

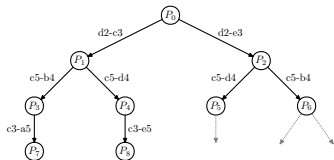
Algoritmus Minimax

Tomáš Kühn

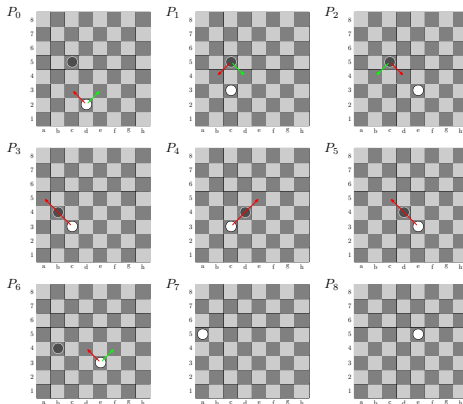
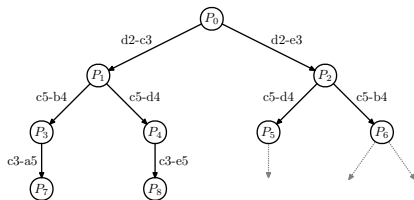
Projektový seminář 1

- **Tah** = přemístění figury hráče na tahu odpovídající pravidlům dané hry.
 - Při tahu může být manipulováno i s figurami soupeře, pokud to odpovídá pravidlům hry (např. odstranění přeskočené figury v dámě).
 - Tah se může skládat z několika dílčích pohybů, pokud to odpovídá pravidlům hry (např. vícenásobný skok v dámě).
 - Pozor, v některé literatuře “náš tah” označován jako *půltah*.
- **Pozice** = stav hry v určitém okamžiku.
 - Pozice je vesměs jednoznačně určena rozmístěním figur na desce a určením hráče na tahu.
 - Občas používáme i pojmy **vyhrávající pozice**, **prohrávající pozice**, **remízová pozice**, **koncová pozice** a **počáteční pozice**.

- Pro zobrazení “všech” možností, jak se může hra z dané pozice vyvíjet, používáme tzv. **herní strom**.
- Bohužel není v silách člověka ani počítače zobrazit či vzít do úvahy celý herní strom. Herní strom tedy běžně zobrazujeme pouze do určité předem dané hloubky (počet zkoumaných po sobě následujících tahů).
- Listy herního stromu nemusí být vždy ve stejné hloubce. Tato situace nastane, pokud v některé větvi nastal konec hry.



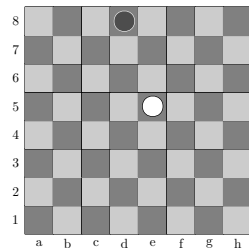
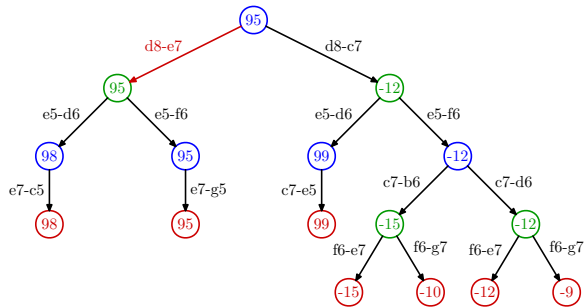
Příklad herního stromu



Princip algoritmu Minimax

- Algoritmus Minimax určuje nejlepší tah na základě prozkoumání herního stromu vycházejícího z aktuální pozice do předem dané hloubky.
- Minimax nejprve ohodnotí listové pozice pomocí heuristické ohodnocovací funkce.
- Ohodnocení pozic blíže ke kořeni herního stromu se pak určí jako
 - *maximum* z ohodnocení jeho následovníků, pokud je v dané pozici na tahu *aktuální hráč*
 - nebo jako *minimum* z ohodnocení následovníků, pokud je v dané pozici na tahu *soupeř*.
- U kořenové pozice nás pak nezajímá její ohodnocení, ale tah vedoucí k nejlépe ohodnocenému následovníkovi.

Ilustrace principu Minimaxu



Implementace algoritmu Minimax

- Algoritmus je realizován rekurzivní funkcí, která “prochází” herní strom do určité hloubky.
- Vstupem funkce je herní pozice a hloubka, do které se má herní strom dále prozkoumávat.
- Výstupem funkce je vypočtené ohodnocení dané pozice.
- Rozhodování z pohledu obou hráčů je realizováno totožným kódem. Využíváme zde toho, že ohodnocení dané pozice z pohledu prvního a druhého hráče se liší pouze znaménkem. Dále je nutné si uvědomit, že platí $\min(a, b) = -\max(-a, -b)$.
- Mezní podmínkou rekurze je dosažení požadované hloubky nebo koncové pozice.

Zjednodušený pseudokód

```
function minimax(pozice, hloubka)
if (pozice je koncová or hloubka = 0) then
    return heuristické ohodnocení pozice
else
    ohod  $\leftarrow -\infty$ 
    for all potomek pozice do
        ohod  $\leftarrow \mathbf{max}$ (ohod,  $-\mathbf{minimax}$ (potomek, hloubka - 1))
    end for
    return ohod
end if
end function
```


Detailní pseudokód (ošetření výhry/prohry)

```
function minimax(pozice, hloubka)
```

```
if je_prohra(pozice) then
```

```
    return -MAX
```

```
end if
```

```
if je_výhra(pozice) then
```

```
    return MAX
```

```
end if
```

```
if je_remíza(pozice) then
```

```
    return 0
```

```
end if
```

```
...
```

Detailní pseudokód (hlavní část)

...

if hloubka = 0 **then**

return ohodnocovaci_funkce(pozice)

else

 tahy \leftarrow generuj_tahy(pozice)

 ohod \leftarrow -MAX

for all tah *v kolekci* tahy **do**

 potomek \leftarrow zahraj(pozice, tah)

 ohod \leftarrow **max**(ohod, -minimax(potomek, hloubka - 1))

end for

...

return ohod

end if

end function

Detailní pseudokód (pozice blízké konci hry)

...

if ohod $>$ MNOHO **then**

ohod \leftarrow ohod $-$ 1

end if

if ohod $<$ $-$ MNOHO **then**

ohod \leftarrow ohod $+$ 1

end if

...

Nalezení nejlepšího tahu

function nej_tah(pozice, hloubka)

tahy \leftarrow generuj_tahy(pozice)

nejlepsi_ohodnoceni \leftarrow $-MAX$

for all tah *v kolekci* tahy **do**

potomek \leftarrow zahraj(pozice, tah)

ohodnoceni \leftarrow $-\text{minimax}(\text{potomek}, \text{hloubka} - 1)$

if ohodnoceni $>$ nejlepsi_ohodnoceni **then**

nejlepsi_ohodnoceni \leftarrow ohodnoceni

nejlepsi_tah \leftarrow tah

end if

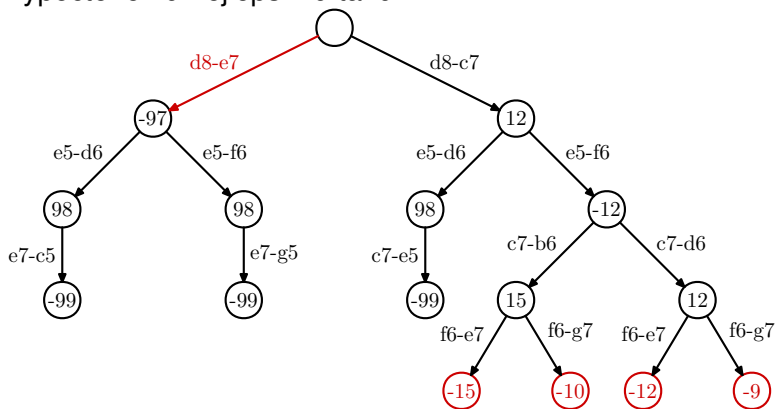
end for

return nejlepsi_tah

end function

Příklad

Při ohodnocování následujícího herního stromu byl použit algoritmus Minimax s hloubkou výpočtu 4 a konstantami $MAX = 99$ a $MNOHO = 90$. Červeně jsou zvýrazněny uzly, které byly ohodnoceny heuristicky, a hrana, které odpovídá vypočtenému nejlepšímu tahu.



- V algoritmu Minimax je potřeba pro danou herní situaci vytvořit kolekci všech legálních tahů, které se dají v této pozici zahrát.
- Tyto tahy jsou pak ve vzájemně jednoznačném vztahu s následovníky dané pozice v herním stromu.
- Při vytváření kolekce bývá dobré postupovat systematicky – procházet hrací desku, případně nějakou pomocnou kolekci figur
- a pro každou figuru vygenerovat všechny možné tahy.
- Algoritmus pro generování tahů je rozumné přizpůsobit pravidlům dané hry.

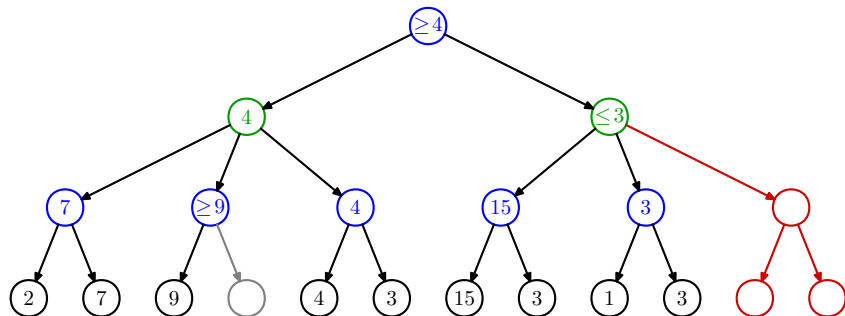
Ohodnocovací funkce

- Vstupem ohodnocovací funkce je ohodnocovaná pozice.
- Výstupem je celé číslo v intervalu $\langle -MNOHO, MNOHO \rangle$.
- Ohodnocovací funkci je rozumné vytvořit z pohledu jednoho hráče, ohodnocení z pohledu druhého hráče pak získáme změnou znaménka.
- Zcela vyrovnaná pozice má tedy ohodnocení rovno nule.
- Co lze hodnotit?
 - materiální složka (např. rozdíl v počtu figur hráčů)
 - statická poziční složka (bonusy a postihy za umístění figur na některá pole)
 - dynamická poziční složka (bloky figur, osamělé figury, ...)
- Ohodnocovací funkce by měla být rychlá a jednoduchá.
- Pomocí změn ohodnocení lze „donutit“ počítačového hráče k větší agresivitě, aktivitě, ochotě dělat výměny a podobně. Lze také vytvořit více ohod. funkcí pro různé fáze hry.

Princip Alfa-beta ořezávání

- V některých situacích nemusí Minimax zkoumat další herní pozice, protože je již zřejmé, že nebudou mít na volbu tahu vliv.
- Typy ořezávání:
 - alfa ořezávání – byla nalezena příliš malá hodnota, tuto větev hráč na tahu nezvolí,
 - beta ořezávání – nalezená hodnota je příliš velká, soupeř tuto větev nezvolí.
- V algoritmu použité hodnoty alfa tedy tvoří dolní mez, hodnoty beta pak horní mez při vyhledávání.
- Hodnoty alfa a beta se získají a upřesňují z ohodnocení dříve prozkoumaných pozic.
- Alfa-beta ořezávání je neúčinnější, pokud se nejprve zkoumají nejsilnější tahy. Někdy se používá heuristika pro seřazení tahů před zkoumáním následovníků dané pozice.

Příklad



Pomocné funkce pro algoritmus Alfa-beta

```
function dal(ohodnoceni)
  if ohodnoceni > MNOHO then
    return ohodnoceni + 1
  end if
  if ohodnoceni < -MNOHO then
    return ohodnoceni - 1
  end if
  return ohodnoceni
end function
```

```
function bliz(ohodnoceni)
  if ohodnoceni > MNOHO then
    return ohodnoceni - 1
  end if
  if ohodnoceni < -MNOHO then
    return ohodnoceni + 1
  end if
  return ohodnoceni
end function
```

Funkce Alfa-beta (část 1)

```
function alfabeta(pozice, hloubka, alfa, beta)
  if je_prohra(pozice) then
    return -MAX
  end if

  if je_vyhra(pozice) then
    return MAX
  end if

  if je_remiza(pozice) then
    return 0
  end if

  if hloubka = 0 then
    return ohodnocovaci_funkce(pozice)
  end if
```

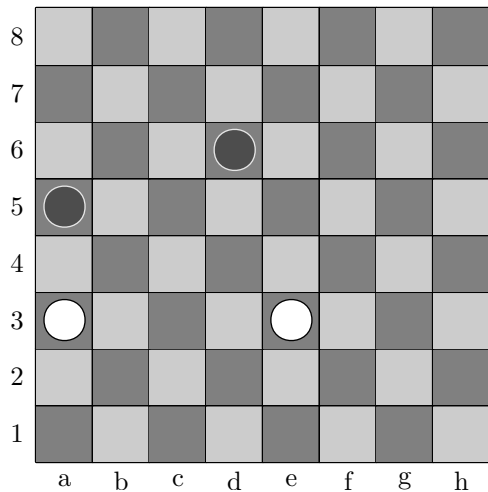
Funkce Alfa-beta (část 2)

```
tahy  $\leftarrow$  generuj_tahy(pozice)
for all tah v kolekci tahy do
  pot  $\leftarrow$  zahraj(pozice, tah)
  ohod  $\leftarrow$  -alfabeta(pot, hloubka - 1, dal(-beta), dal(-alfa))
  ohod  $\leftarrow$  bliz(ohod)
  if ohod > alfa then
    alfa  $\leftarrow$  ohod
    if ohod  $\geq$  beta then
      return beta
    end if
  end if
end for
return alfa
end function
```

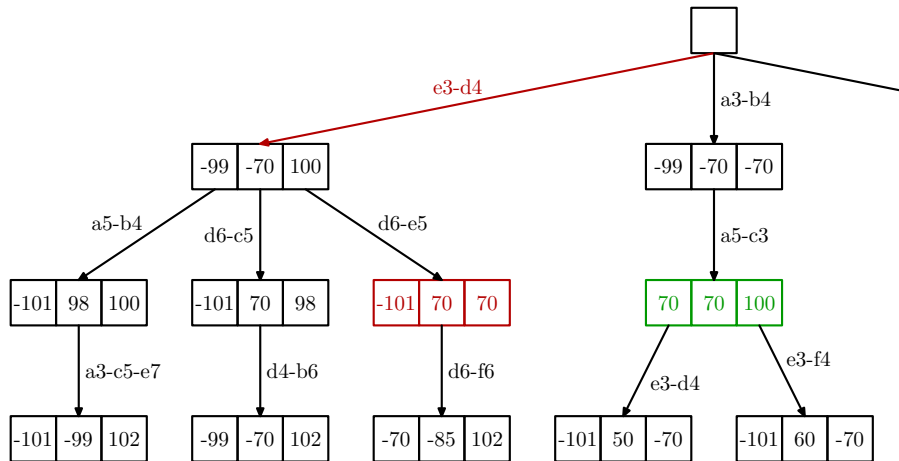
Zjištění nejlepšího tahu

```
function nej_tah(pozice, hloubka)
  tahy  $\leftarrow$  generuj_tahy(pozice)
  alfa  $\leftarrow$  -MAX
  for all tah v kolekci tahy do
    pot  $\leftarrow$  zahraj(pozice, tah)
    ohod  $\leftarrow$  -alfabeta(pot, hloubka - 1, -MAX, dal(-alfa))
    ohod  $\leftarrow$  bliz(ohod)
    if ohod > alfa then
      alfa  $\leftarrow$  ohod
      nejlepsi_tah  $\leftarrow$  tah
    end if
  end for
  return nejlepsi_tah
end function
```

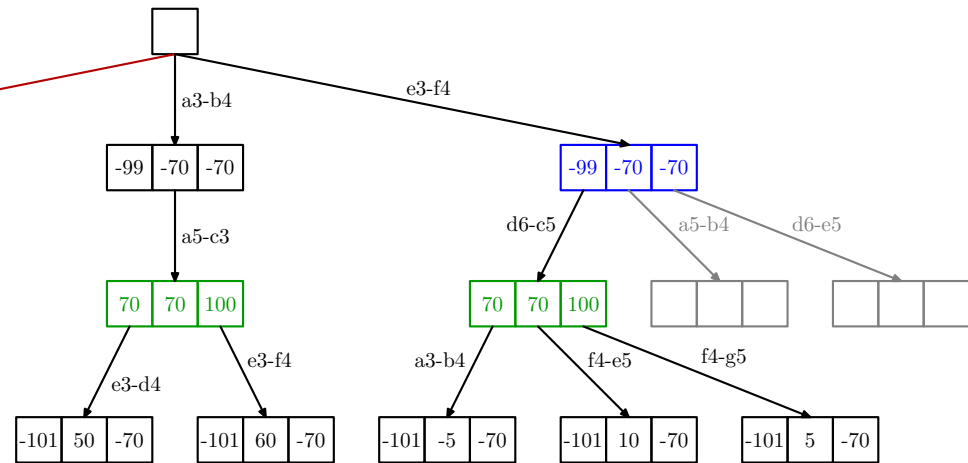
Příklad



Příklad



Příklad



- Dieter Steinwender, Frederic A. Friedel: *Šachy na PC*. Unis Publishing, Přerov, 1997.
- *Minimax (algoritmus) – Wikipedie, otevřená encyklopedie* [online], poslední revize 1. 9. 2010 (citováno 6. 9. 2010). Dostupné na adrese [http://cs.wikipedia.org/wiki/Minimax_\(algoritmus\)](http://cs.wikipedia.org/wiki/Minimax_(algoritmus)).
- *Alpha-beta pruning (anglicky, německy, česky) – Wikipedie, otevřená encyklopedie* [online], citováno 19. 10. 2010. Dostupné na adrese http://en.wikipedia.org/wiki/Alpha-beta_pruning.
- Jan Němec: Šachové myšlení. *Linux Software* [online], poslední revize 8. 3. 2006 (citováno 6. 9. 2010). Dostupné na adrese http://www.linuxsoft.cz/article.php?id_article=1109.