Hry a UI – historie

- Babbage, 1846 – počítač porovnává přínos různých herních tahů
- von Neumann, 1944 – algoritmy perfektní hry
- Zuse, Wiener, Shannon, 1945–50 – přibližné vyhodnocování
- Turing, 1951 – první šachový program (jen na papíře)
- Samuel, 1952–57 – strojové učení pro zpřesnění vyhodnocování
- McCarthy, 1956 – prořezávání pro možnost hlubšího prohledávání

Hry a UI – historie

- Babbage, 1846 – počítač porovnává přínos různých herních tahů
- von Neumann, 1944 – algoritmy perfektní hry
- Zuse, Wiener, Shannon, 1945–50 – přibližné vyhodnocování
- Turing, 1951 – první šachový program (jen na papíře)
- Samuel, 1952–57 – strojové učení pro zpřesnění vyhodnocování
- McCarthy, 1956 – prořezávání pro možnost hlubšího prohledávání

řešení her je zajímavým předmětem studia $\leftarrow$ je obtížné:
průměrný faktor větvení v šachách $b = 35$
pro 50 tahů 2 hráčů ...
prohledávací strom $\approx 35^{100} \approx 10^{154}$ uzlů ($\approx 10^{40}$ stavů)

Hry a UI – aktuální výsledky

- Othello – od 1980 světoví šampioni odmítají hrát s počítači, protože stroje jsou příliš dobré. Othello (též Reversi) pro dva hráče na desce 8×8 – snaží se mezi své dva kameny uzavřít soupeřovy, které se přebarví. Až se zaplní deska, spočítají se kameny.
- dáma – 1994 program *Chinook* porazil světovou šampionku Marion Tinsley. Používá úplnou databázi tahů pro ≤ 8 figur (443 748 401 247 pozic).
- šachy – 1997 porazil stroj *Deep Blue* světového šampiona Gary Kasparova $3^1/_2 : 2^1/_2$. Stroj počítá 200 mil. pozic/s, sofistikované vyhodnocování a nezveřejněné metody pro prozkoumávání některých tahů až do hloubky 40 tahů. 2006 porazil program *Deep Fritz* na PC světového šampiona Vladimíra Kramnika 2:4. V současnosti vyhrávají turnaje i programy na slabším hardware mobilních telefonů s 20 tis. pozic/s.
- Go – do roku 2008 světoví šampioni odmítali hrát s počítači, protože stroje jsou příliš slabé. V Go je $b > 300$, takže počítače mohou používat téměř pouze znalostní bázi vzorových her.
  od 2009 – první programy dosahují pokročilejší amatérské úrovně (zejména na desce 9×9, nižší úroveň i na 19×19).

Typy her

	deterministické	*s náhodou*
perfektní znalosti	šachy, dáma, Go, Othello	backgammon, monopoly
nepřesné znalosti		bridge, poker, scrabble

Hledání optimálního tahu

2 hráči – MAX (△) a MIN (▽)

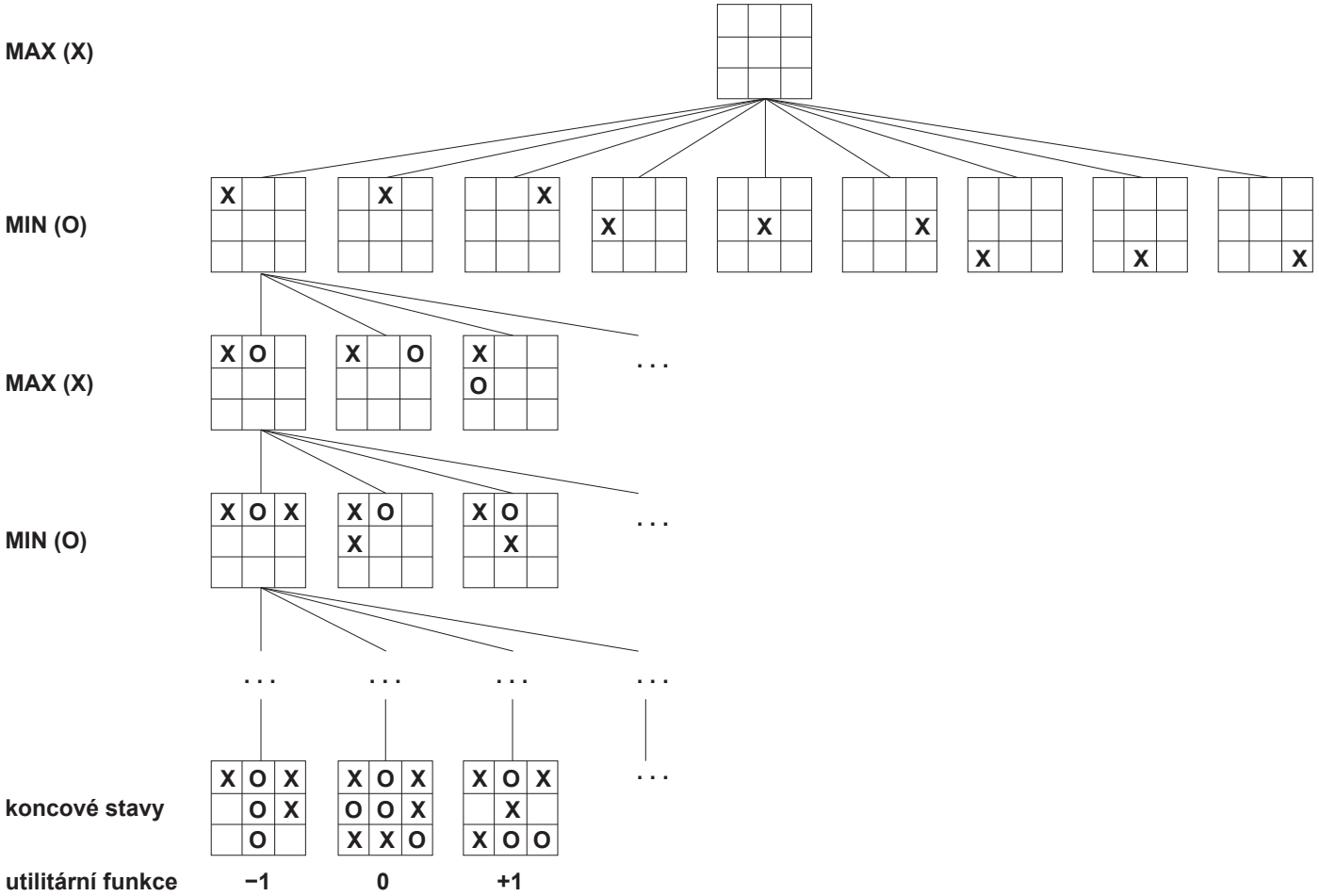
MAX je první na tahu a pak se střídají až do konce hry

hra = prohledávací problém:

- počáteční stav – počáteční herní situace + kdo je na tahu
- přechodová funkce – vrací dvojice (legální tah, výsledný stav)
- ukončovací podmínka – určuje, kdy hra končí, označuje koncové stavy
- utilitární funkce – numerické ohodnocení koncových stavů

Hledání optimálního tahu – pokrač.

počáteční stav a přechodová funkce definují herní strom:



Algoritmus Minimax

Hráč MAX (△) musí *prohledat* herní strom pro zjištění nejlepšího tahu proti hráči MIN (▽)
→ zjistit nejlepší hodnotu minimax – zajišťuje *nejlepší výsledek* proti *nejlepšímu protivníkovi*

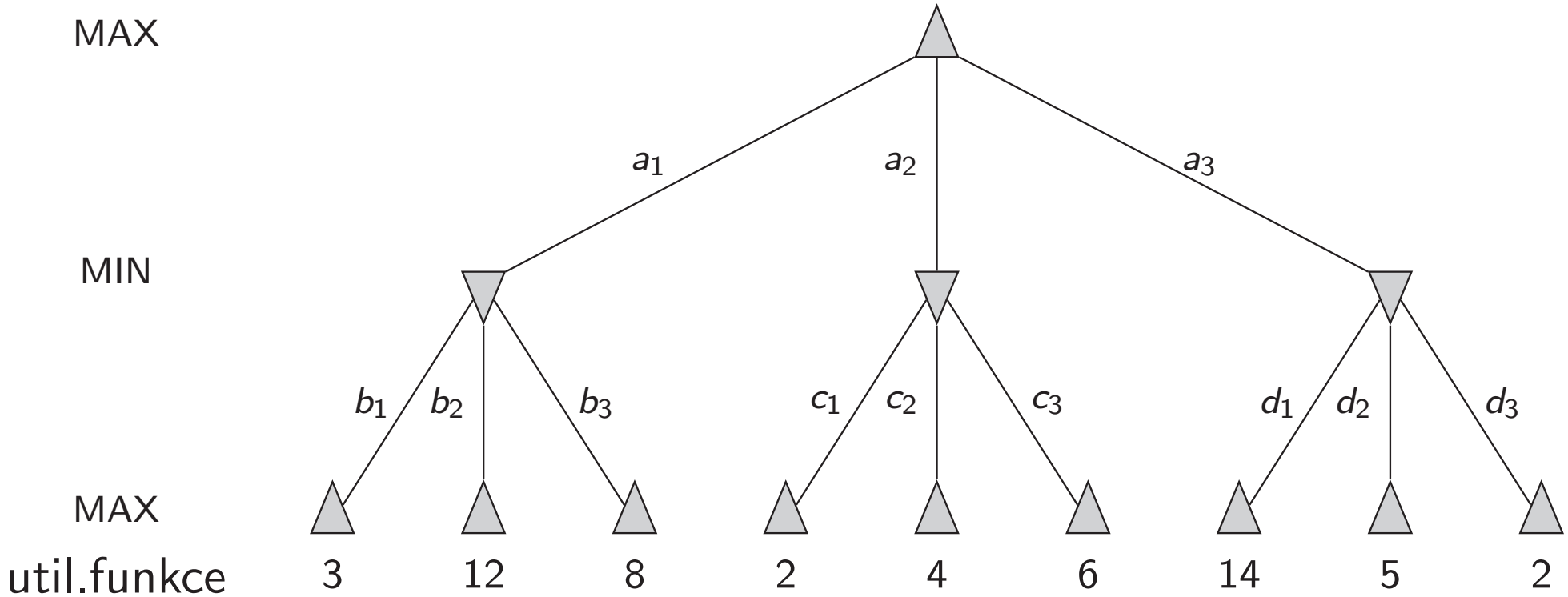
$$\text{Hodnota minimax}(n) = \begin{cases} \text{utility}(n), & \text{pro koncový stav } n \\ \max_{s \in \text{moves}(n)} \text{Hodnota minimax}(s), & \text{pro MAX uzel } n \\ \min_{s \in \text{moves}(n)} \text{Hodnota minimax}(s), & \text{pro MIN uzel } n \end{cases}$$

Algoritmus Minimax – pokrač.

příklad – hra jen na jedno kolo = 2 tahy (půlkola)

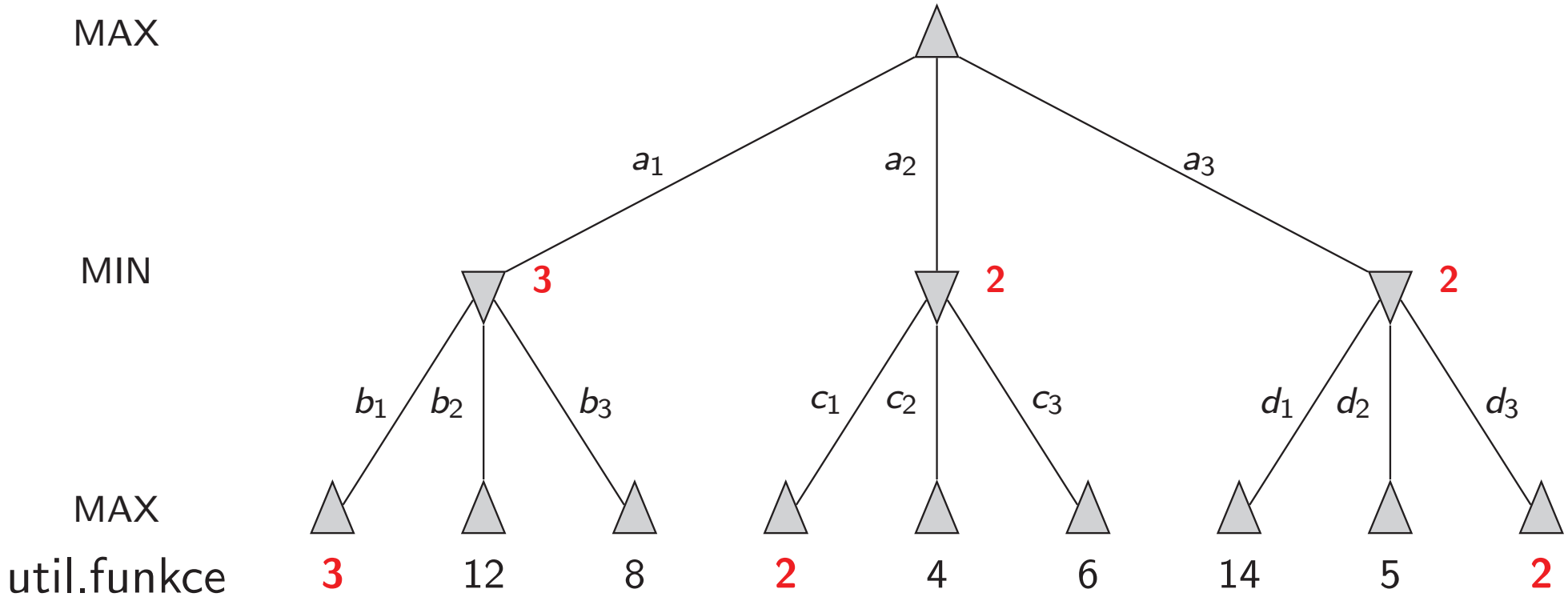
Algoritmus Minimax – pokrač.

příklad – hra jen na jedno kolo = 2 tahy (půlkola)



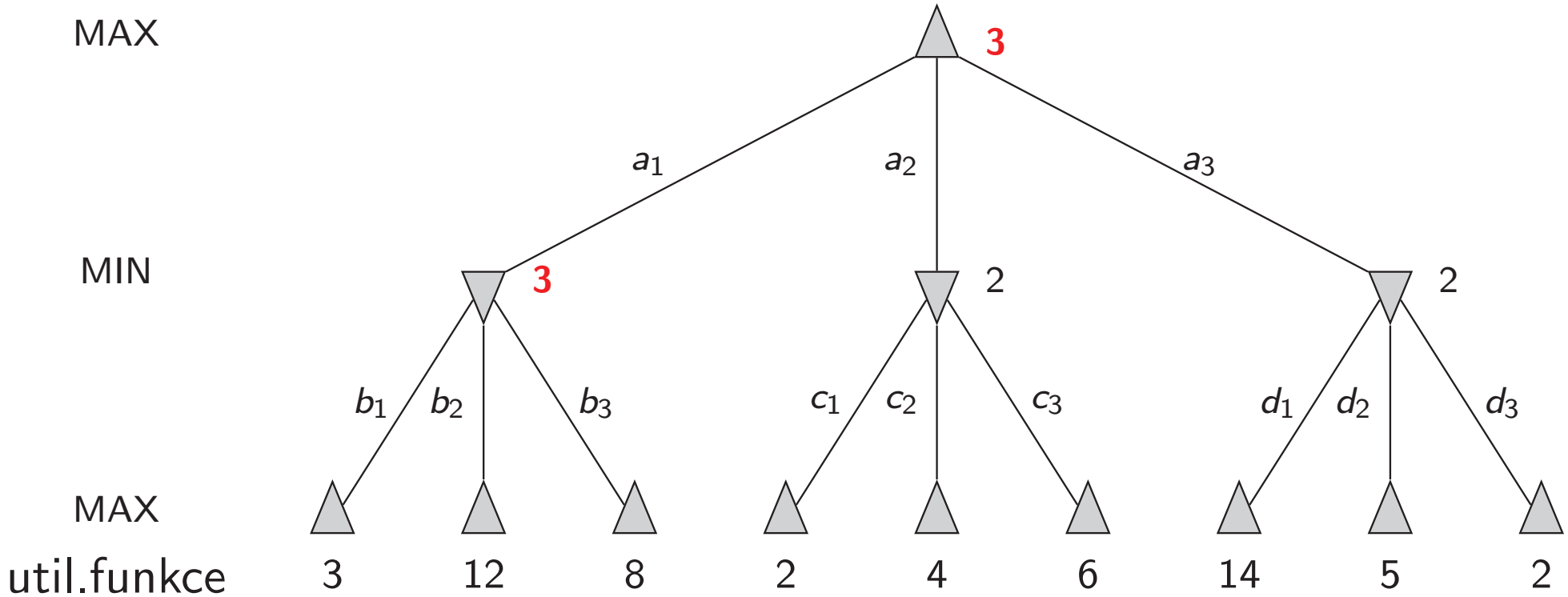
Algoritmus Minimax – pokrač.

příklad – hra jen na jedno kolo = 2 tahy (půlkola)



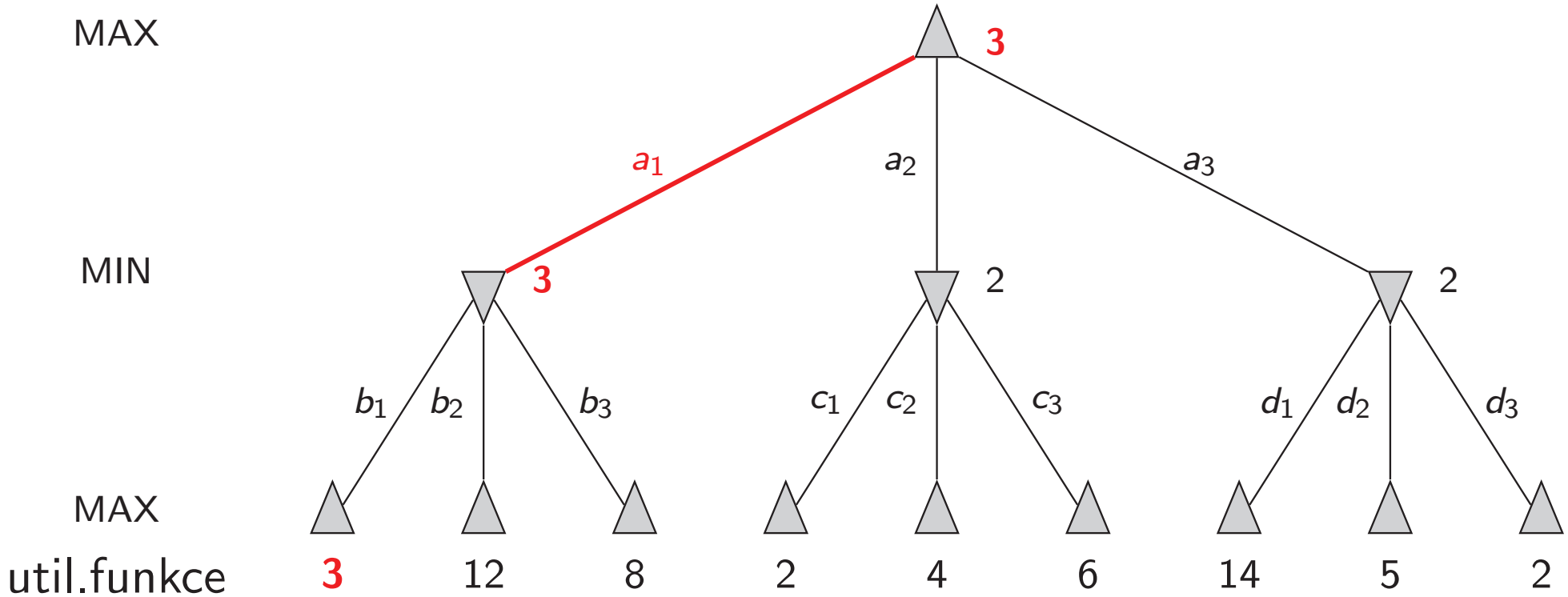
Algoritmus Minimax – pokrač.

příklad – hra jen na jedno kolo = 2 tahy (půlkola)



Algoritmus Minimax – pokrač.

příklad – hra jen na jedno kolo = 2 tahy (půlkola)



Algoritmus Minimax – vlastnosti

úplnost
optimálnost
časová složitost
prostorová složitost

Algoritmus Minimax – vlastnosti

úplnost	úplný pouze pro konečné stromy
optimálnost	
časová složitost	
prostorová složitost	

Algoritmus Minimax – vlastnosti

úplnost	úplný pouze pro konečné stromy
optimálnost	je optimální proti optimálnímu oponentovi
časová složitost	
prostorová složitost	

Algoritmus Minimax – vlastnosti

úplnost	úplný pouze pro konečné stromy
optimálnost	je optimální proti optimálnímu oponentovi
časová složitost	$O(b^m)$
prostorová složitost	

Algoritmus Minimax – vlastnosti

úplnost	úplný pouze pro konečné stromy
optimálnost	je optimální proti optimálnímu oponentovi
časová složitost	$O(b^m)$
prostorová složitost	$O(bm)$, prohledávání do hloubky

Algoritmus Minimax – vlastnosti

úplnost	úplný pouze pro konečné stromy
optimálnost	je optimální proti optimálnímu oponentovi
časová složitost	$O(b^m)$
prostorová složitost	$O(bm)$, prohledávání do hloubky

šachy ... $b \approx 35, m \approx 100 \Rightarrow$ přesné řešení není možné

např. $b^m = 10^6, b = 35 \Rightarrow m \approx 4$

4-tahy $\approx$ člověk-nováček

8-tahů $\approx$ člověk-mistr, typické PC

12-tahů $\approx$ Deep Blue, Kasparov

Časové omezení

předpokládejme, že máme 100 sekund + prozkoumáme 10^4 uzlů/s
$\Rightarrow$ 10^6 uzlů na 1 tah

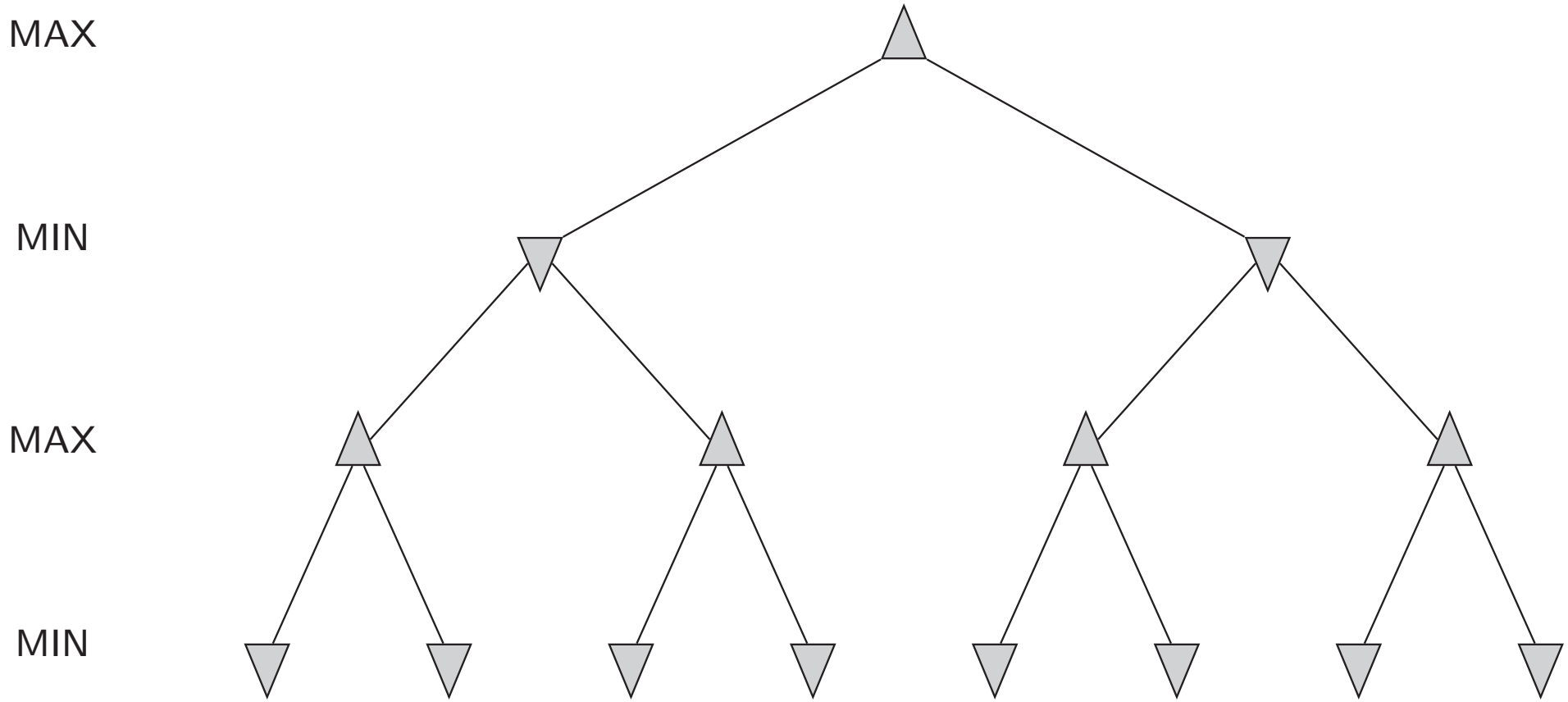
řešení **minimax_cutoff**:

- ohodnocovací funkce odhad přínosu pozice
  nahradí utilitární funkci
- ořezávací test (*cutoff test*) – např. hloubka nebo hodnota ohodnocovací funkce
  nahradí koncový test

Algoritmus Alfa-Beta prořezávání

Příklad stromu, který zpracuje predikát **minimax**

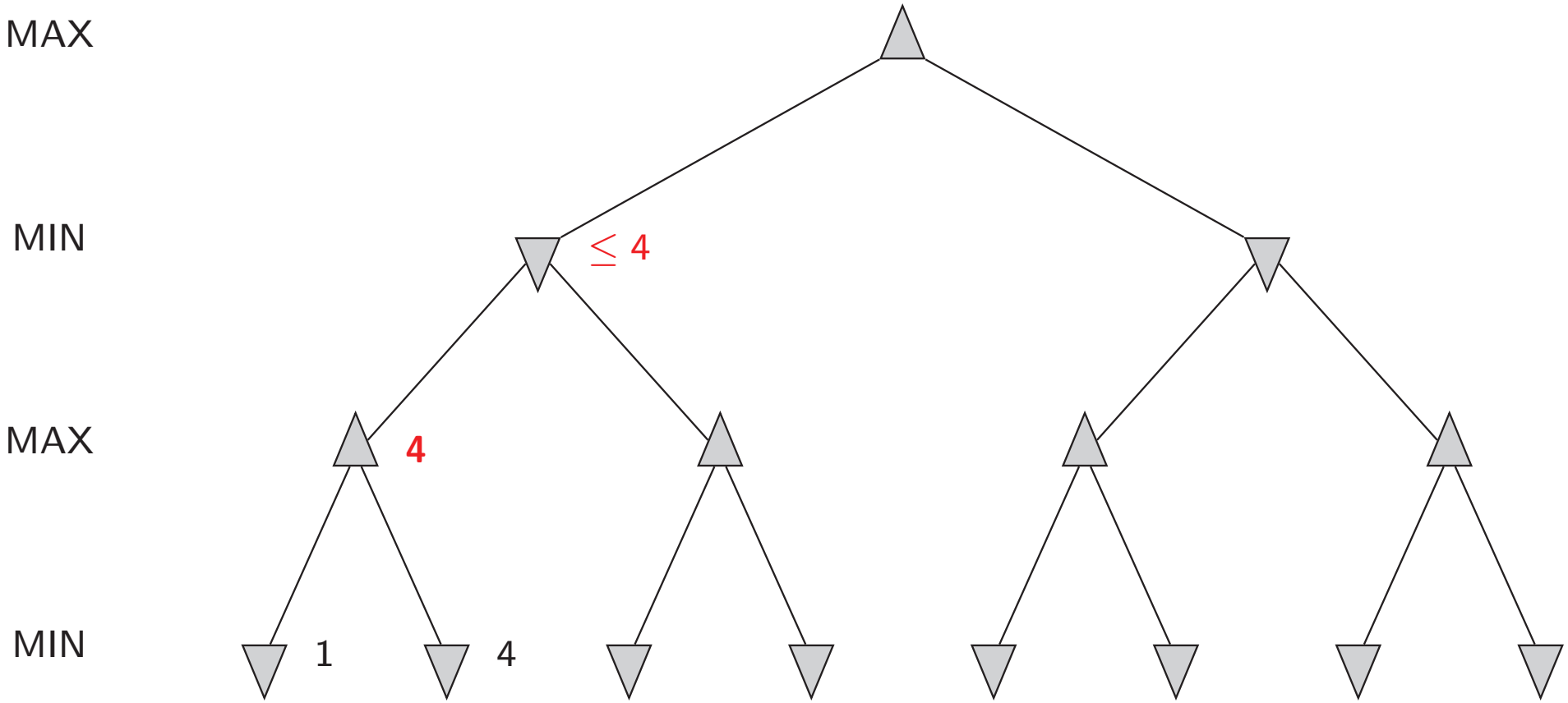
Alfa-Beta odřízne expanzi některý uzlů ⇒ Alfa-Beta procedura je efektivnější variantou minimaxu



Algoritmus Alfa-Beta prořezávání

Příklad stromu, který zpracuje predikát **minimax**

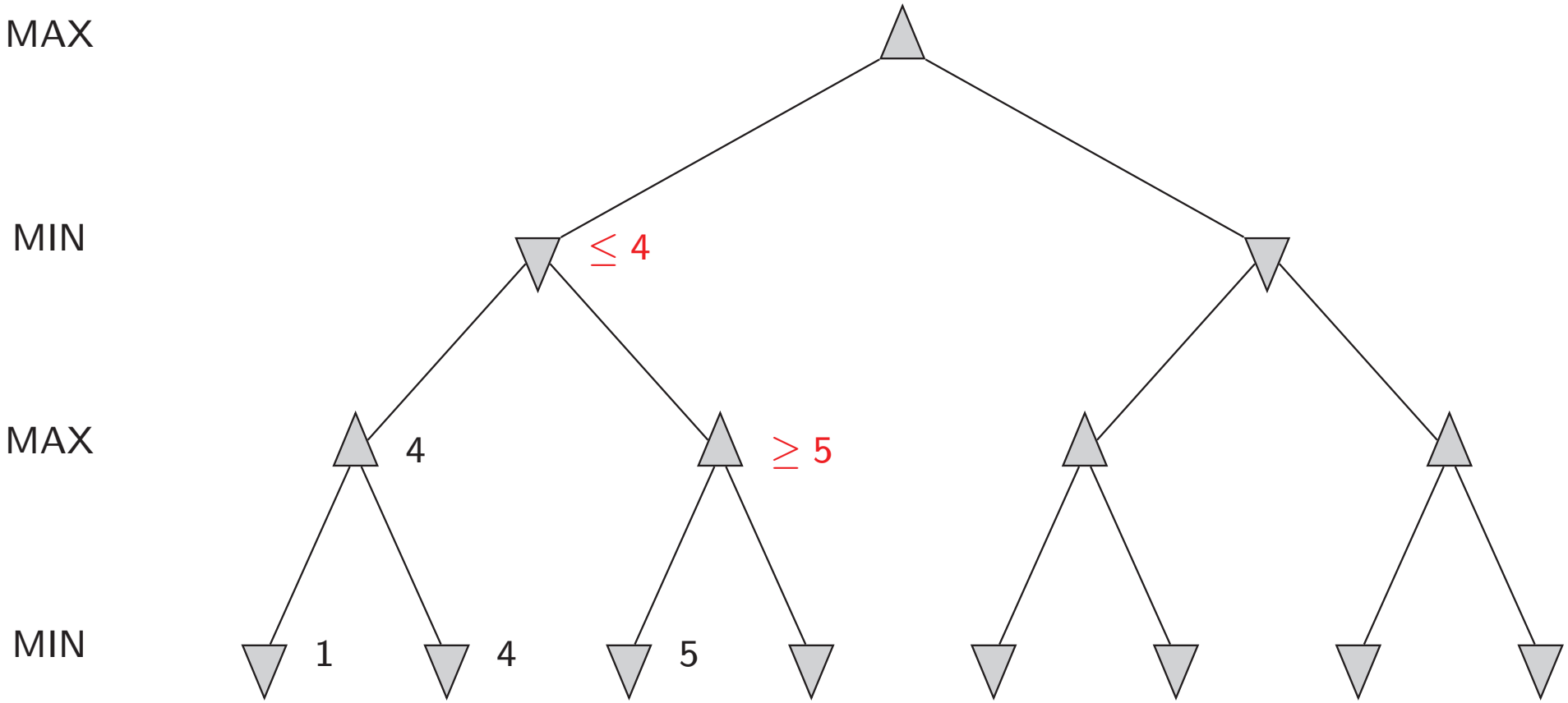
Alfa-Beta odřízne expanzi některý uzlů $\Rightarrow$ Alfa-Beta procedura je efektivnější variantou minimaxu



Algoritmus Alfa-Beta prořezávání

Příklad stromu, který zpracuje predikát **minimax**

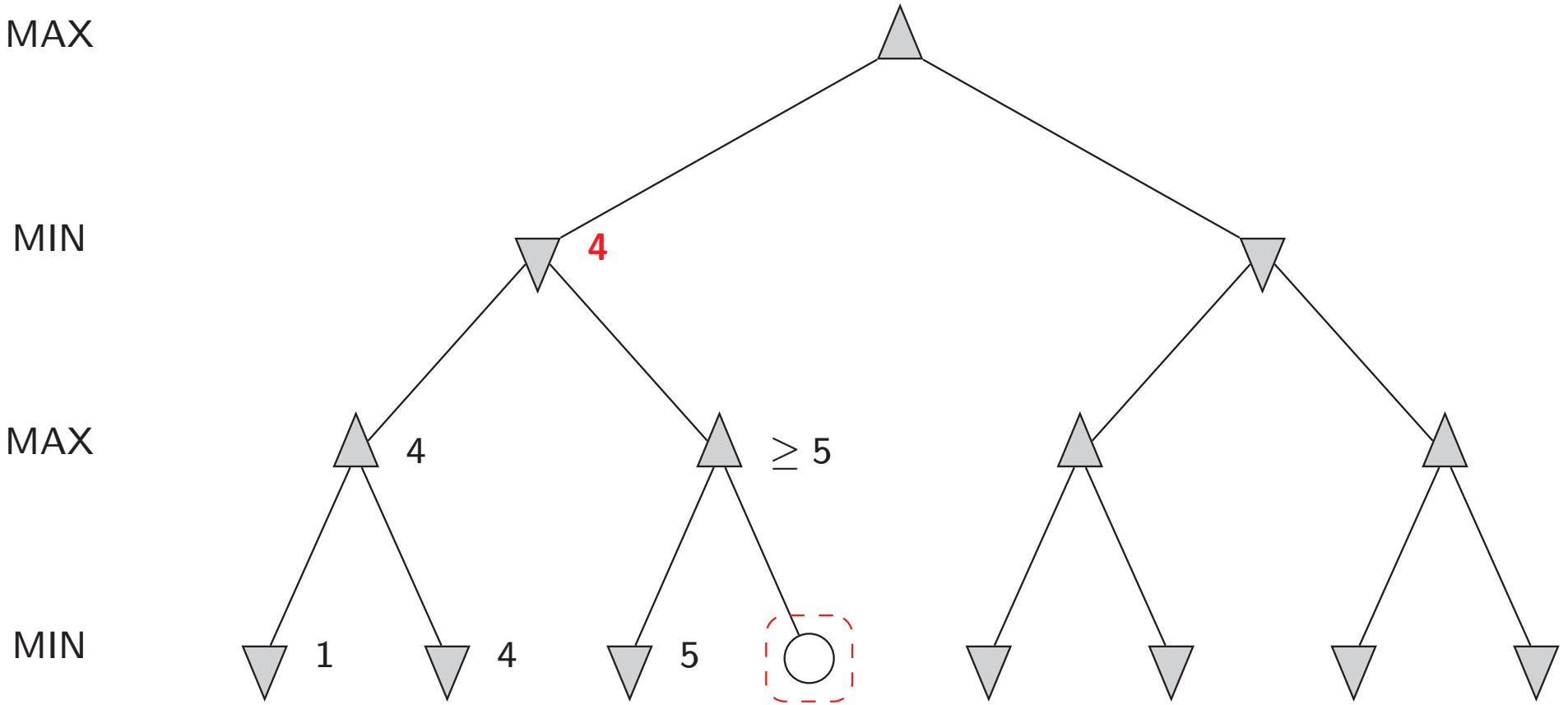
Alfa-Beta odřízne expanzi některý uzlů $\Rightarrow$ Alfa-Beta procedura je efektivnější variantou minimaxu



Algoritmus Alfa-Beta prořezávání

Příklad stromu, který zpracuje predikát **minimax**

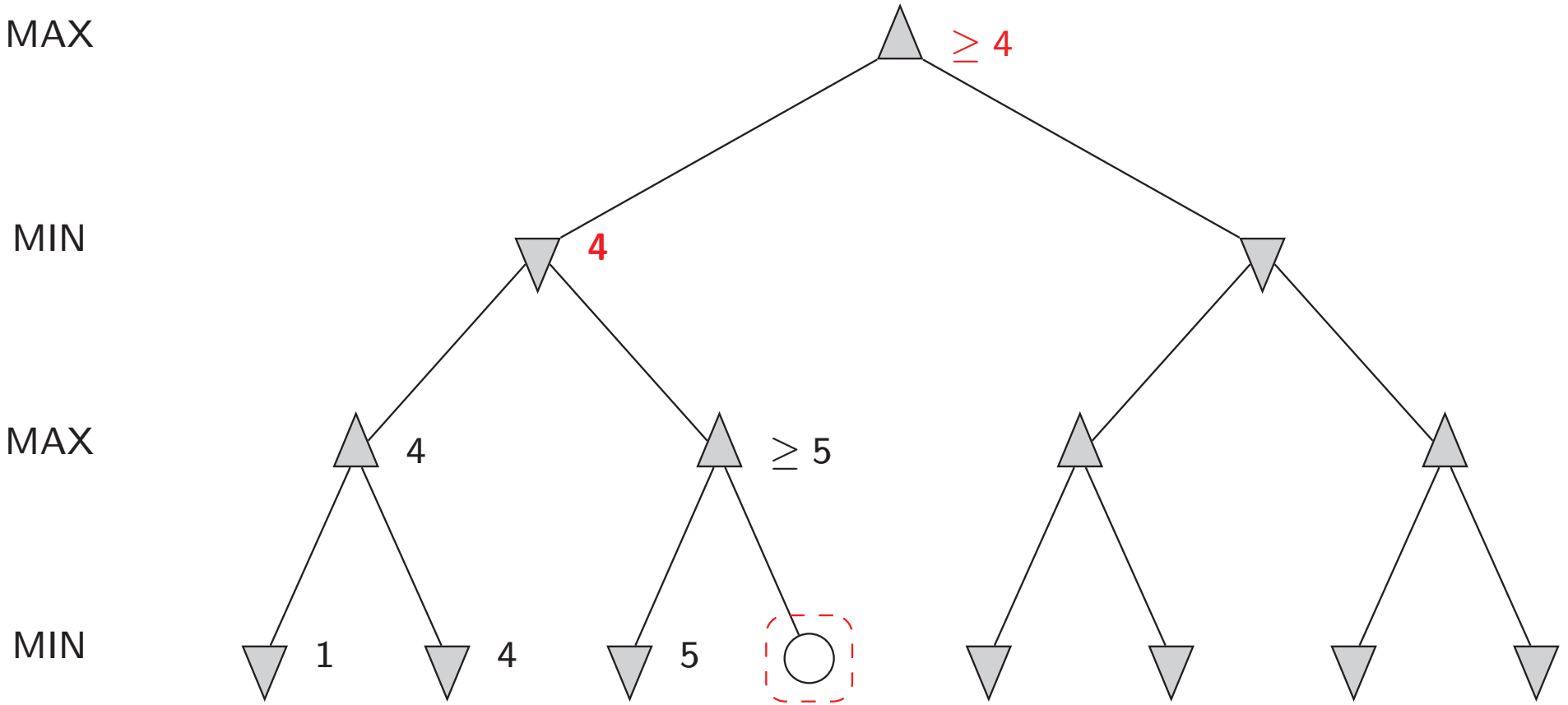
Alfa-Beta odřízne expanzi některý uzlů ⇒ Alfa-Beta procedura je efektivnější variantou minimaxu



Algoritmus Alfa-Beta prořezávání

Příklad stromu, který zpracuje predikát **minimax**

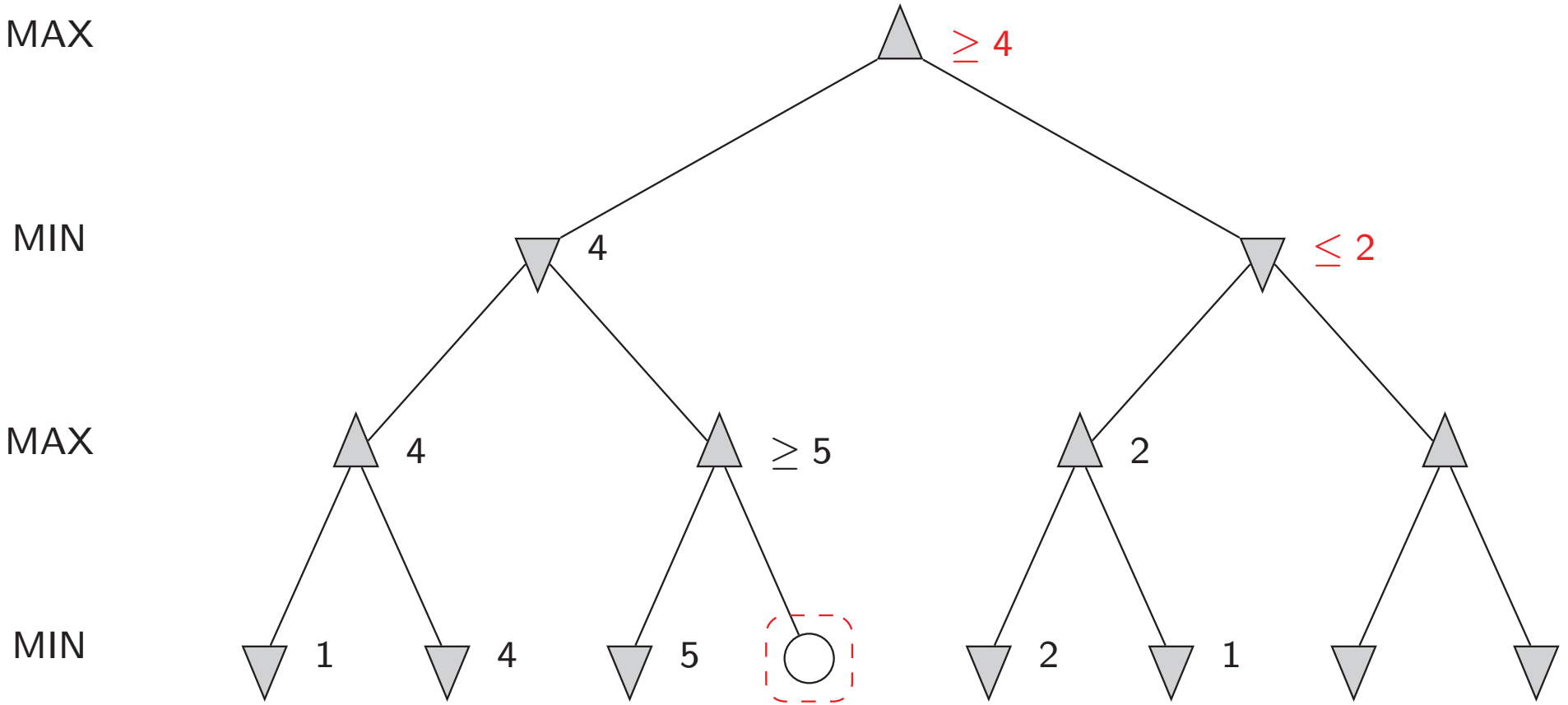
Alfa-Beta odřízne expanzi některý uzlů ⇒ Alfa-Beta procedura je efektivnější variantou minimaxu



Algoritmus Alfa-Beta prořezávání

Příklad stromu, který zpracuje predikát **minimax**

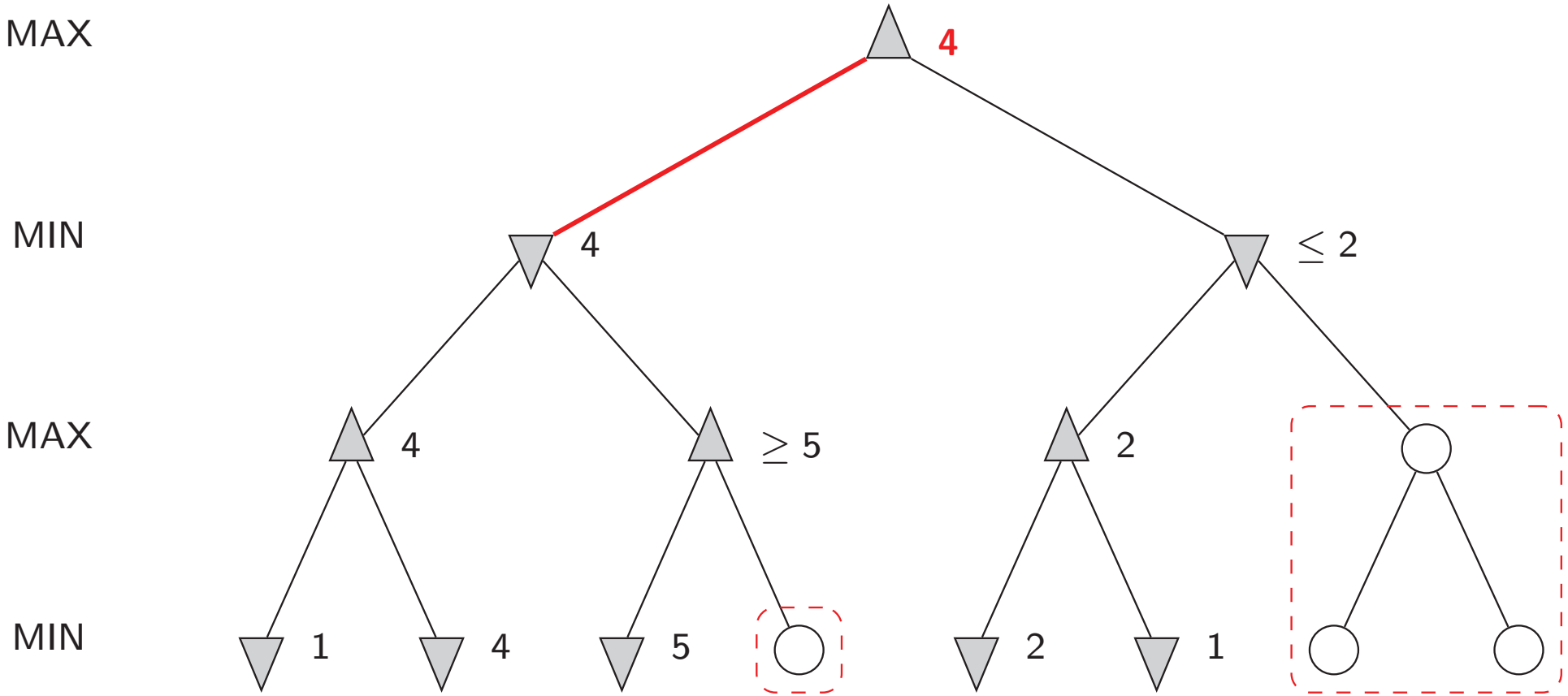
Alfa-Beta odřízne expanzi některý uzlů $\Rightarrow$ Alfa-Beta procedura je efektivnější variantou minimaxu



Algoritmus Alfa-Beta prořezávání

Příklad stromu, který zpracuje predikát **minimax**

Alfa-Beta odřízne expanzi některý uzlů ⇒ Alfa-Beta procedura je efektivnější variantou minimaxu



Algoritmus Alfa-Beta prořezávání – vlastnosti

- prořezávání neovlivní výsledek $\Rightarrow$ je stejný jako u minimaxu
- dobré uspořádání přechodů (možných tahů) ovlivní efektivitu prořezávání
- v případě "nejlepšího" uspořádání časová složitost$= O(b^{m/2})$
  $\Rightarrow$ zdvojí hloubku prohledávání
  $\Rightarrow$ může snadno dosáhnout hloubky 8 v šachu, což už je použitelná úroveň