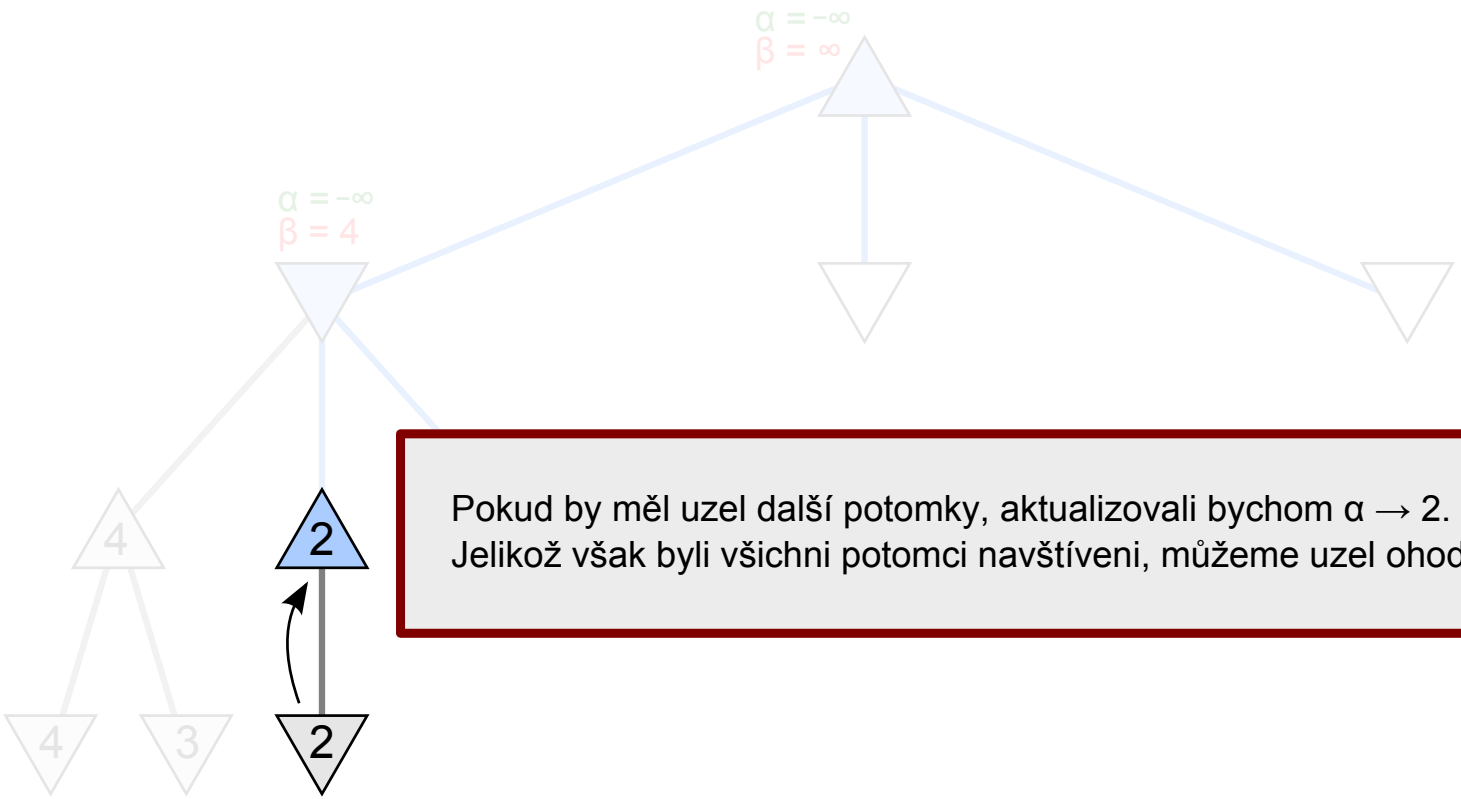


Minimax s alfa-beta prořezáváním: Příklad



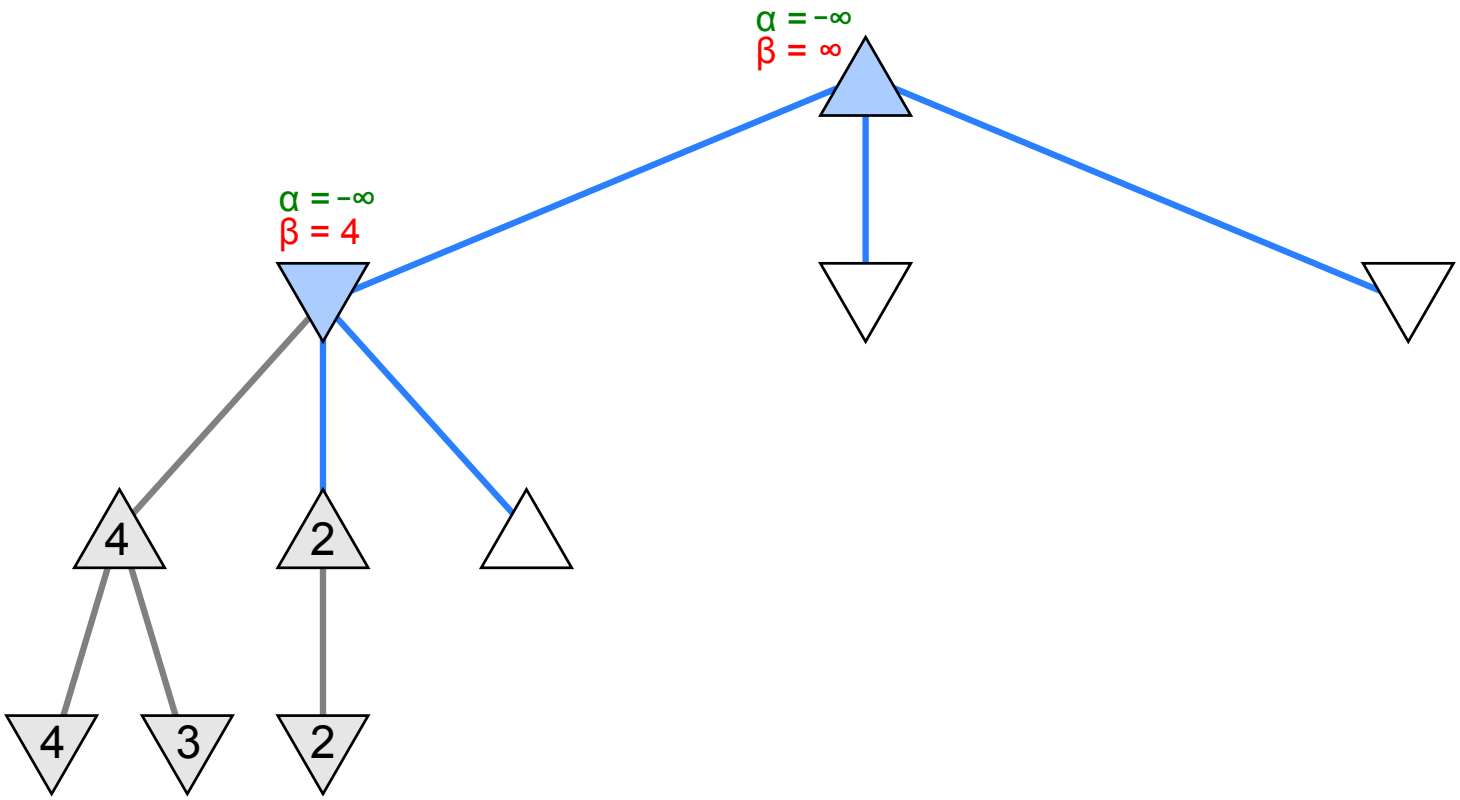
Pokud by měl uzel další potomky, aktualizovali bychom $\alpha \rightarrow 2$.
Jelikož však byli všichni potomci navštíveni, můžeme uzel ohodnotit.

Minimax s alfa-beta prořezáváním: Příklad

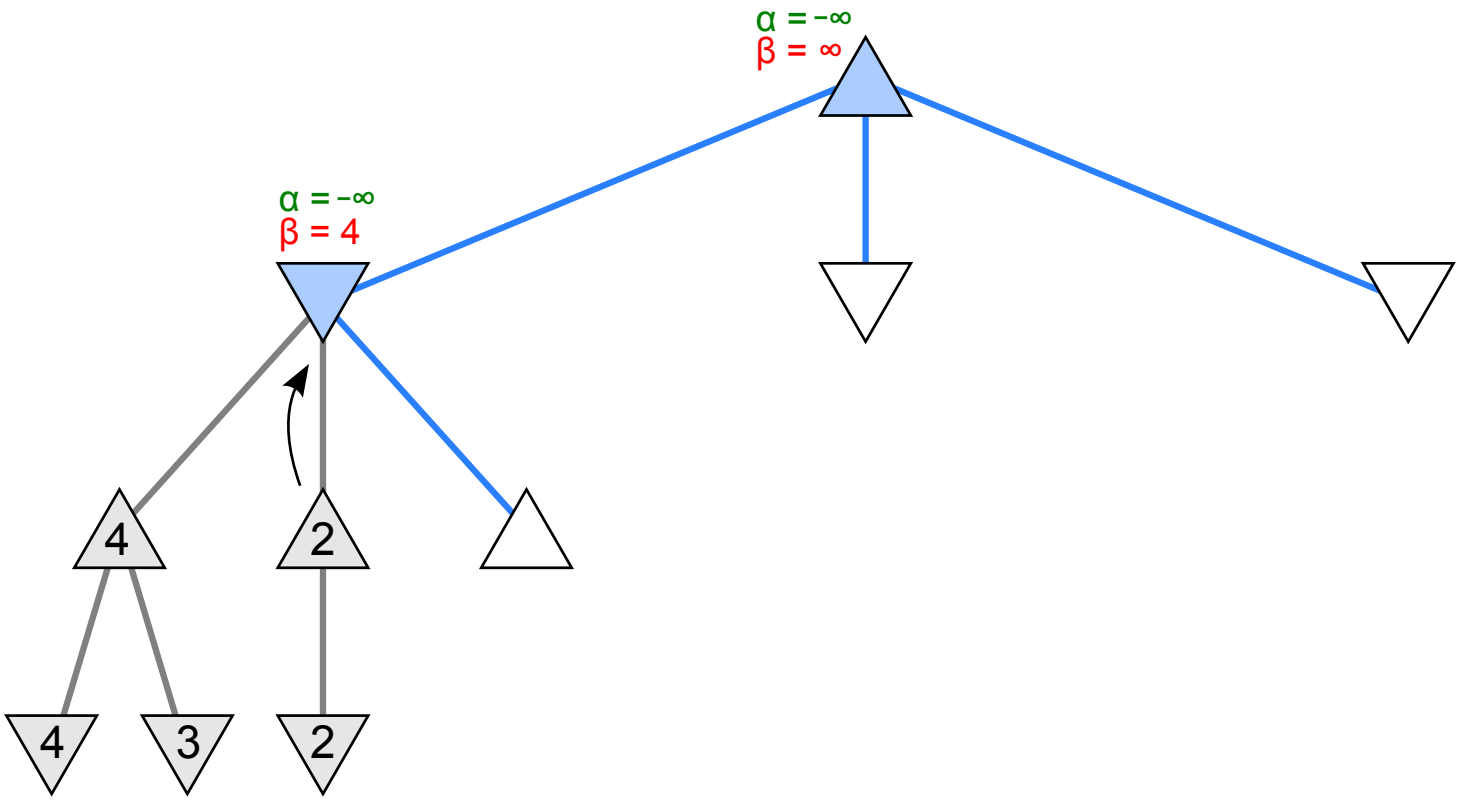


Pokud by měl uzel další potomky, aktualizovali bychom $\alpha \rightarrow 2$.
Jelikož však byli všichni potomci navštíveni, můžeme uzel ohodnotit.

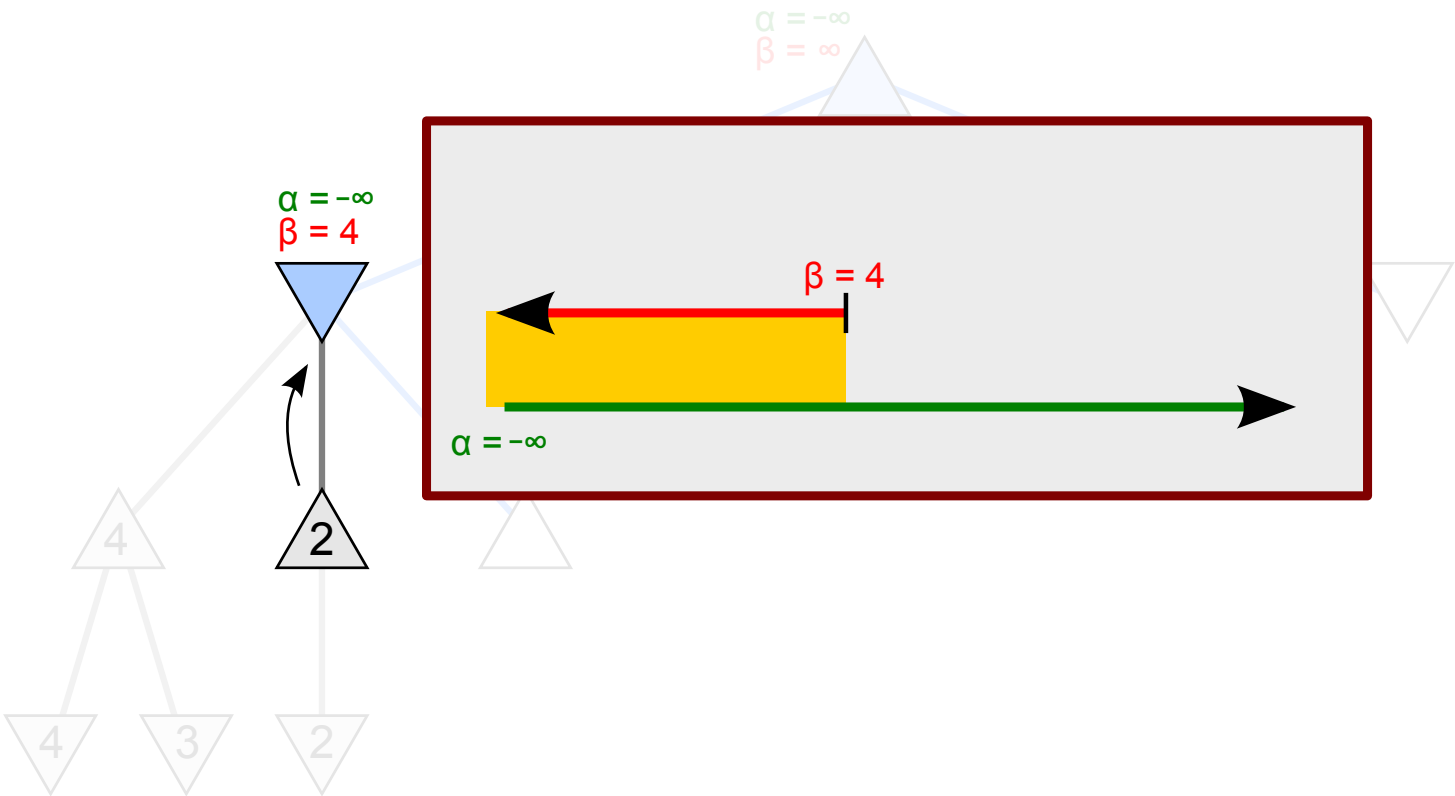
Minimax s alfa-beta prořezáváním: Příklad



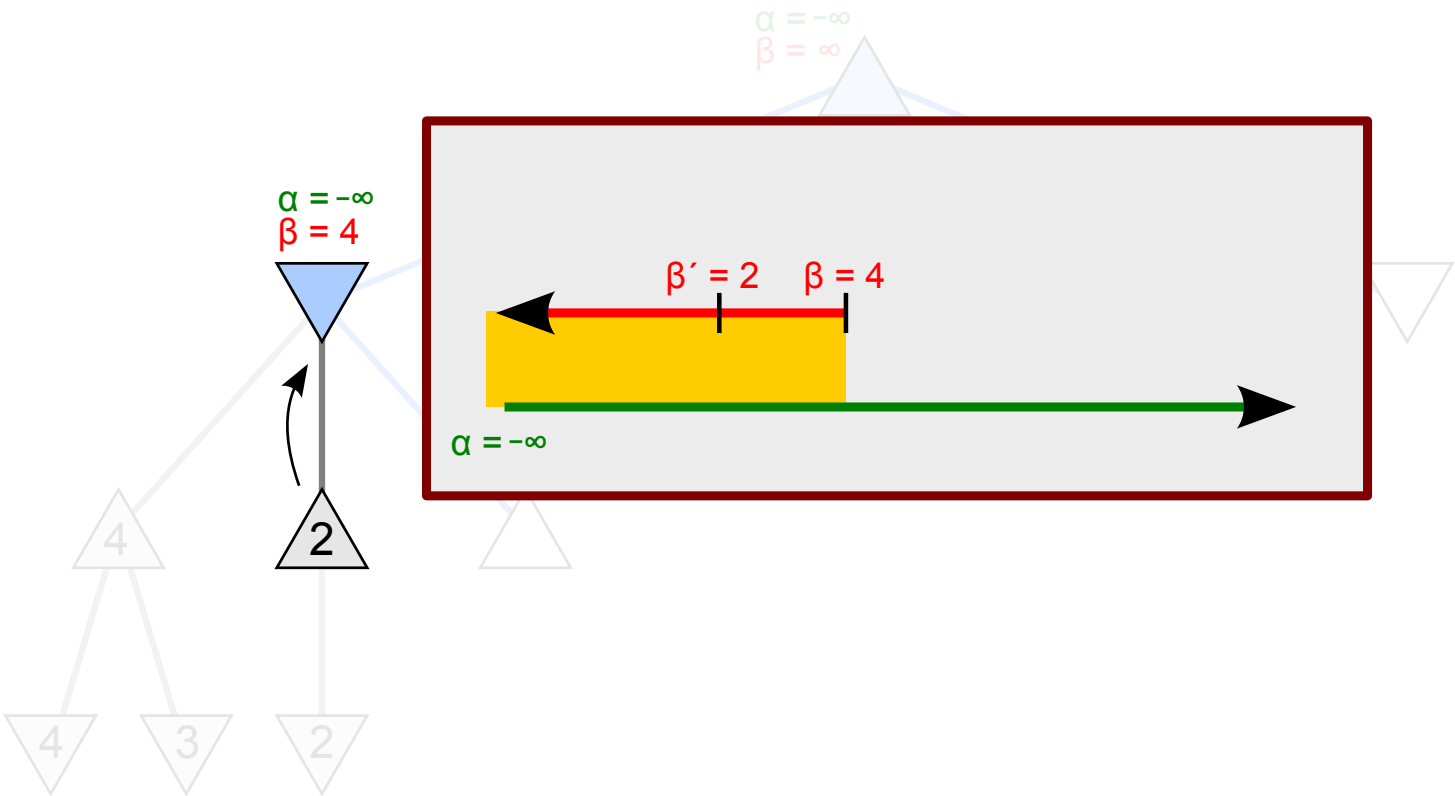
Minimax s alfa-beta prořezáváním: Příklad



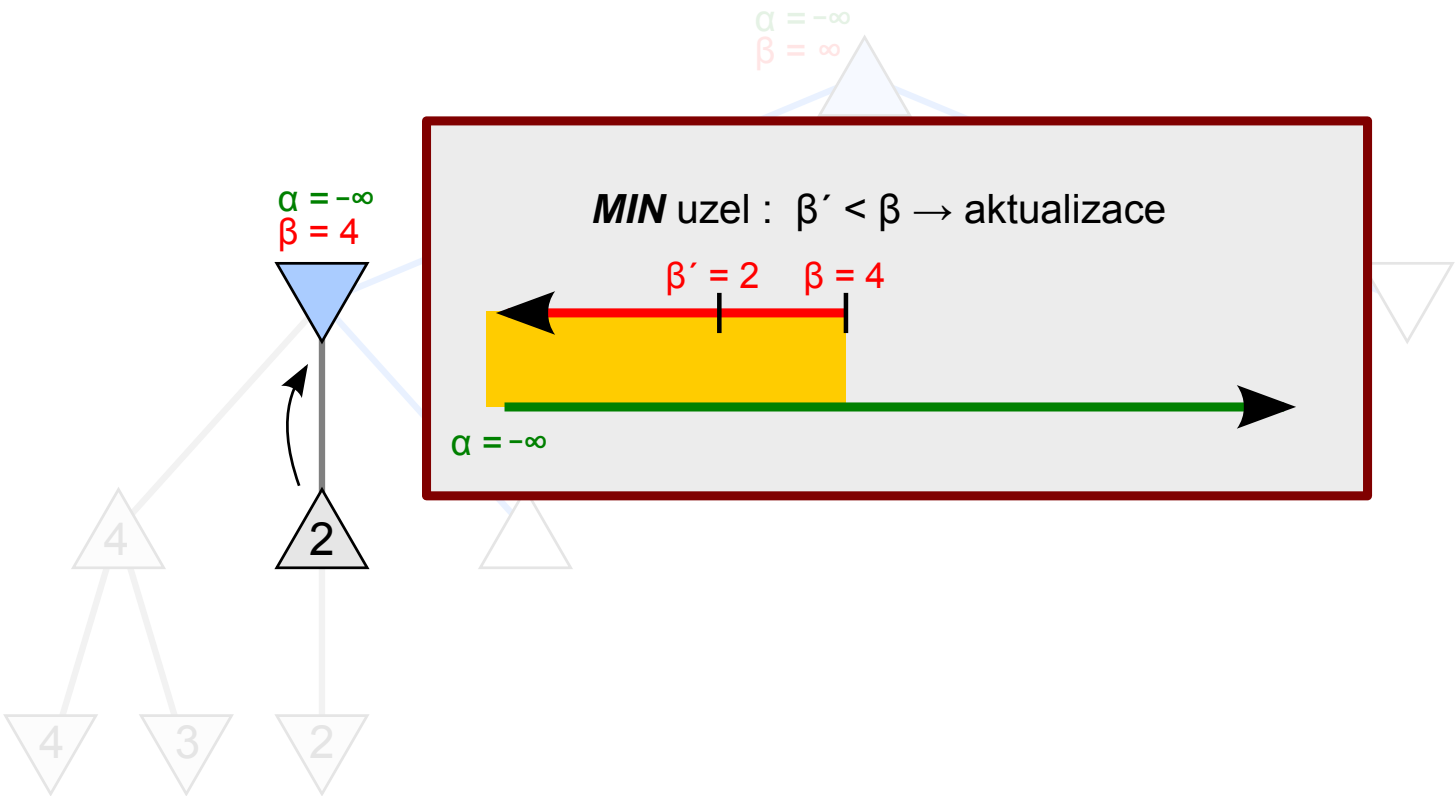
Minimax s alfa-beta prořezáváním: Příklad



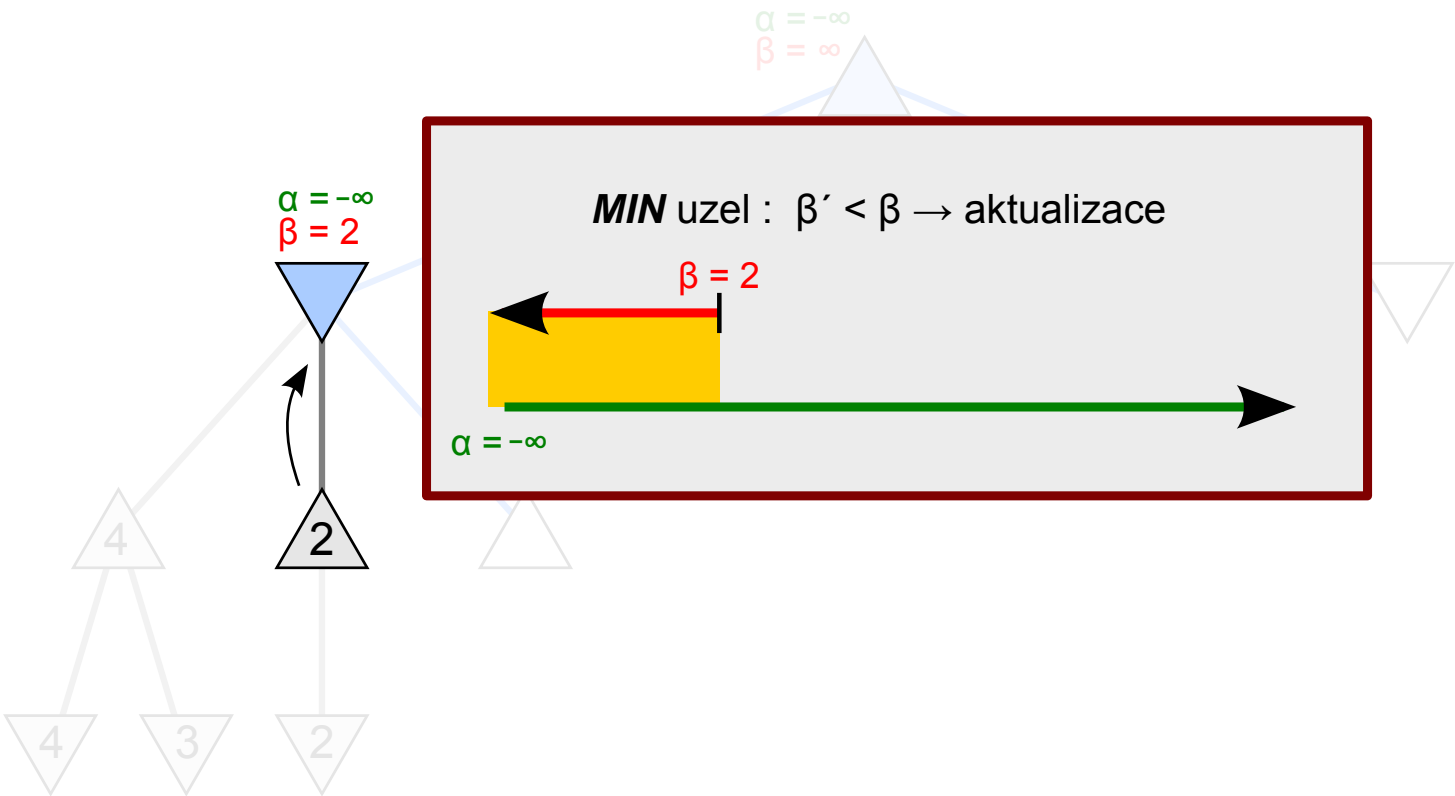
Minimax s alfa-beta prořezáváním: Příklad



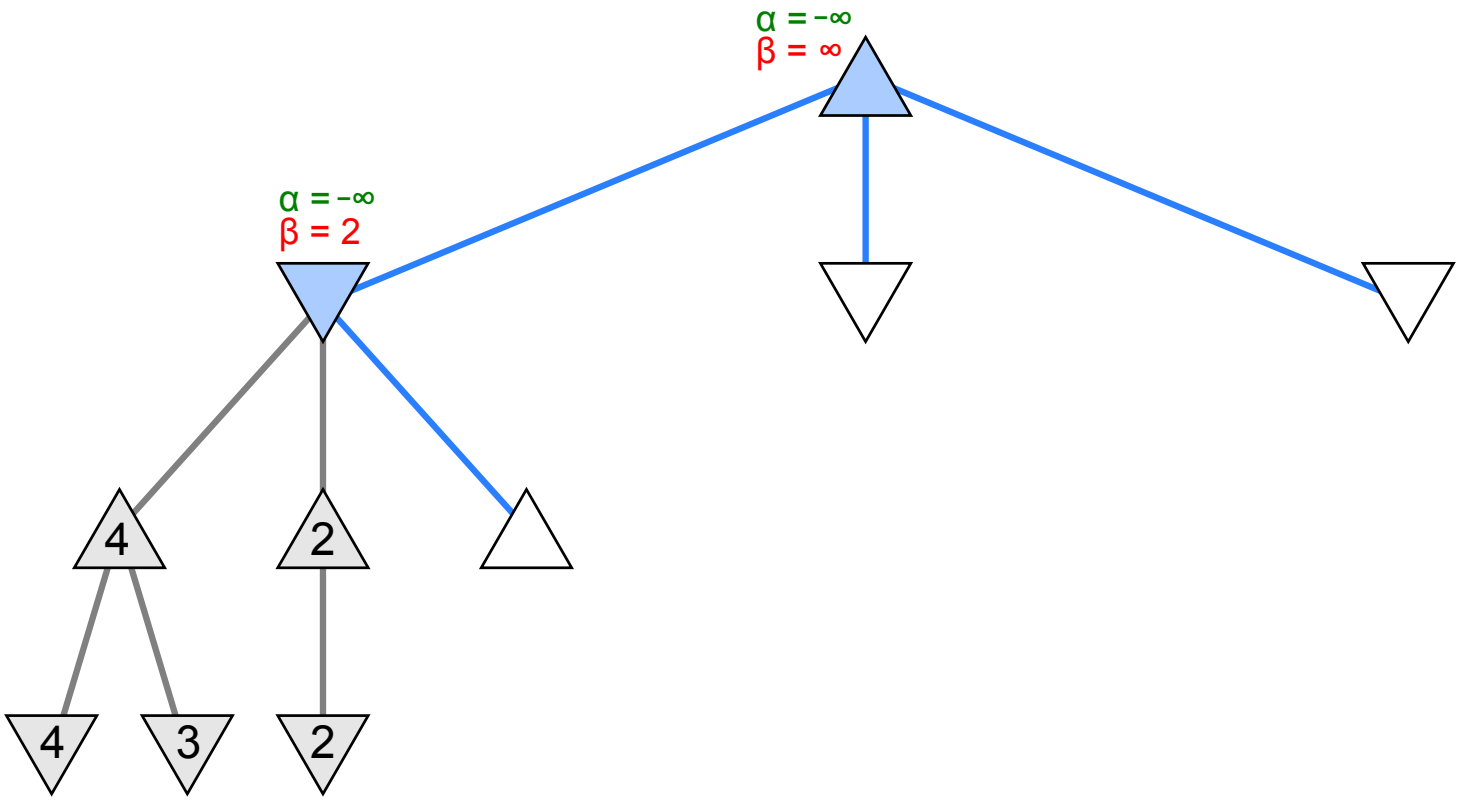
Minimax s alfa-beta prořezáváním: Příklad



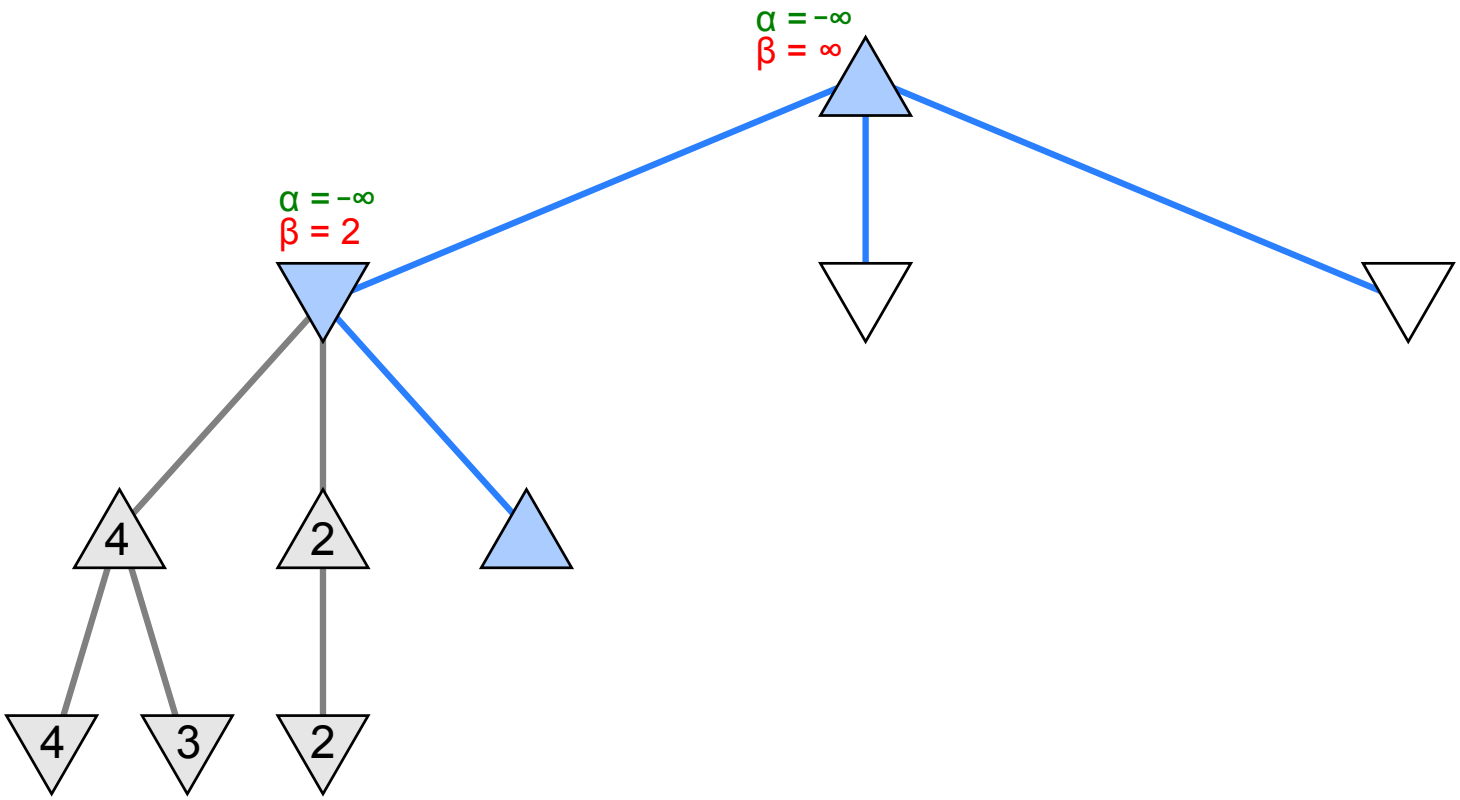
Minimax s alfa-beta prořezáváním: Příklad



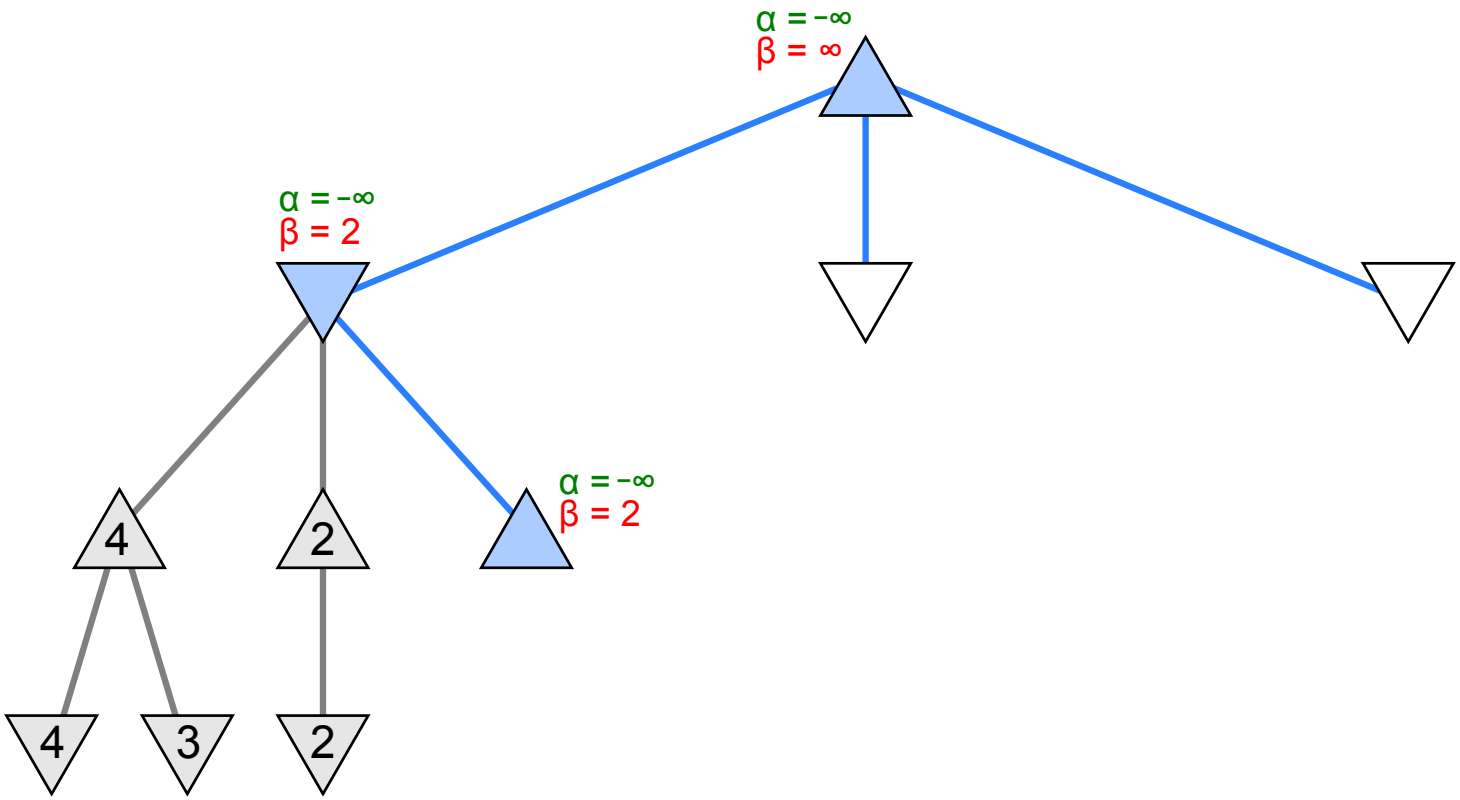
Minimax s alfa-beta prořezáváním: Příklad



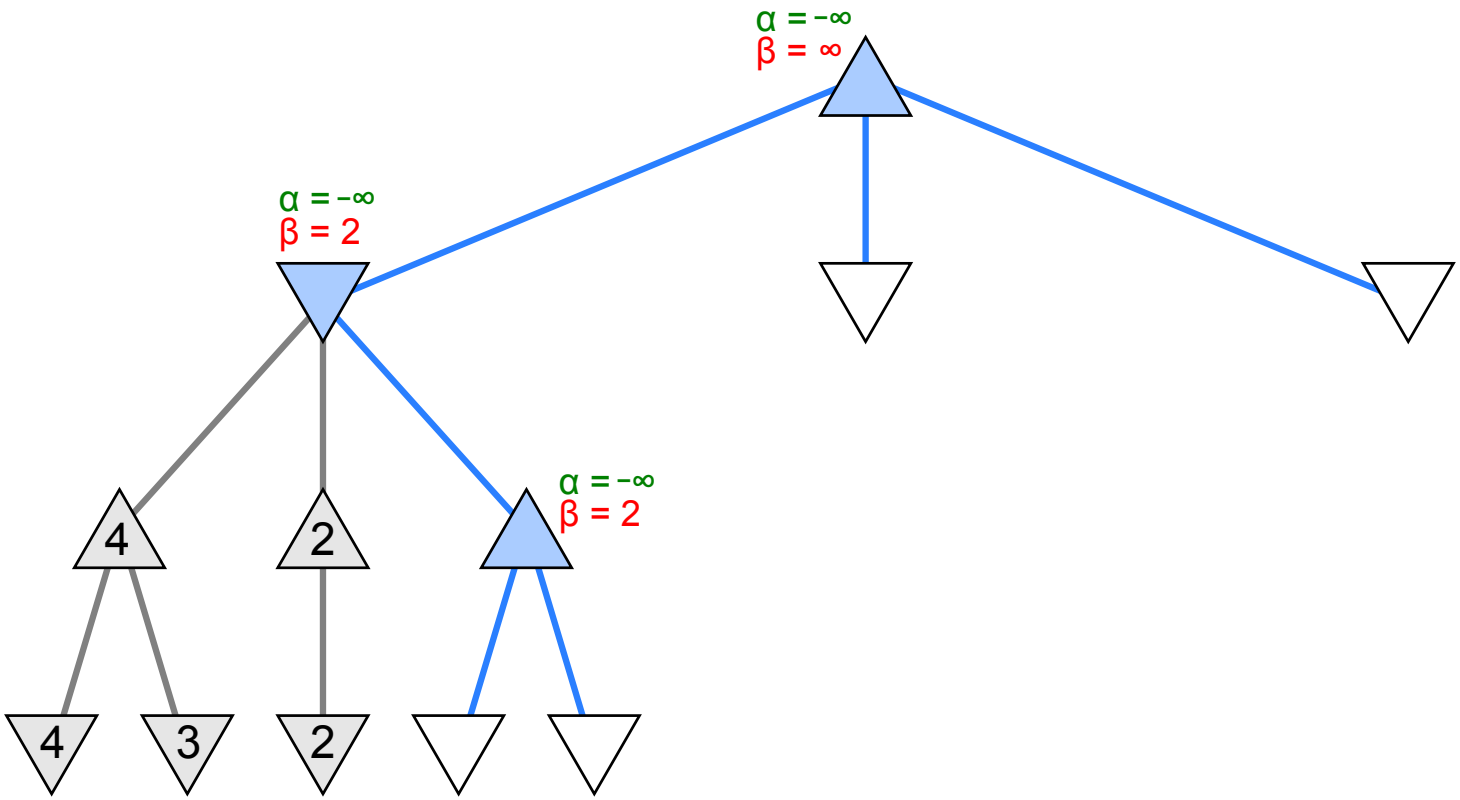
Minimax s alfa-beta prořezáváním: Příklad



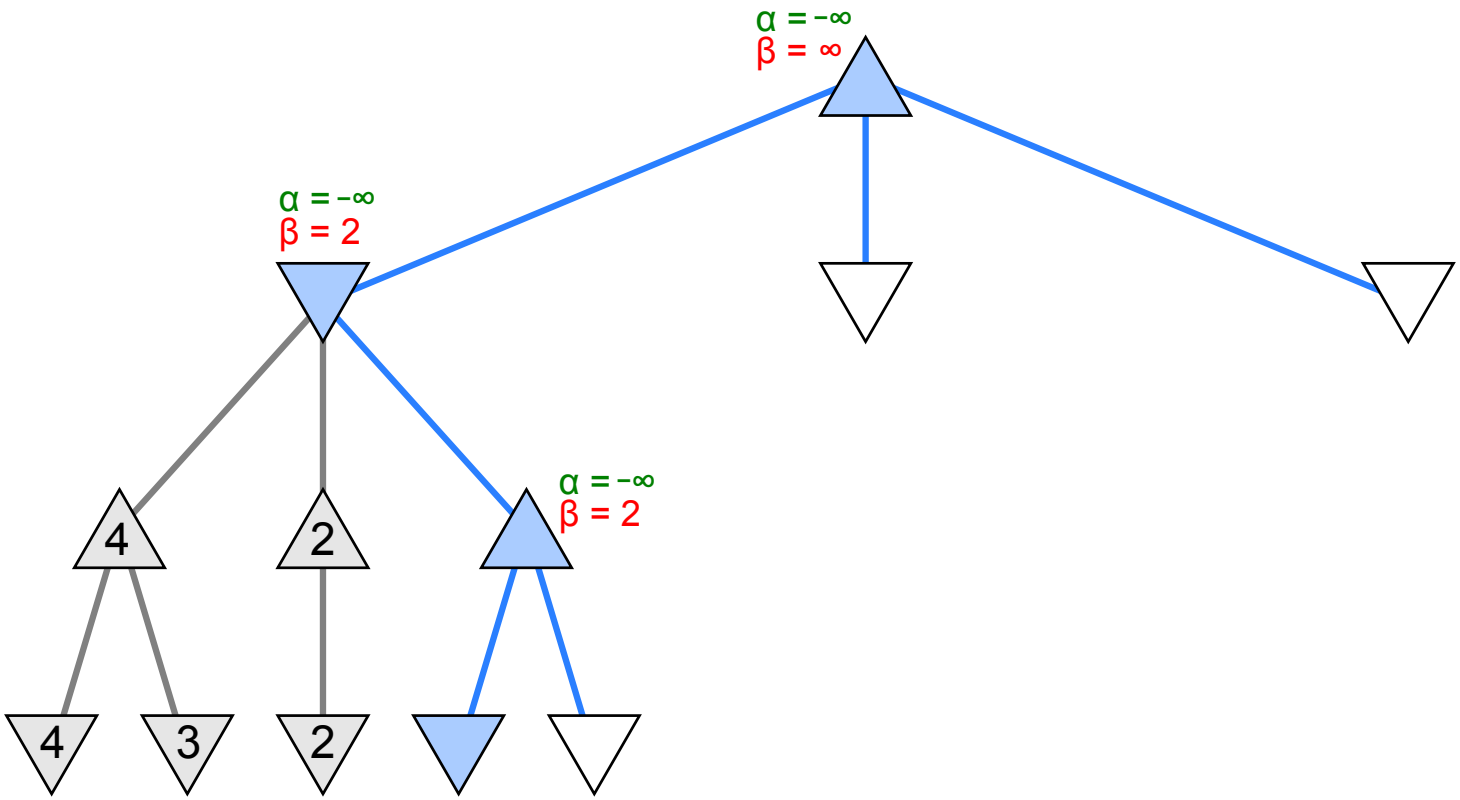
Minimax s alfa-beta prořezáváním: Příklad



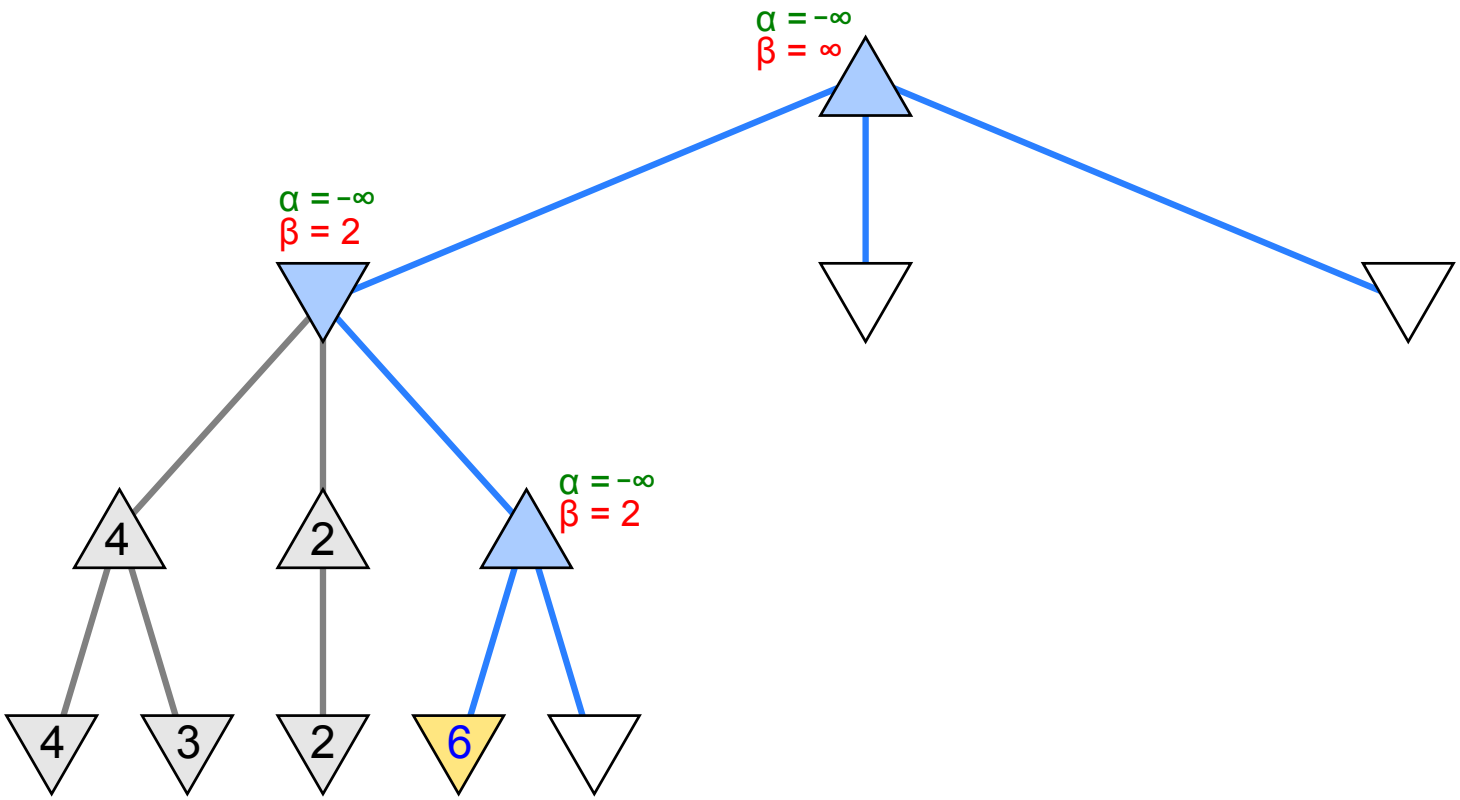
Minimax s alfa-beta prořezáváním: Příklad



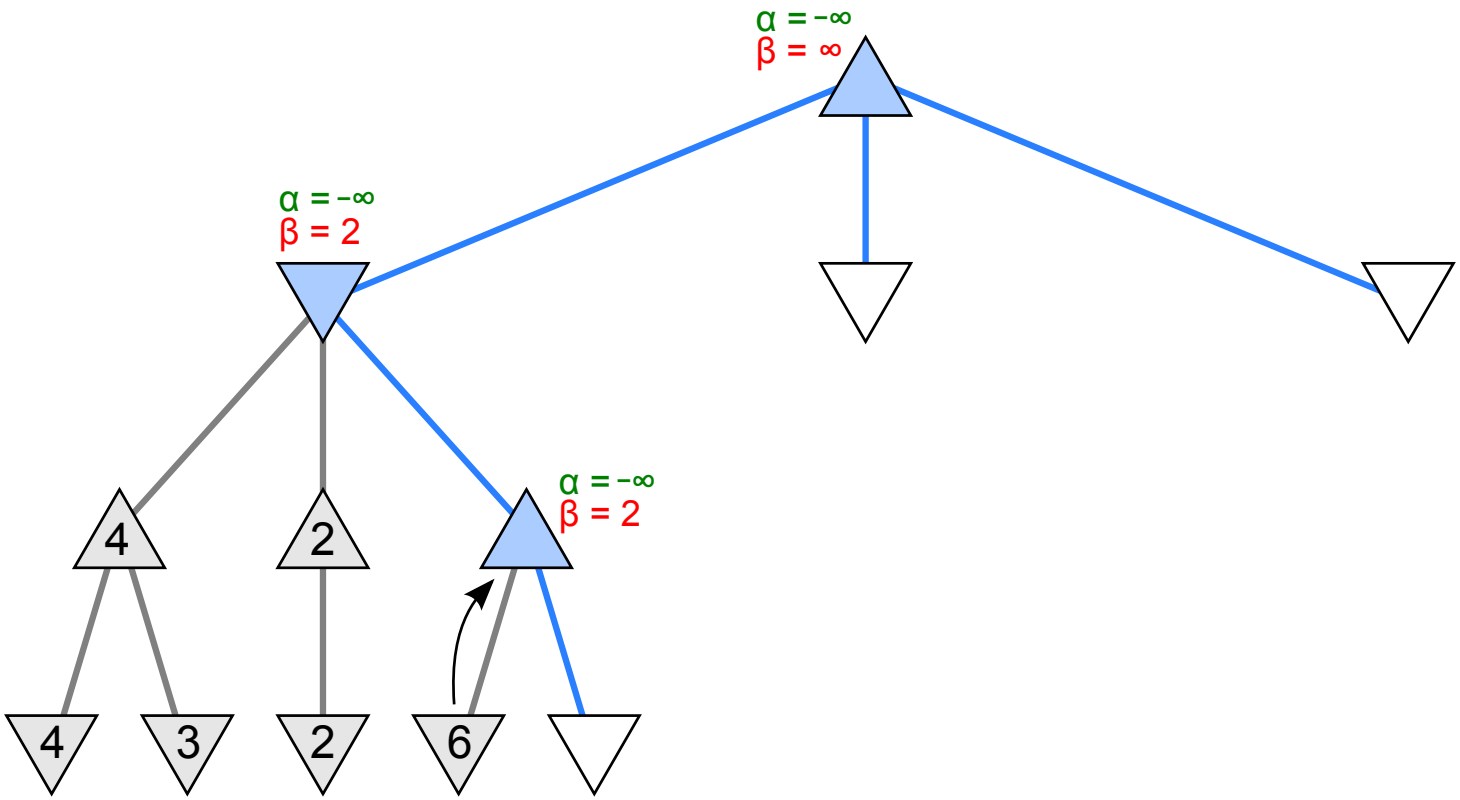
Minimax s alfa-beta prořezáváním: Příklad



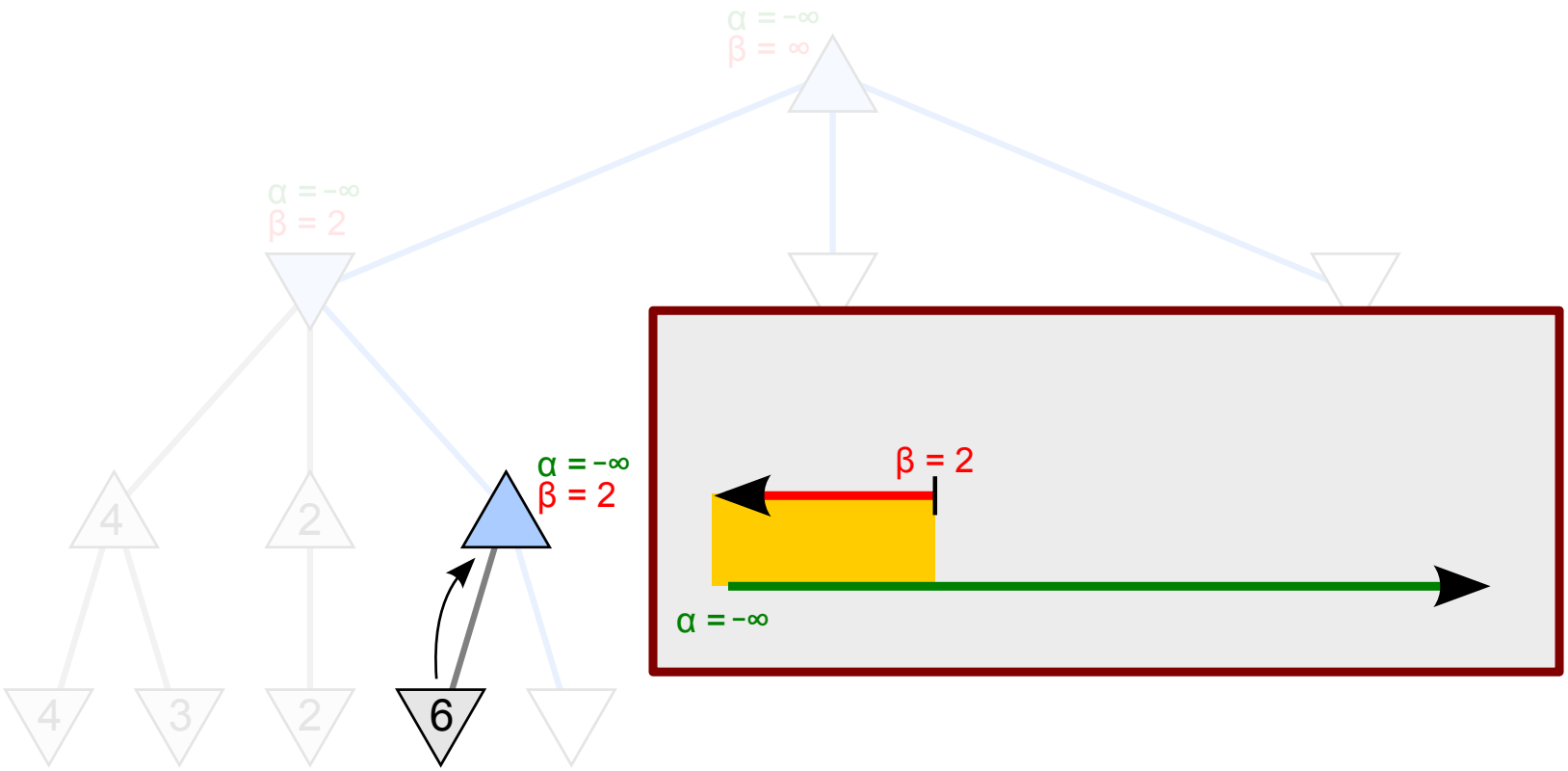
Minimax s alfa-beta prořezáváním: Příklad



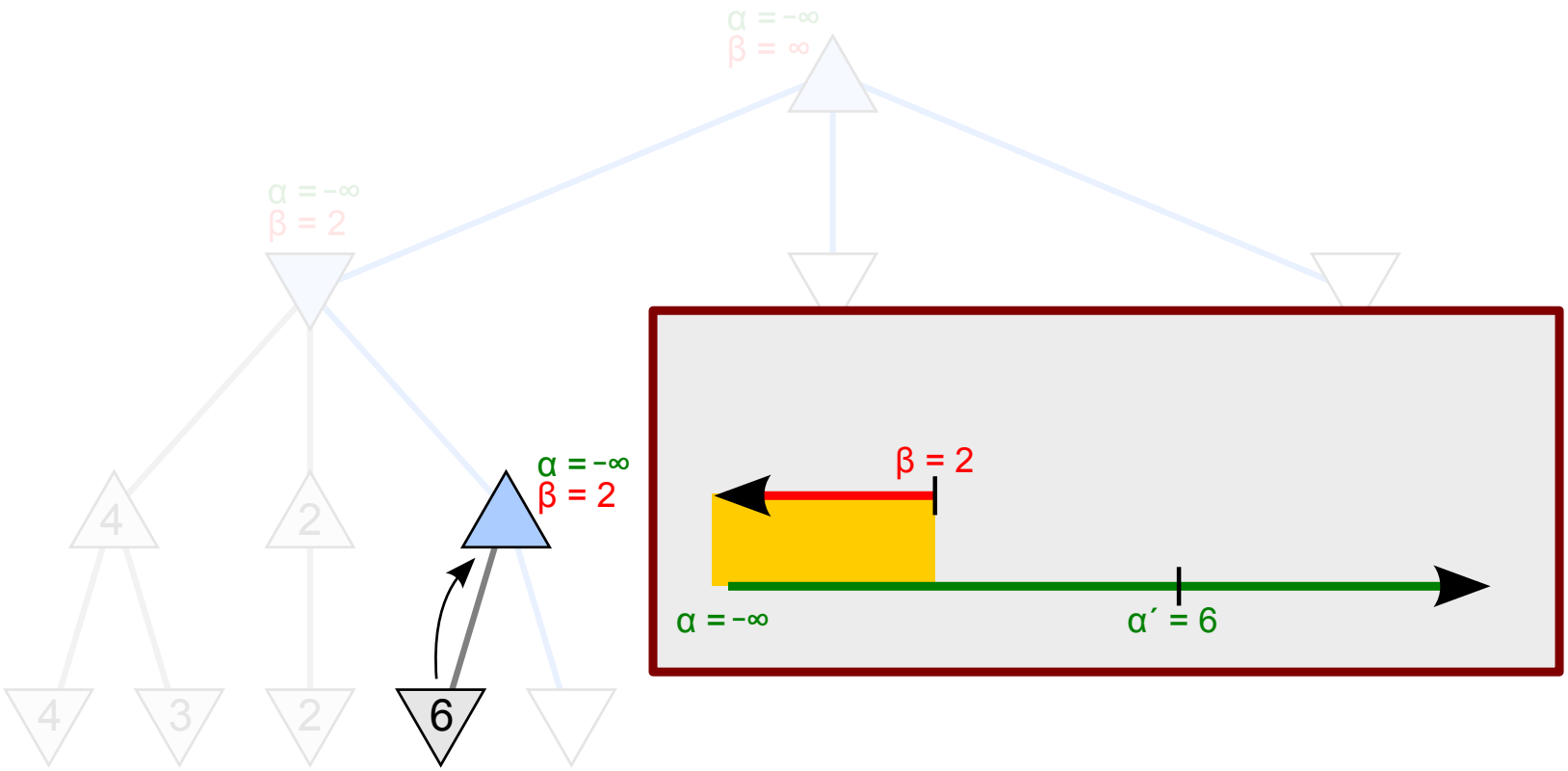
Minimax s alfa-beta prořezáváním: Příklad



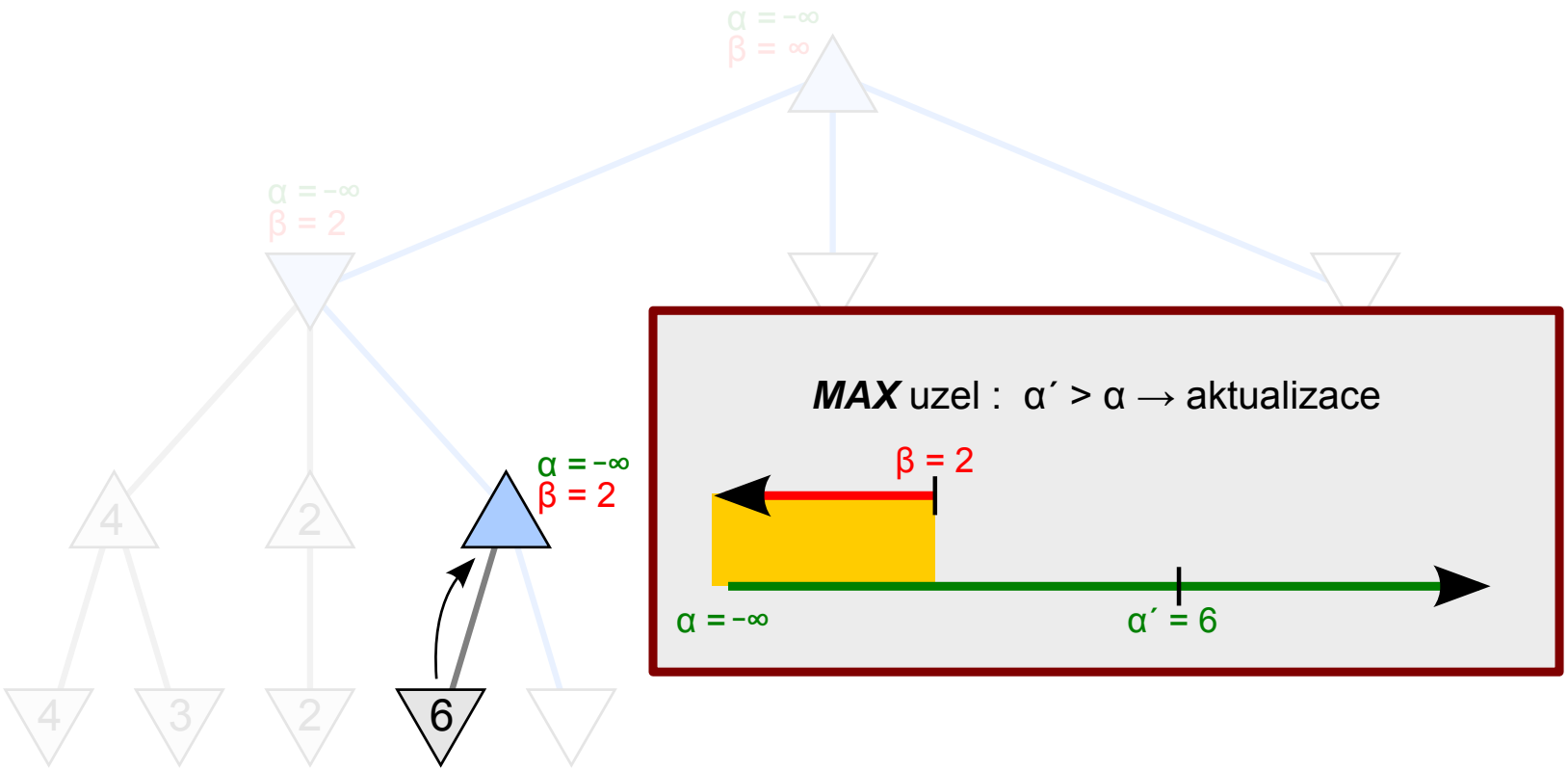
Minimax s alfa-beta prořezáváním: Příklad



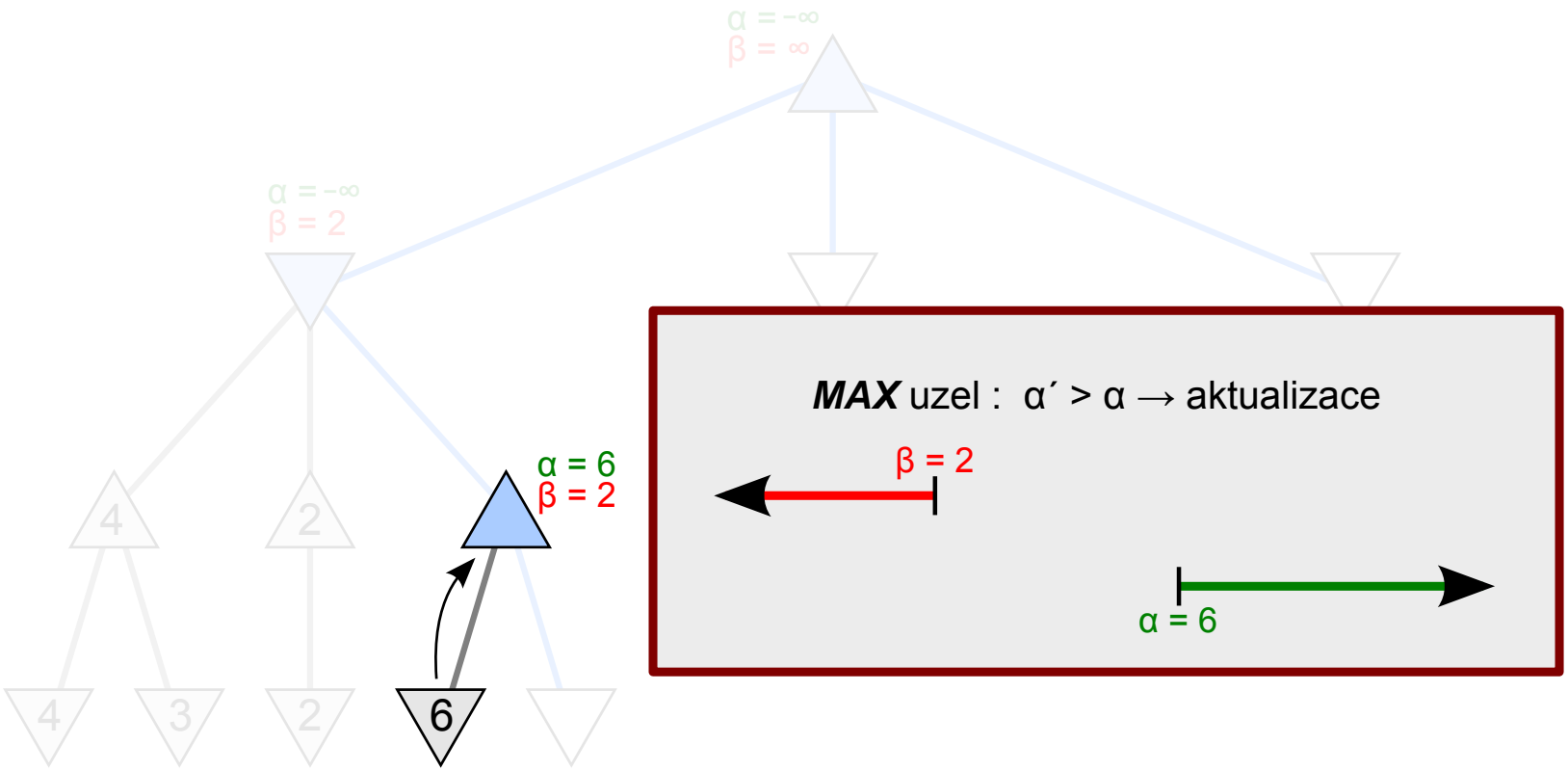
Minimax s alfa-beta prořezáváním: Příklad



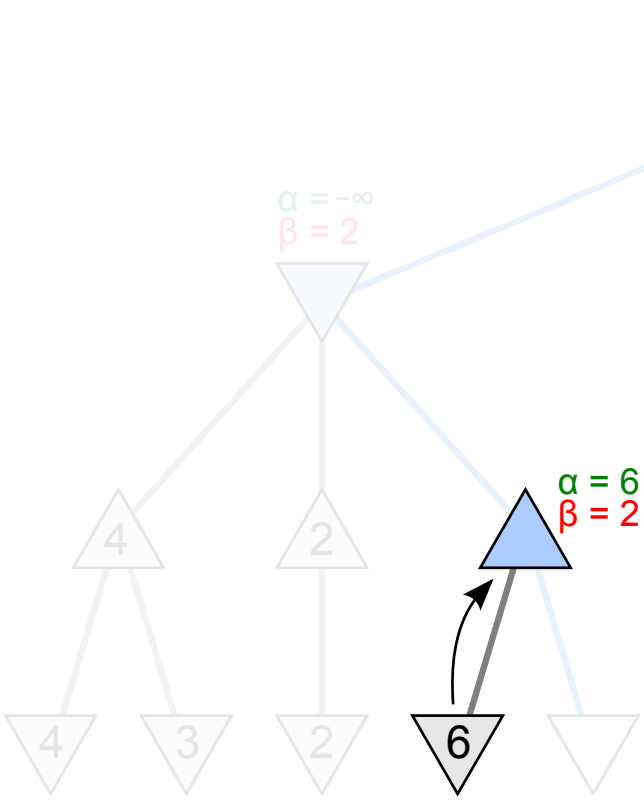
Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad

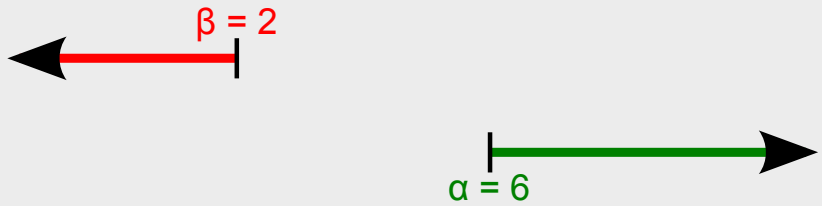


Minimax s alfa-beta prořezáváním: Příklad

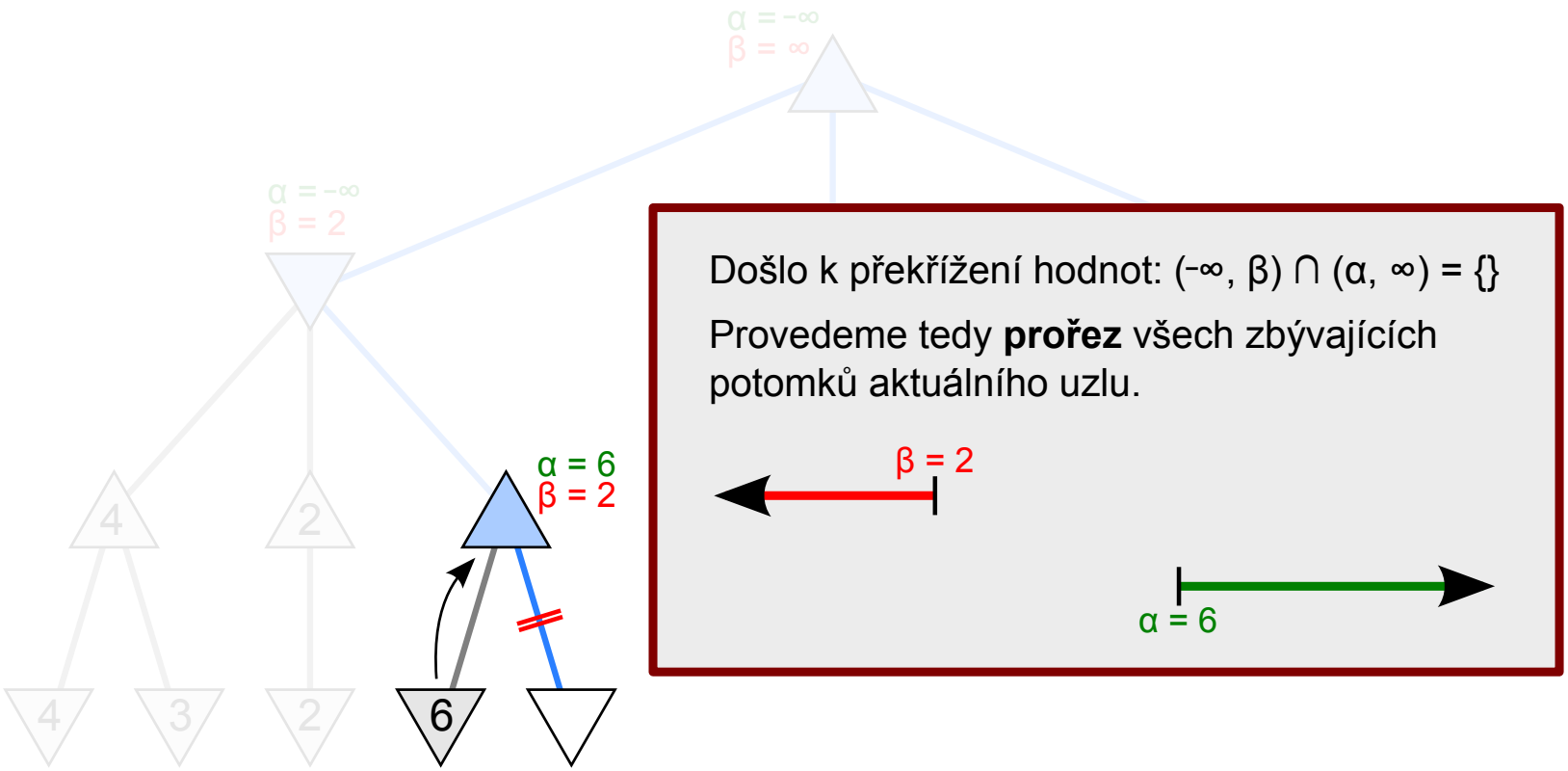


Došlo k překřížení hodnot: $(-\infty, \beta) \cap (\alpha, \infty) = \{\}$

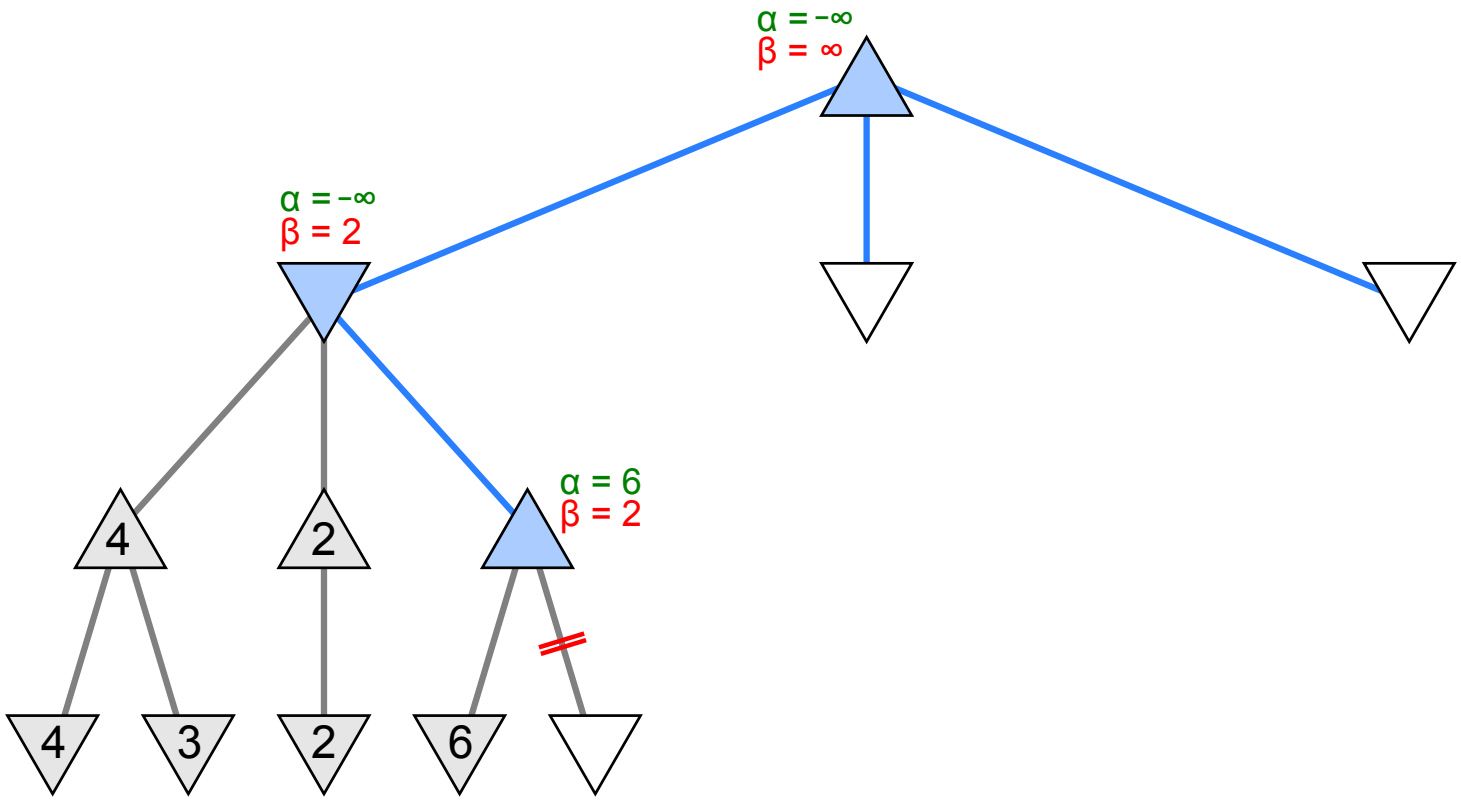
Provedeme tedy **prořez** všech zbývajících potomků aktuálního uzlu.



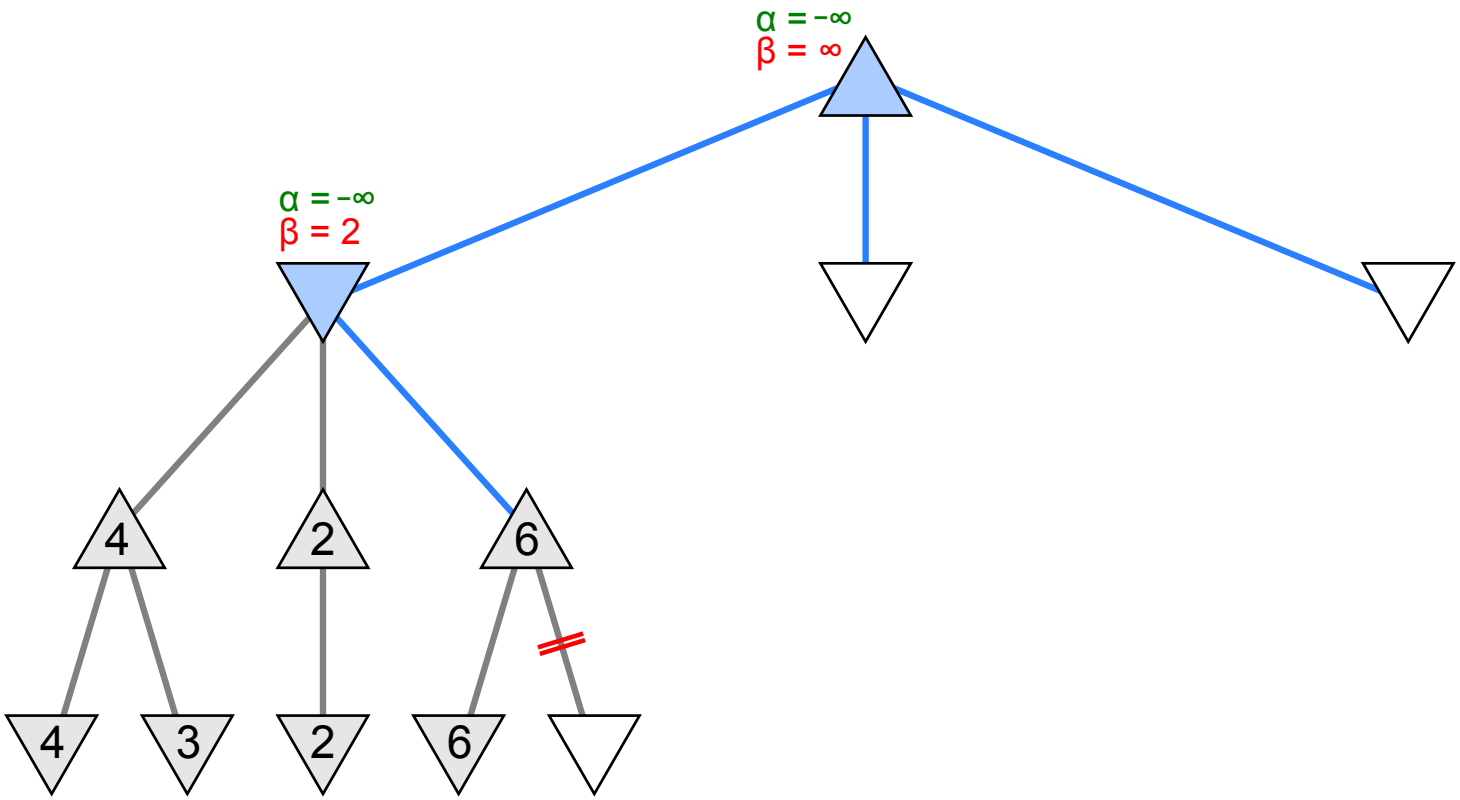
Minimax s alfa-beta prořezáváním: Příklad



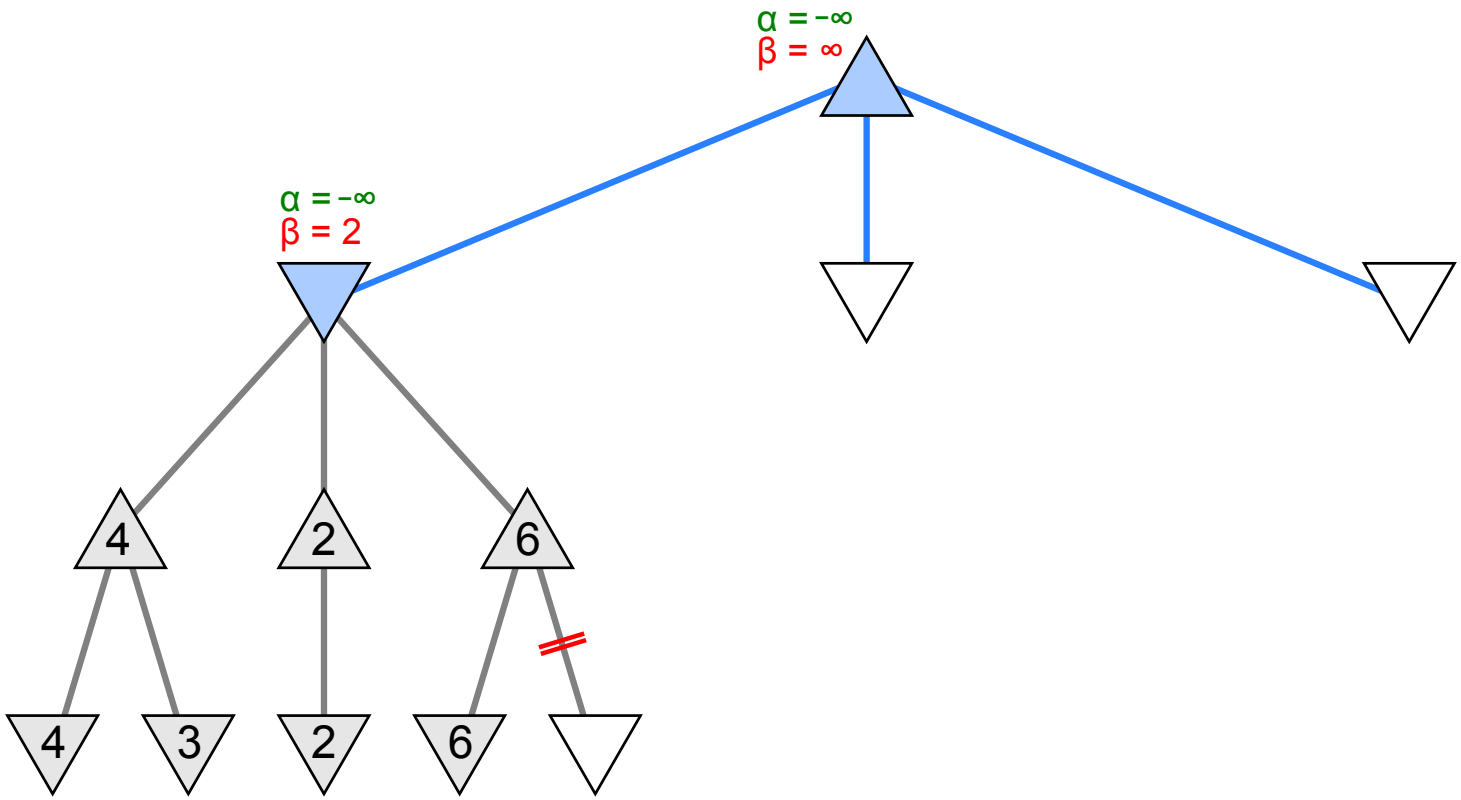
Minimax s alfa-beta prořezáváním: Příklad



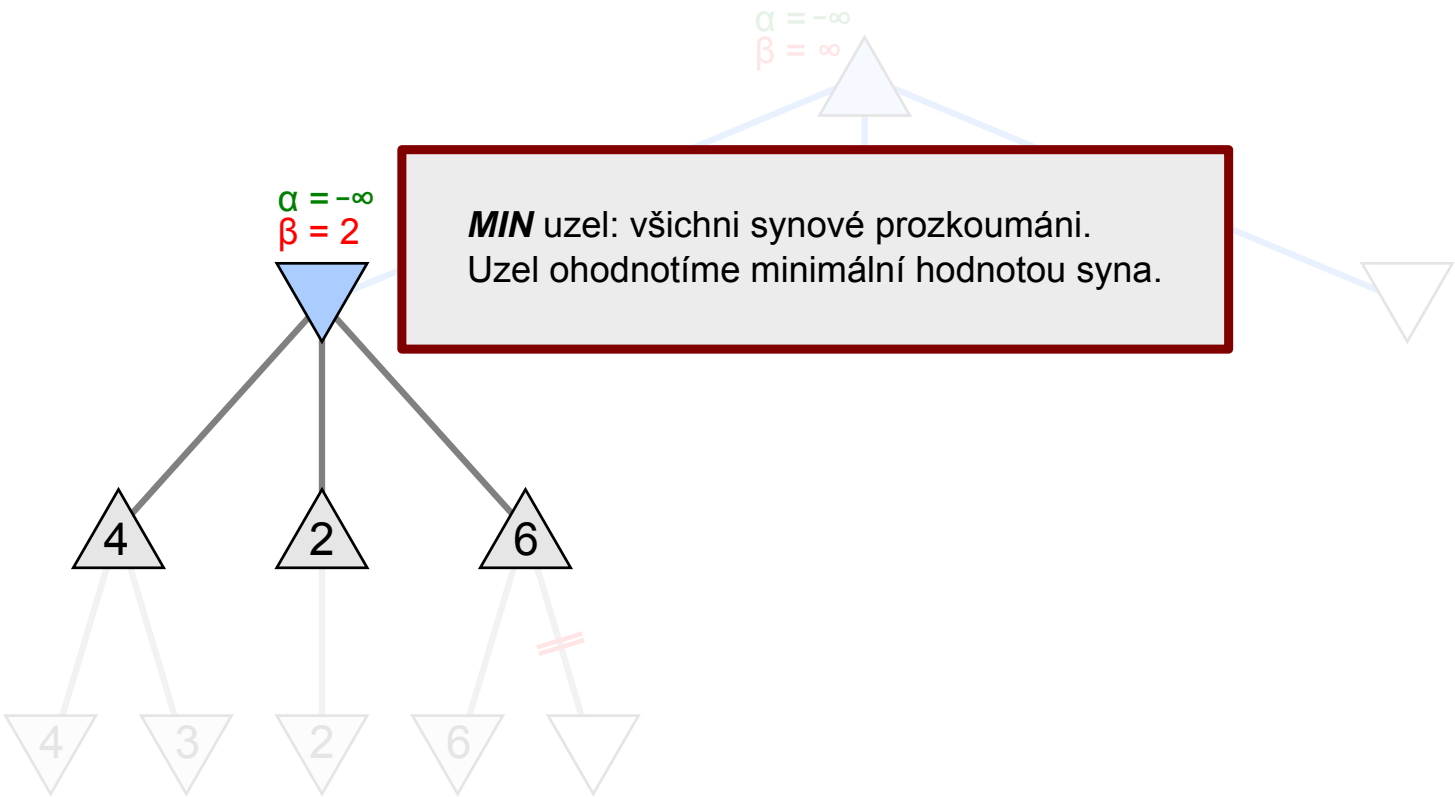
Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad

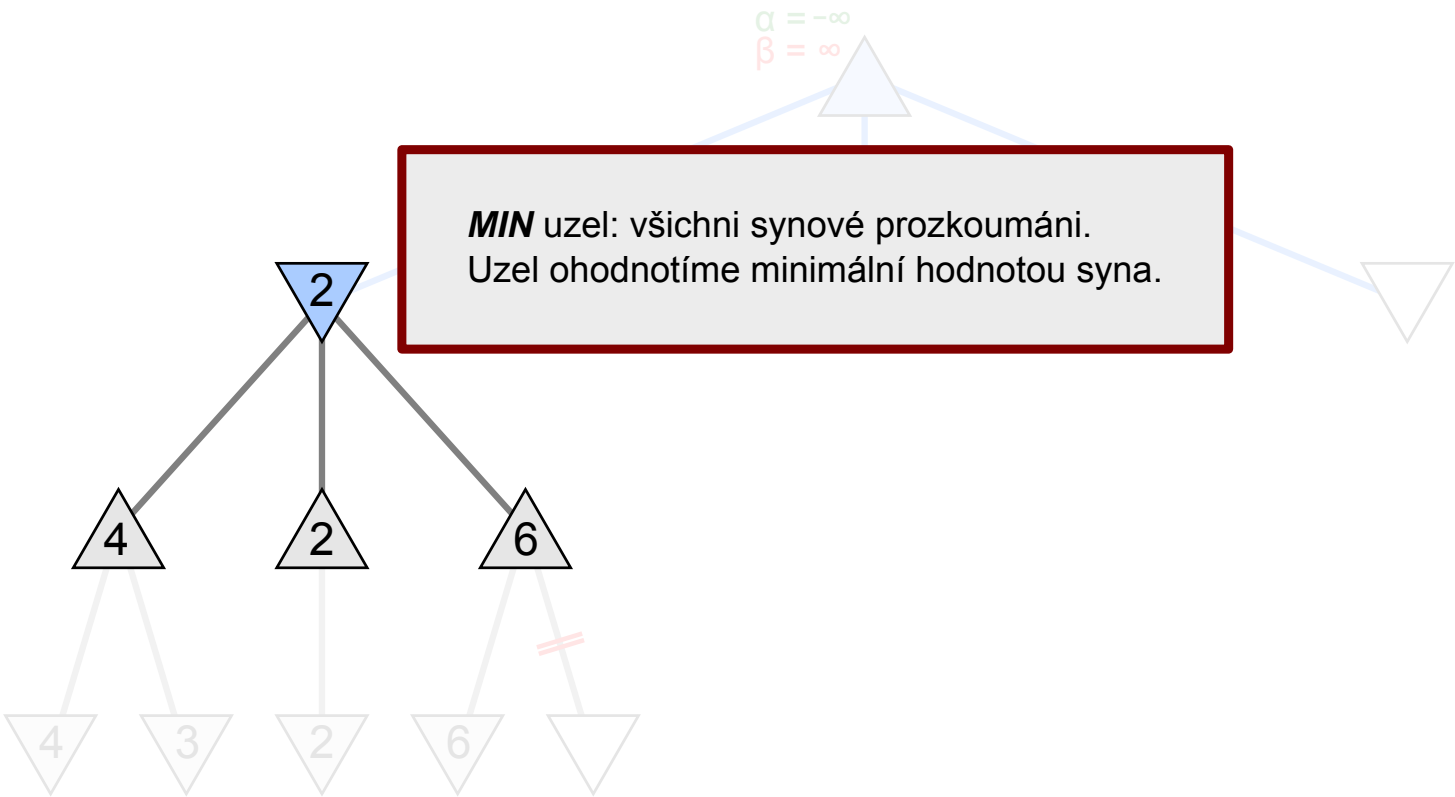


Minimax s alfa-beta prořezáváním: Příklad

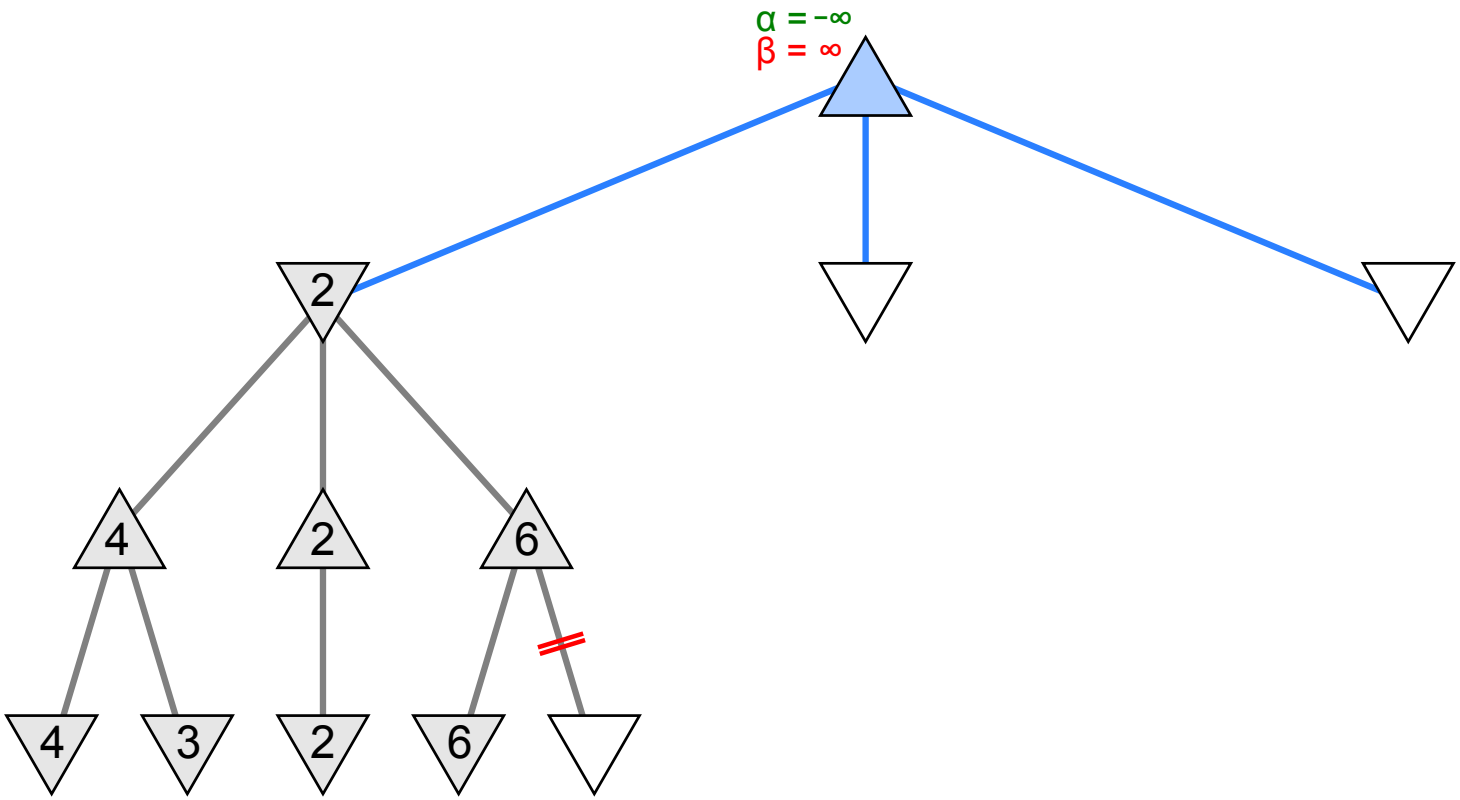


MIN uzel: všichni synové prozkoumáni.
Uzel ohodnotíme minimální hodnotou syna.

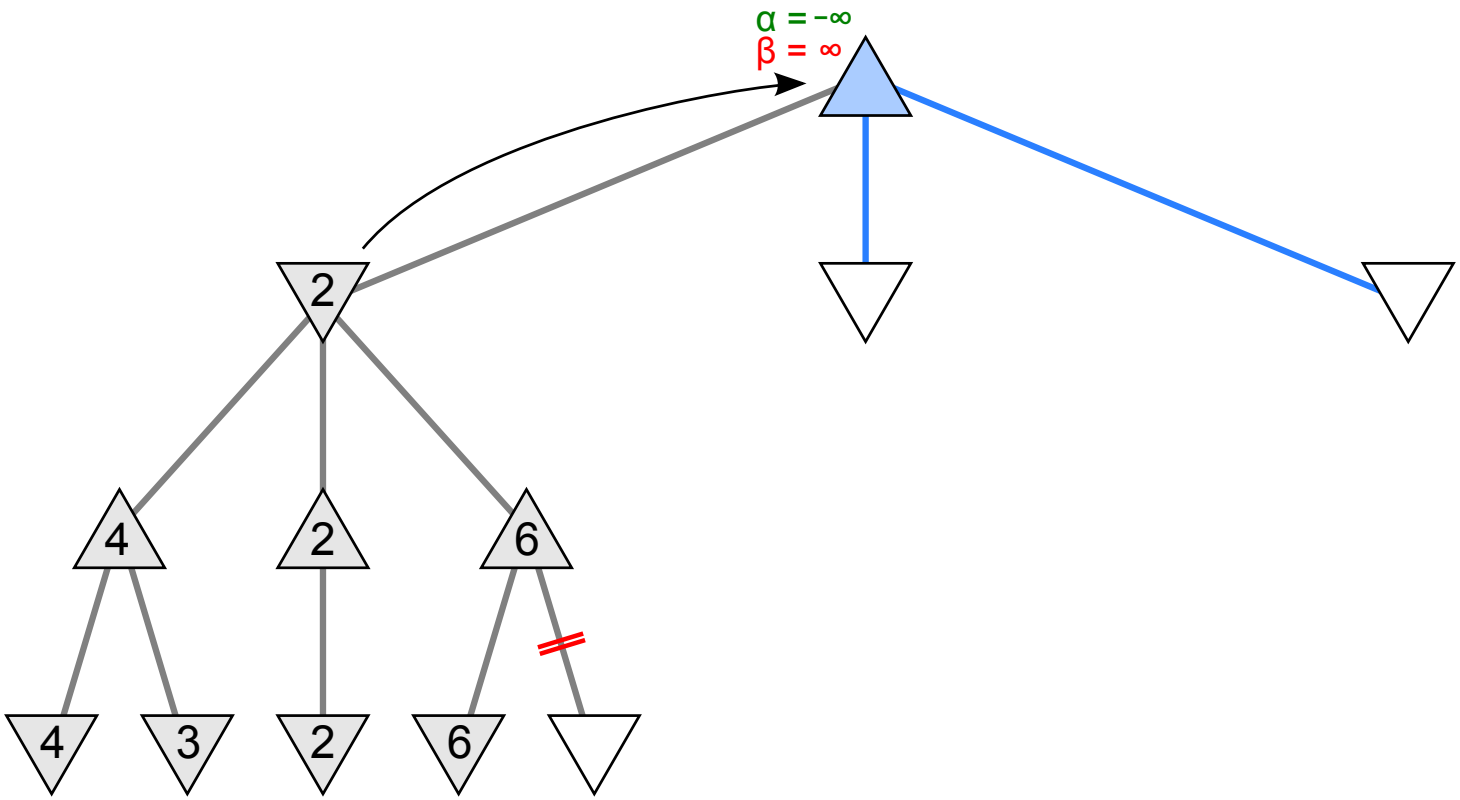
Minimax s alfa-beta prořezáváním: Příklad



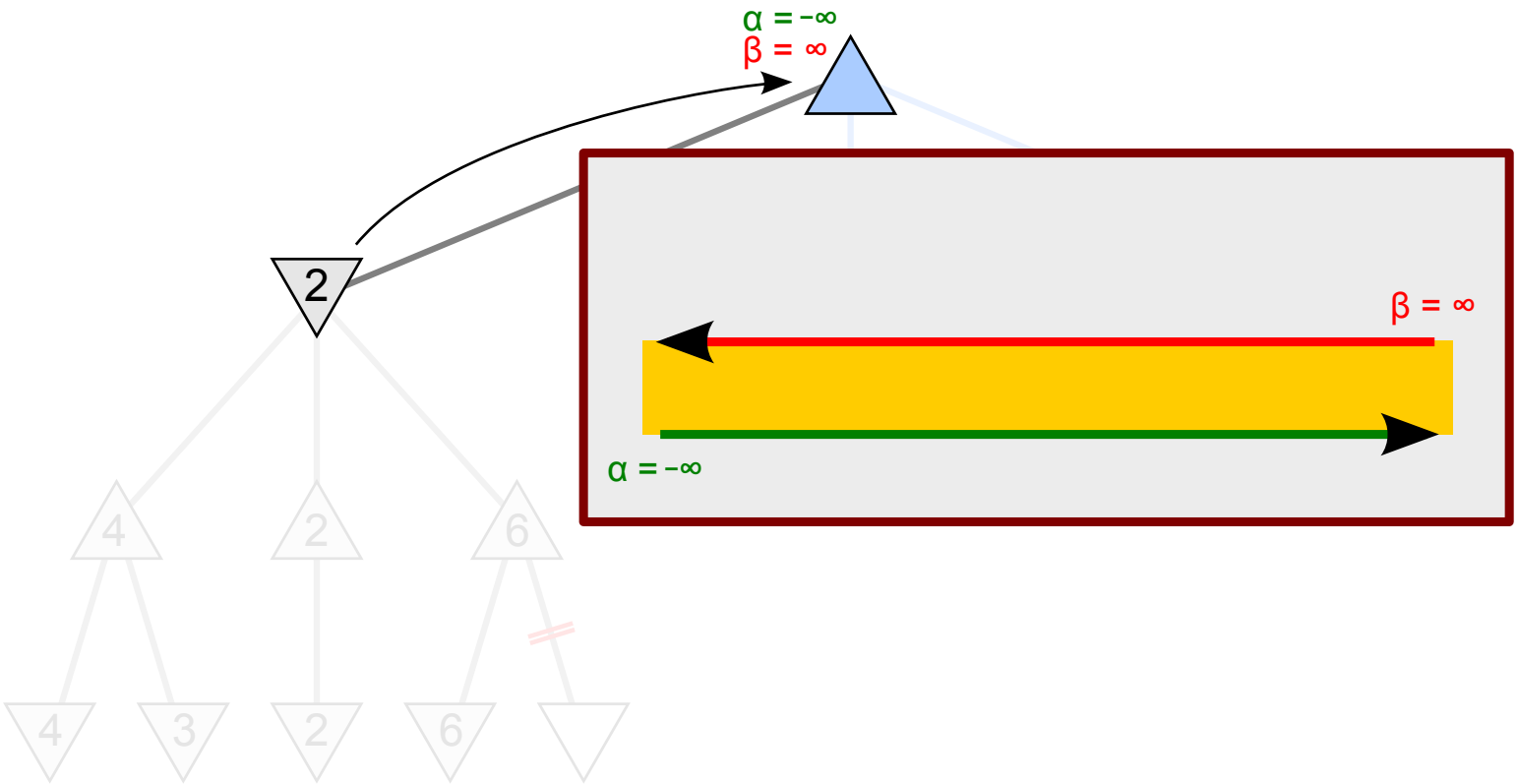
Minimax s alfa-beta prořezáváním: Příklad



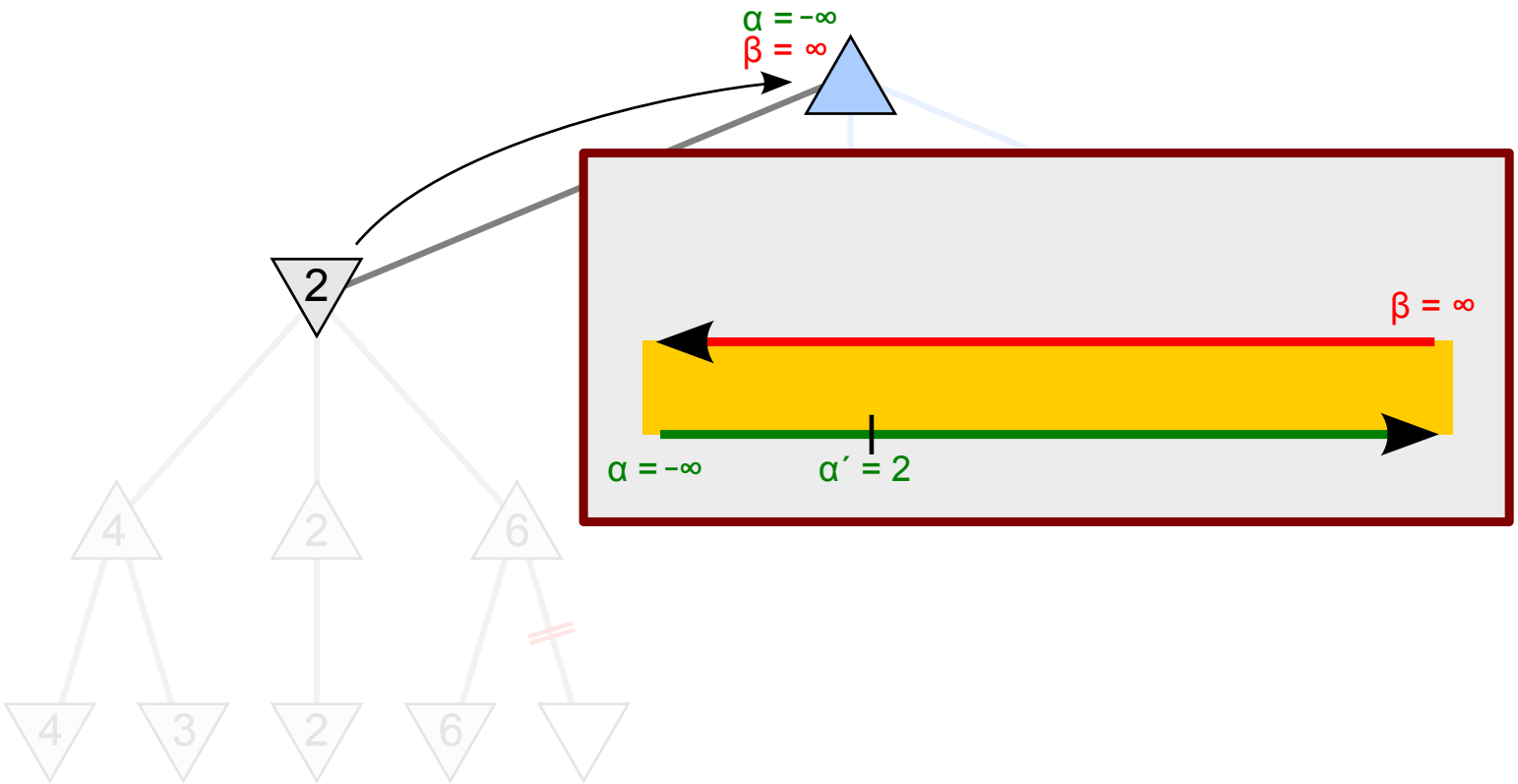
Minimax s alfa-beta prořezáváním: Příklad



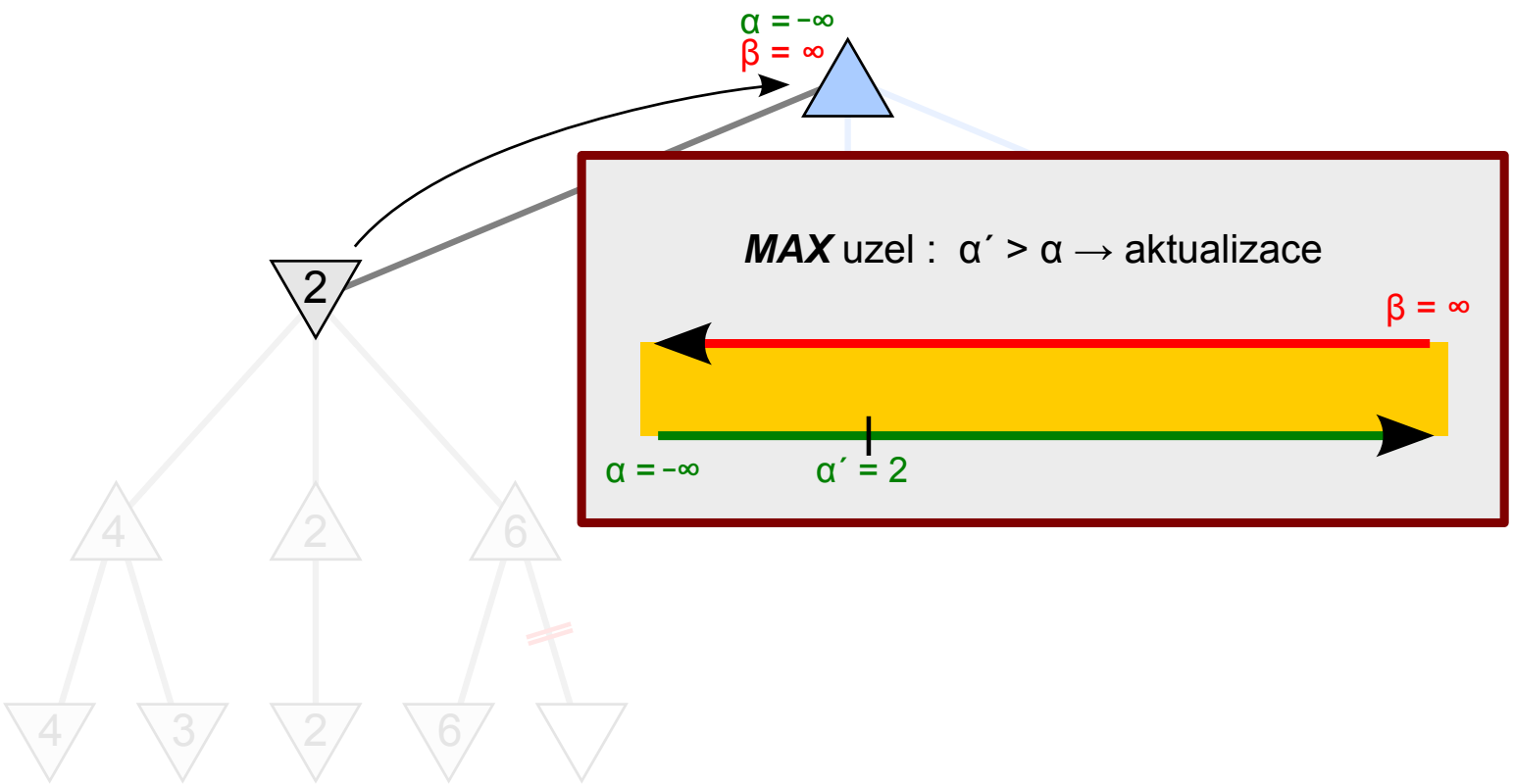
Minimax s alfa-beta prořezáváním: Příklad



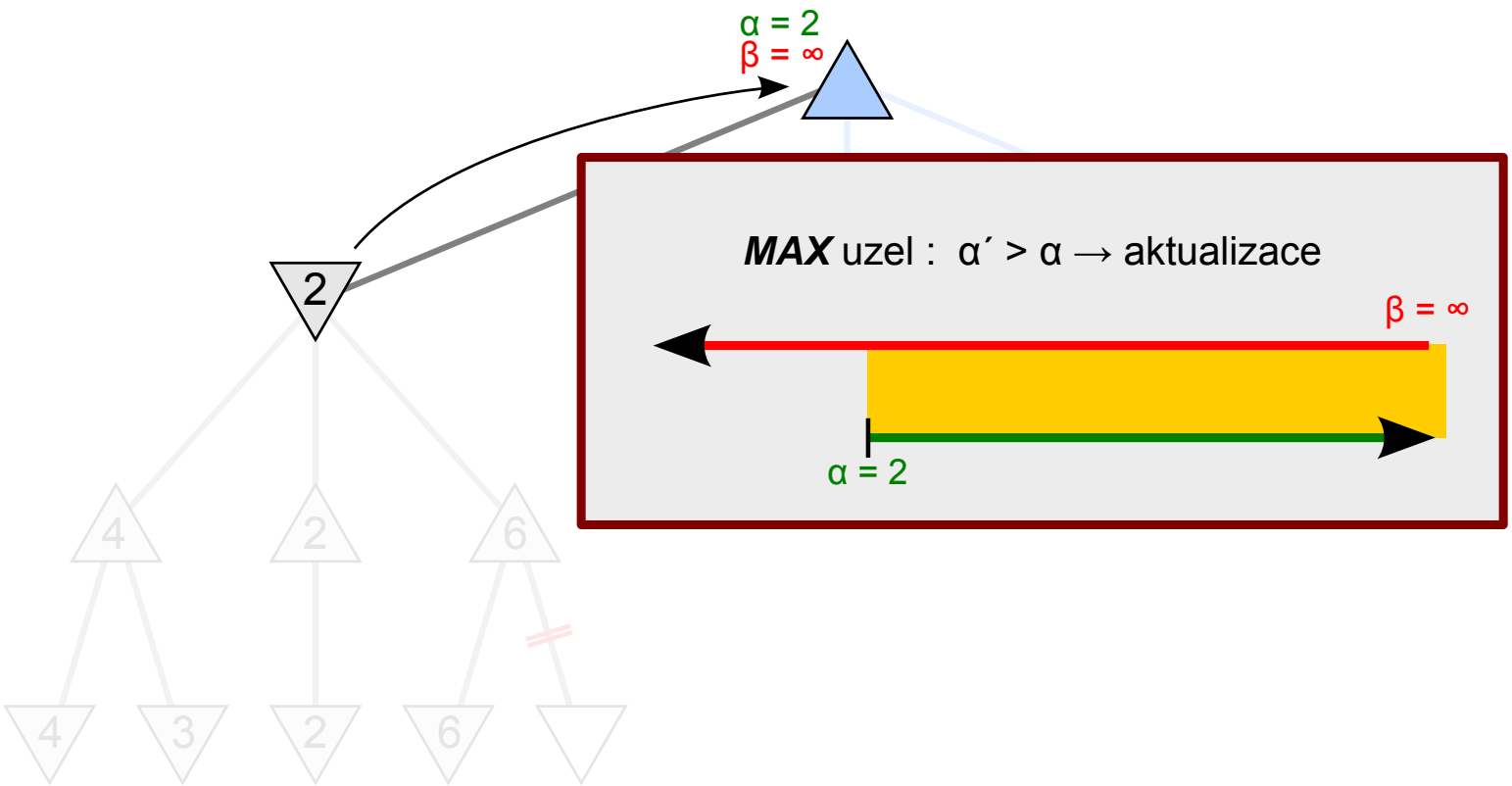
Minimax s alfa-beta prořezáváním: Příklad



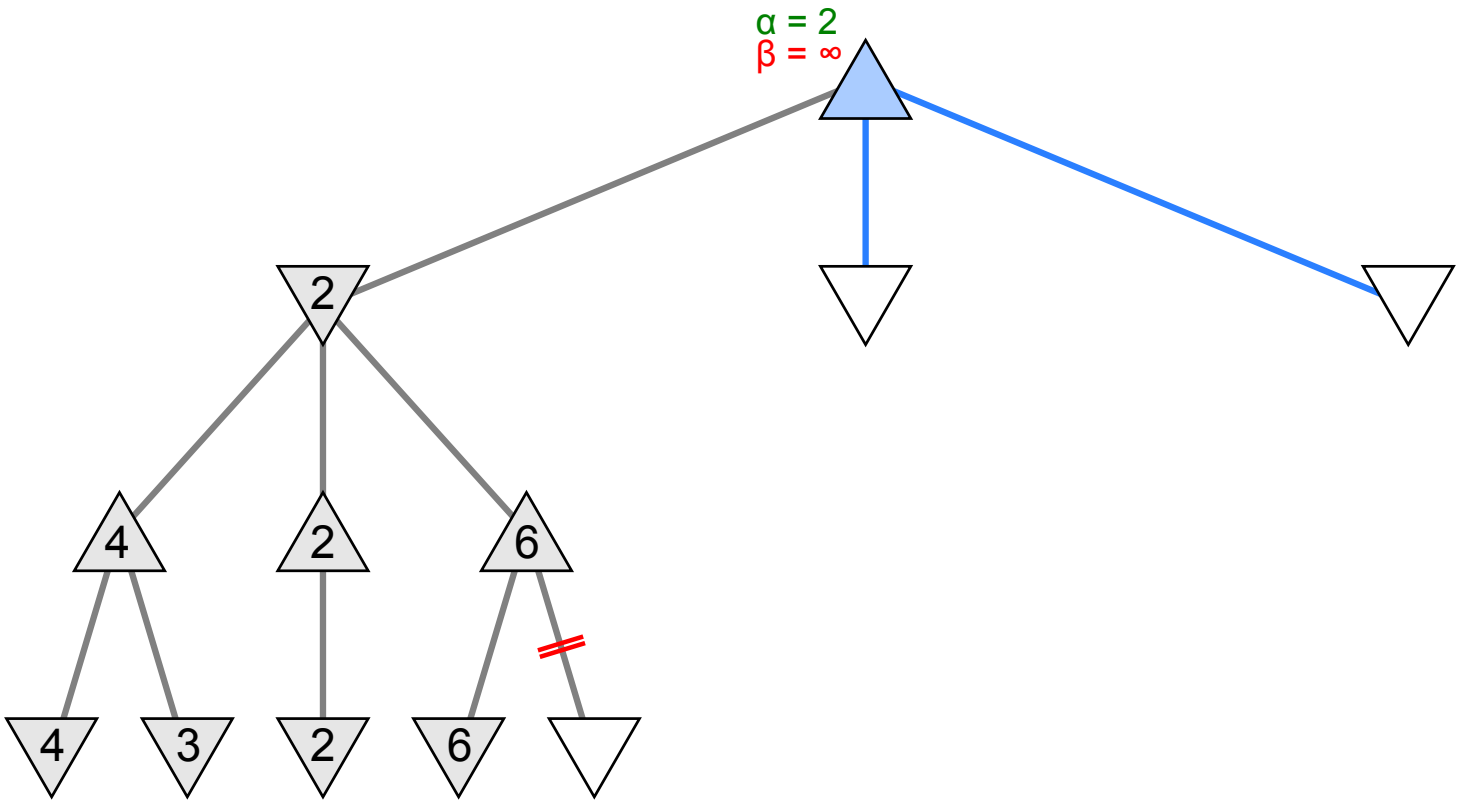
Minimax s alfa-beta prořezáváním: Příklad



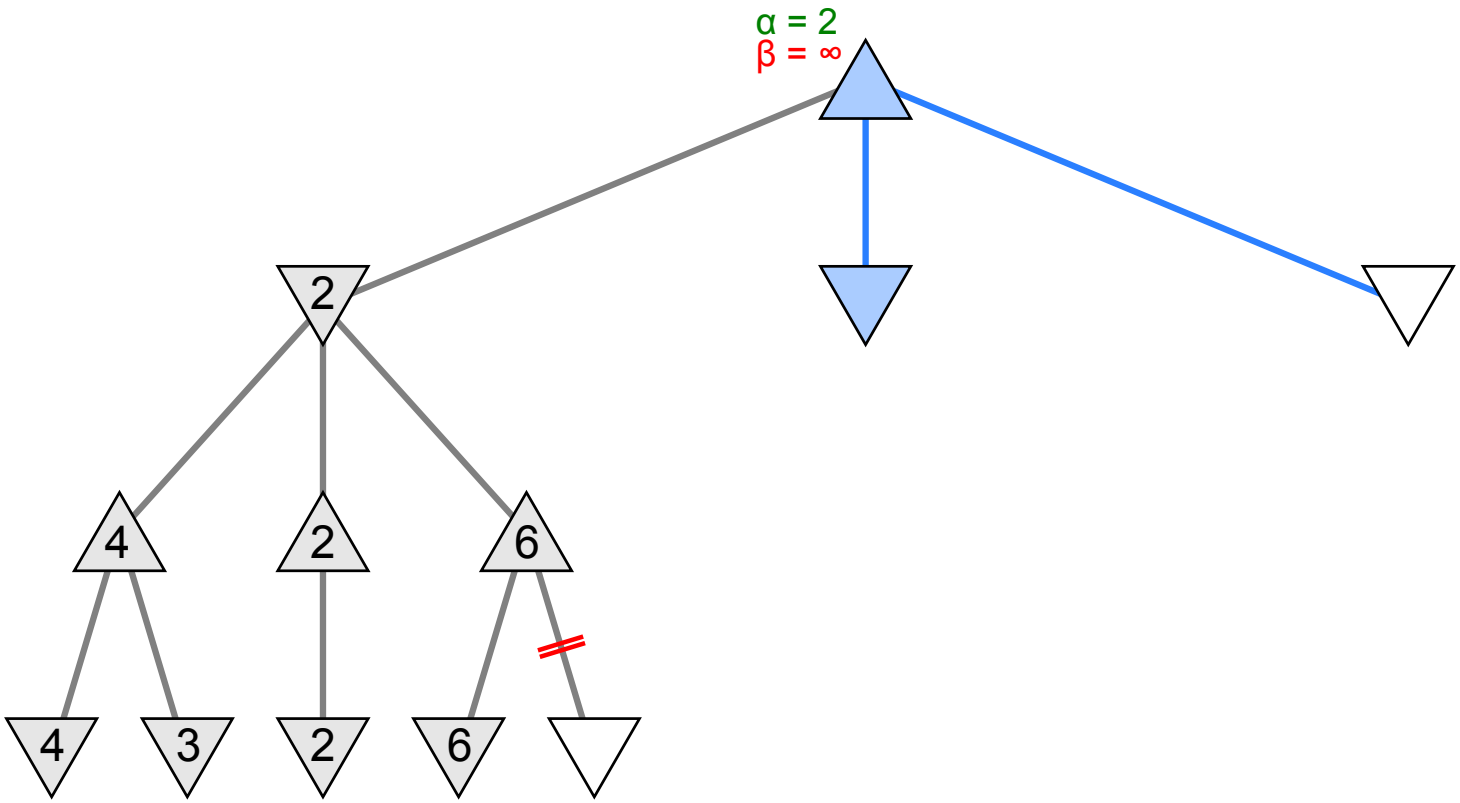
Minimax s alfa-beta prořezáváním: Příklad



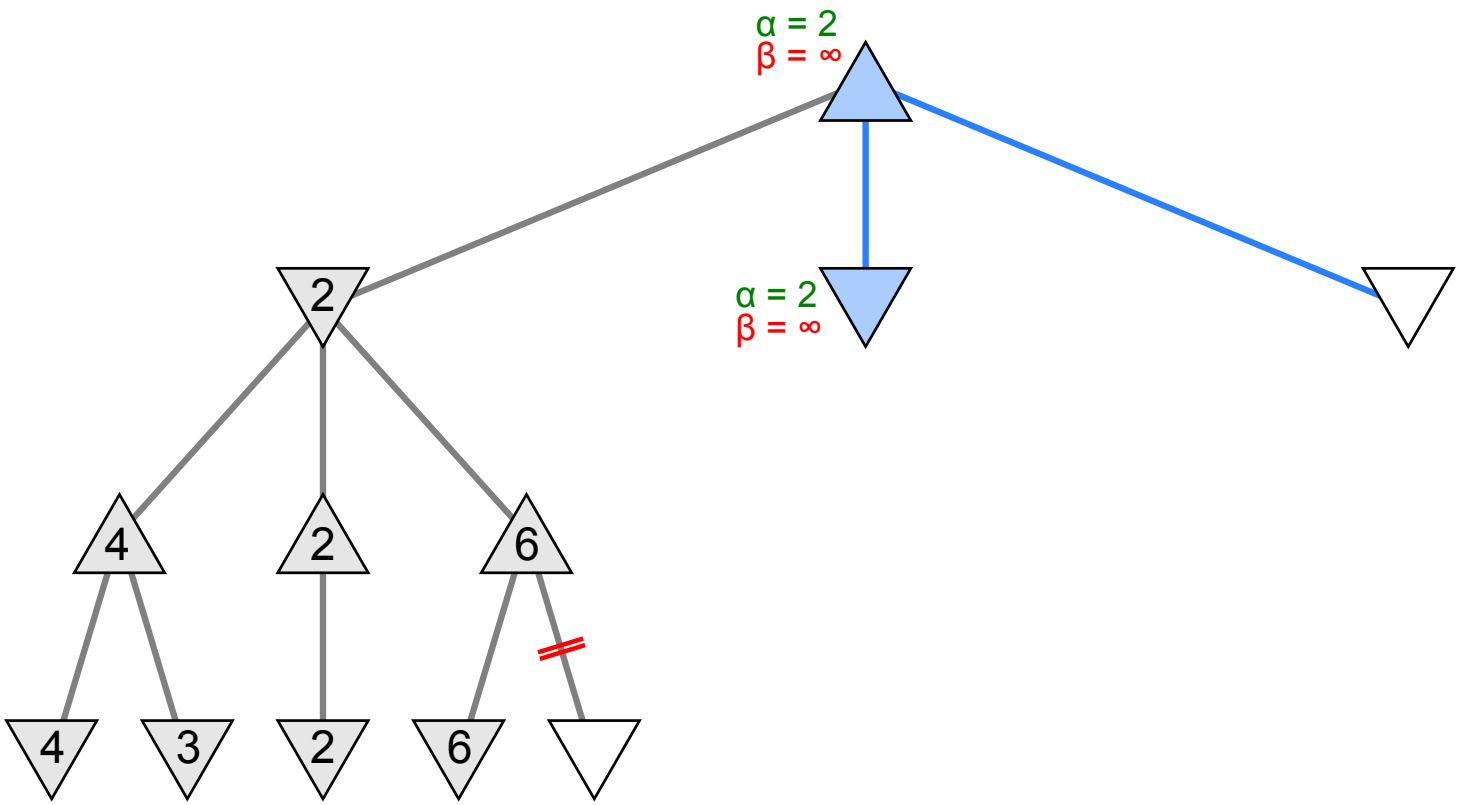
Minimax s alfa-beta prořezáváním: Příklad



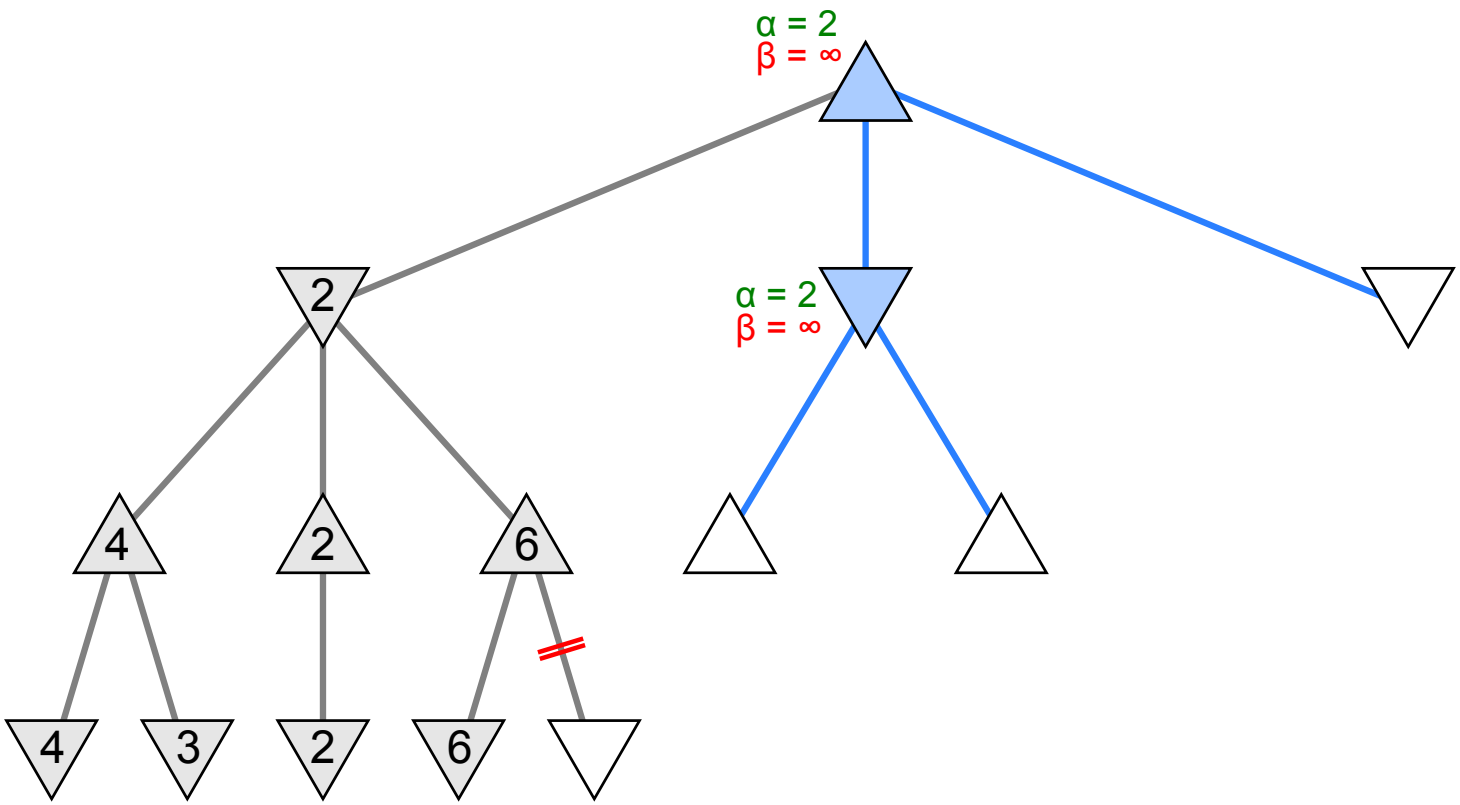
Minimax s alfa-beta prořezáváním: Příklad



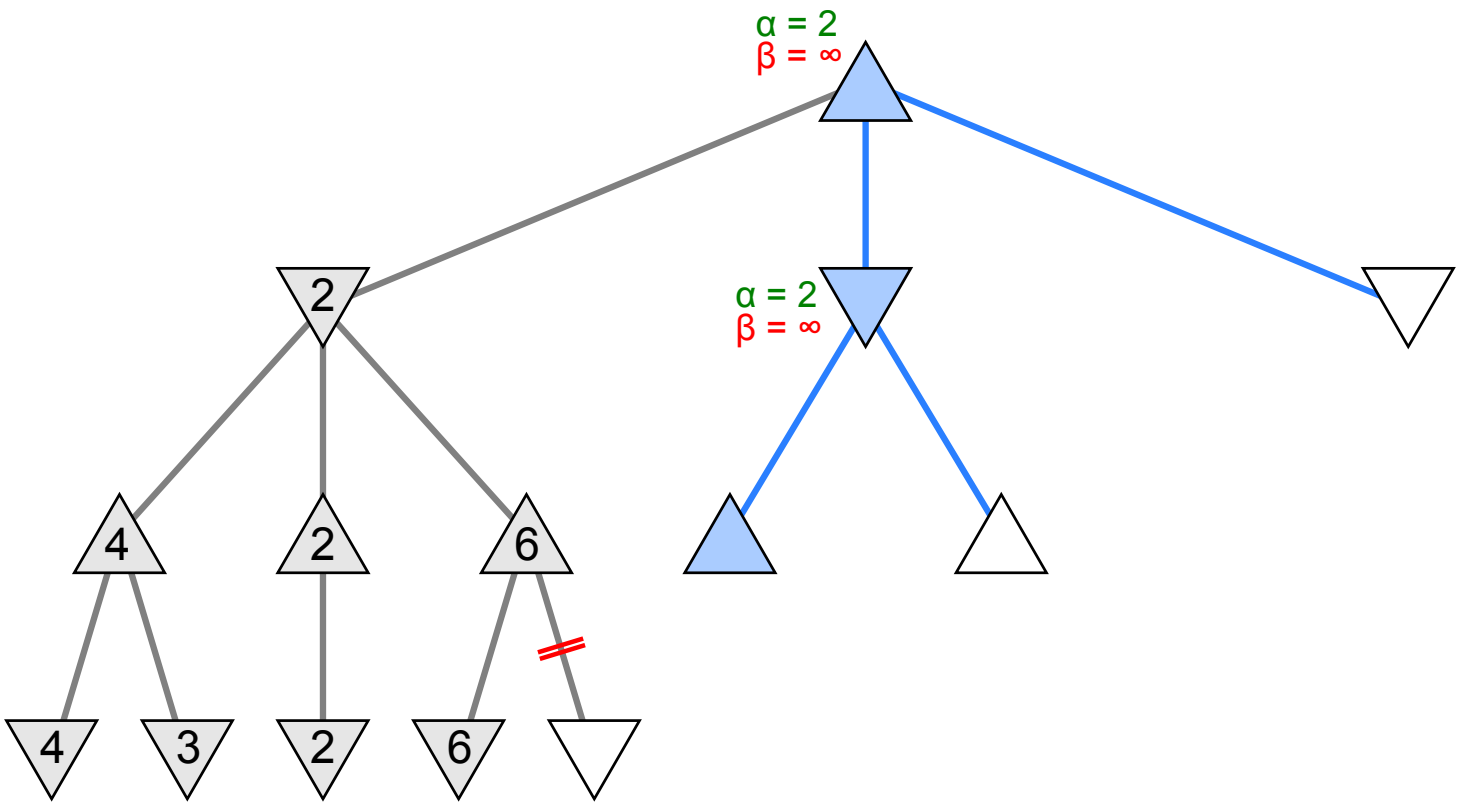
Minimax s alfa-beta prořezáváním: Příklad



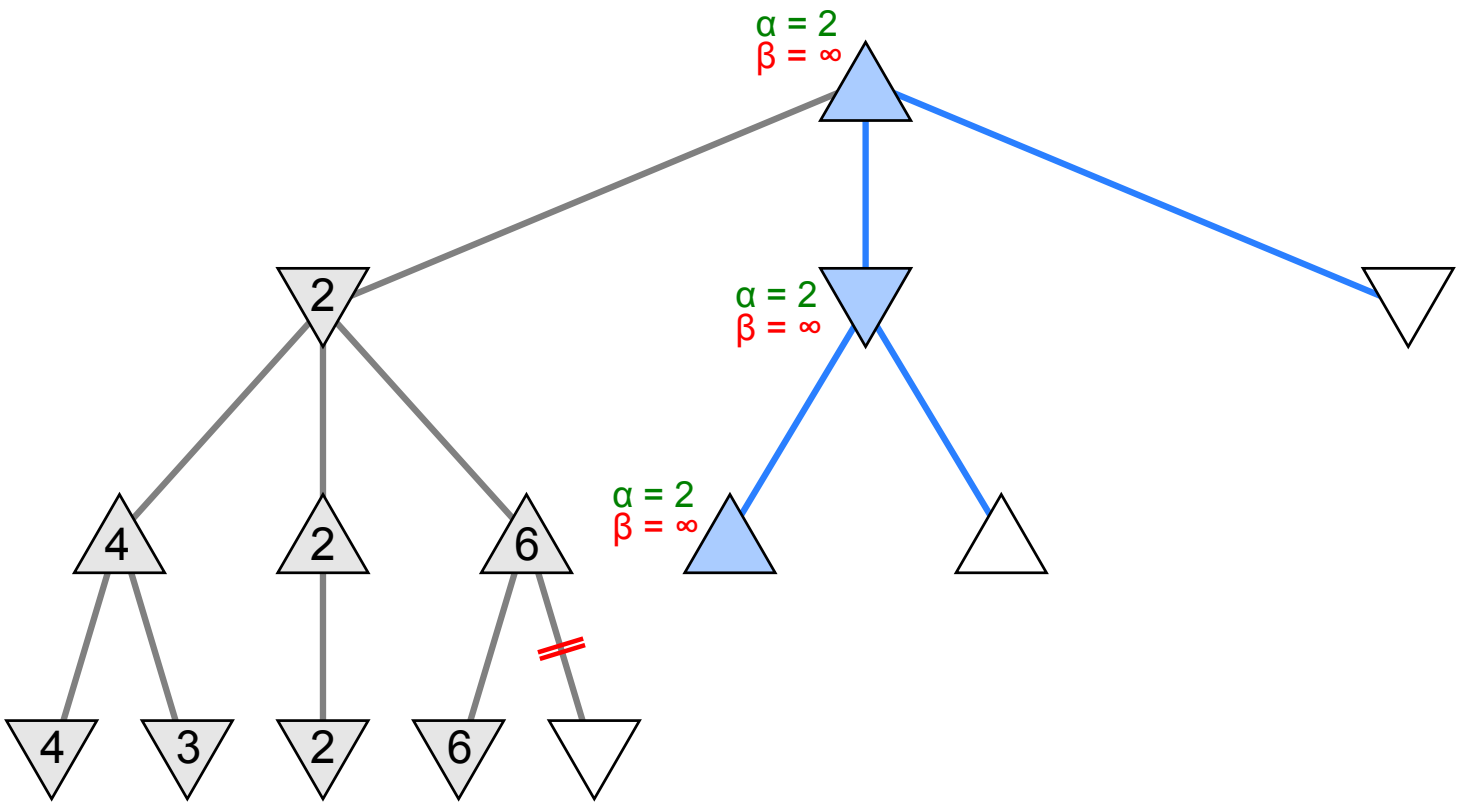
Minimax s alfa-beta prořezáváním: Příklad



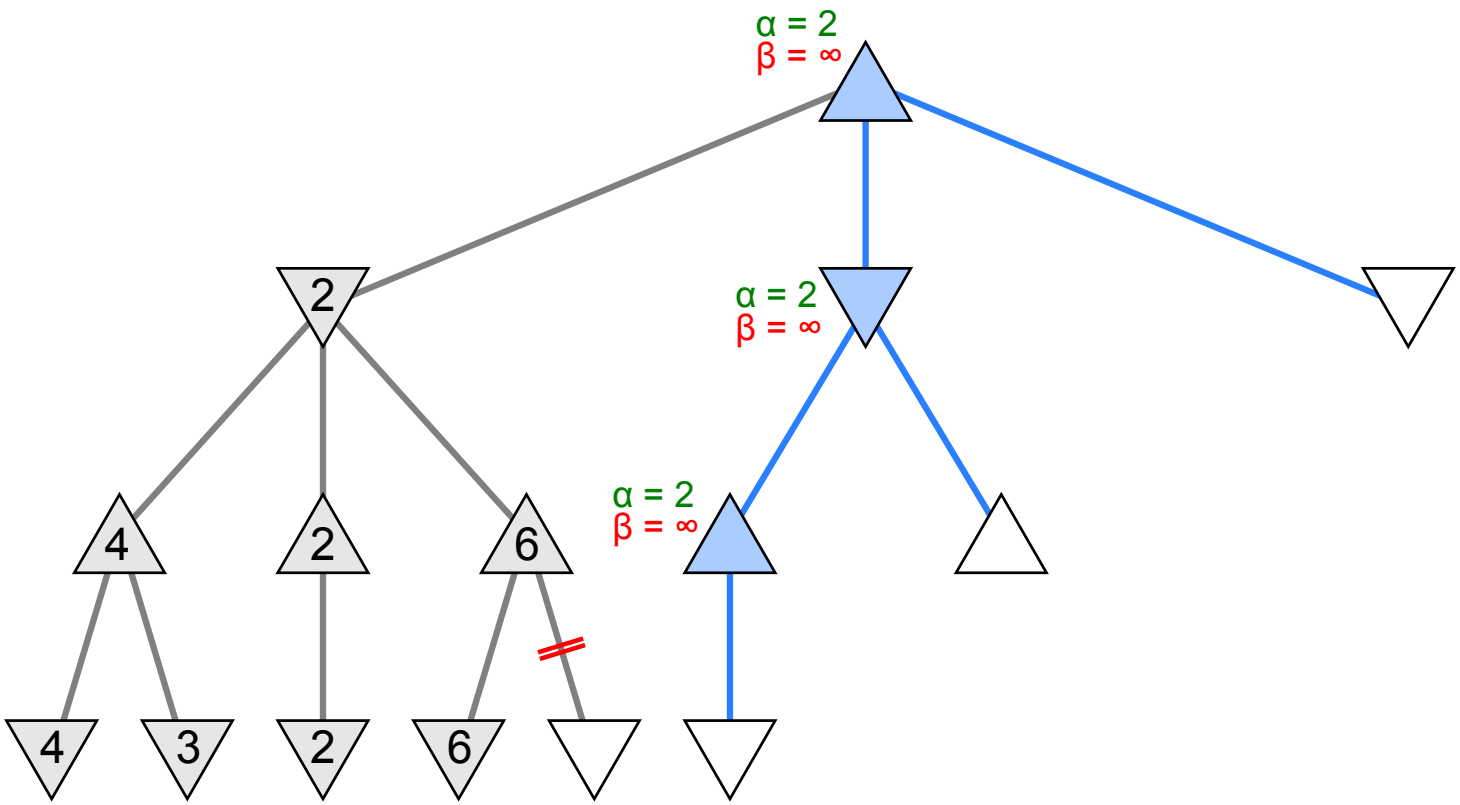
Minimax s alfa-beta prořezáváním: Příklad



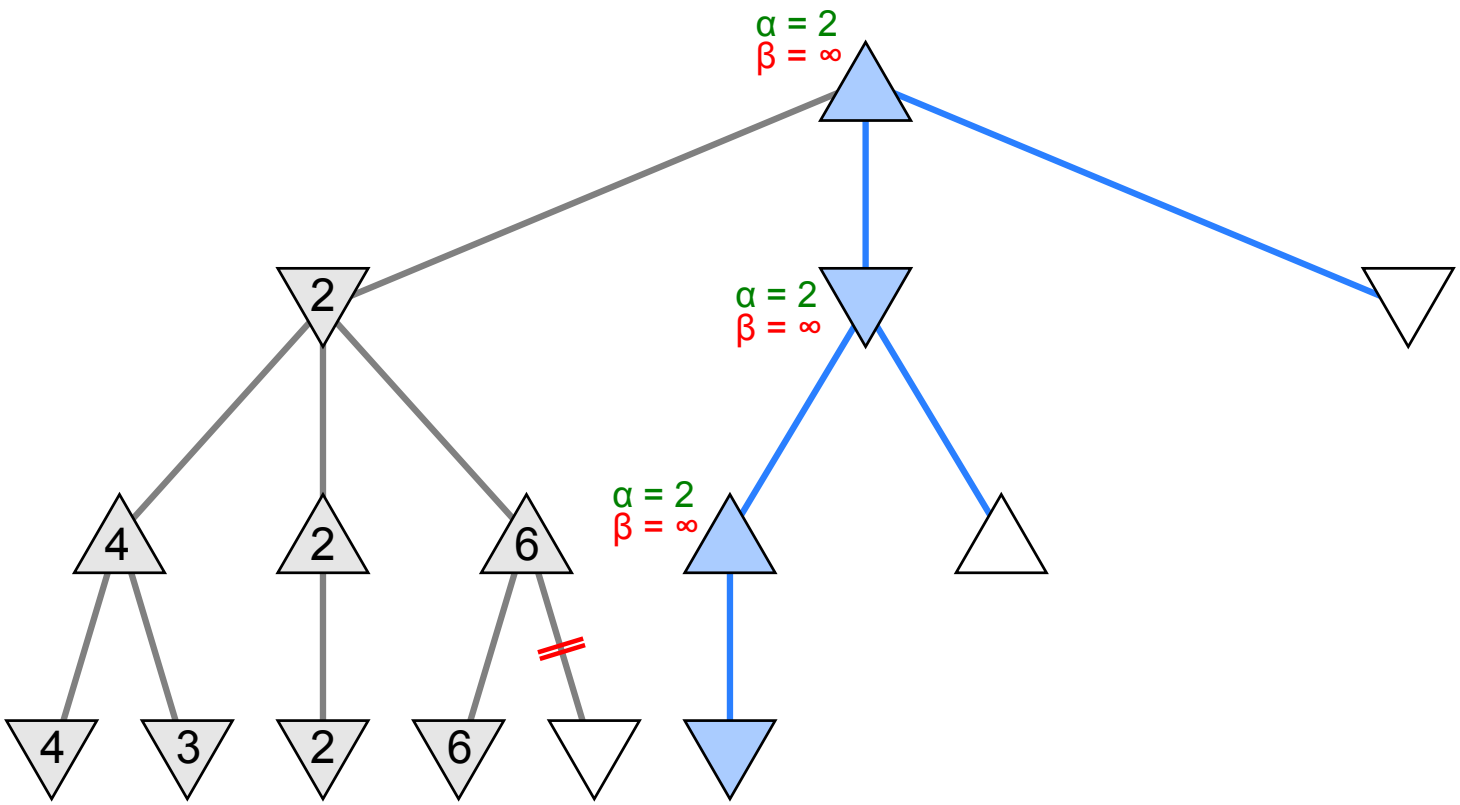
Minimax s alfa-beta prořezáváním: Příklad



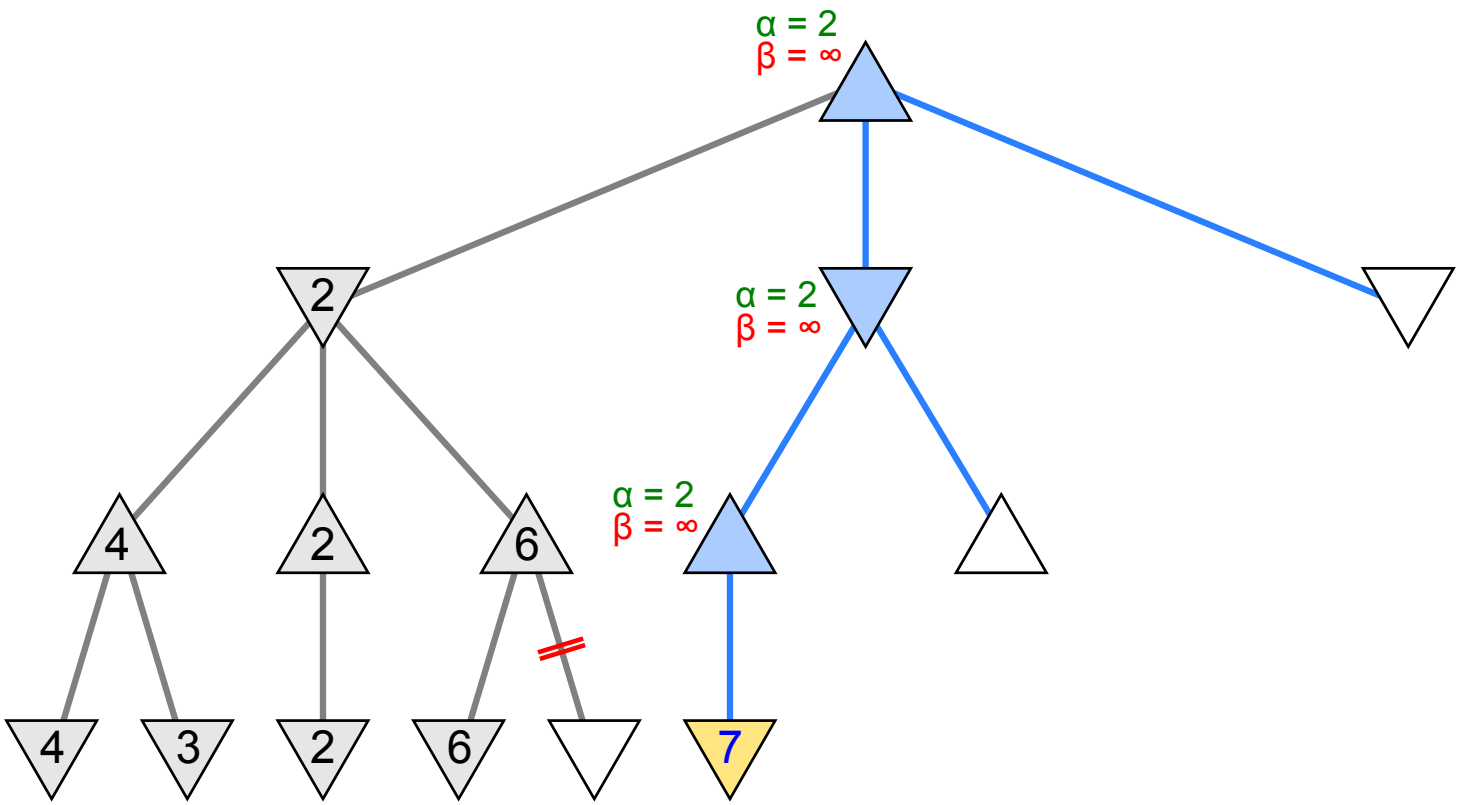
Minimax s alfa-beta prořezáváním: Příklad



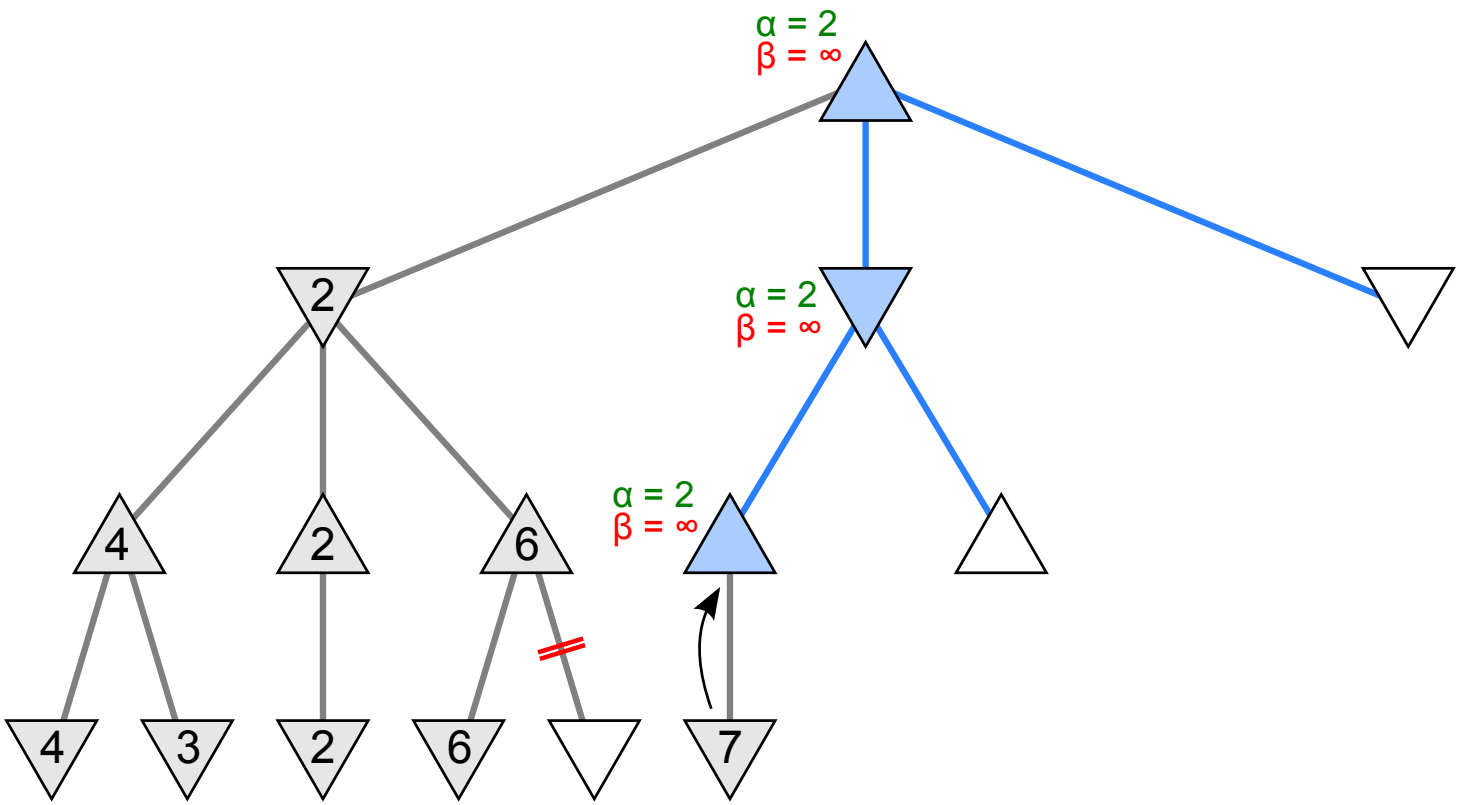
Minimax s alfa-beta prořezáváním: Příklad



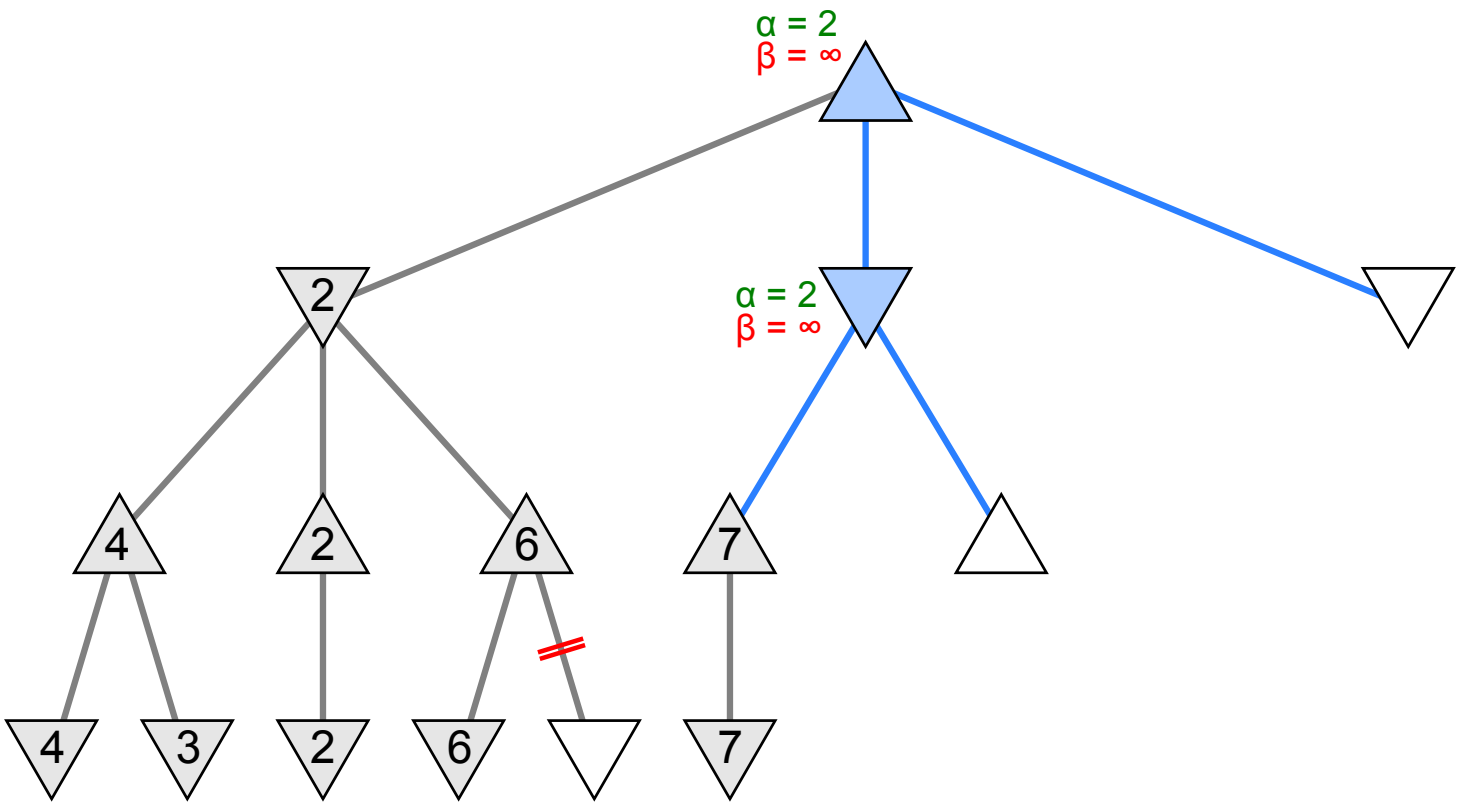
Minimax s alfa-beta prořezáváním: Příklad



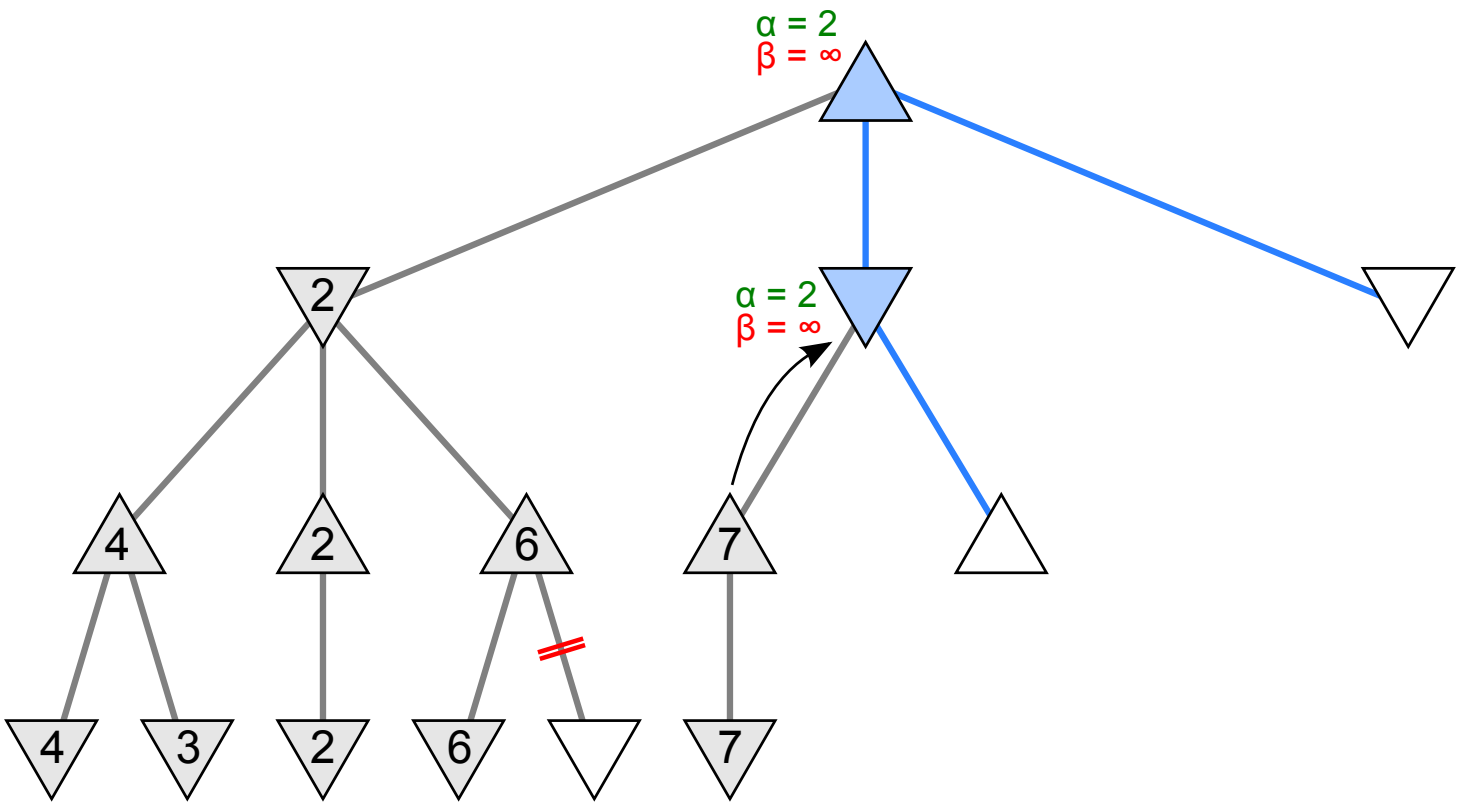
Minimax s alfa-beta prořezáváním: Příklad



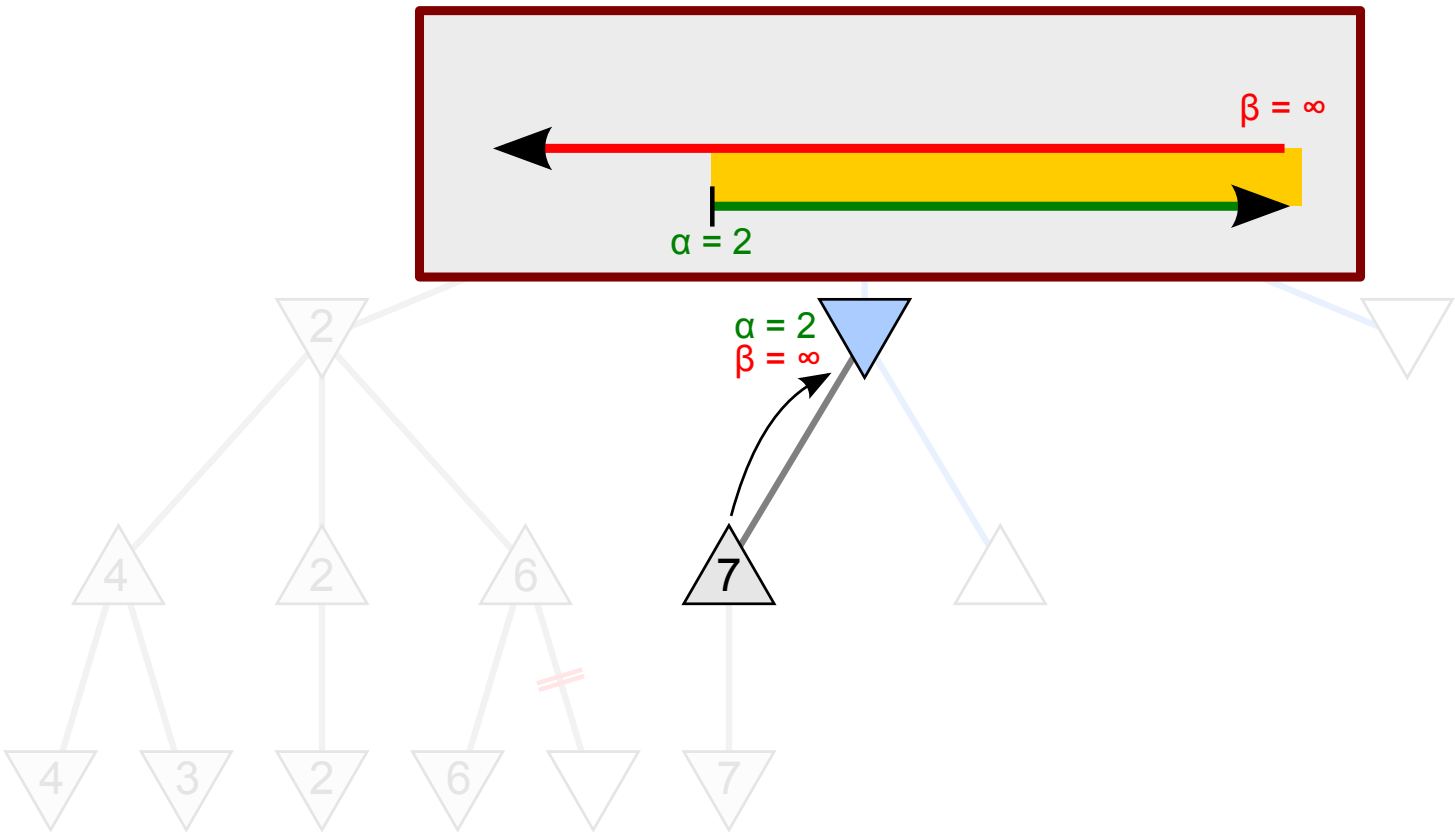
Minimax s alfa-beta prořezáváním: Příklad



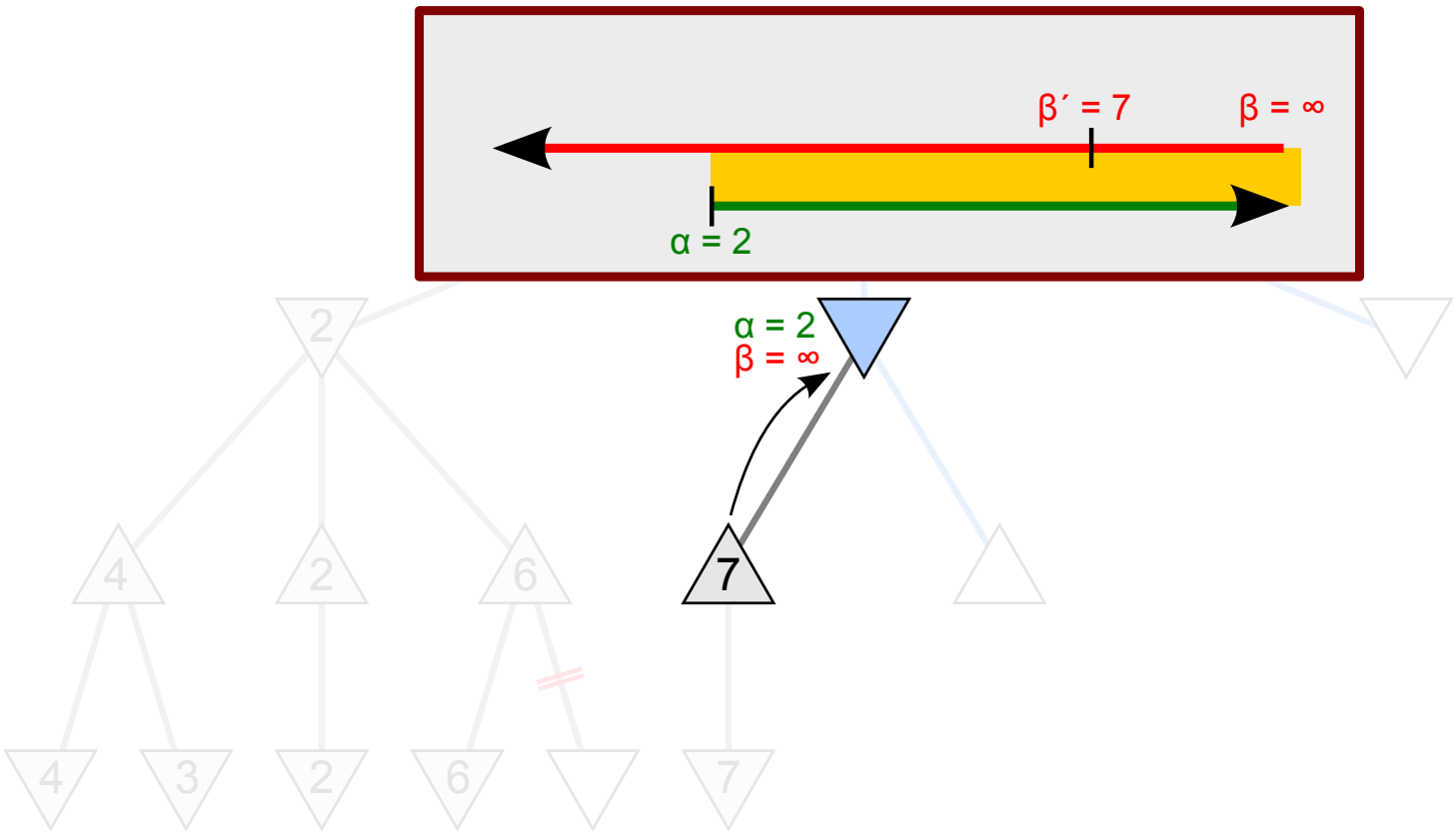
Minimax s alfa-beta prořezáváním: Příklad



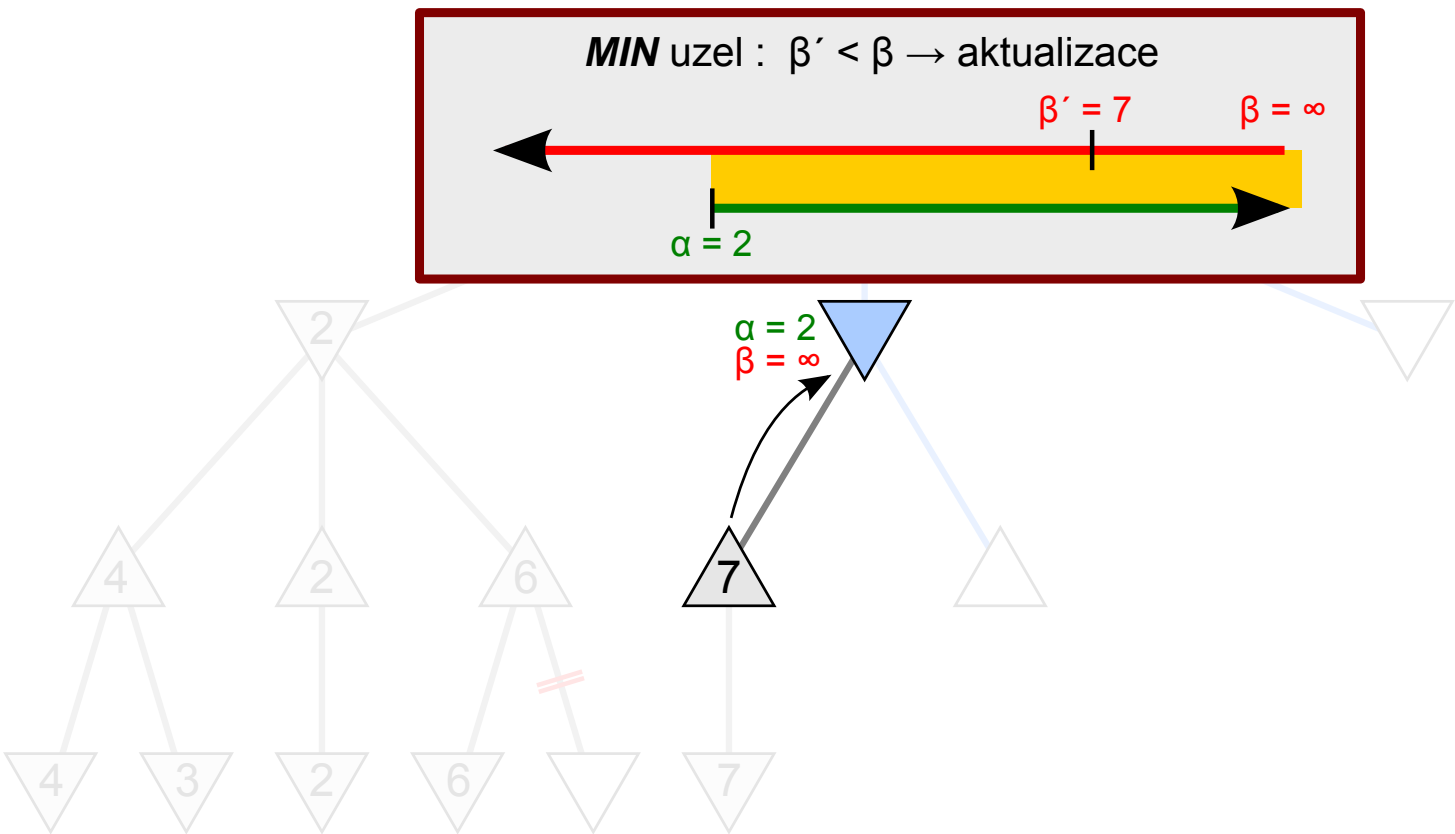
Minimax s alfa-beta prořezáváním: Příklad



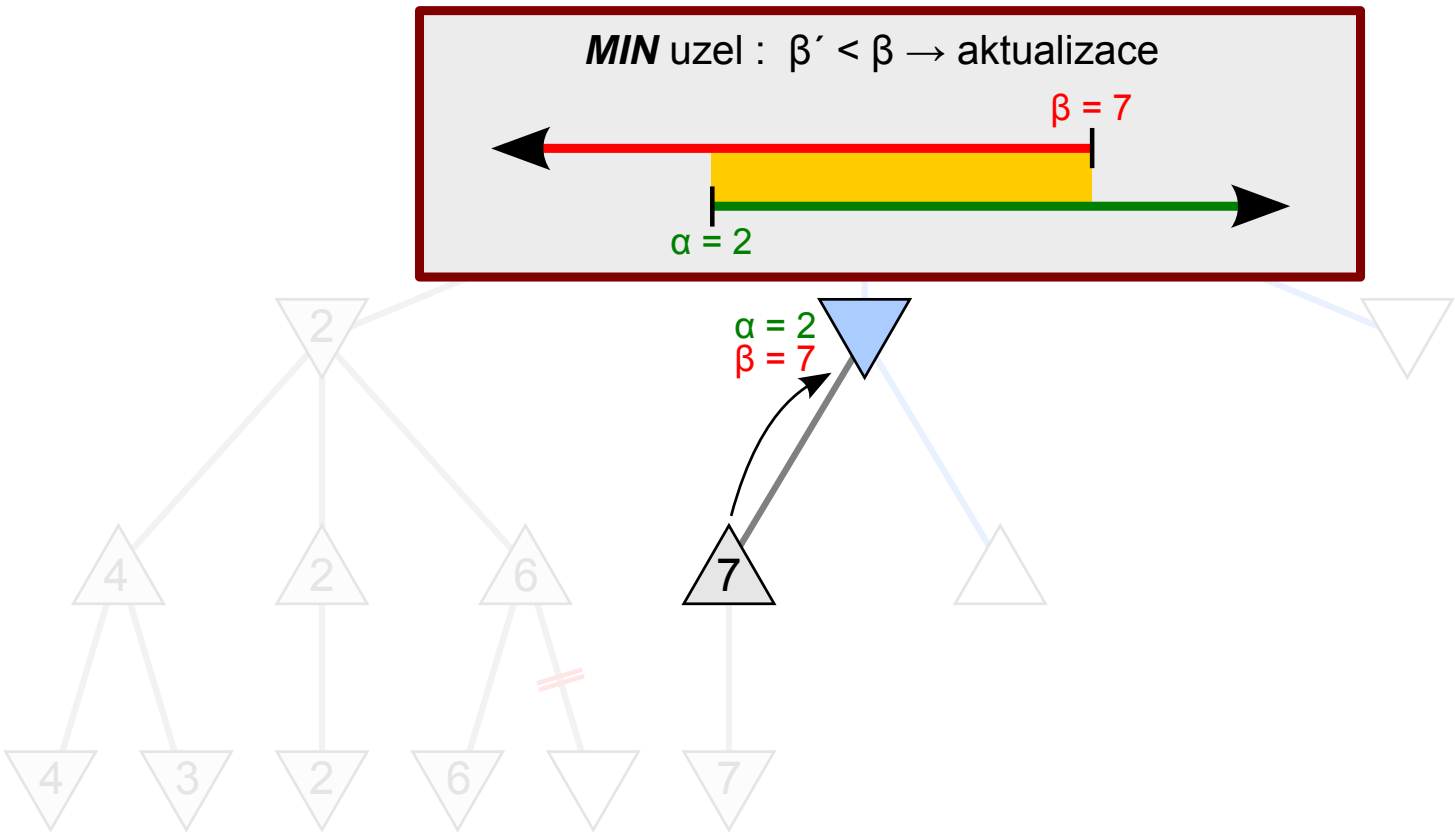
Minimax s alfa-beta prořezáváním: Příklad



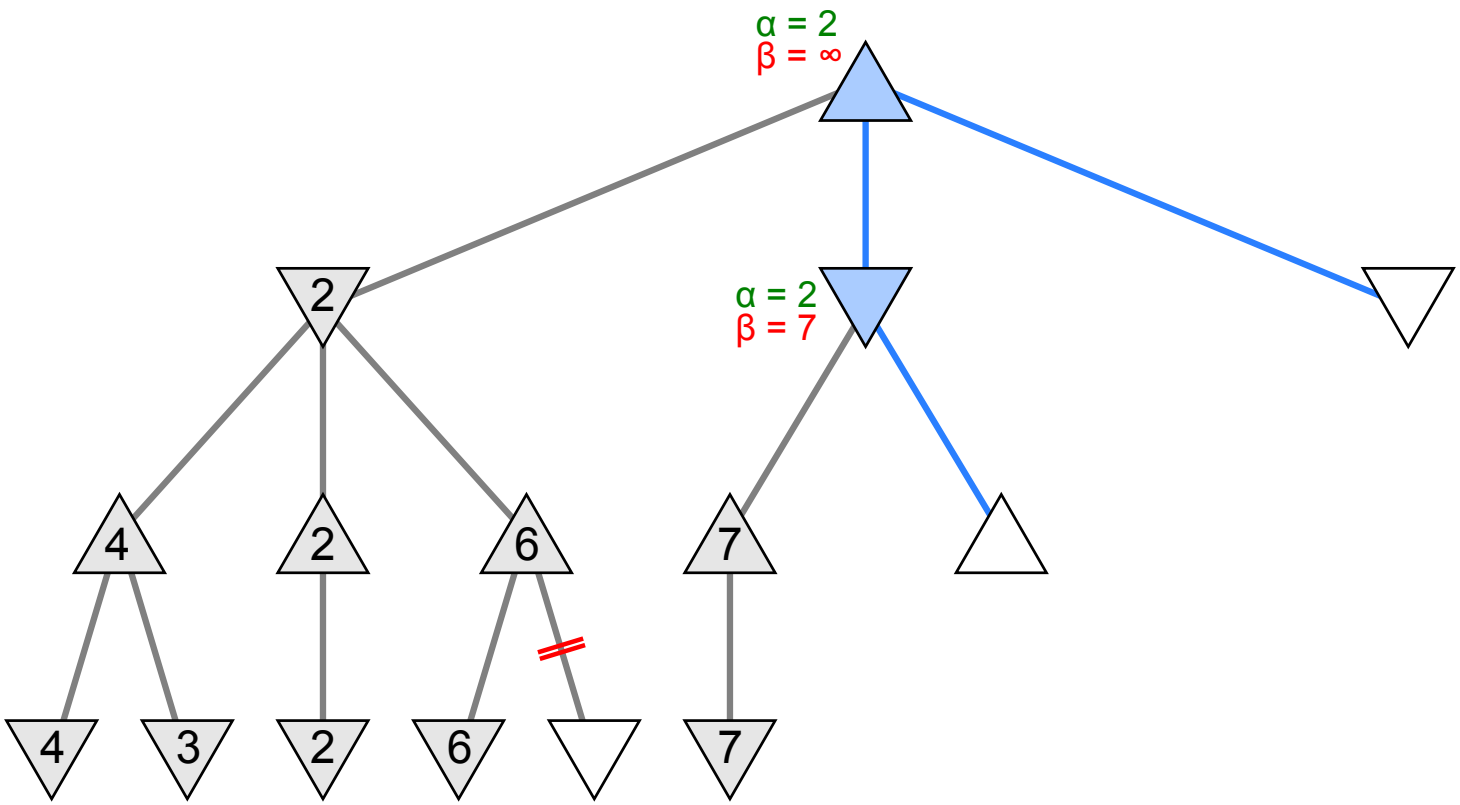
Minimax s alfa-beta prořezáváním: Příklad



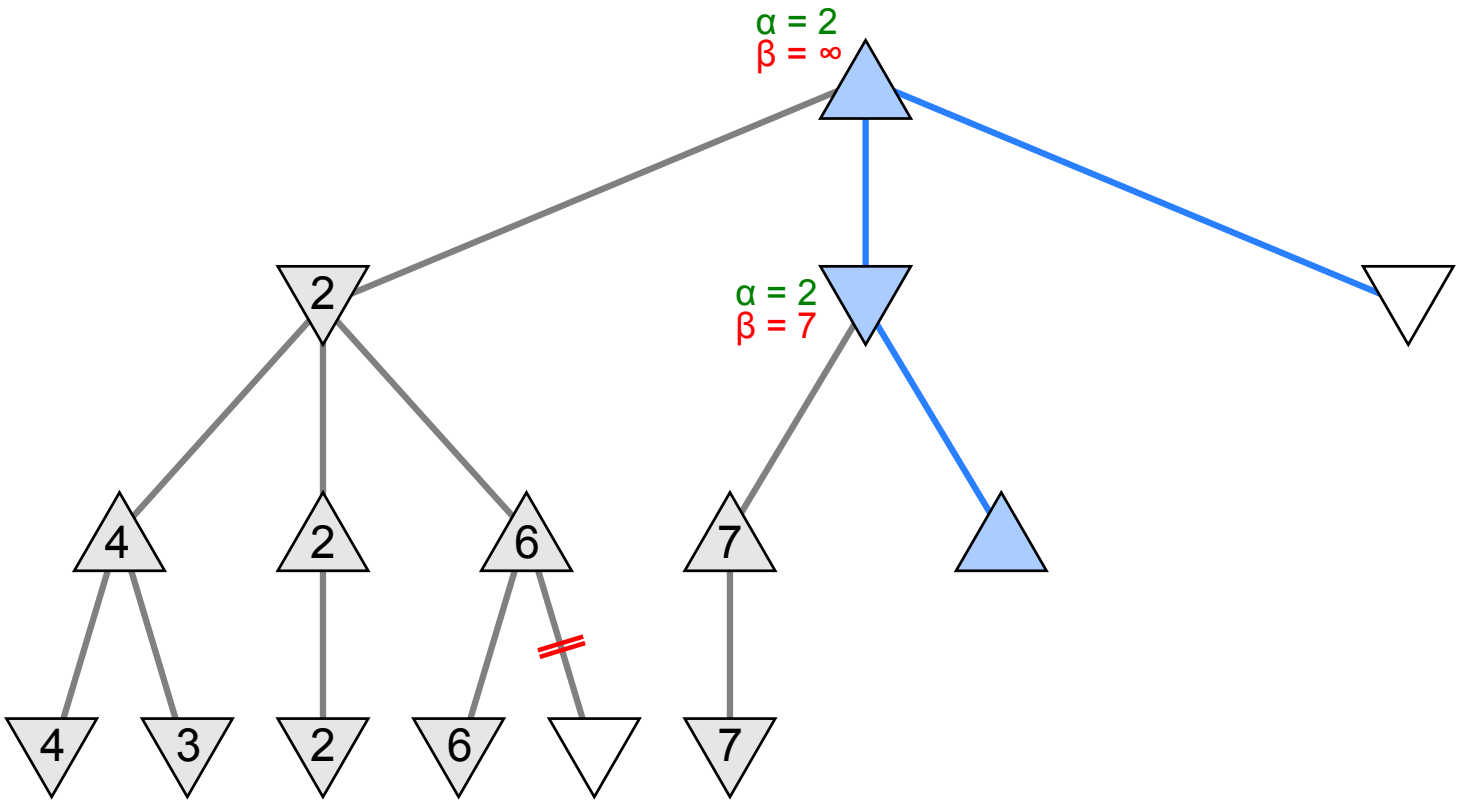
Minimax s alfa-beta prořezáváním: Příklad



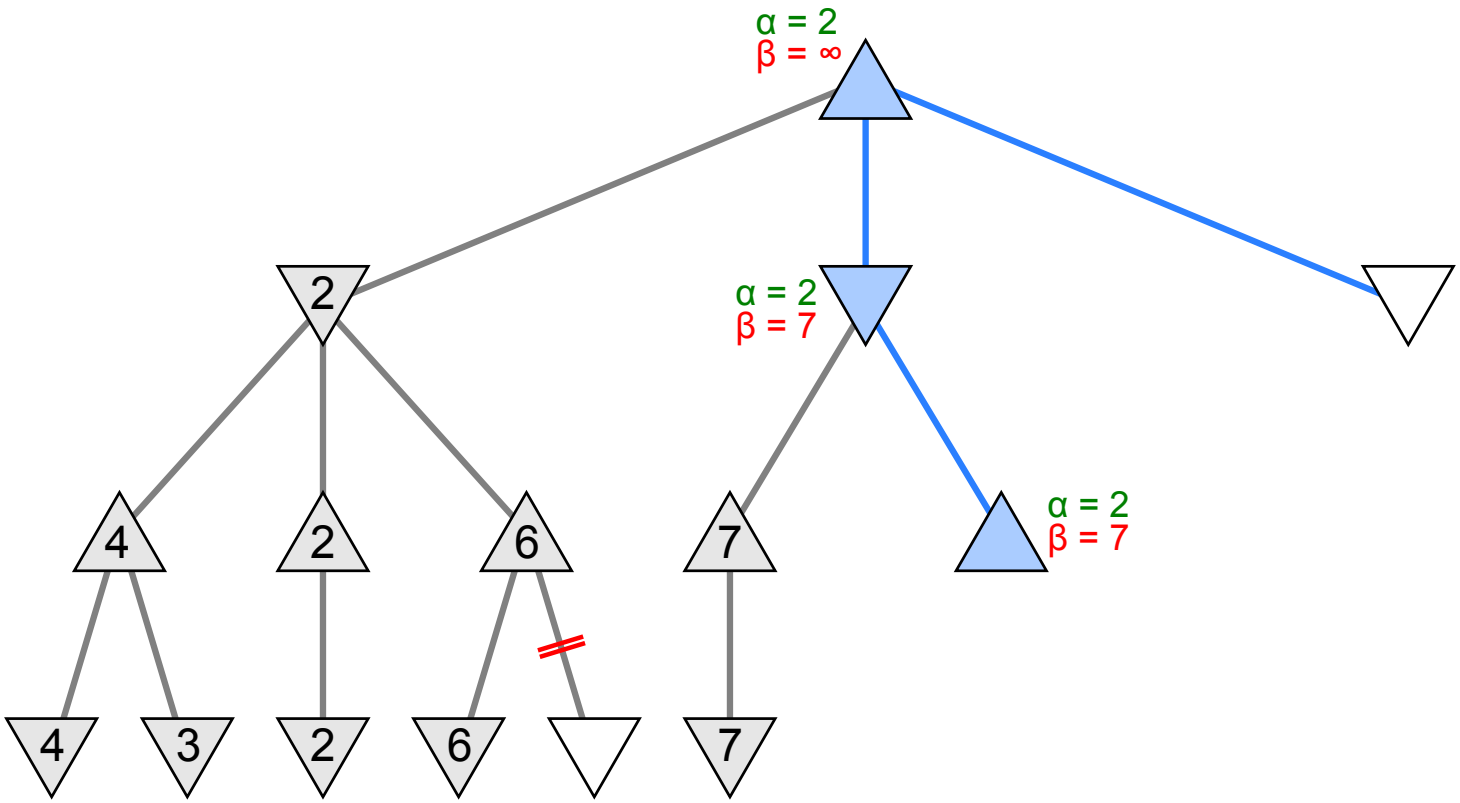
Minimax s alfa-beta prořezáváním: Příklad



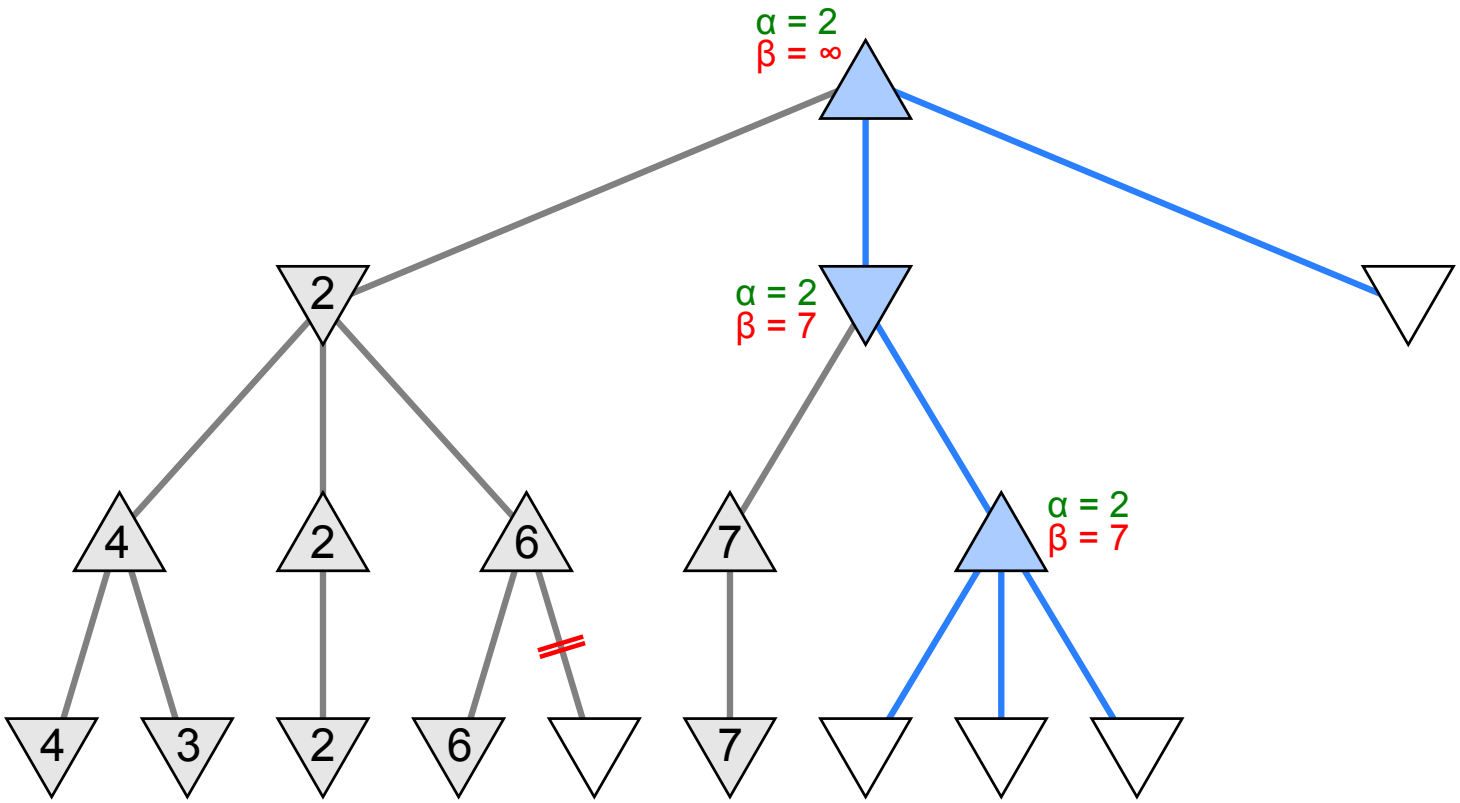
Minimax s alfa-beta prořezáváním: Příklad



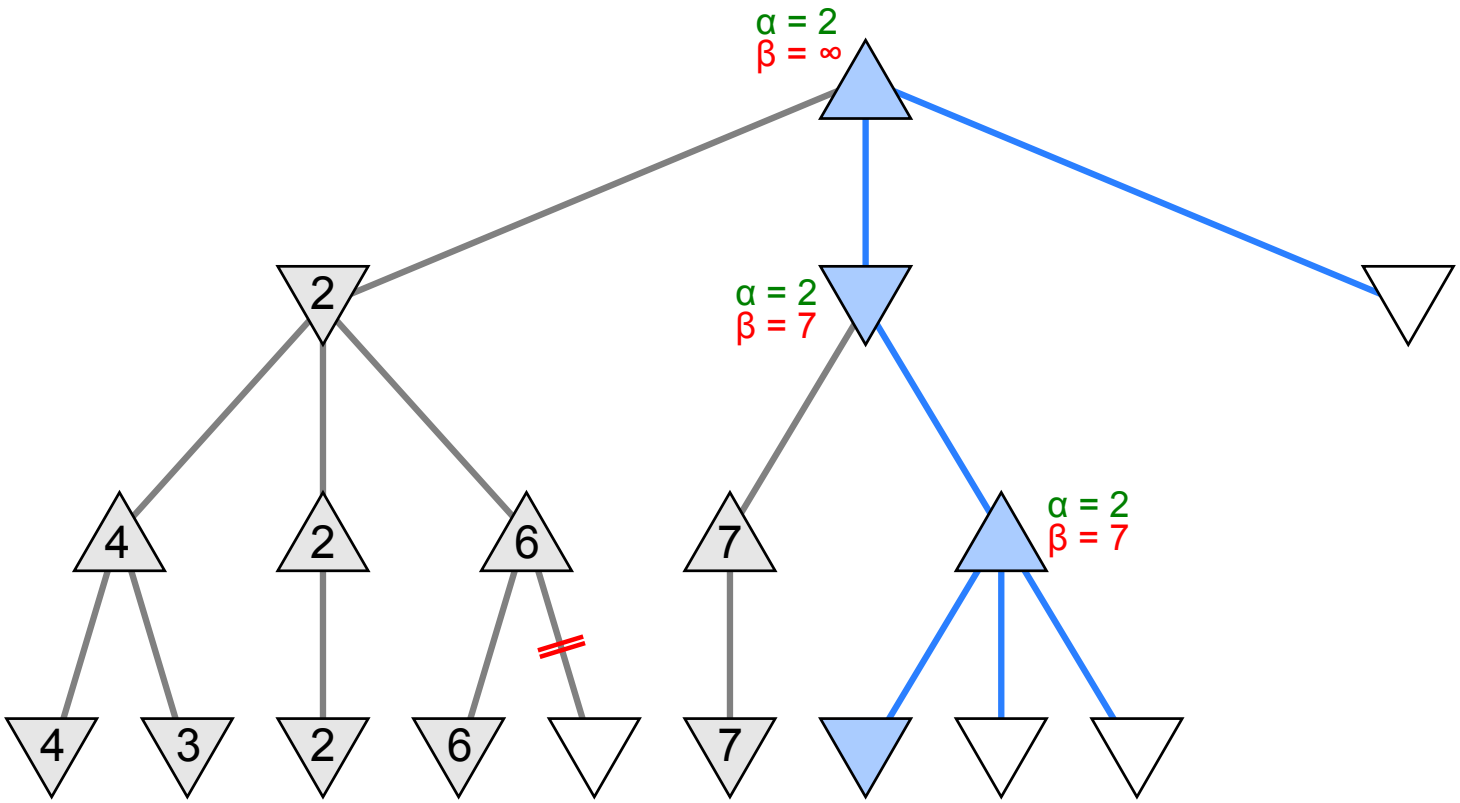
Minimax s alfa-beta prořezáváním: Příklad



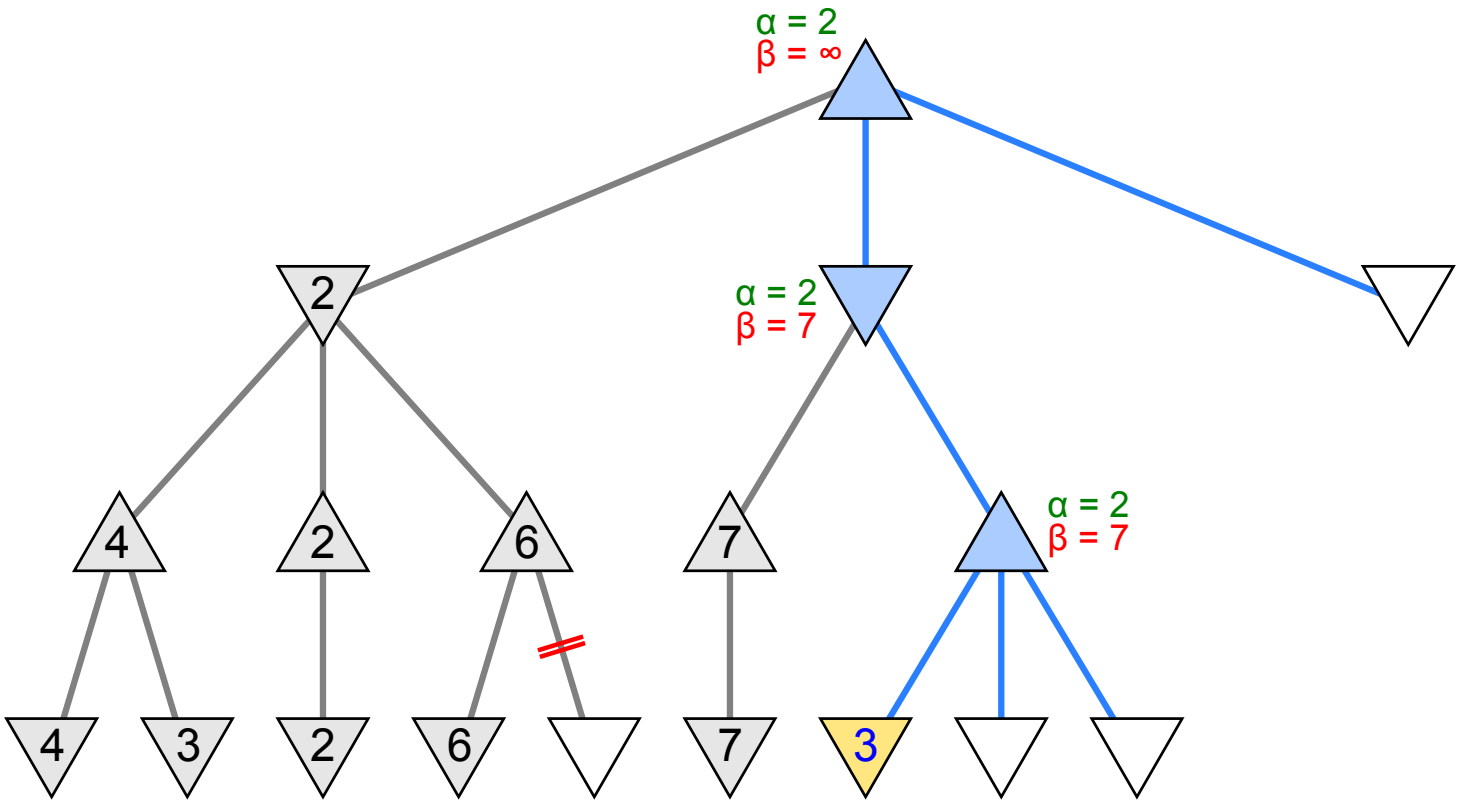
Minimax s alfa-beta prořezáváním: Příklad



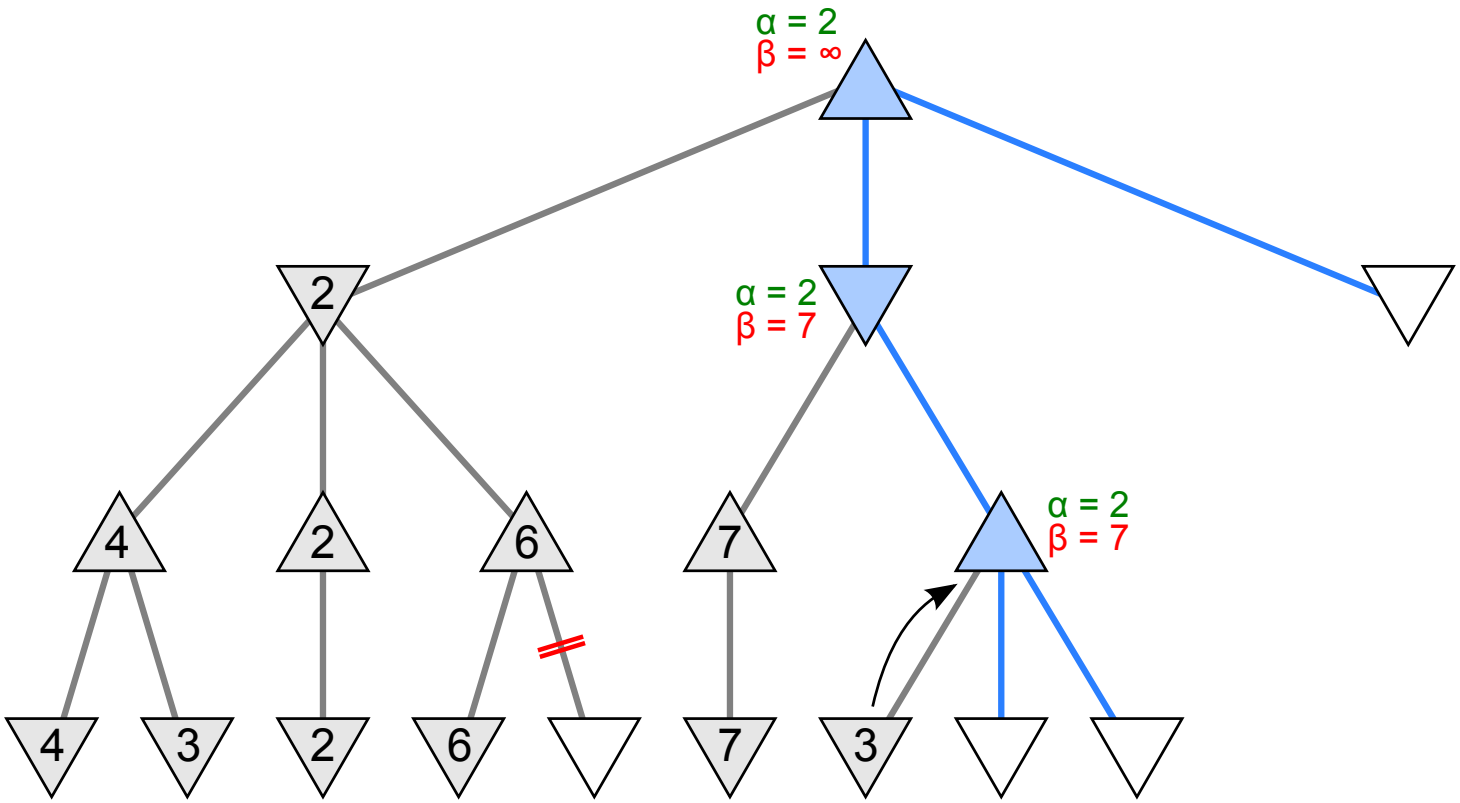
Minimax s alfa-beta prořezáváním: Příklad



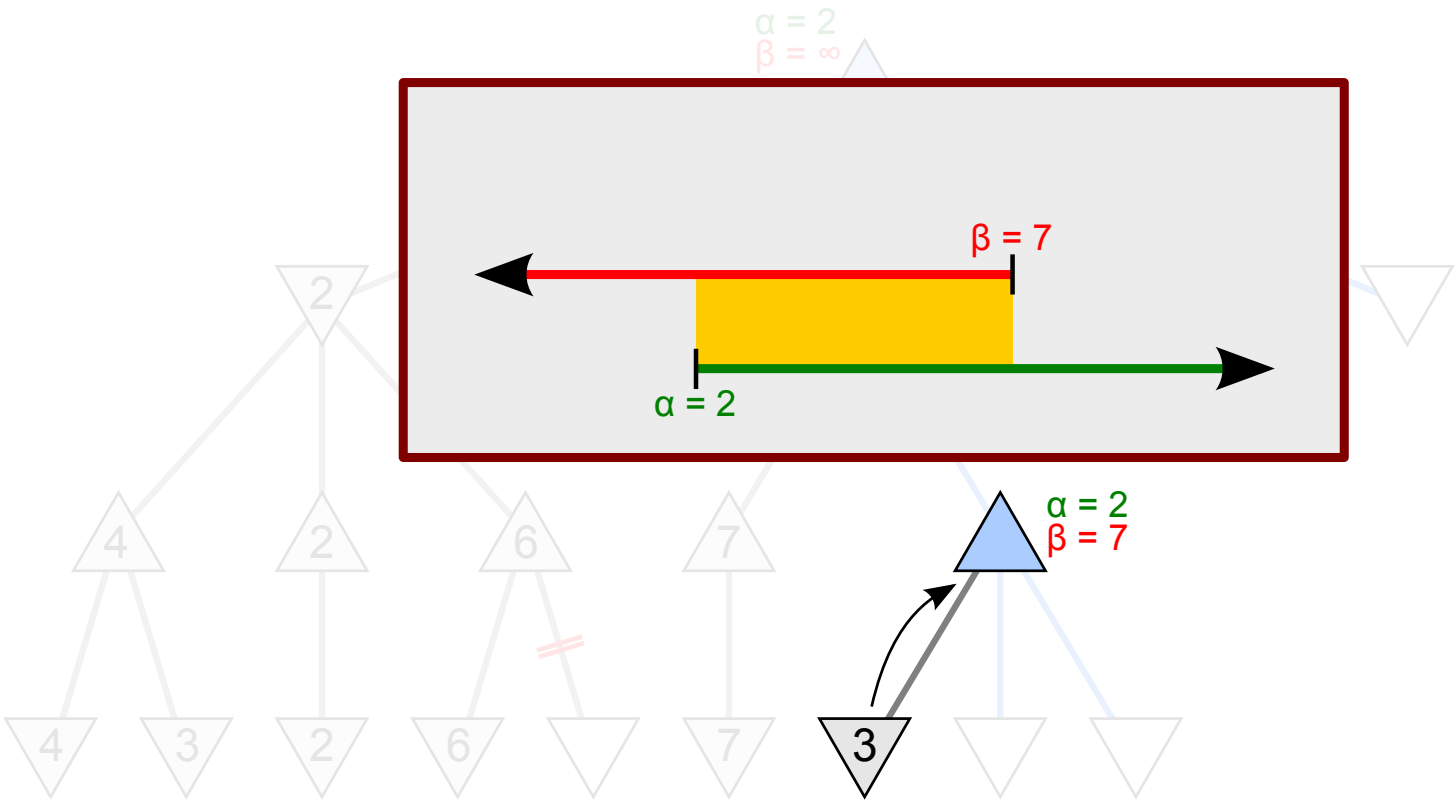
Minimax s alfa-beta prořezáváním: Příklad



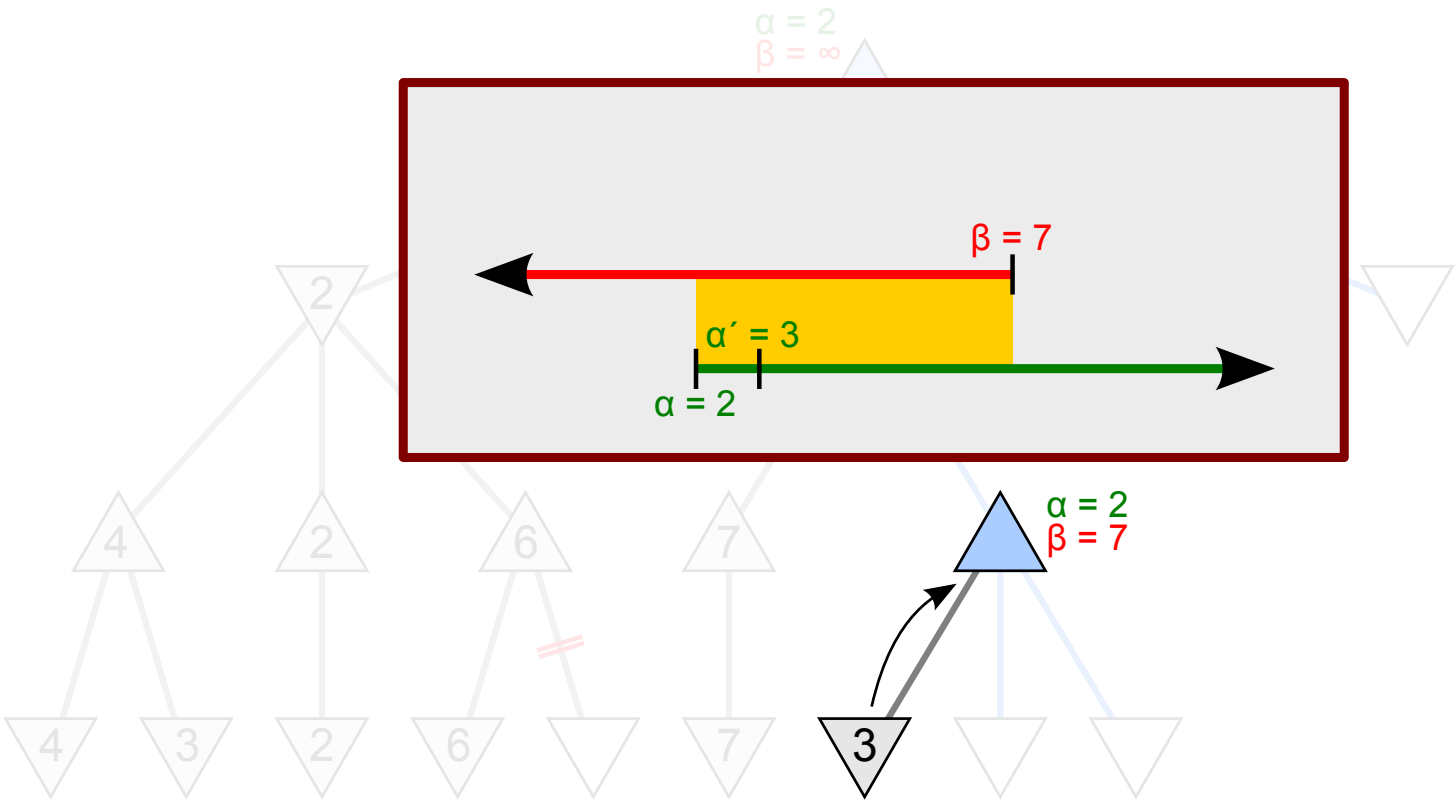
Minimax s alfa-beta prořezáváním: Příklad



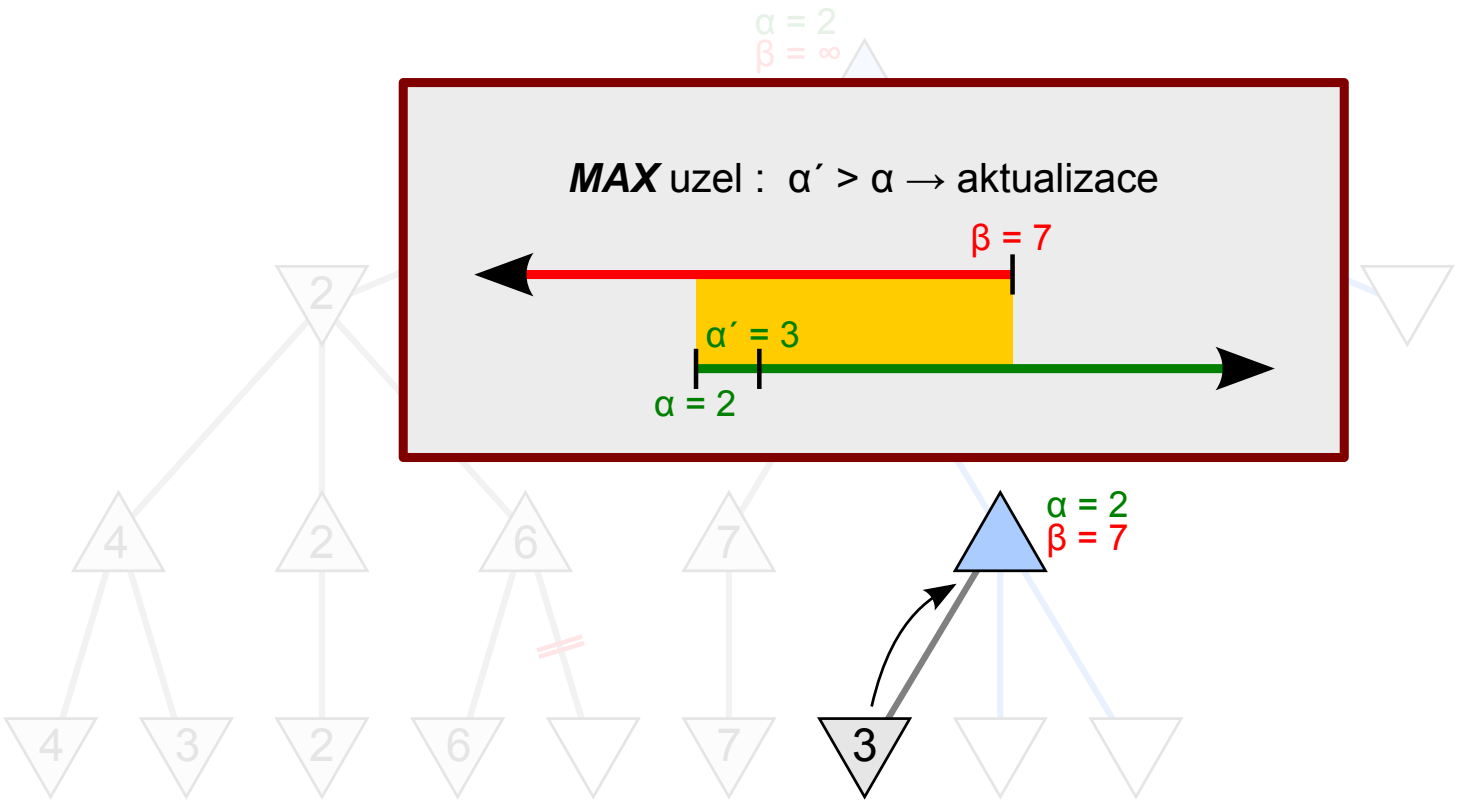
Minimax s alfa-beta prořezáváním: Příklad



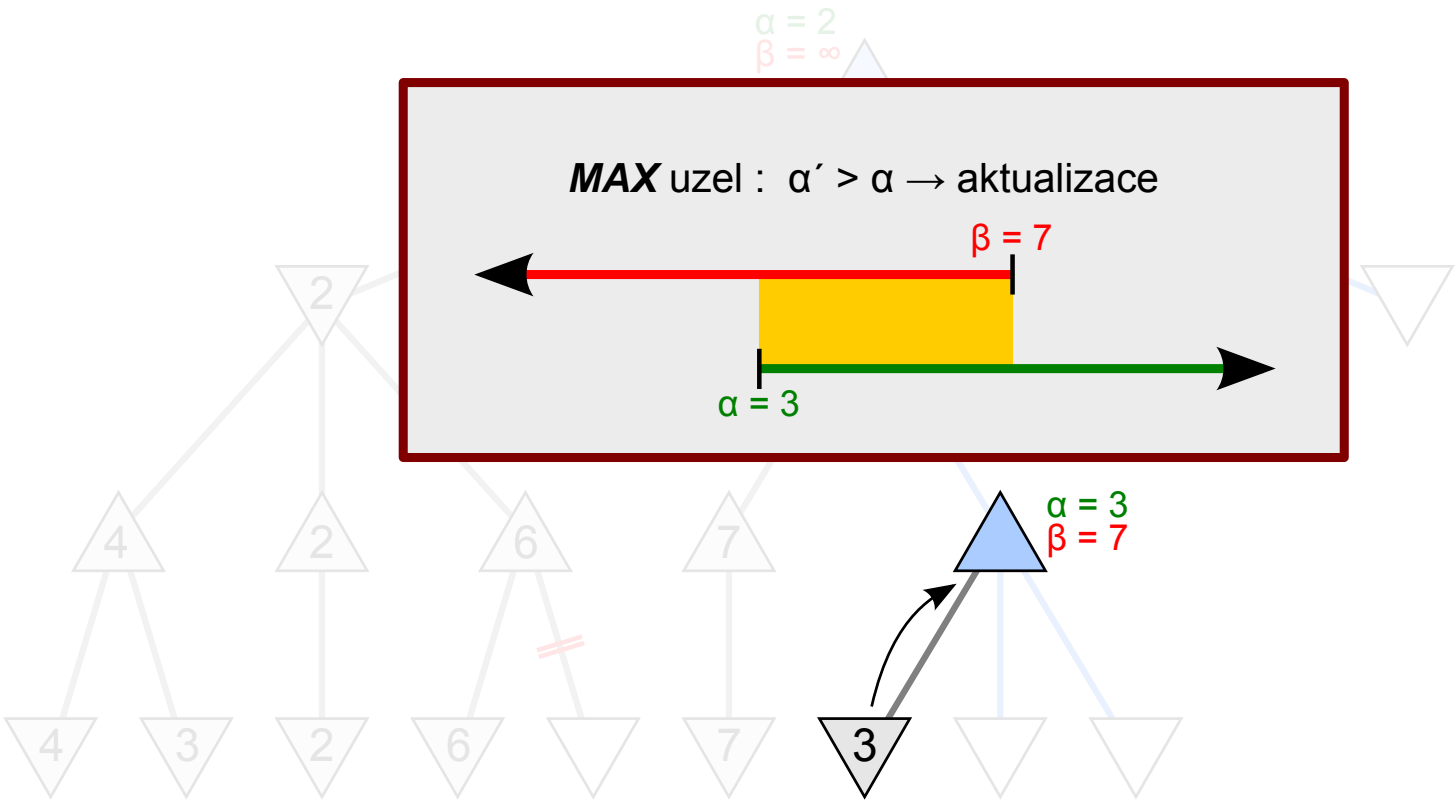
Minimax s alfa-beta prořezáváním: Příklad



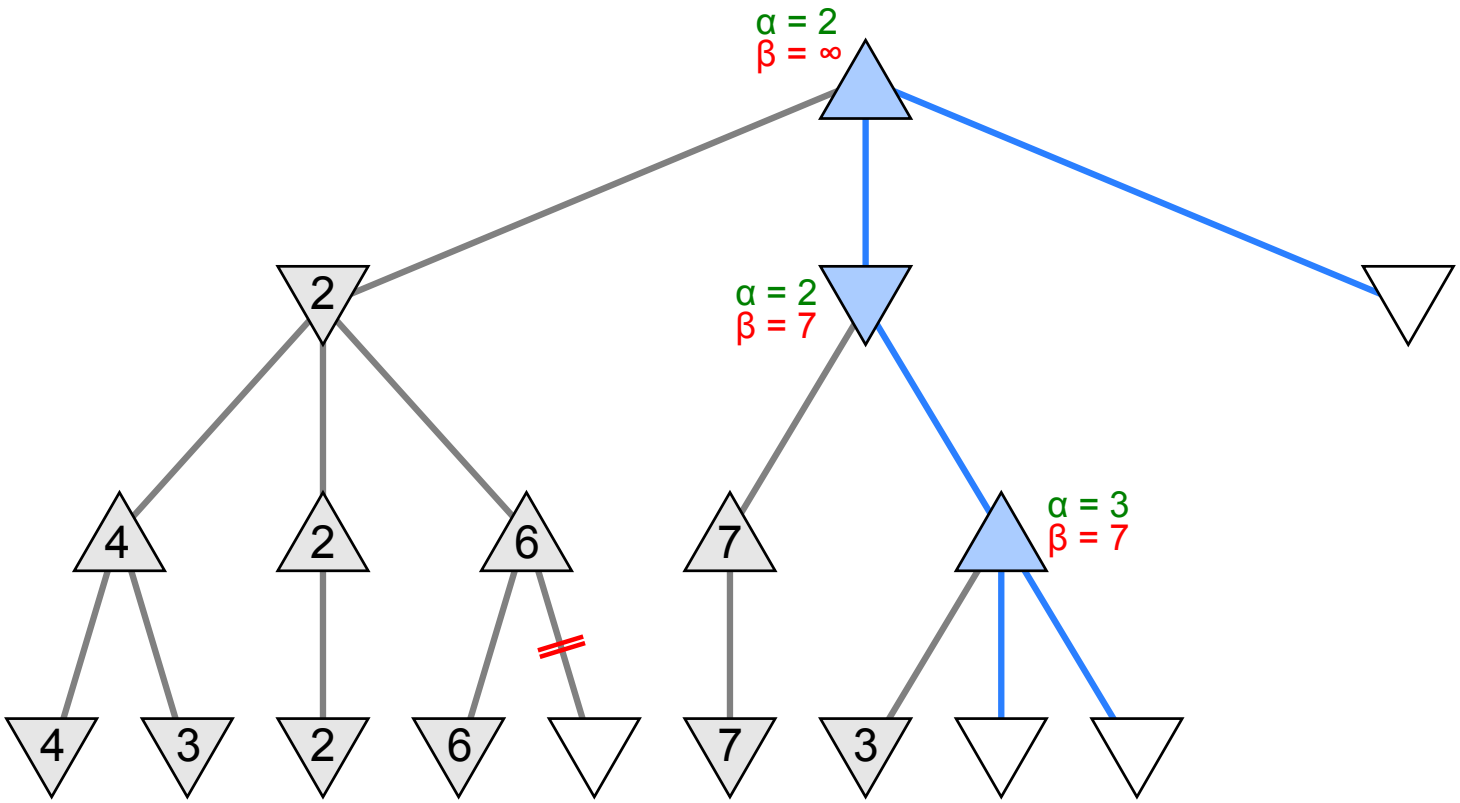
Minimax s alfa-beta prořezáváním: Příklad



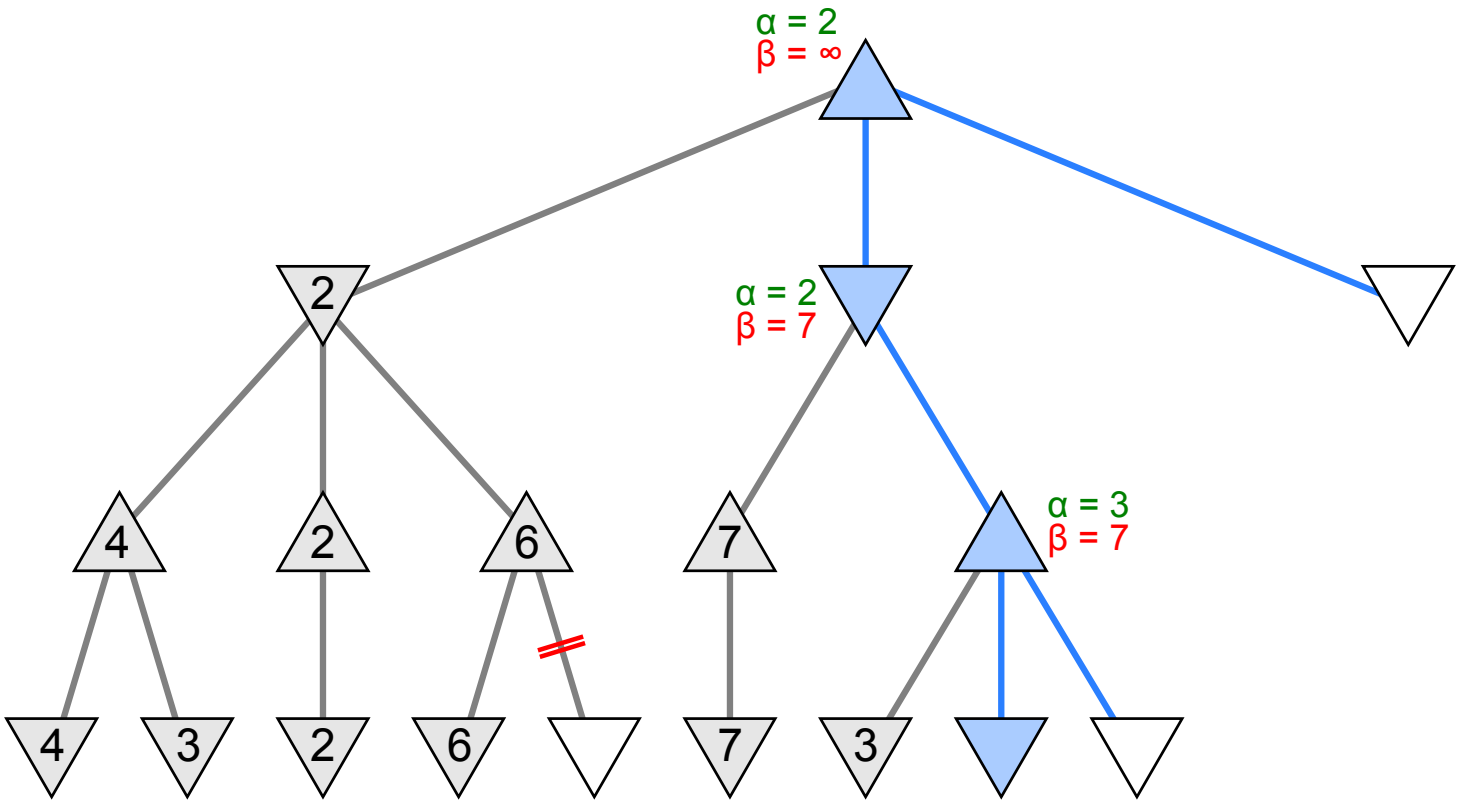
Minimax s alfa-beta prořezáváním: Příklad



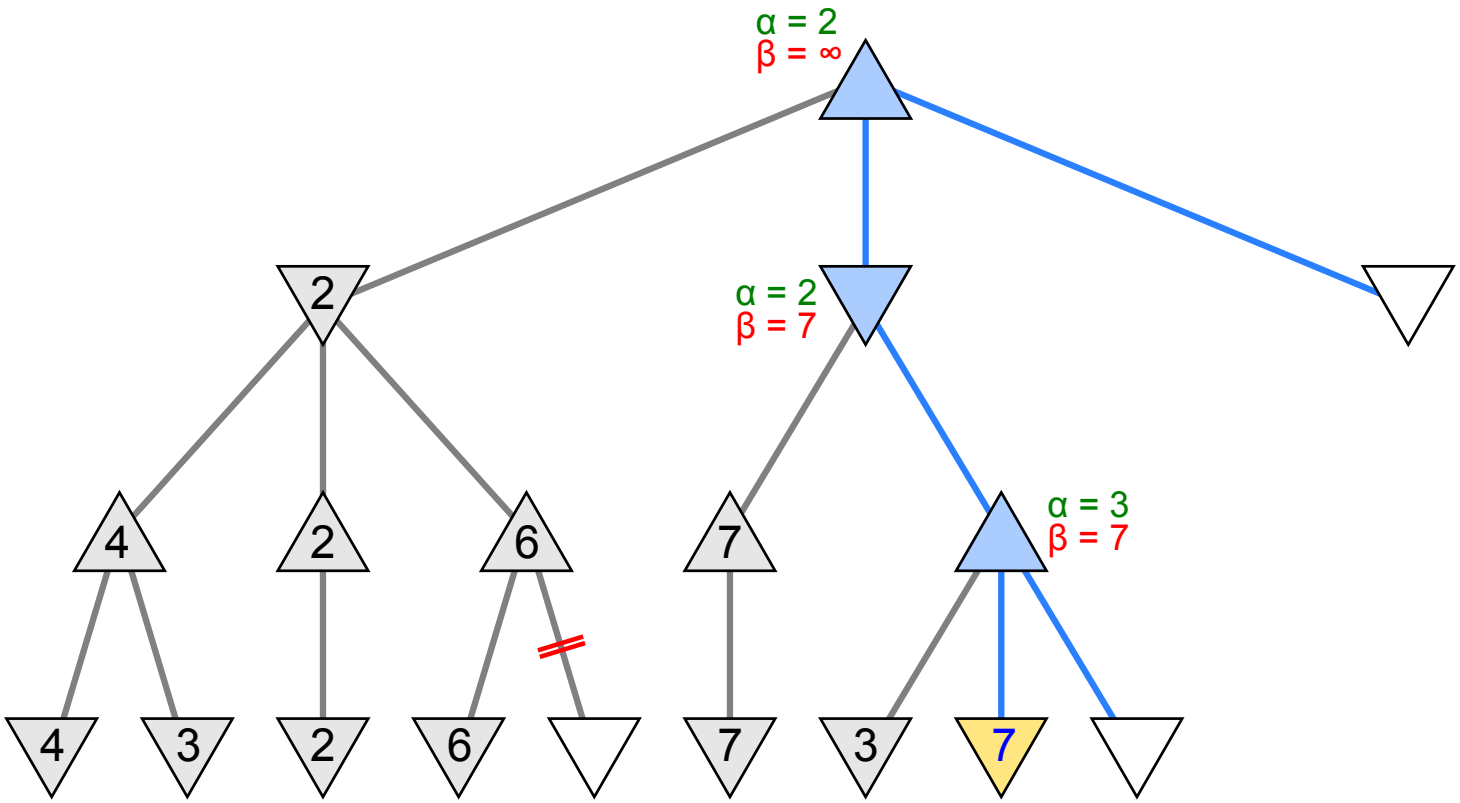
Minimax s alfa-beta prořezáváním: Příklad



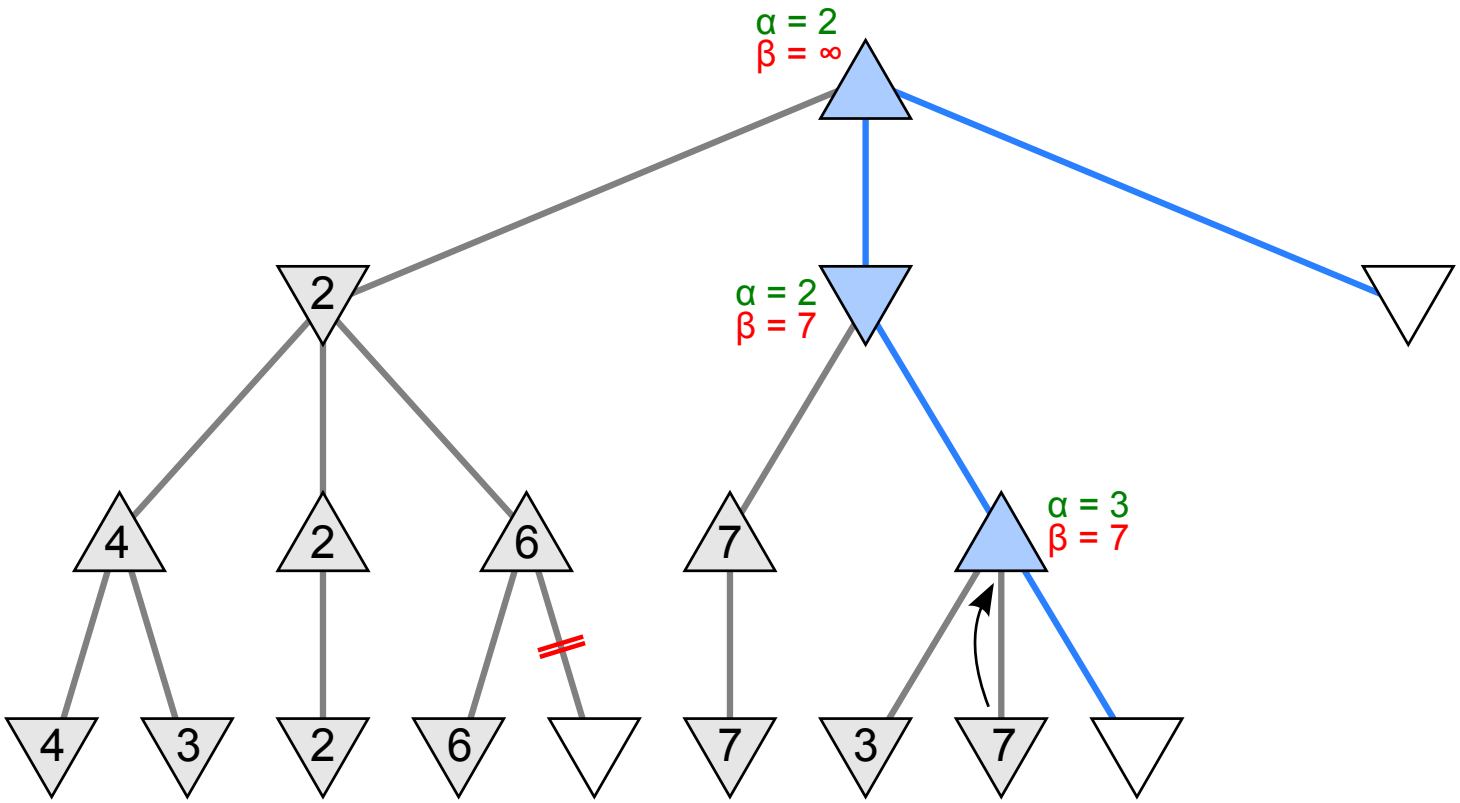
Minimax s alfa-beta prořezáváním: Příklad



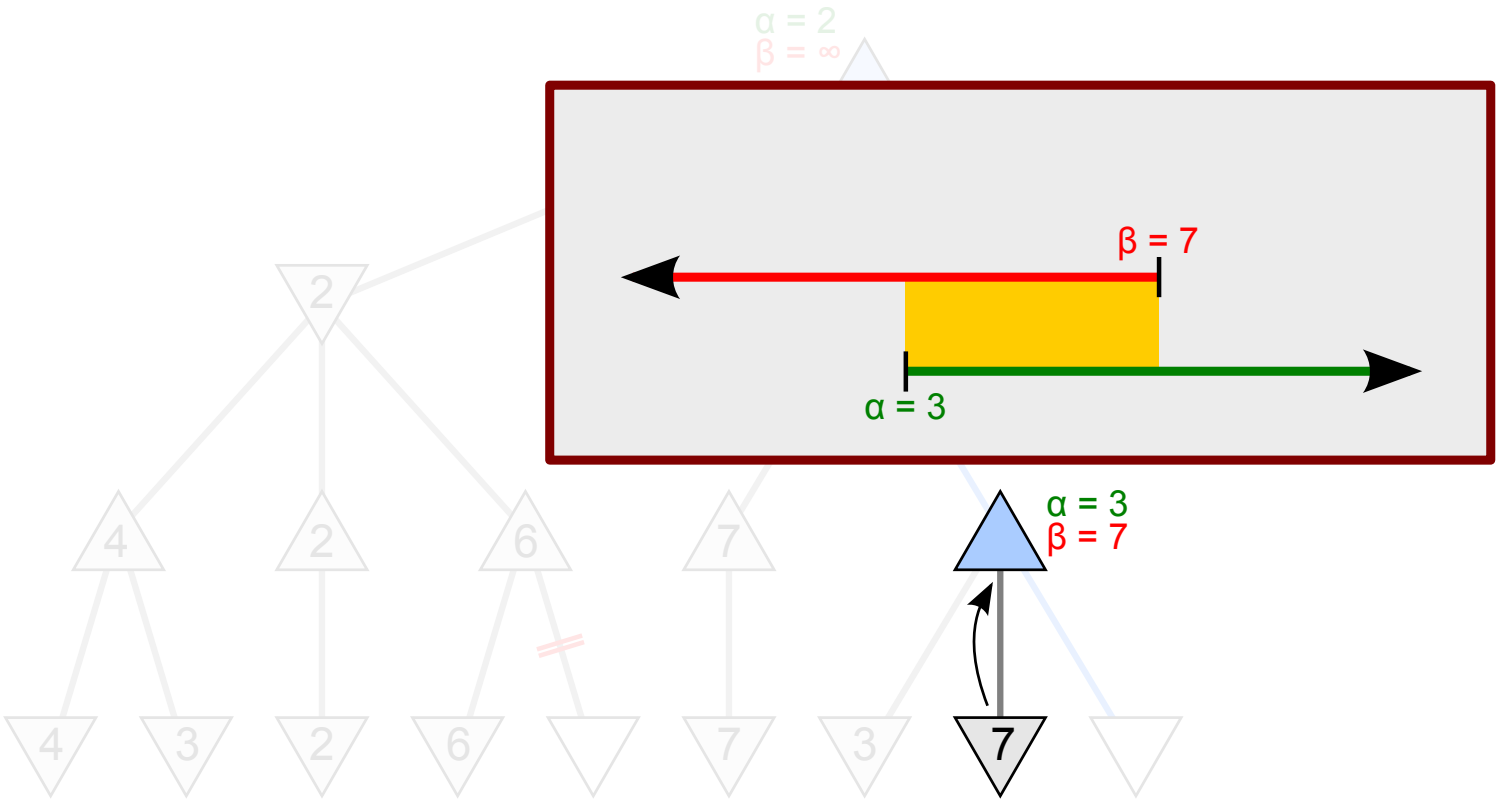
Minimax s alfa-beta prořezáváním: Příklad



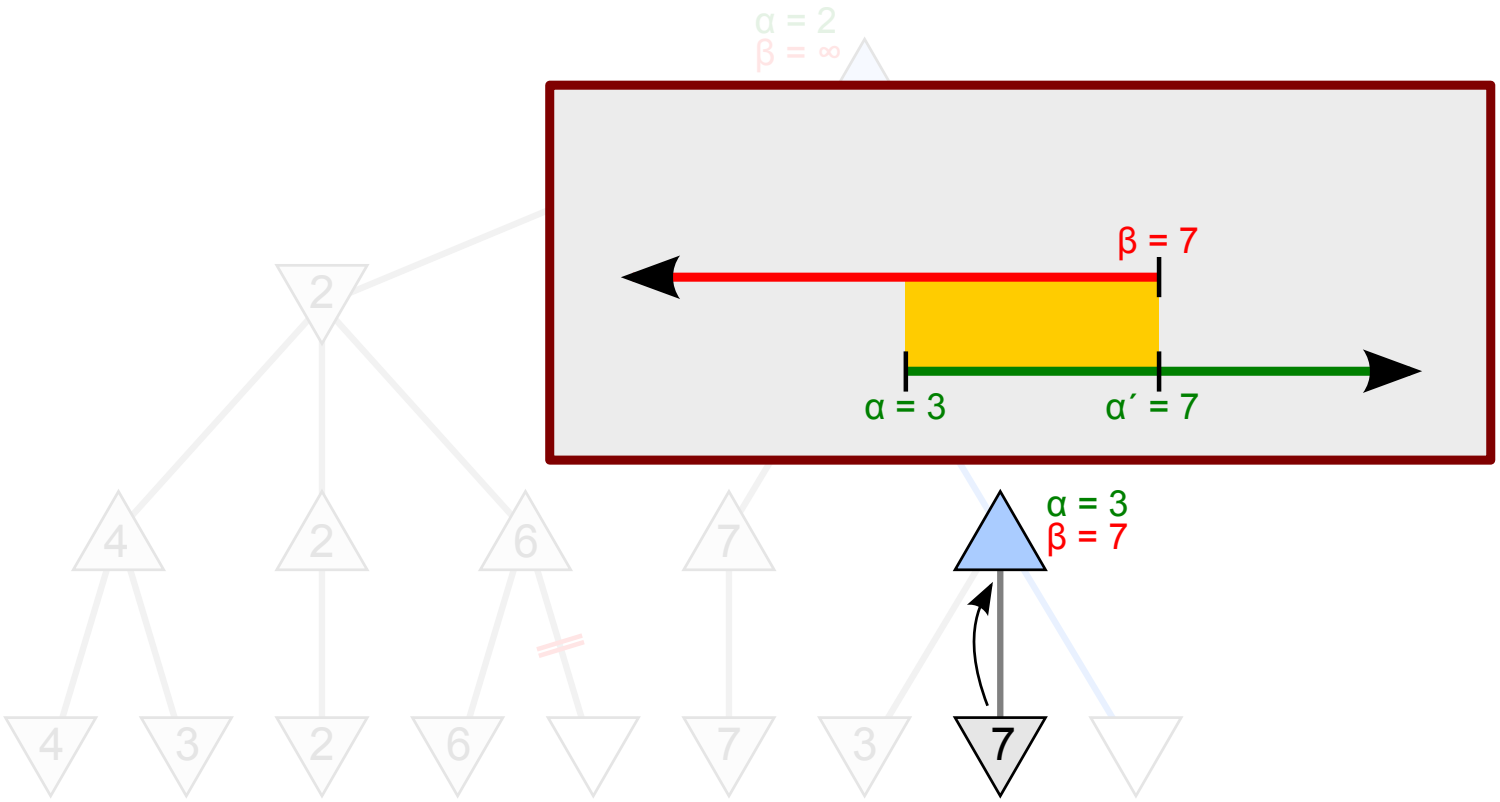
Minimax s alfa-beta prořezáváním: Příklad



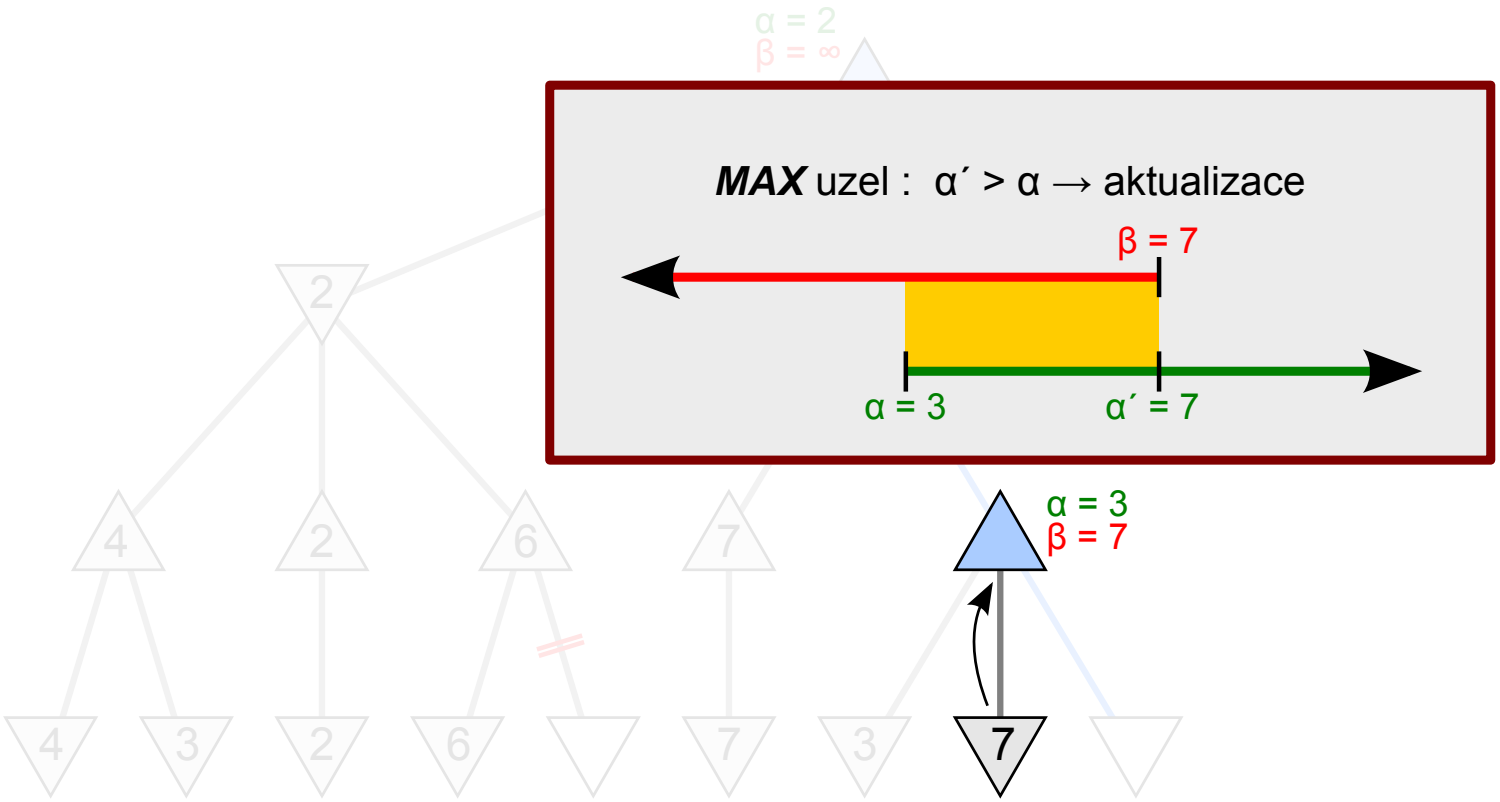
Minimax s alfa-beta prořezáváním: Příklad



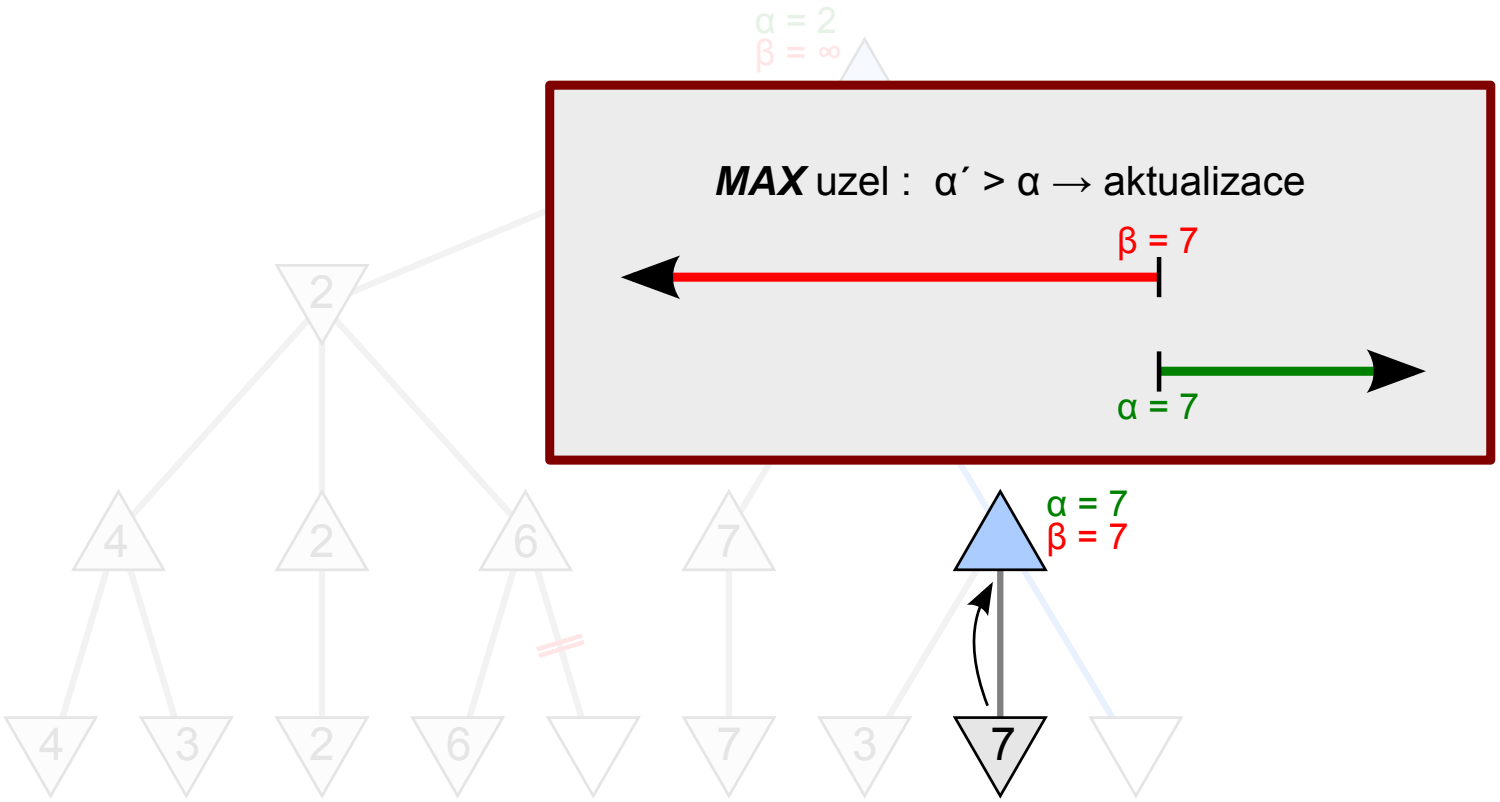
Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad



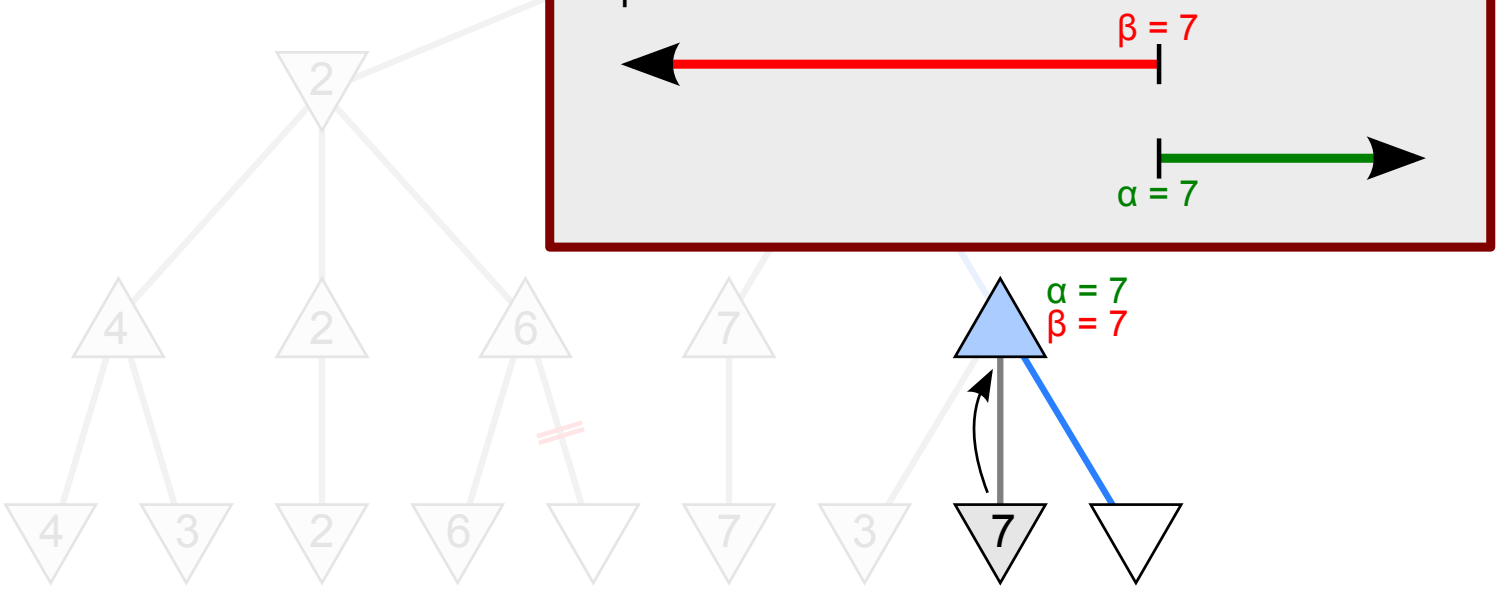
Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad

Došlo k překřížení hodnot: $(-\infty, \beta) \cap (\alpha, \infty) = \{\}$
(speciální případ pro $\alpha = \beta$)

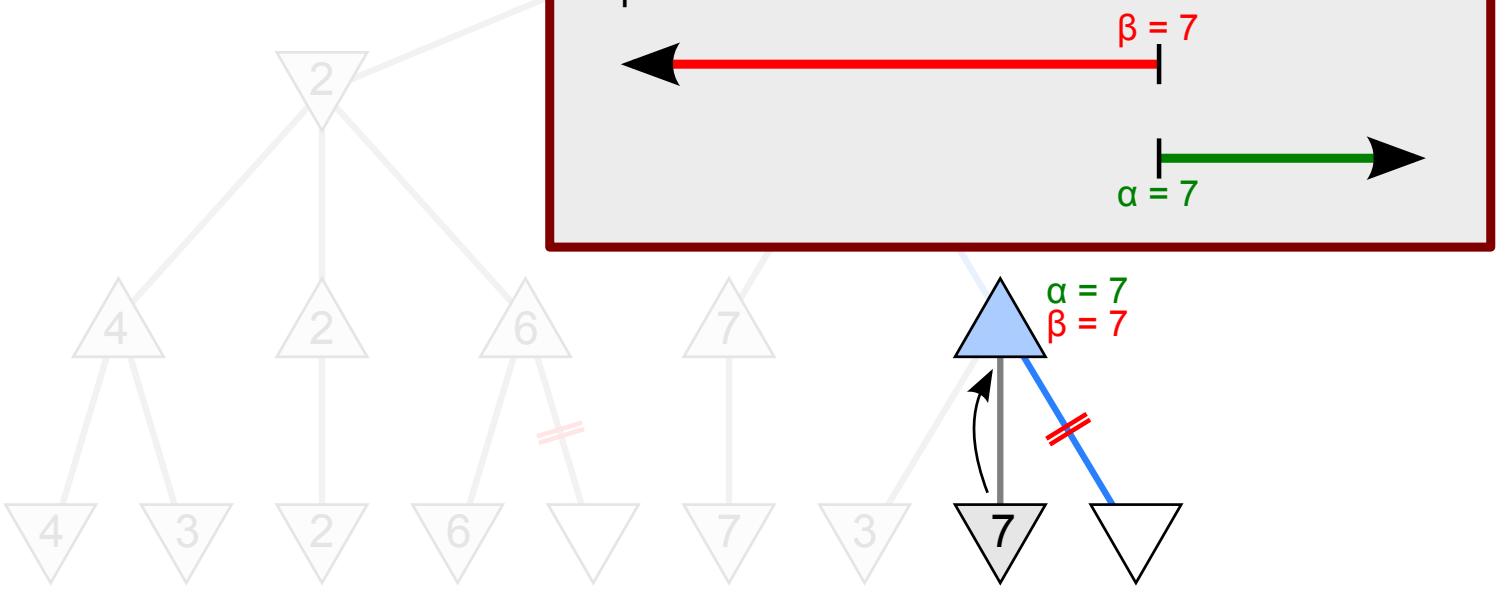
Provedeme tedy **prořez** všech zbývajících potomků aktuálního uzlu.



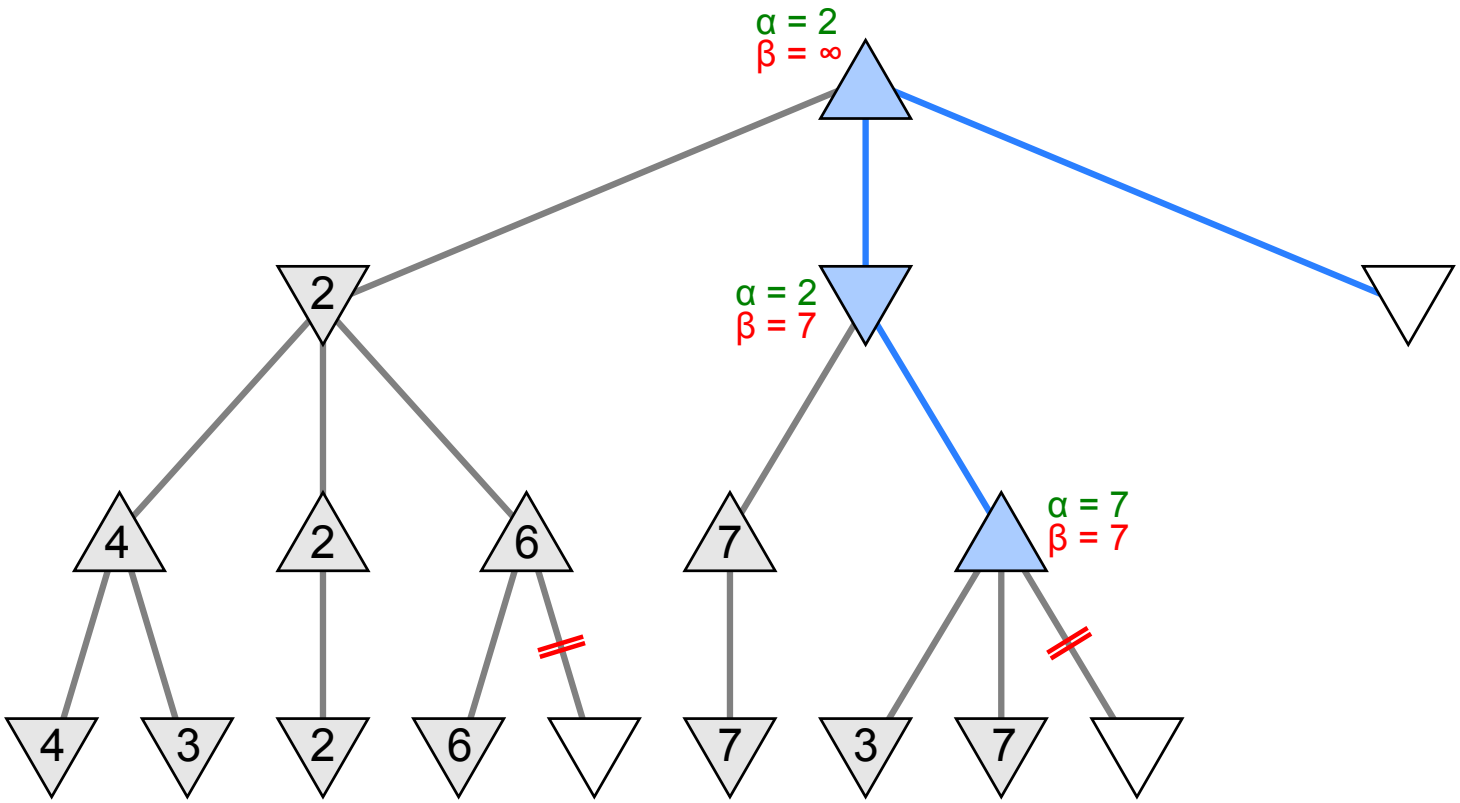
Minimax s alfa-beta prořezáváním: Příklad

Došlo k překřížení hodnot: $(-\infty, \beta) \cap (\alpha, \infty) = \{\}$
(speciální případ pro $\alpha = \beta$)

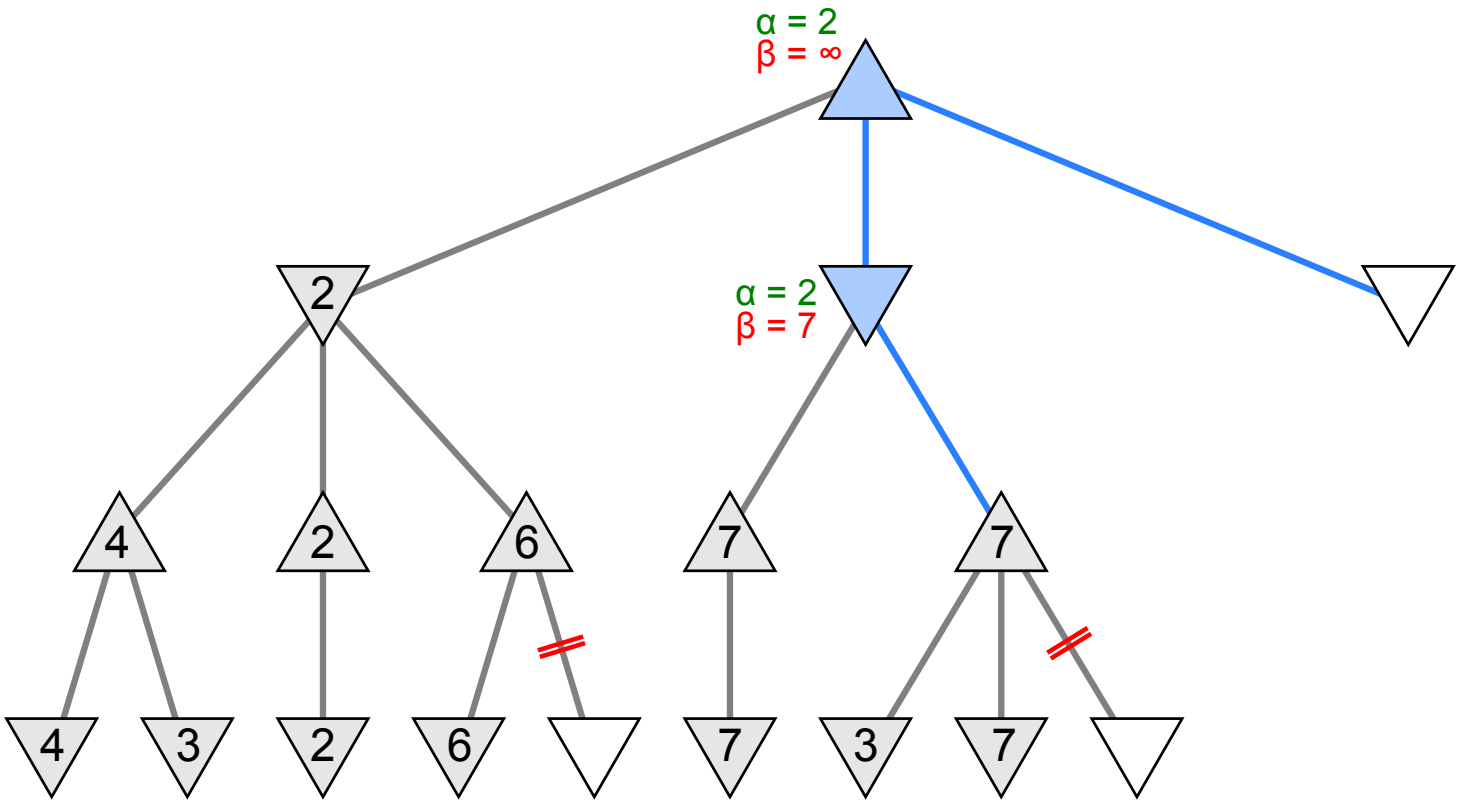
Provedeme tedy **prořez** všech zbývajících potomků aktuálního uzlu.



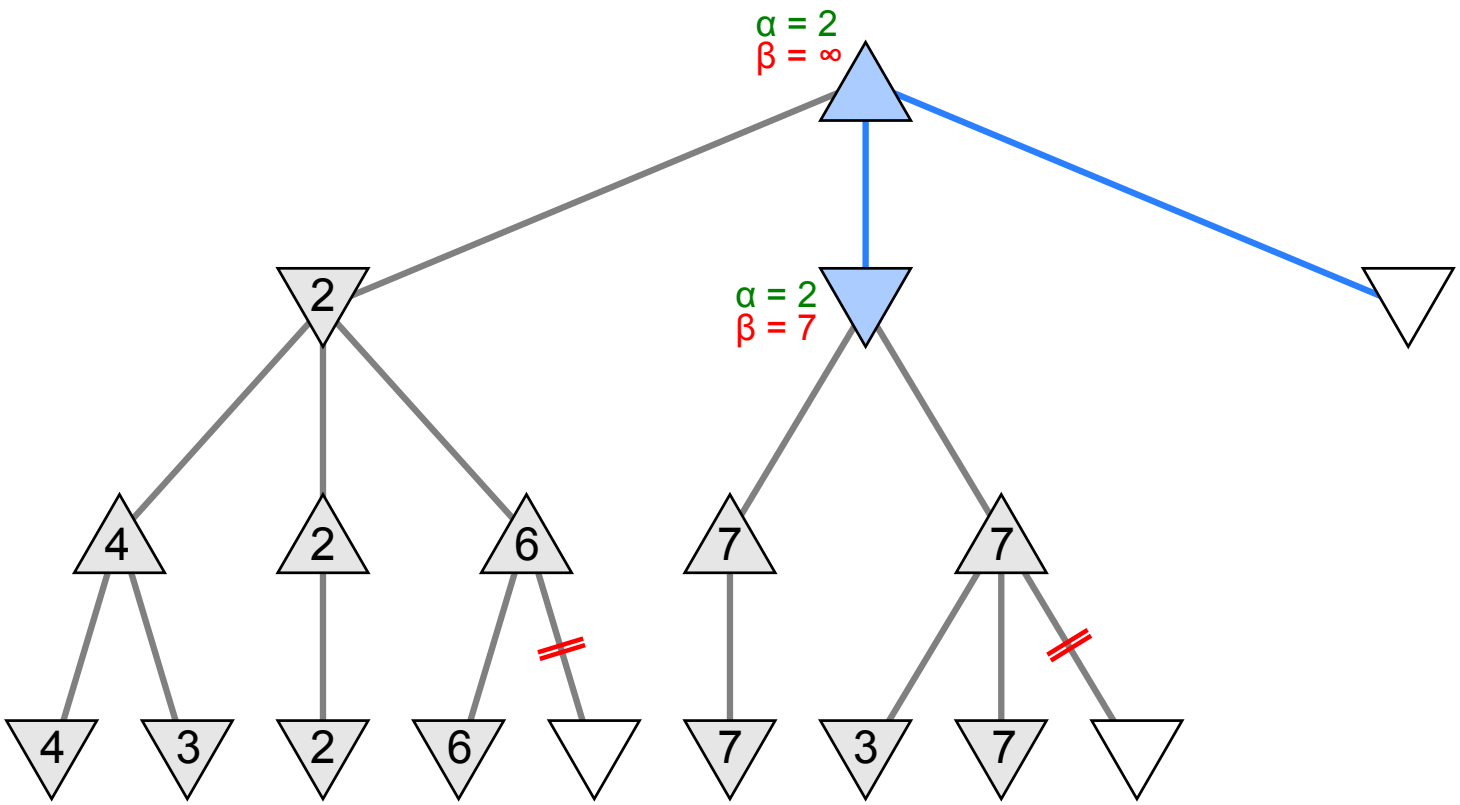
Minimax s alfa-beta prořezáváním: Příklad



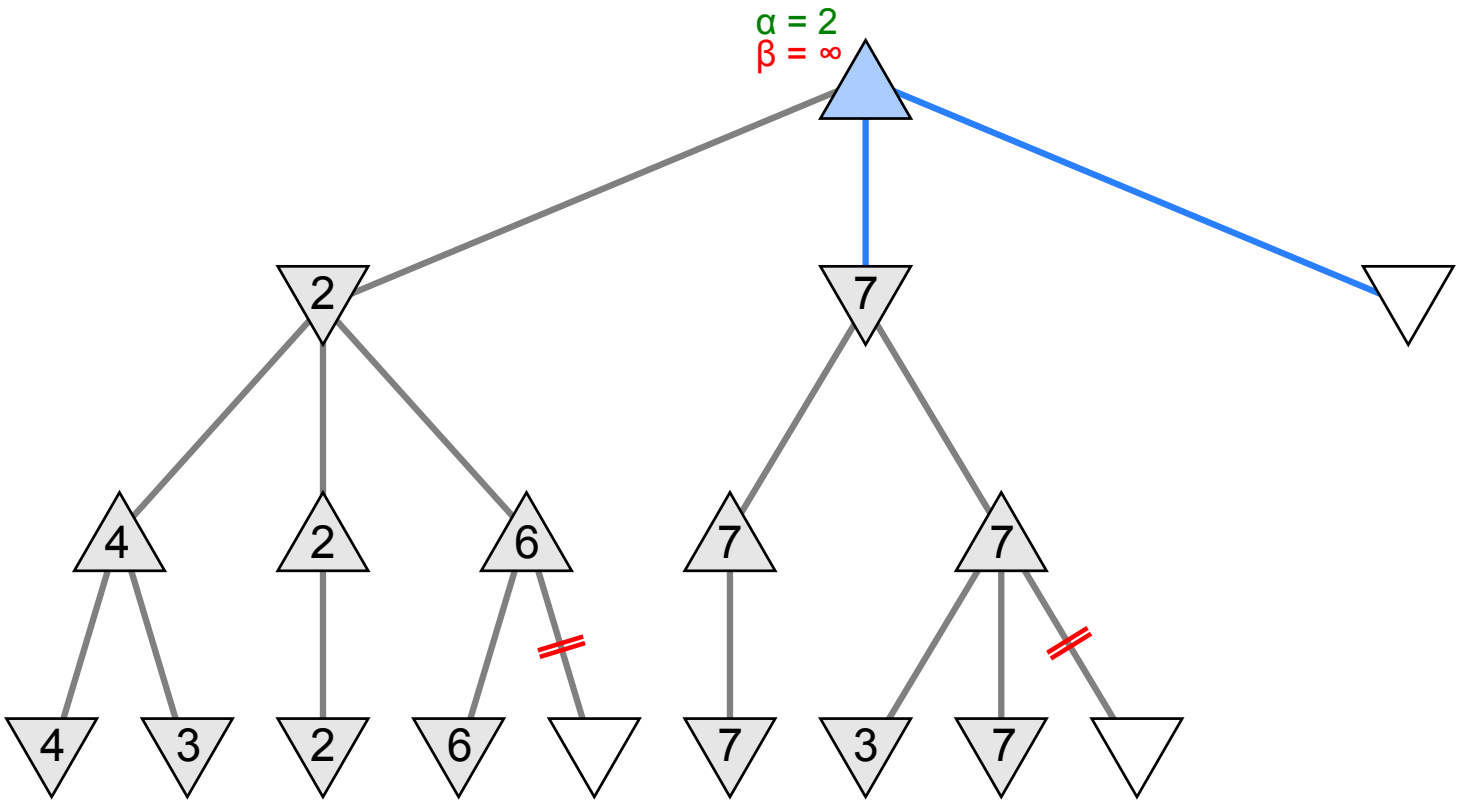
Minimax s alfa-beta prořezáváním: Příklad



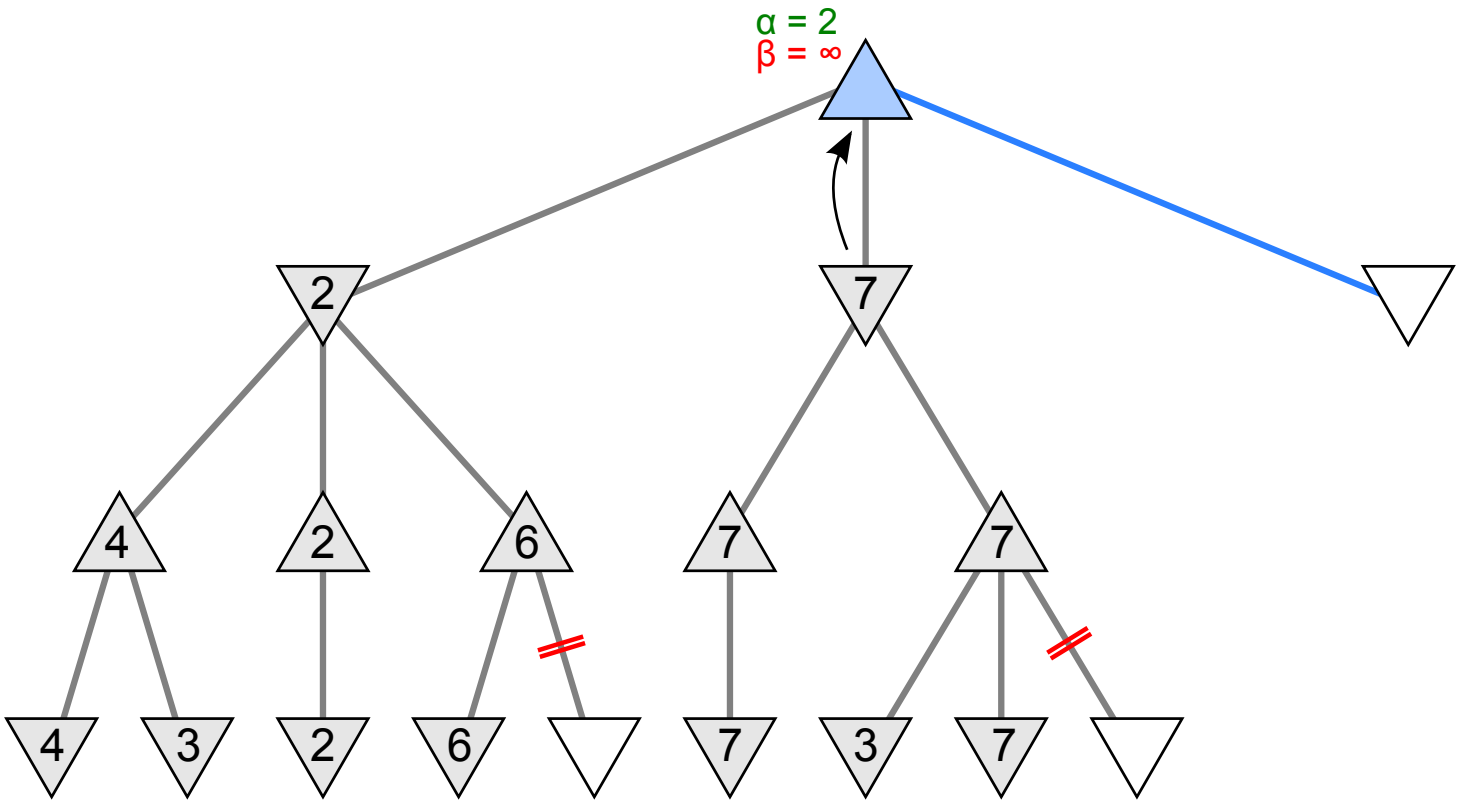
Minimax s alfa-beta prořezáváním: Příklad



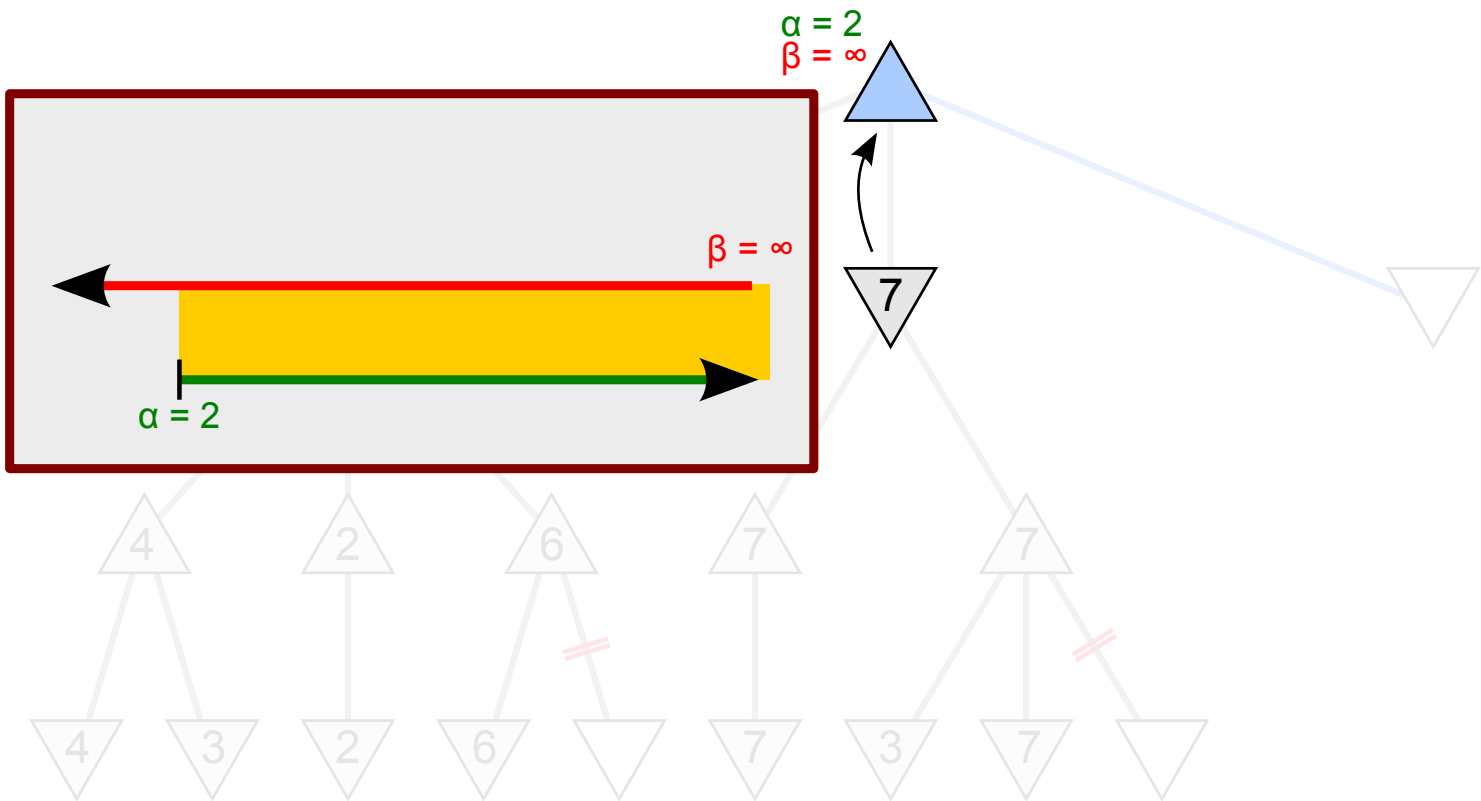
Minimax s alfa-beta prořezáváním: Příklad



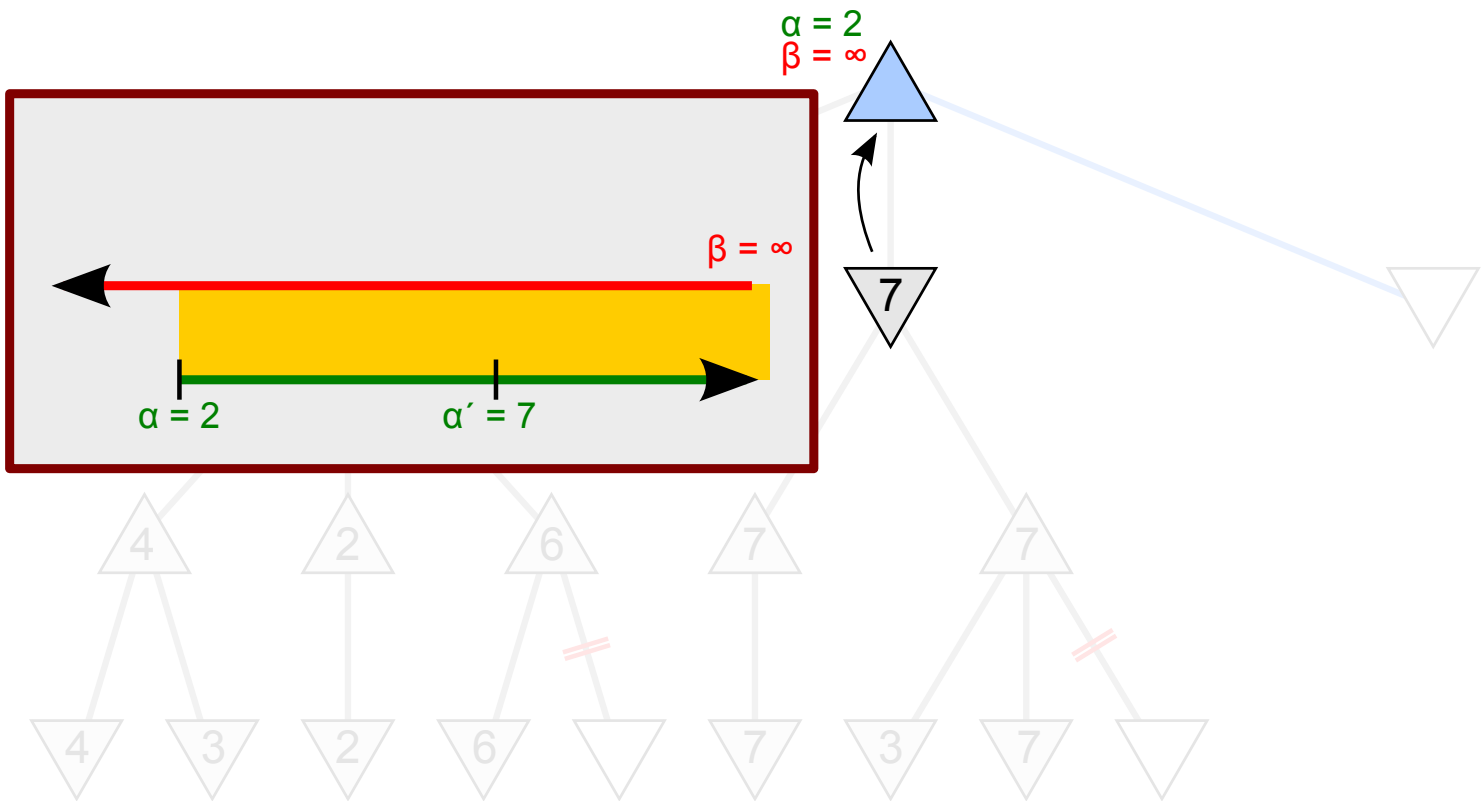
Minimax s alfa-beta prořezáváním: Příklad



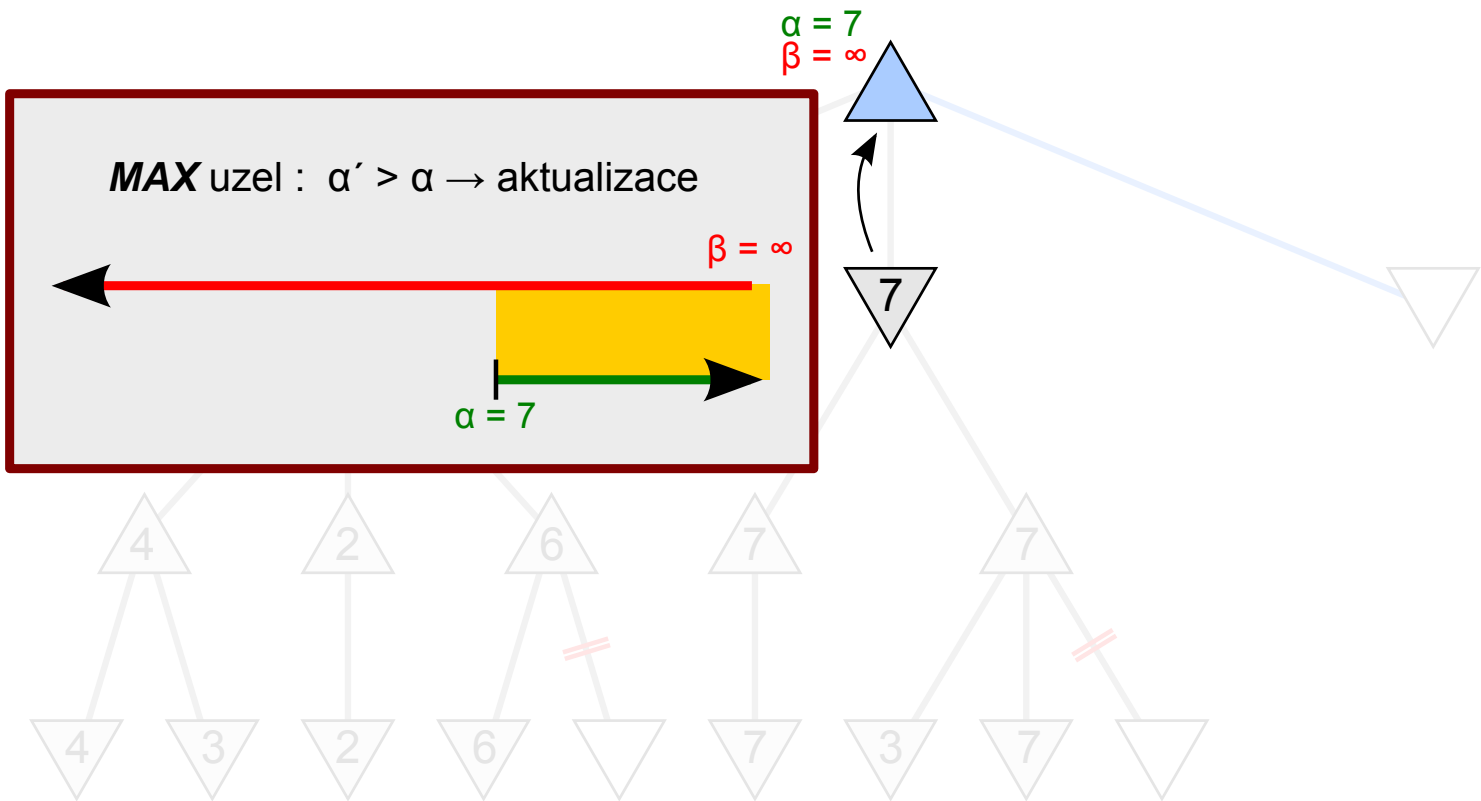
Minimax s alfa-beta prořezáváním: Příklad



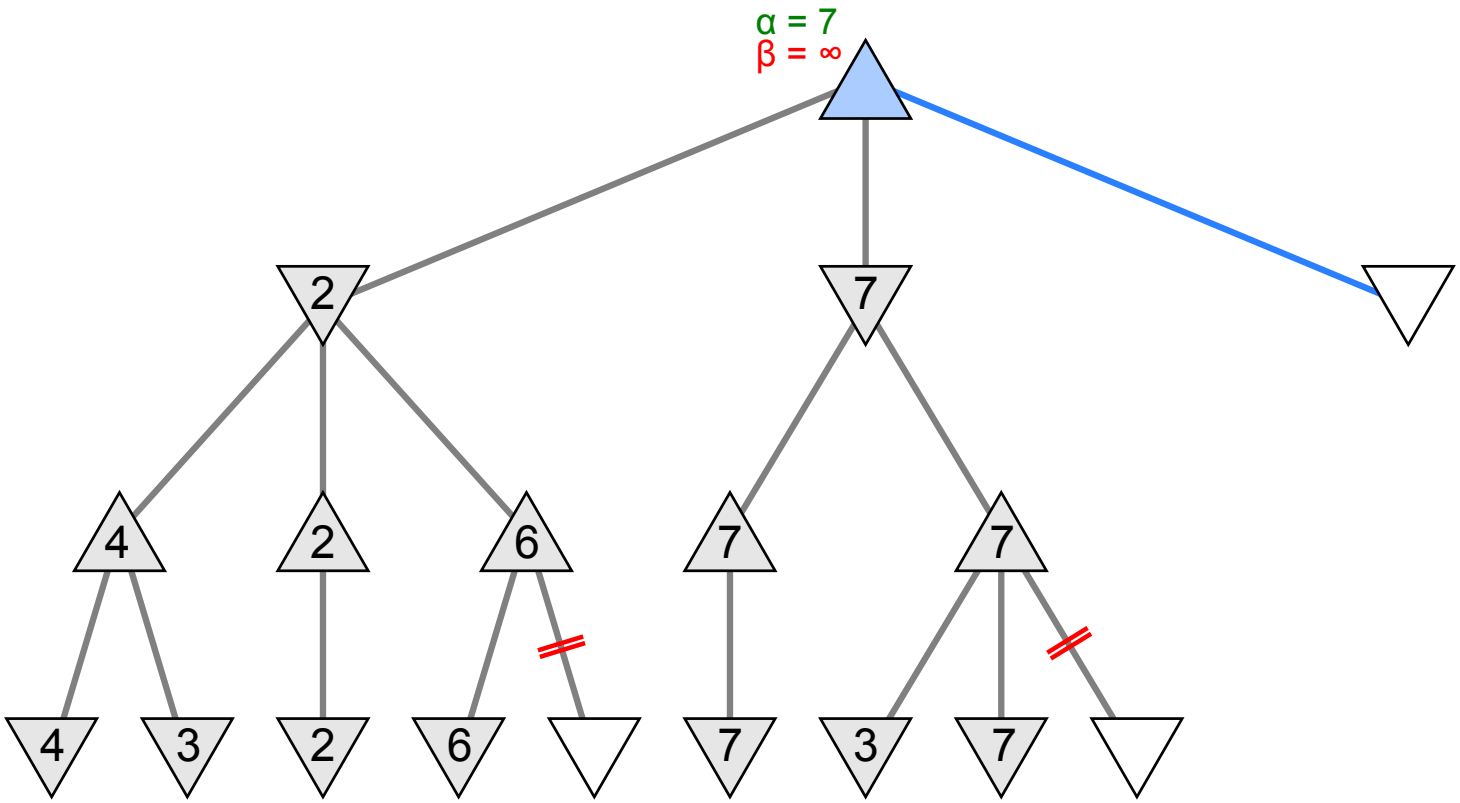
Minimax s alfa-beta prořezáváním: Příklad



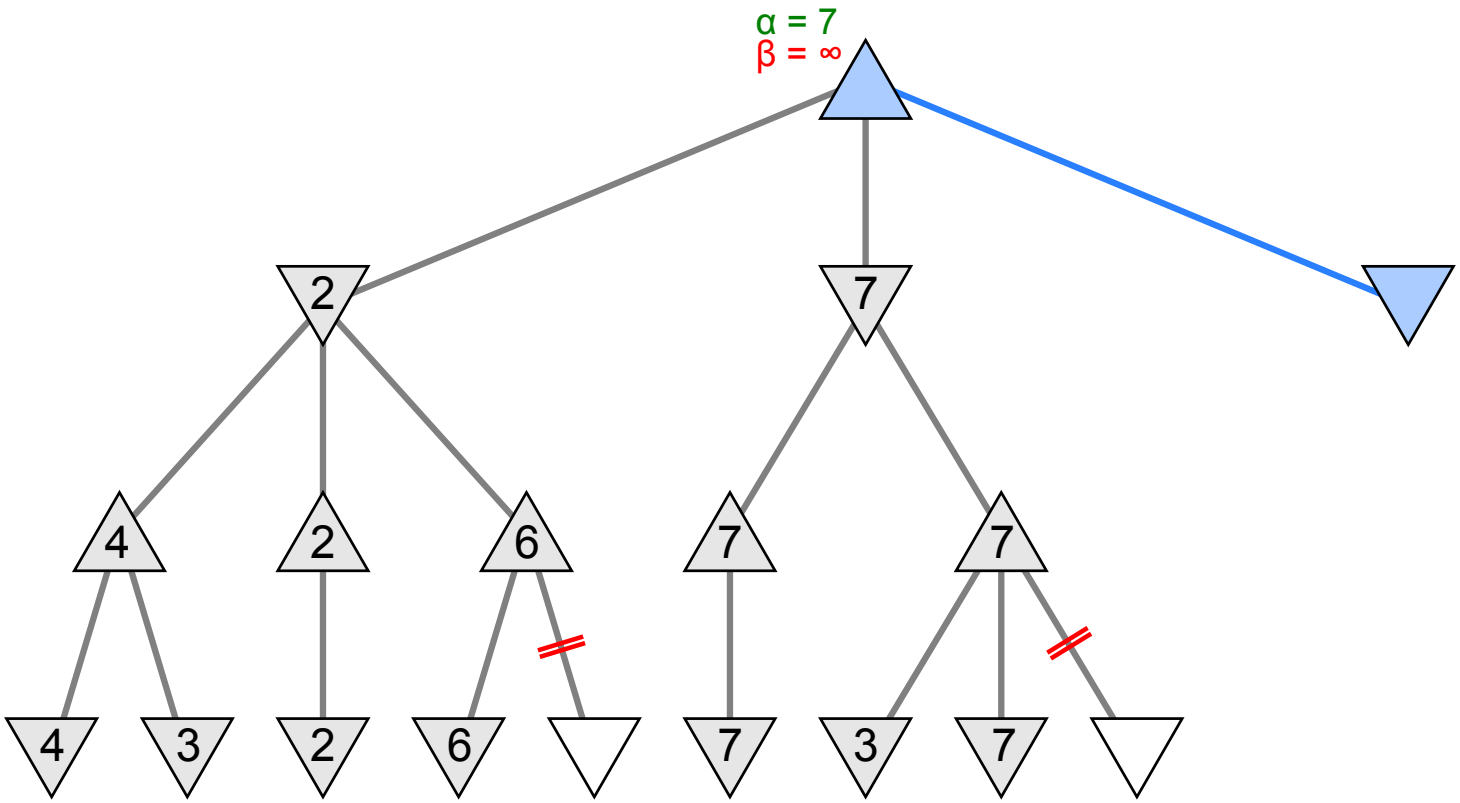
Minimax s alfa-beta prořezáváním: Příklad



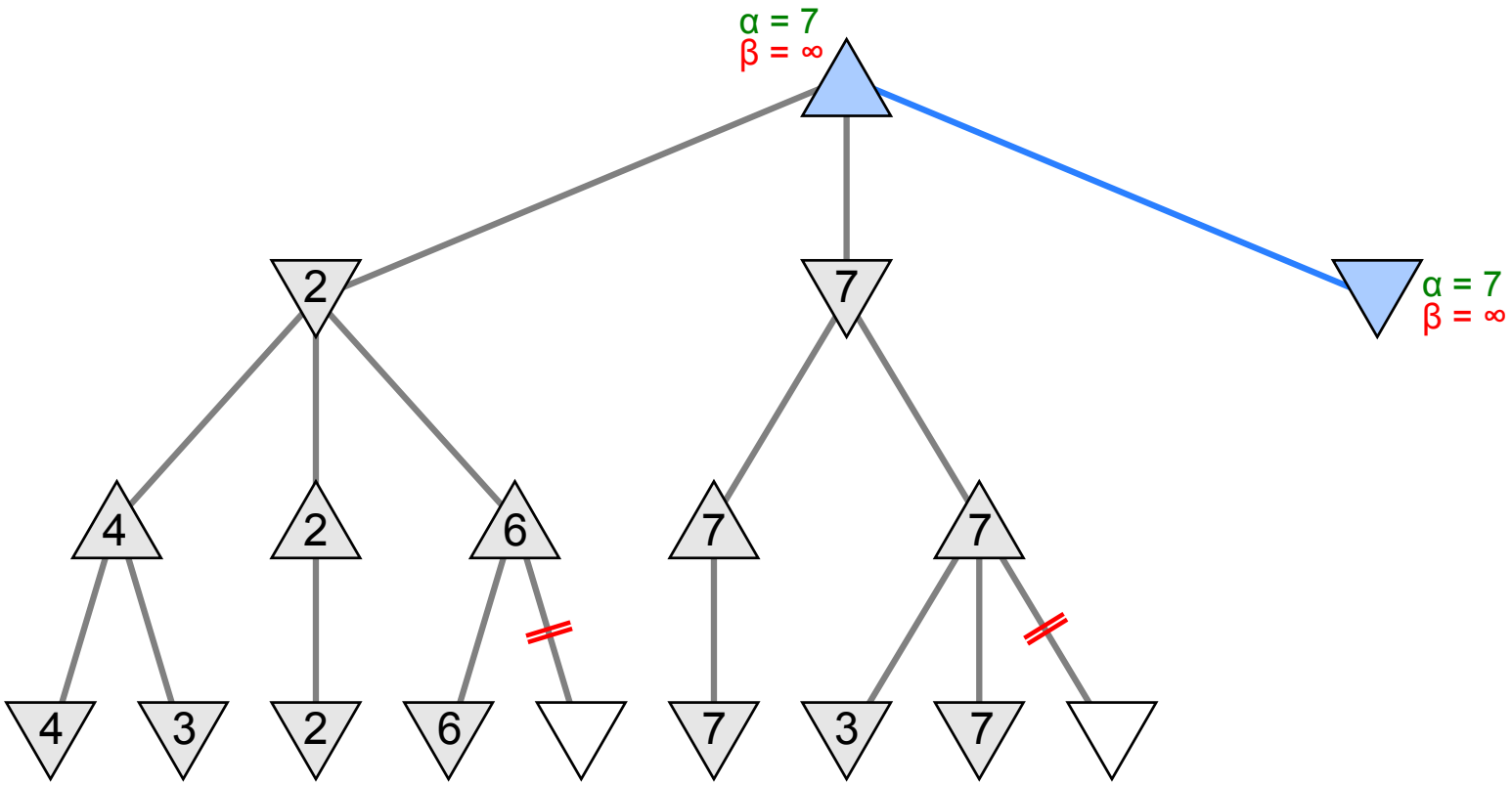
Minimax s alfa-beta prořezáváním: Příklad



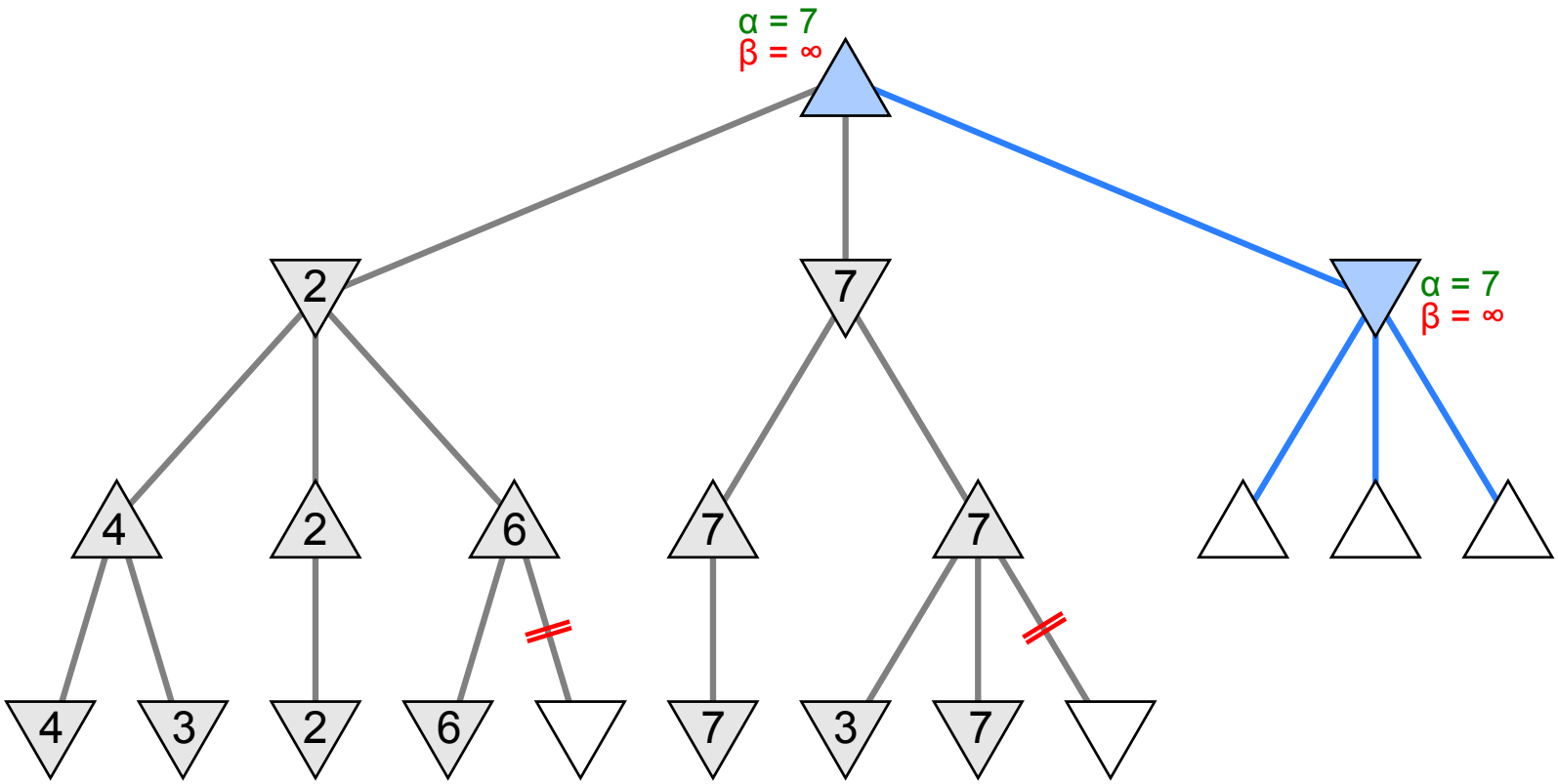
Minimax s alfa-beta prořezáváním: Příklad



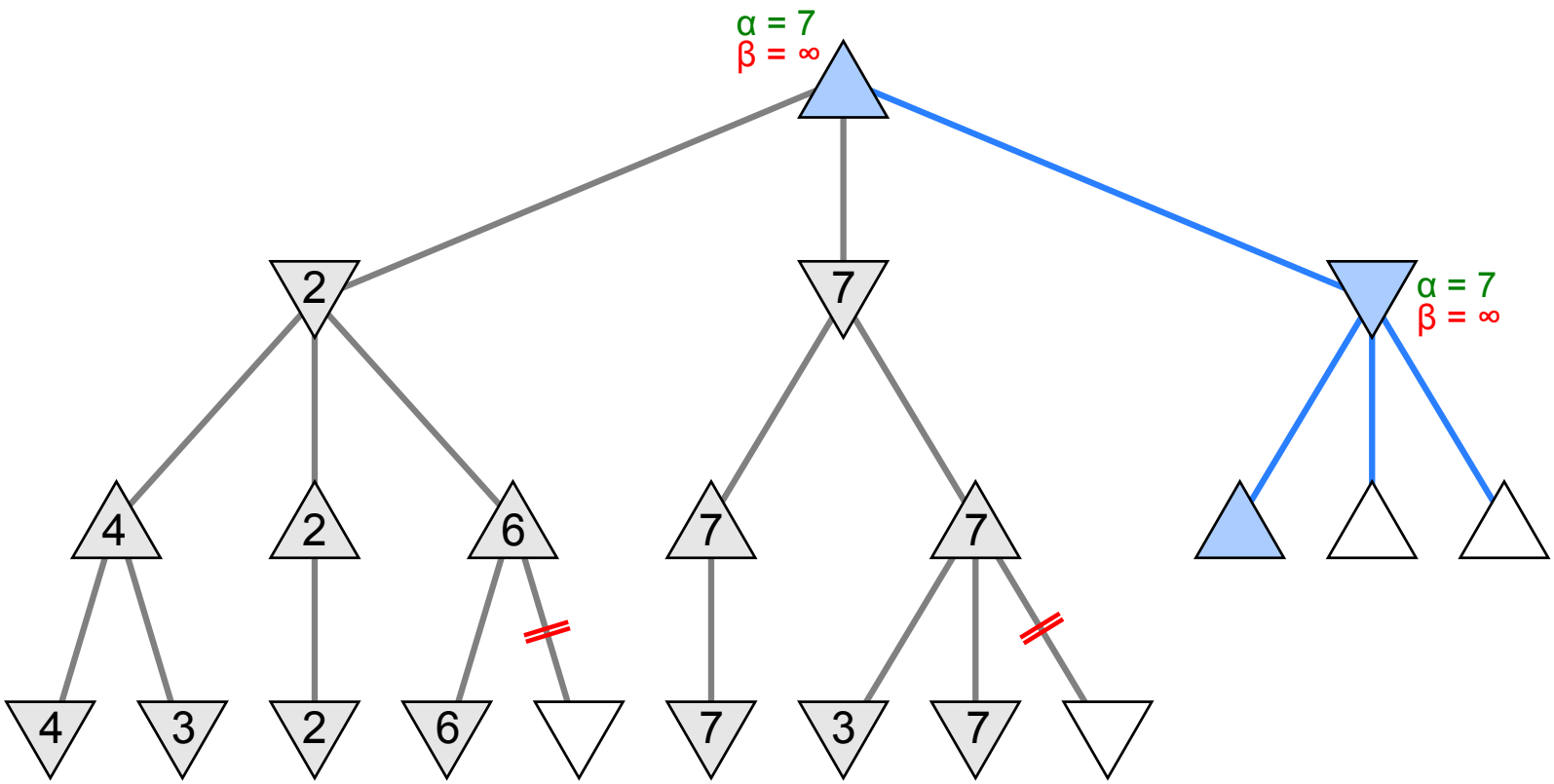
Minimax s alfa-beta prořezáváním: Příklad



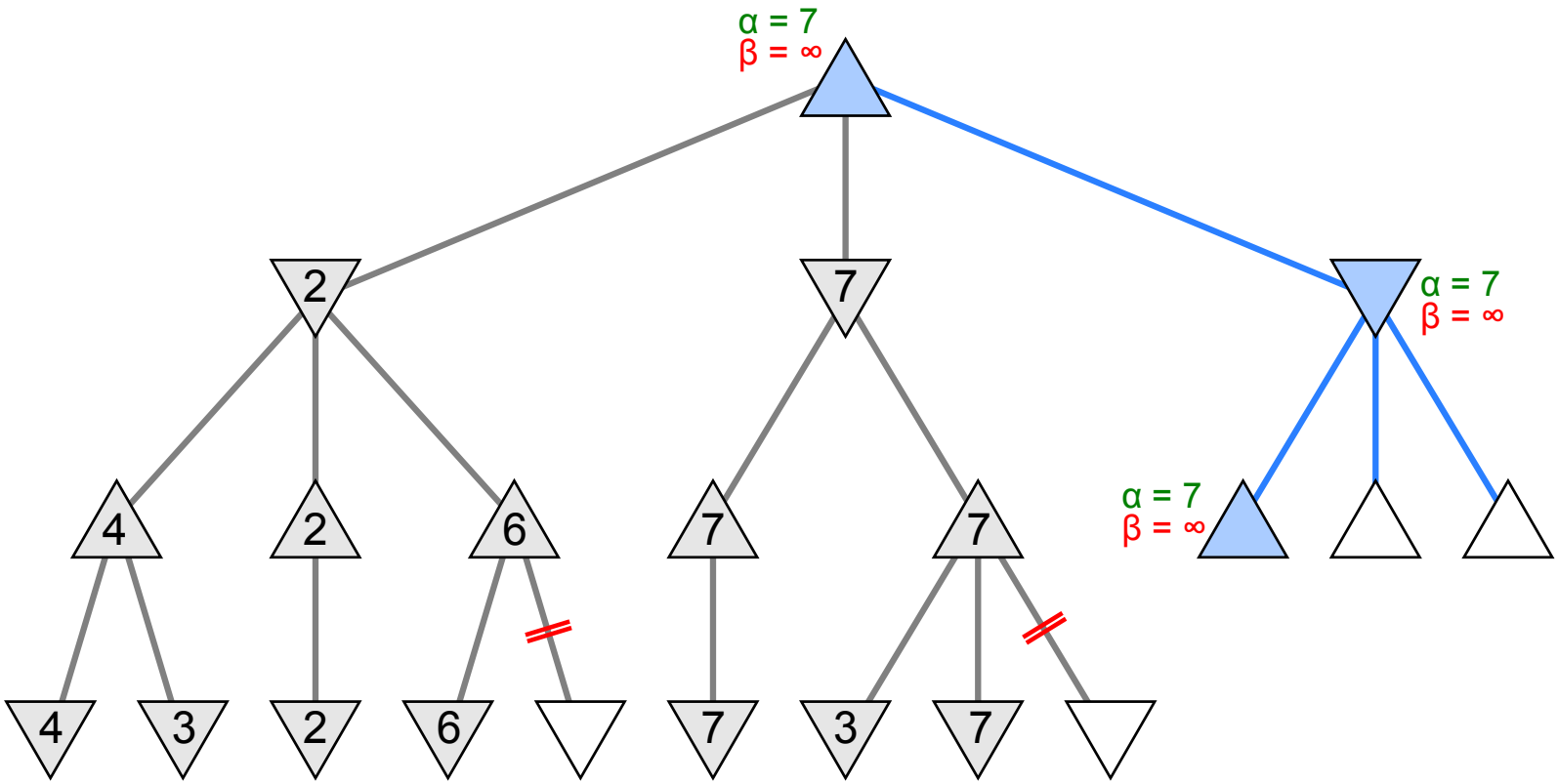
Minimax s alfa-beta prořezáváním: Příklad



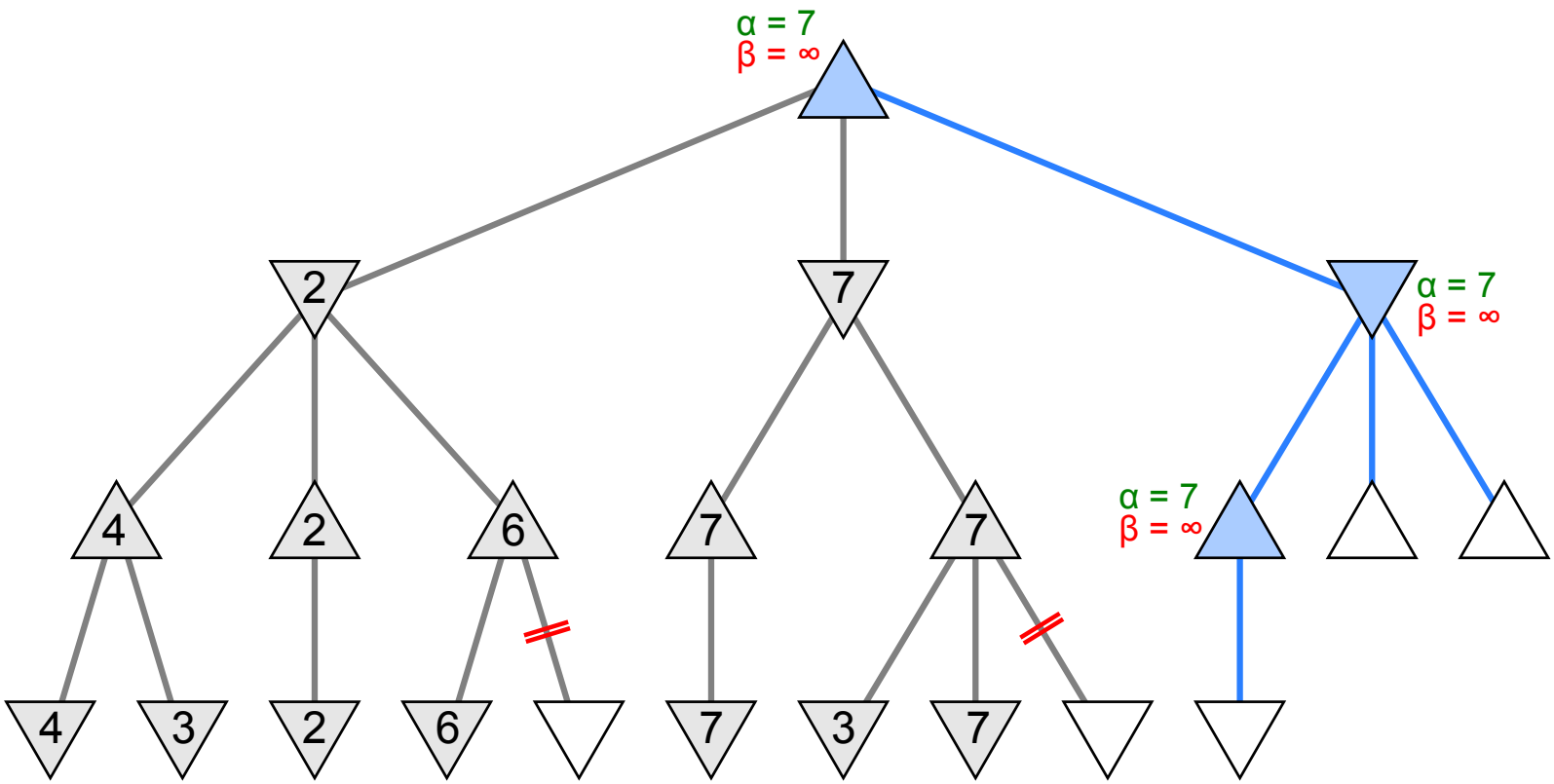
Minimax s alfa-beta prořezáváním: Příklad



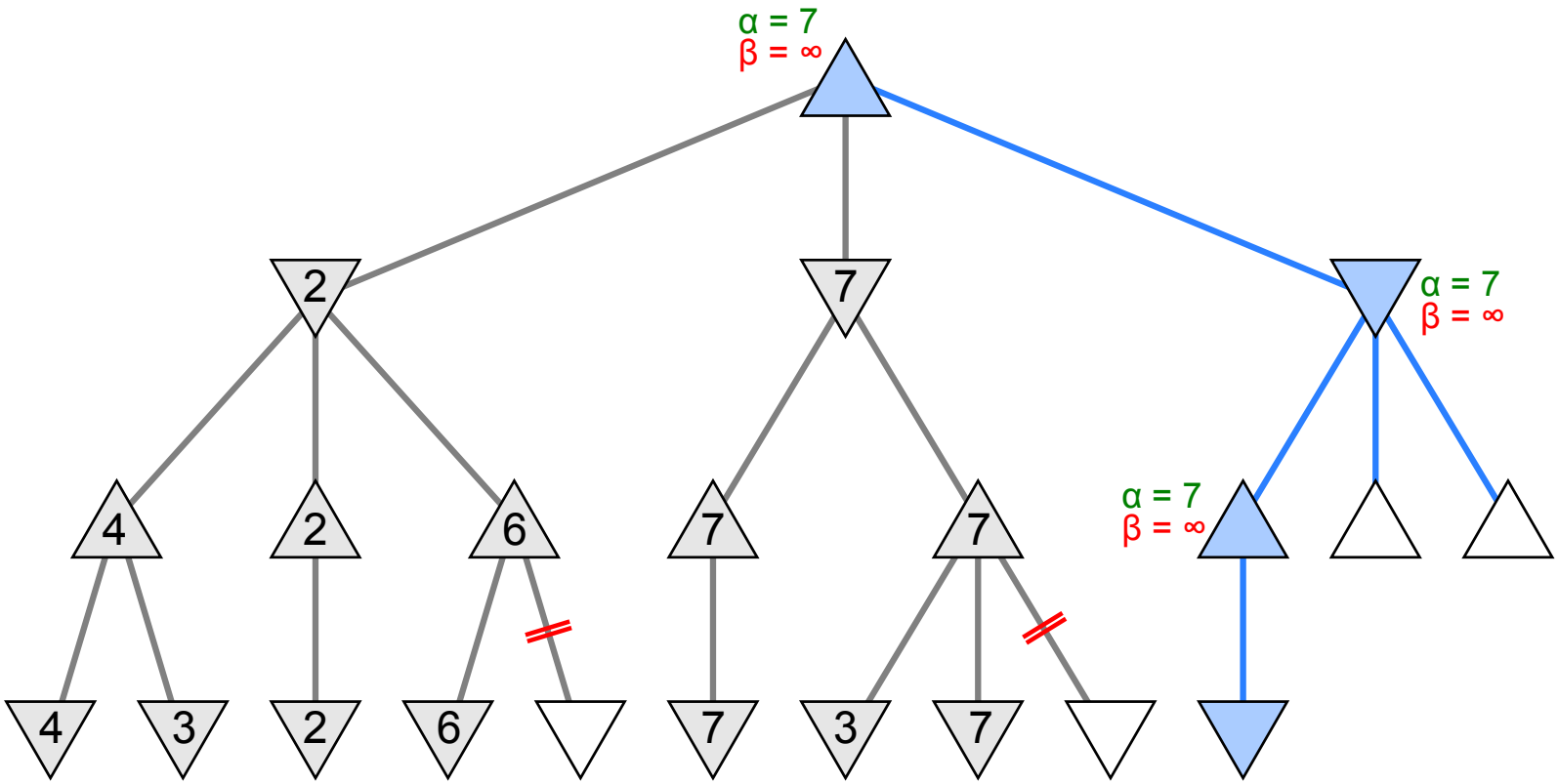
Minimax s alfa-beta prořezáváním: Příklad



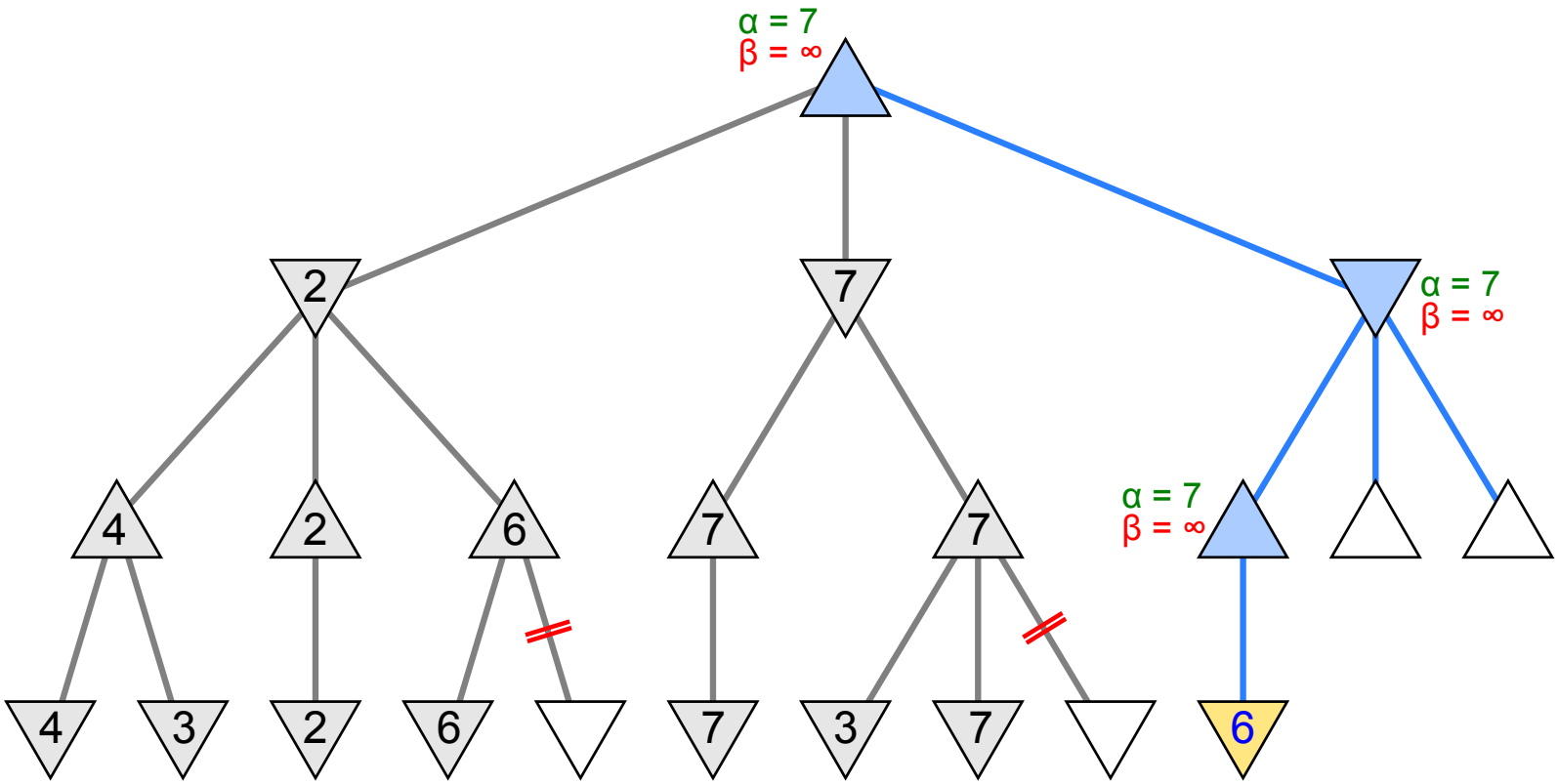
Minimax s alfa-beta prořezáváním: Příklad



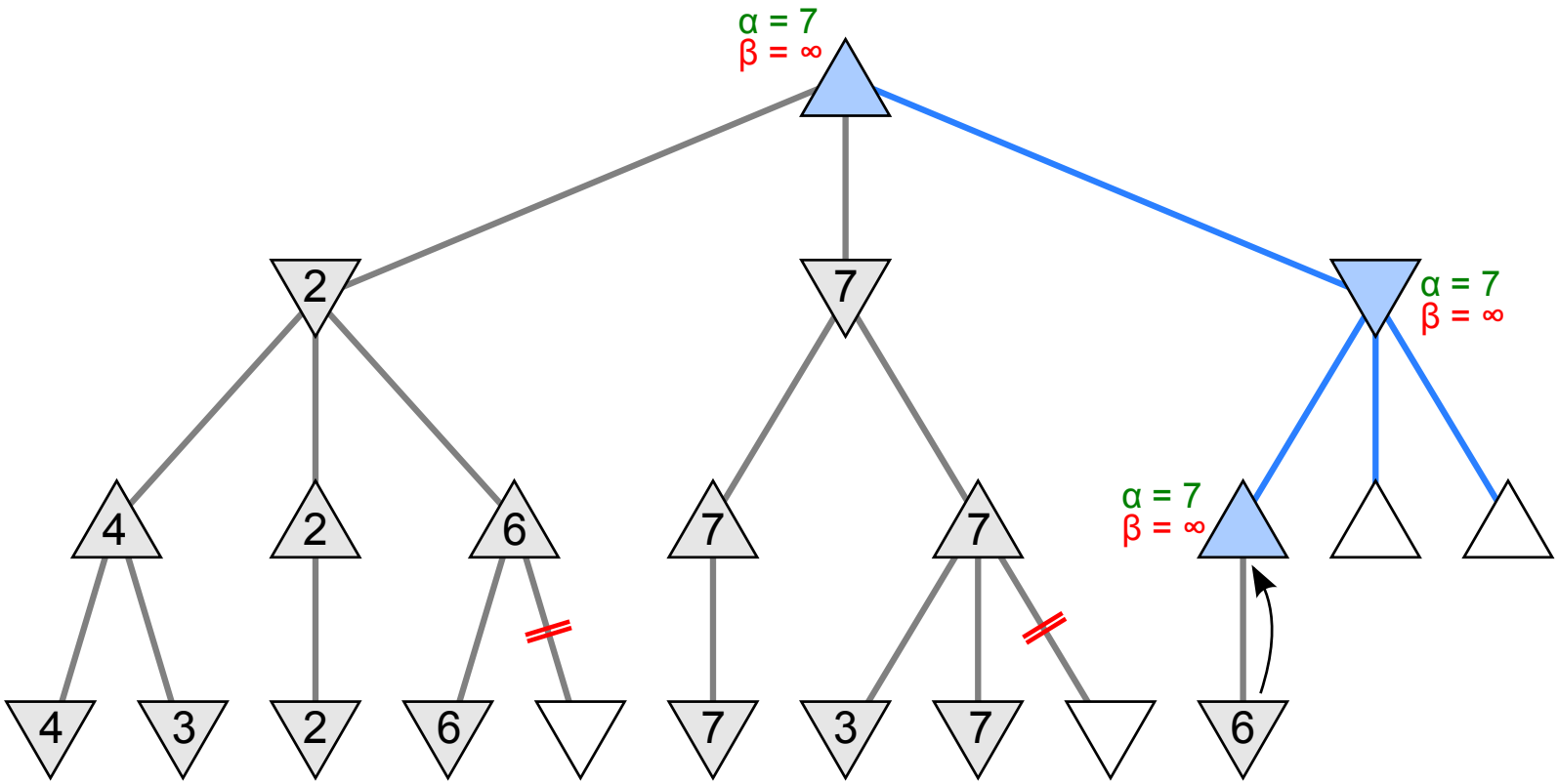
Minimax s alfa-beta prořezáváním: Příklad



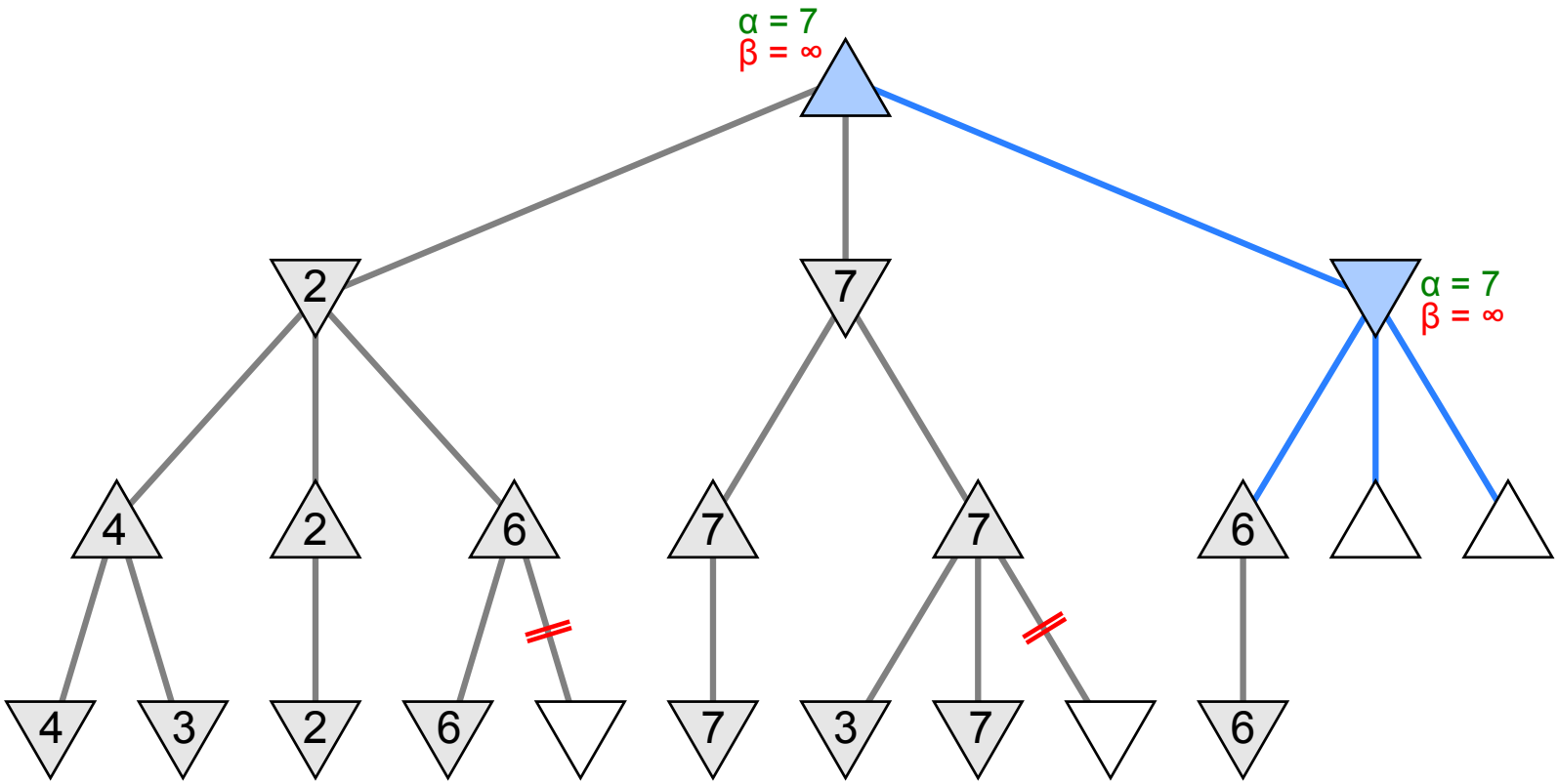
Minimax s alfa-beta prořezáváním: Příklad



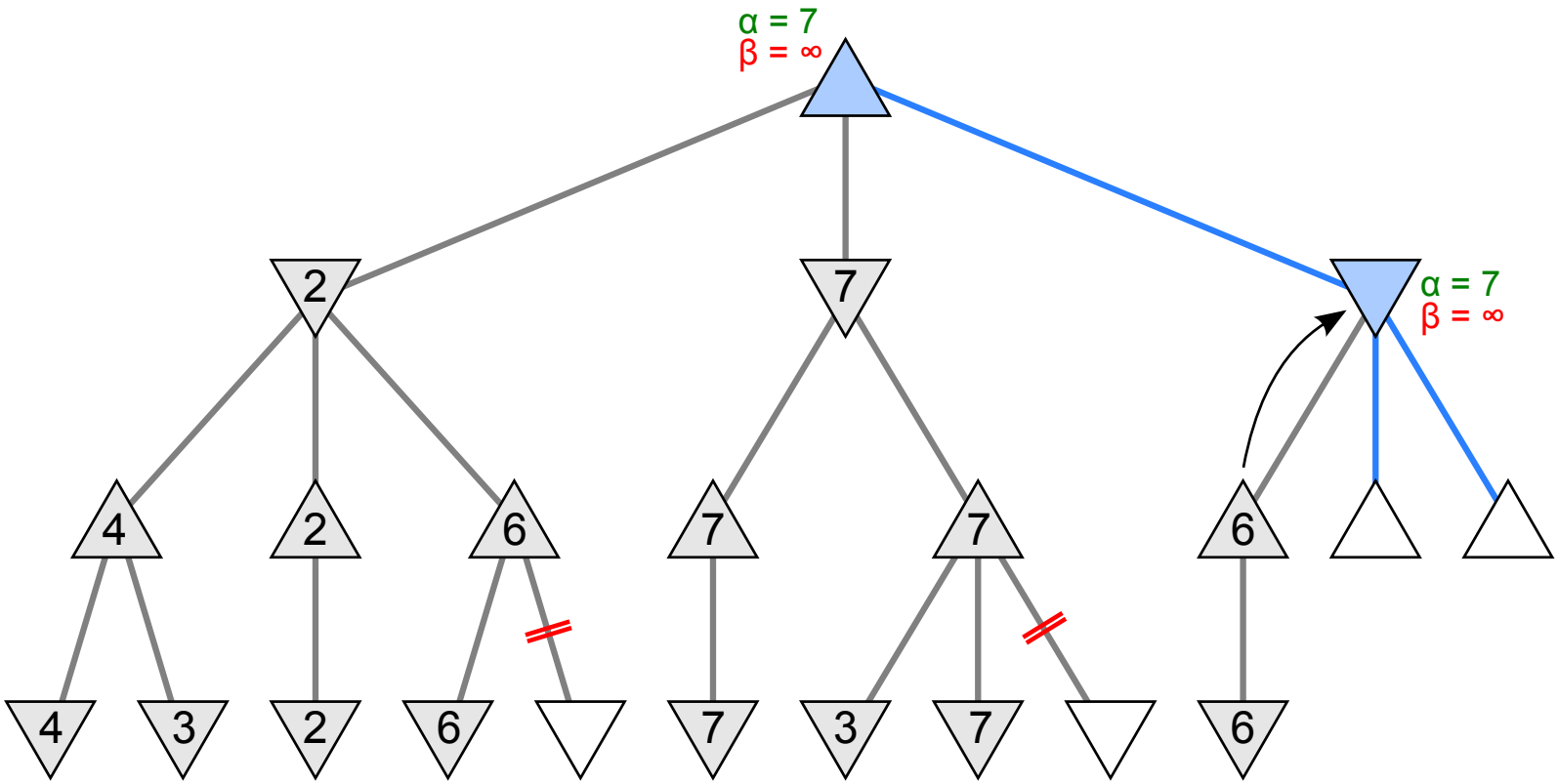
Minimax s alfa-beta prořezáváním: Příklad



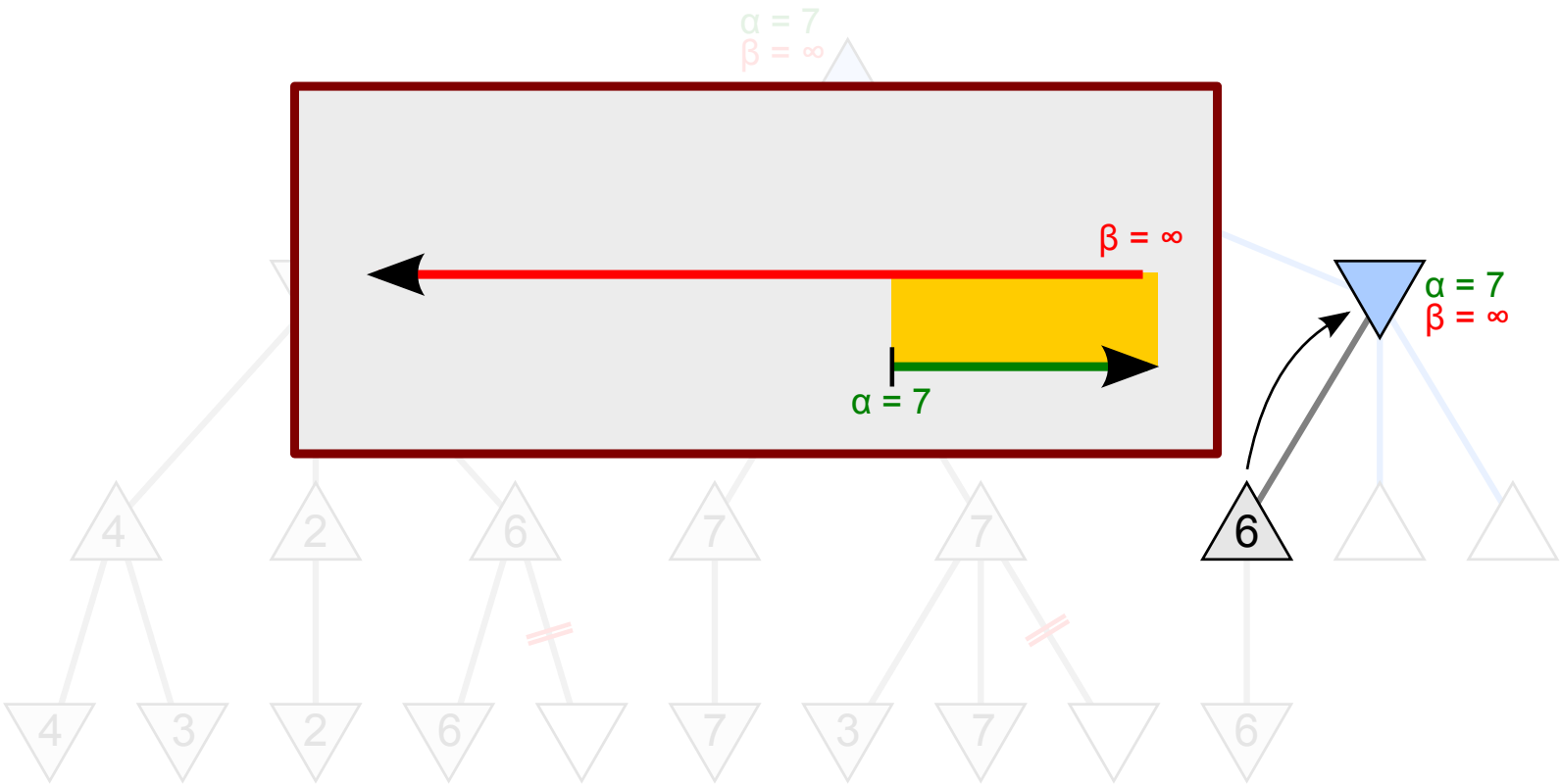
Minimax s alfa-beta prořezáváním: Příklad



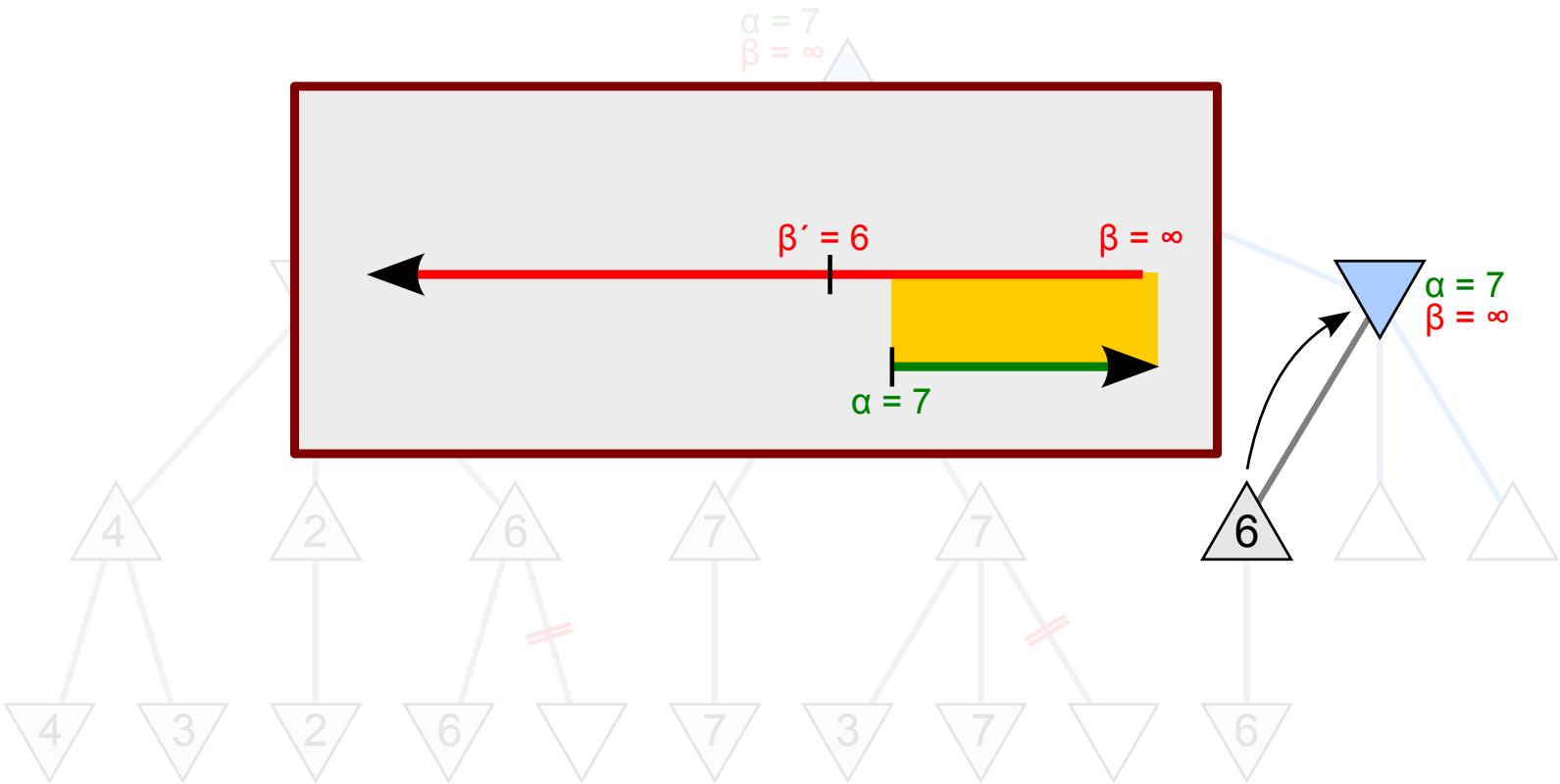
Minimax s alfa-beta prořezáváním: Příklad



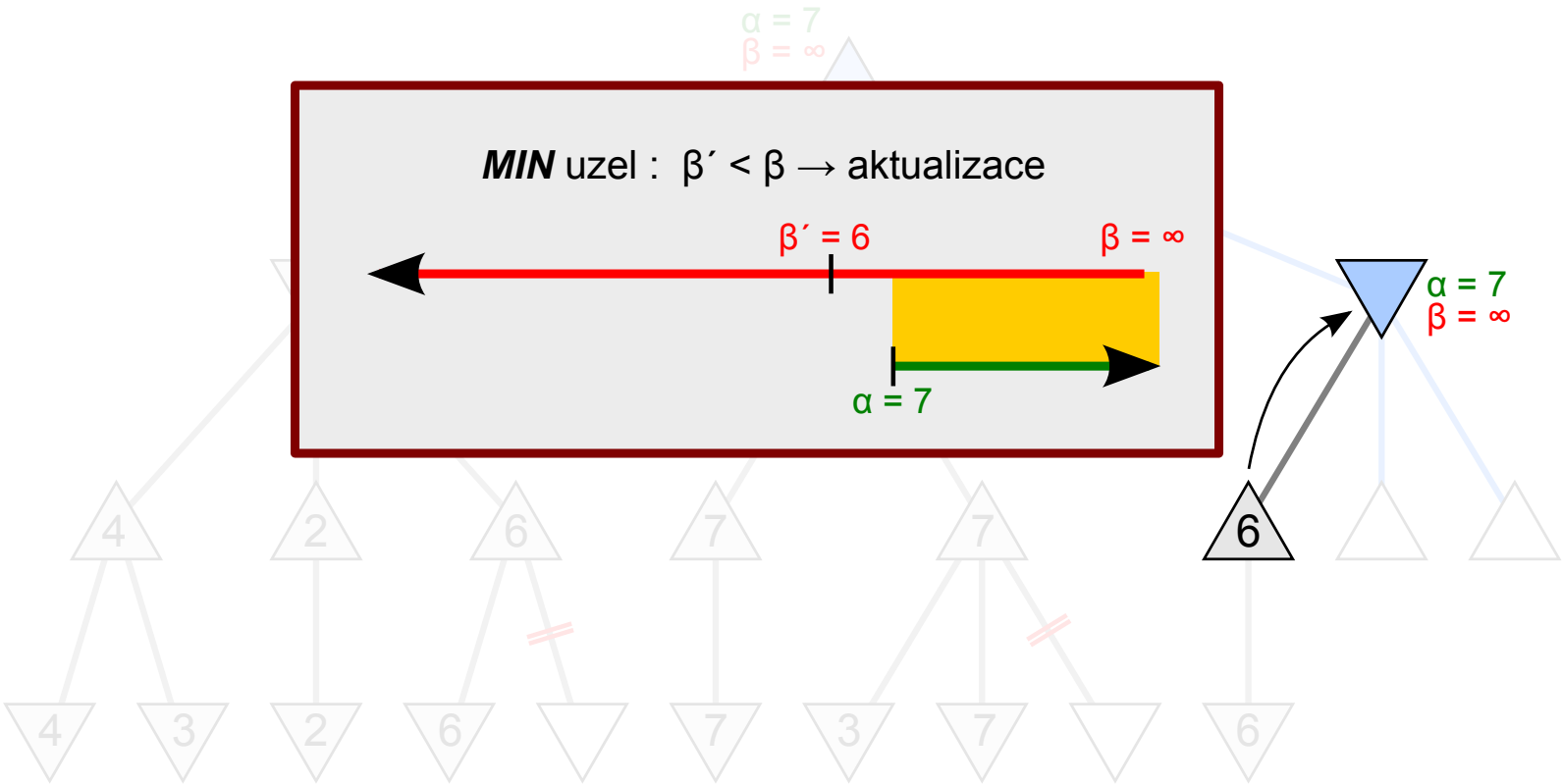
Minimax s alfa-beta prořezáváním: Příklad



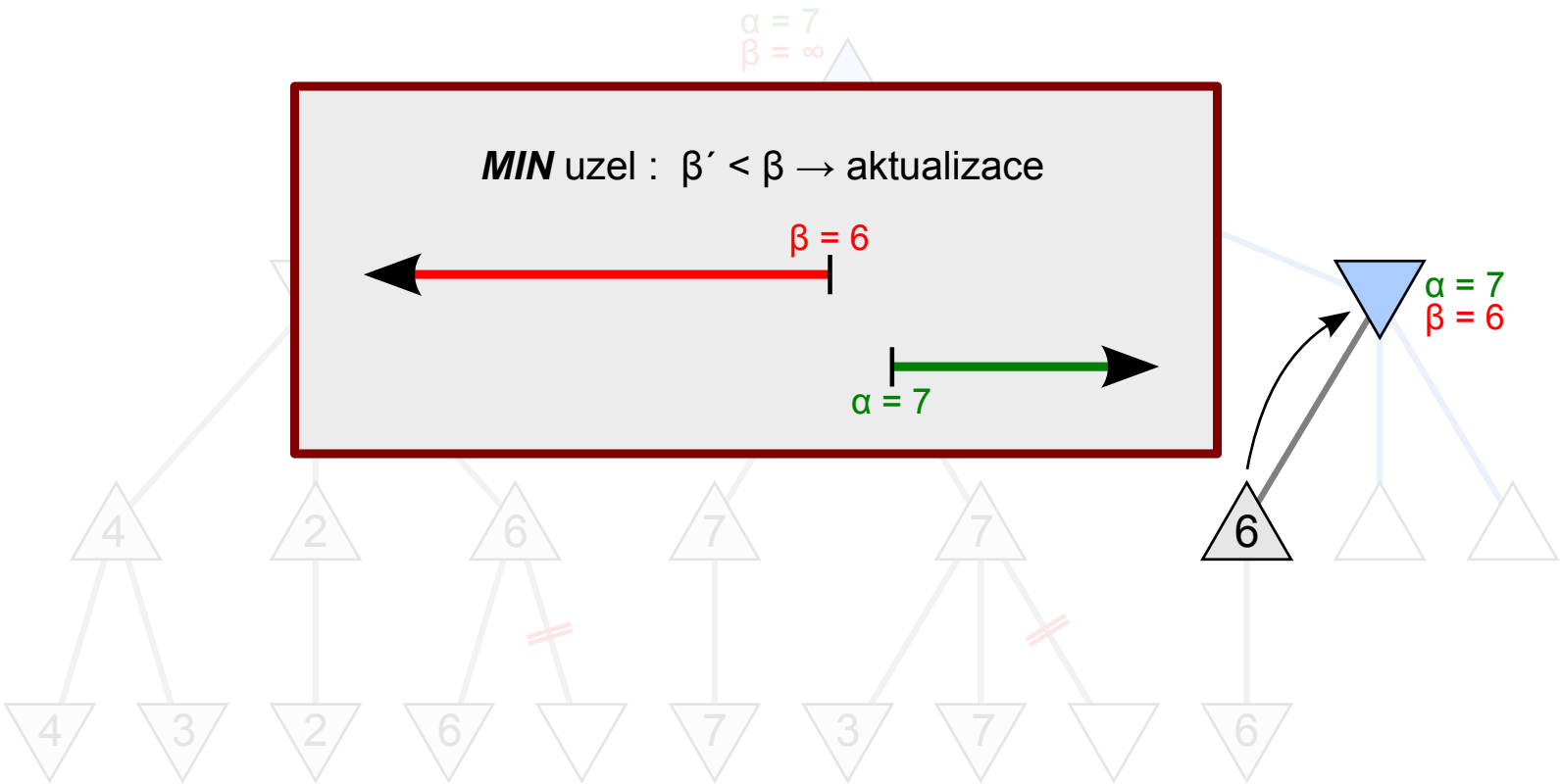
Minimax s alfa-beta prořezáváním: Příklad



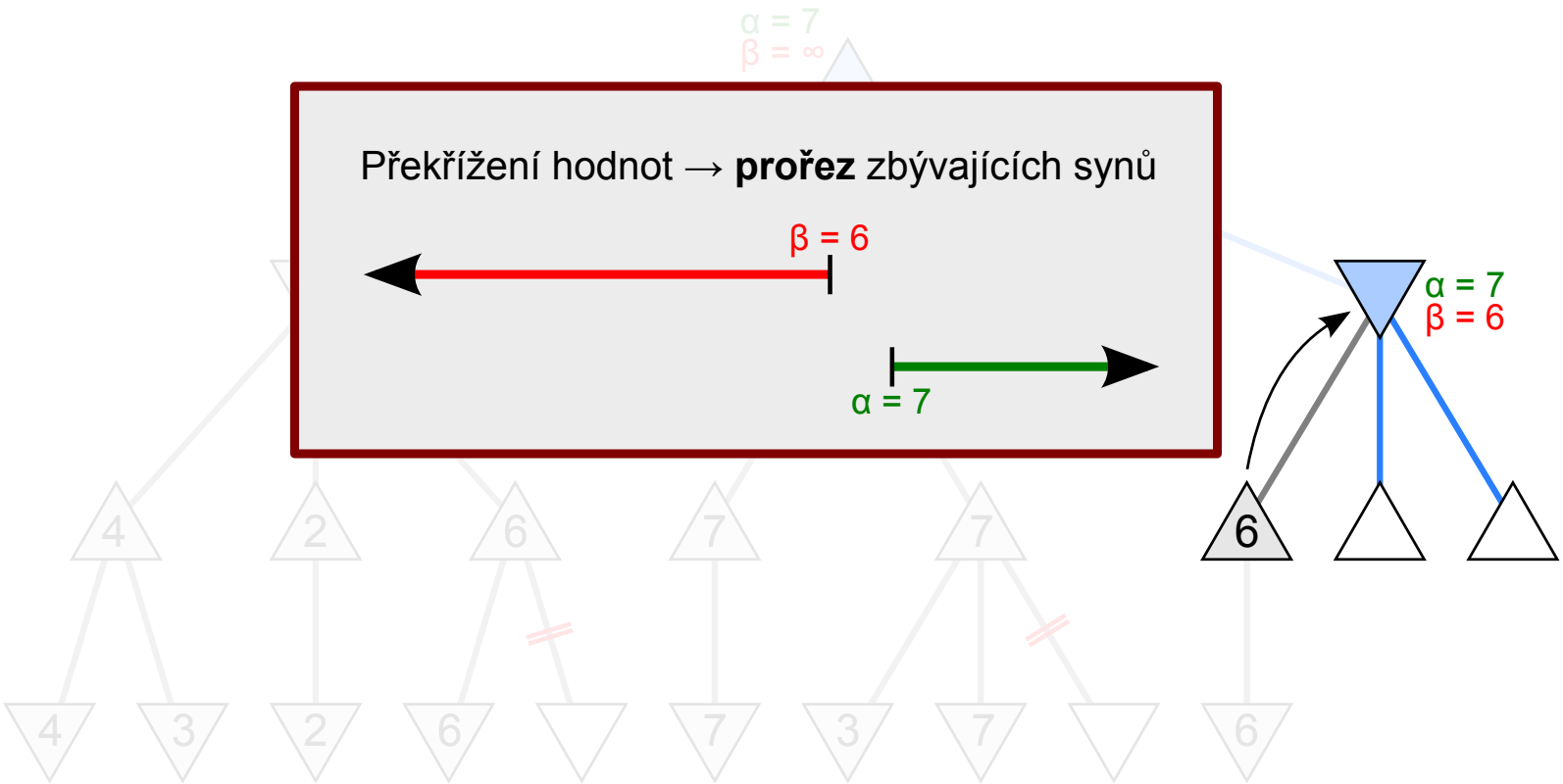
Minimax s alfa-beta prořezáváním: Příklad



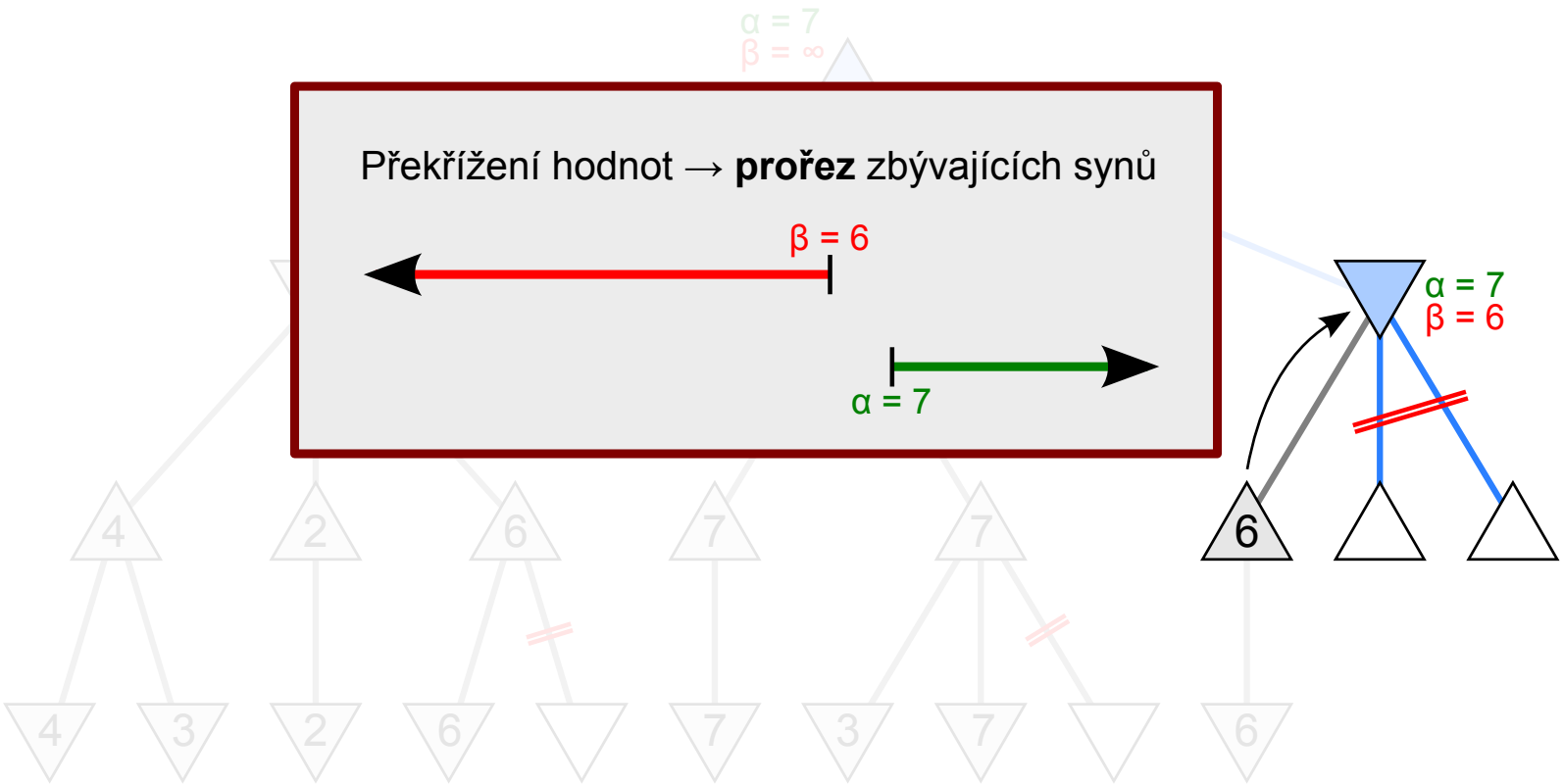
Minimax s alfa-beta prořezáváním: Příklad



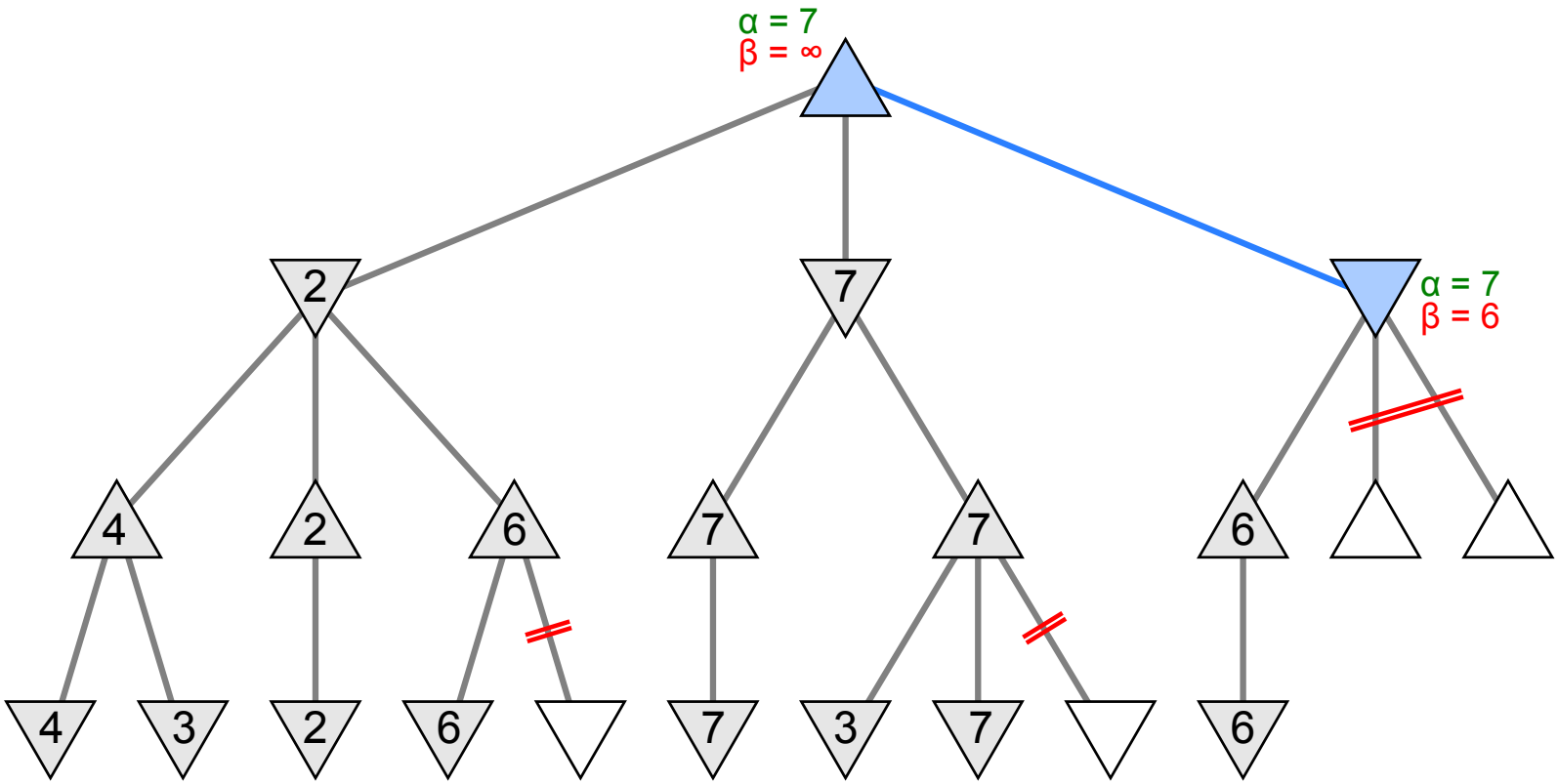
Minimax s alfa-beta prořezáváním: Příklad



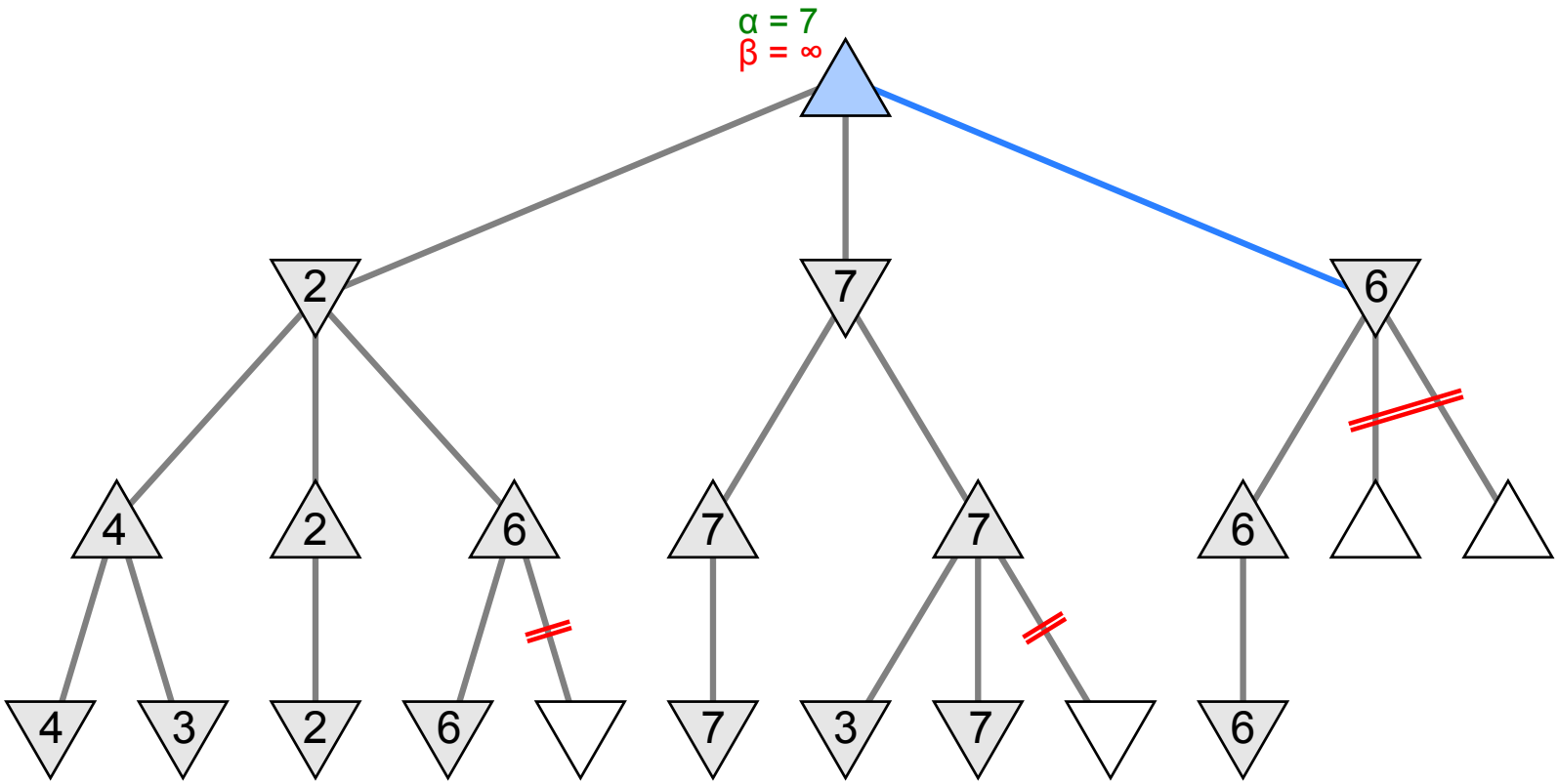
Minimax s alfa-beta prořezáváním: Příklad



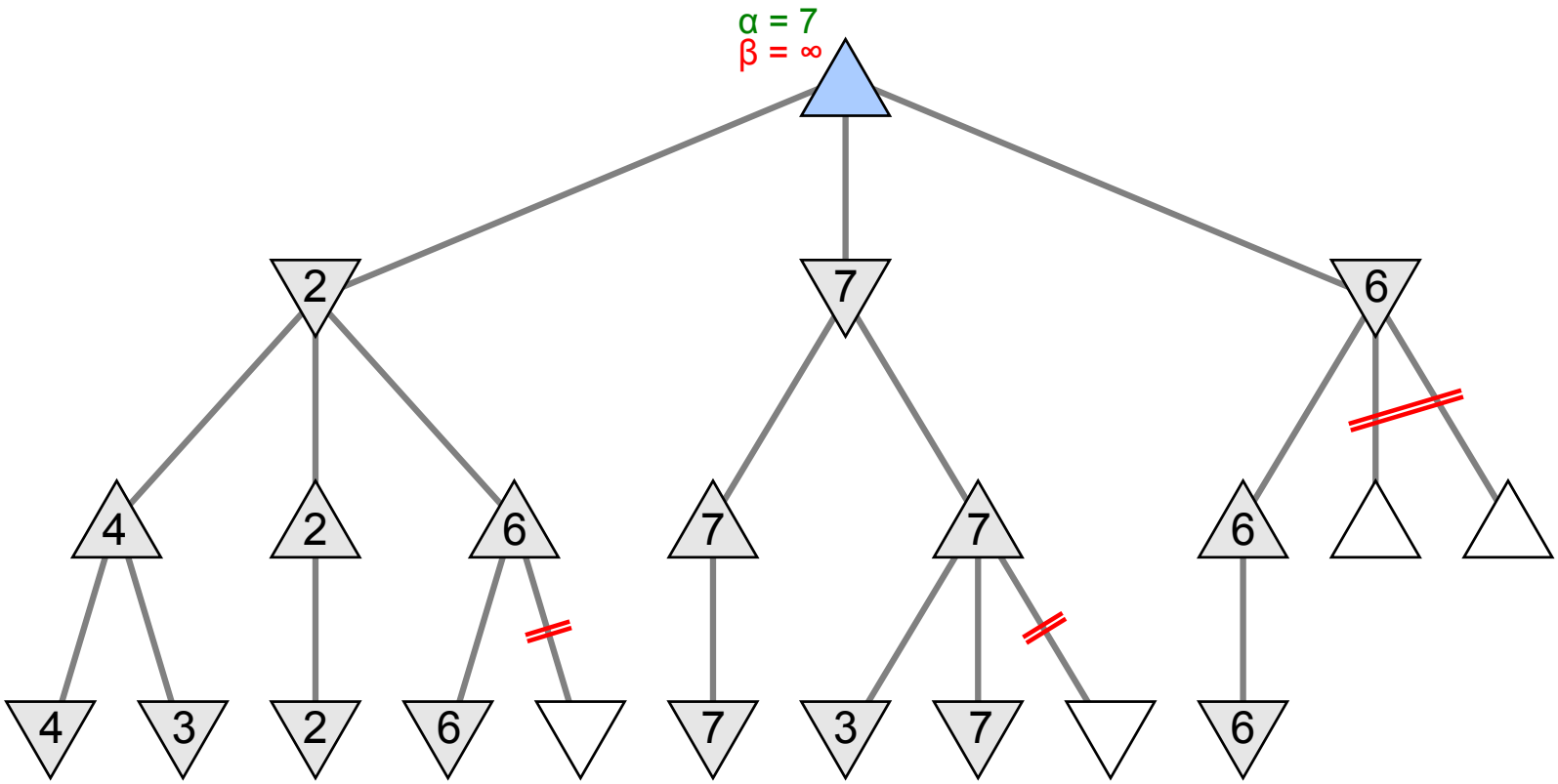
Minimax s alfa-beta prořezáváním: Příklad



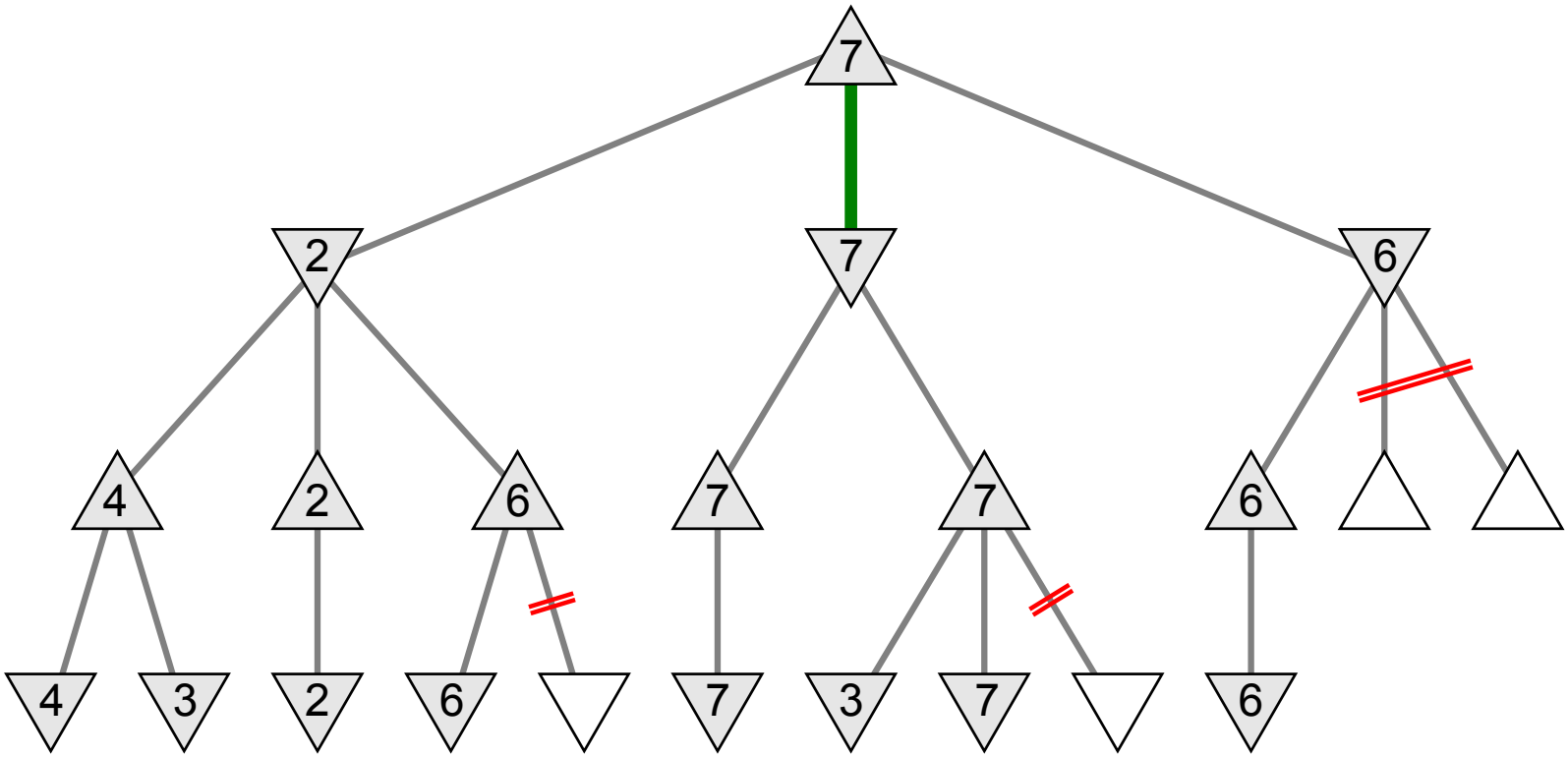
Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad



Minimax s alfa-beta prořezáváním: Příklad



Algoritmus Minimax: Složitost

Základní varianta algoritmu Minimax má složitost $\mathcal{O}(b^d)$, kde

- b je průměrný větvicí faktor,
- d je hloubka stromu.

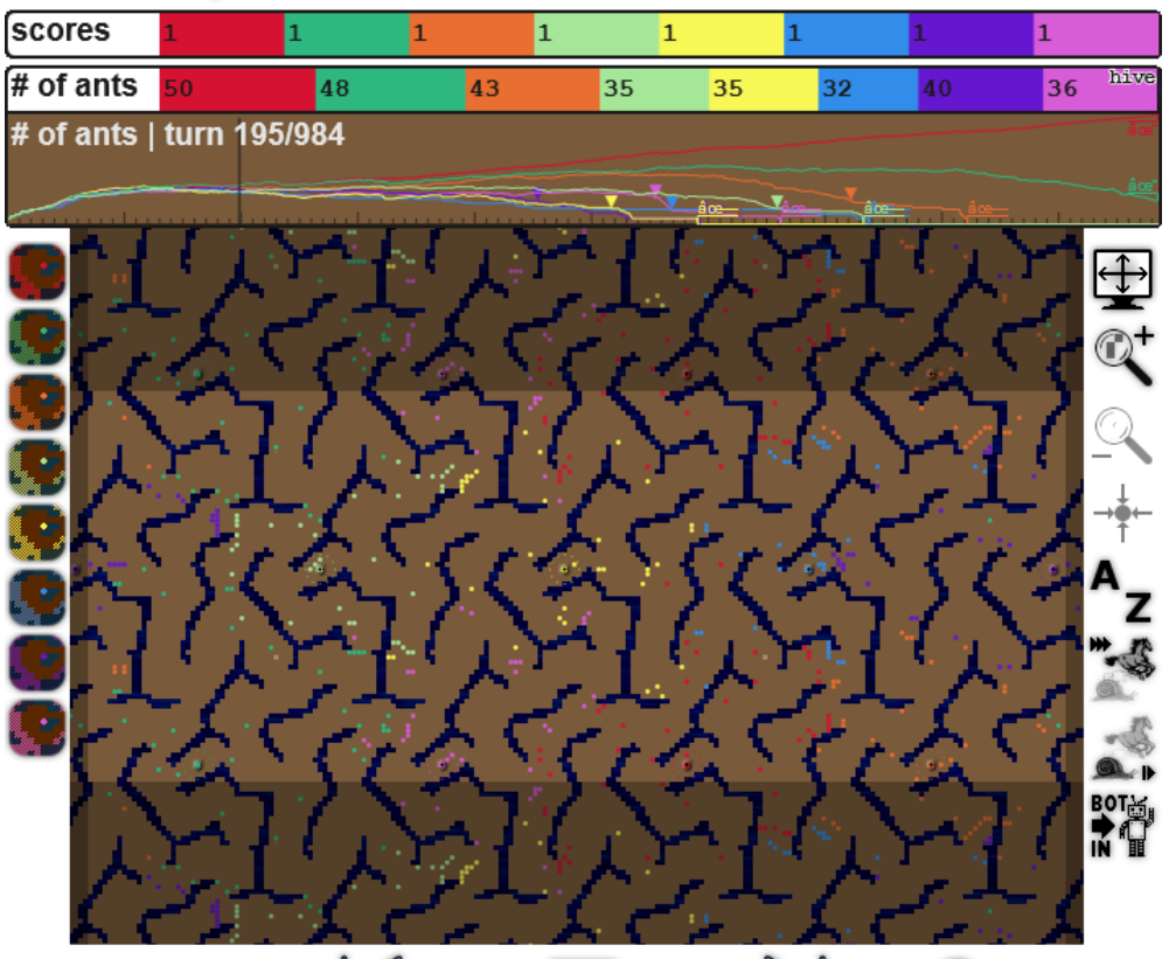
Zlepšení pomocí alfa-beta prořezávání záleží na **pořadí** expanze uzlů:

- přibližně $\mathcal{O}(b^{\frac{3}{4}d})$, pokud je pořadí náhodné a b je relativně malé (< 1000),
- $\mathcal{O}(b^{\frac{d}{2}})$, pokud je pořadí optimální
  - optimální = nejlepší akce nejdříve,
  - lze se mu přiblížit např. heuristickým uspořádáním akcí (útok $>$ defenzíva),
    - ★ **killer heuristic** – jestliže daný tah (např. vzetí dámy) způsobil prořez někde jinde ve stromu, prioritizuj tento tah,
  - redukuje větvicí faktor b na $\sqrt{b}$ a umožňuje tak zdvojnásobit hloubku prohledávání

Evaluační funkce

- algoritmus Minimax s alfa-beta prořezáváním volí optimální akci v kořenu **vzhledem k ohodnocení listů**,
  - vše tedy závisí na kvalitě evaluační funkce!
- požadavky na evaluační funkci?
  - **rychlost** – nejdůležitější charakteristika,
  - **spolehlivost** – hodnota skutečně koreluje s kvalitou stavu
- při prohledávání s omezenou hloubkou narážíme na **problém horizontu**
  - v hloubce $d = 5$ je „materiální“ evaluace vysoká, ovšem v hloubce $d = 6$ může protihráč zareagovat vzetím dámy!
  - **quiescence search** – expanzi stromu končíme pouze ve stavech, po nichž bezprostředně nehrozí „divoké výměny“ figur

Příklad aplikace algoritmu Minimax v AI Challenge 2011



http://xathis.com/posts/ai-challenge-2011-ants.html

Minimax s alfa-beta prořezáváním: Alternativy

- algoritmus Minimax s alfa-beta prořezáváním se stal standardní metodou pro řešení her v extenzivní formě,
- dolní odhad složitosti $\mathcal{O}\left(b^{\frac{d}{2}}\right)$ může být frustrující
  - zlepšení lze dosáhnout především návrhem dobrých heuristik,
- **ne**jedná se však o jediný možný postup!
  - K. Chellapilla, D. B. Fogel: evoluce umělých neuronových sítí pro hraní dámy,
  - „vyšlechtěný“ hráč pod pseudonymem „Blondie24“ porazil na online portálu *The Zone* 99.61 % soupeřů
  - vybavování umělé neuronové sítě je řádově mnohem rychlejší, než deterministické algoritmy typu Minimax

Březen 2016: Úspěch AlphaGo

- Lee Sedol, jeden z nejlepších světových hráčů Go, poražen softwarem od společnosti Google DeepMind (skóre 4:1),
  - 2014 akvizice společnosti DeepMind Googlem za $500M
- Při expanzi herního stromu použit Monte Carlo Tree Search s heuristickou evaluací kvality stavu pomocí metod hlubokého + posilovaného učení (Deep learning, Reinforcement learning)
  - trénovací množina 30M konfigurací s expertní evaluací,
  - fáze posilovaného učení: obrovské množství her proti jiným instancím sebe sama za cílem dalšího zlepšení,
  - distribuovaný výpočet (1920 CPUs + 280 GPUs),
  - poměrně rychlá hra: 2 vteřiny na jeden tah