UNIVERZITA PALACKÉHO V OLOMOUCI
KATEDRA INFORMATIKY

Tomáš Kühr

ALGORITMY REALIZUJÍCÍ POČÍTAČOVÉHO HRÁČE

v jednoduchých deskových hrách

Říjen 2011

Abstrakt

Následující text obsahuje detailní popis algoritmu Minimax, který se používá při realizaci rozhodování počítačového hráče v jednoduchých deskových hrách, a jeho vylepšení – Alfa-Beta ořezávání, které značně snižuje časovou náročnost Minimaxu. Pro oba algoritmy zde uvádíme jejich podrobně okomentovaný pseudokód a několik konkrétních příkladů výpočtu nejlepšího tahu. U čtenářů se předpokládají pouze základní znalosti procedurálního paradigma programování (větvení, cykly, funkce), které jsou nutné pro porozumění některým částem tohoto textu.

Text vznikl na Katedře informatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci především pro potřeby studentů předmětů Projektový seminář 1 a 2. Autor může být kontaktován elektronickou poštou na adrese `<tomas.kuhr@upol.cz>`*. Pokud zjistíte jakékoli překlepy, chyby či jiné nedostatky, dejte mi prosím vědět.*

Obsah

1. Průběh hry a jeho zobrazení **1**
1.1. Základní pojmy 1
1.2. Česká dáma 1
1.3. Herní strom 2

2. Algoritmus Minimax **5**
2.1. Princip algoritmu 5
2.2. Implementace algoritmu 5
2.3. Ohodnocovací funkce 10
2.4. Generátor tahů 13

3. Alfa-beta ořezávání **15**
3.1. Princip algoritmu 15
3.2. Implementace algoritmu 16

Literatura **21**

Seznam obrázků

1. Základní postavení kamenů ve hře Česká dáma. 2
2. Příklad pozice s bílým hráčem na tahu, šipky naznačují možné tahy. 3
3. Ukázka jednoho z možných skoků bílé dámy. 3
4. Herní strom. 4
5. Pozice odpovídající uzlům herního stromu na obrázku 4. 4
6. Pozice z hry Česká dáma, černý na tahu. 6
7. Herní strom odpovídající pozici z obrázku 6. 6
8. Herní strom ilustrující výpočty zjednodušeného Minimaxu. 7
9. Herní strom ilustrující výpočty komplexního Minimaxu. 9
10. Pozice ze hry Česká dáma, na tahu je černý hráč 10
11. Herní strom ilustrující výpočet Minimaxu pro pozici z obrázku 10. 10
12. Statické "tabulky" pro ohodnocování umístění bílého pěšce (vlevo nahoře), černého pěšce (vpravo nahoře) bílé dámy (vlevo dole) a černé dámy (vpravo dole). 11
13. Ilustrace přidělování bonusů za umístění v blízkosti jiných figur. 12
14. Herní pozice pro ilustraci výpočtu ohodnocení. 12
15. Ilustrace beta-řezu. 15
16. Příklad alfa-ořezání. 16
17. Počáteční pozice k příkladu 3.2., na tahu je bílý hráč. 19
18. Herní strom z příkladu 3.2. 20

1. Průběh hry a jeho zobrazení

V této kapitole se blíže seznámíme se základními pojmy, které souvisí s problematikou deskových her a používají se při popisu algoritmů pro výpočet nejlepšího tahu počítačového hráče. Ukážeme si také způsob, jakým lze snadno zobrazit a popsat "všechny" možné herní situace. Toto zobrazení průběhu hry nám pak pomůže ilustrovat výběr nejlepšího tahu probíranými algoritmy. V této úvodní kapitole je pro úplnost uveden i souhrn pravidel hry Česká dáma, která bude v rámci tohoto textu použita pro ilustrace průběhů výpočtů popisovaných algoritmů.

1.1. Základní pojmy

Abychom mohli algoritmy uvedené v následujících kapitolách popsat stručně a přitom zcela jednoznačně, ujasníme si zde nejprve několik pojmů, které budeme později při popisování herních situací a jednotlivých kroků algoritmů používat. Zde je nutné také podotknout, že algoritmy popisované v tomto textu jsou použitelné pouze pro hry 2 hráčů, kteří spolu soupeří (výhra prvního hráče znamená automaticky prohru hráče druhého a naopak). Algoritmy dále vyžadují, aby se oba hráči při svých tazích pravidelně střídali.

Tahem bude v následujícím textu vždy myšleno přemístění figury patřící hráči, který je aktuálně na tahu. Toto přemístění musí samozřejmě odpovídat pravidlům dané hry. Pokud to pravidla dané hry vyžadují může být během tahu jednoho hráče manipulováno i s kameny hráče druhého (např. odstranění přeskočených soupeřových kamenů z desky). Tato "definice" tahu není bohužel jediná, se kterou se čtenáři mohou v literatuře popisující tuto problematiku setkat. V některých publikacích se námi "definovaný" tah označuje pojmem půltah, tahem se pak rozumí dvojice půltahů.

Pojmem **pozice** budeme dále rozumět stav hry v daném okamžiku. Pozice je tedy jednoznačně určena rozmístěním figur na hrací desce a určením hráče, který je právě na tahu. **Počáteční pozicí** rozumíme pozici před provedením prvního tahu. Pojmy **vyhrávající pozice**, **prohrávající pozice** a **remízová pozice** se pak používají k označení pozic na konci hry, obecně o nich také hovoříme jako o **koncových pozicích**. U vyhrávající a prohrávající pozice je pochopitelně nutné uvést, který hráč vyhrál či prohrál (např. "vyhrávající pozice z pohledu bílého hráče").

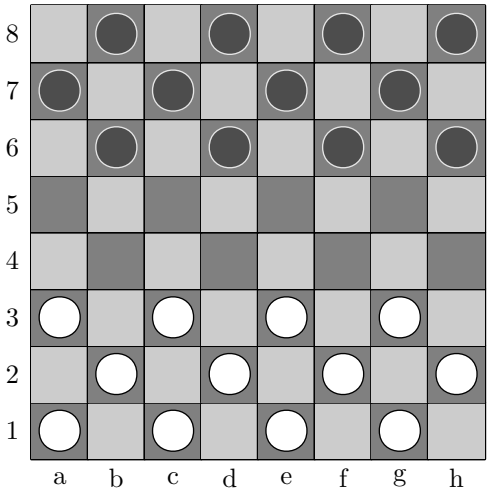
1.2. Česká dáma

Česká dáma je jednou z mnoha variant světoznámé hry Dáma. Jak již bylo v úvodu kapitoly zmíněno, budeme tuto hru v následujícím textu používat pro ilustrace herních situací a průběhu výpočtu popisovaných algoritmů. K tomuto účelu byla česká dáma zvolena především pro svá jednoduchá pravidla, svou rozšířenost a podobnost s většinou deskových, jejichž implementace bývá náplní projektových seminářů. Níže uvedená pravidla hry jsou převzata z webových stránek České federace dámy [1], kde je také možné snadno dohledat kompletní verzi pravidel hry Česká dáma.

Stručná pravidla české dámy

- Hraje se na desce s rozměrem 8×8 polí, každý hráč má na začátku hry 12 kamenů.
- Rozestavení kamenů na začátku hry ukazuje obrázek 1.
- Kameny se táhne po diagonálách, vždy o jedno pole vpřed.
- Dokončí-li kámen tah na poslední řadě, mění se v dámu. Dáma se pohybuje po diagonálách dopředu i dozadu.
- Kameny se skáče jen dopředu, dáma může skákat přes libovolný kámen na diagonále dopředu i dozadu a dokončit skok na kterémkoli poli za přeskočeným kamenem.

- Skákání je povinné a figura po dokončení tahu nesmí mít další možnost skoku. Je-li více možností skákání, může si hráč vybrat bez ohledu na množství přeskočených kamenů soupeře.
- Dáma má při skákání vždy přednost před kamenem.
- Partii vyhrává hráč, jehož soupeř nemůže provést tah podle pravidel (nemá již žádné figury nebo jsou jeho figury zablokovány).
- Partie skončí nerozhodně po dohodě hráčů nebo když se ve hře vyskytne 3-krát stejná pozice.



Obrázek 1. Základní postavení kamenů ve hře Česká dáma.

Zápis tahů

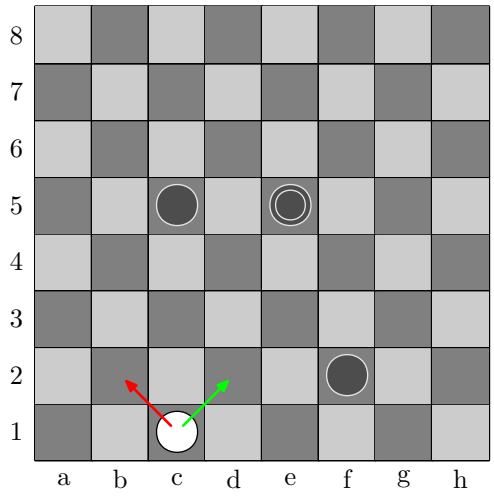
Při popisování průběhu hry narazíme na problém, jak jednoznačně a dostatečně stručně popsat prováděné tahy. U her probíhajících na čtvercové nebo obdélníkové desce se nejčastěji využívá počátečních písmen anglické abecedy k jednoznačnému označení sloupců a čísel k jednoznačné identifikaci řádku. Dvojice písmene a čísla pak jednoznačně určuje pole na hrací desce. Posloupnost "identifikátorů" polí pak opět jednoznačně popíše celý provedený tah.

▷ Příklad 1.1. V pozici na obrázku 2. je bílý kámen umístěn na poli **c1**, černé kameny na polích **c5** a **f2**, černá dáma je pak na poli **e5**. Bílý hráč z této pozice může provést tah **c1-b2** nebo **c1-d2**. □

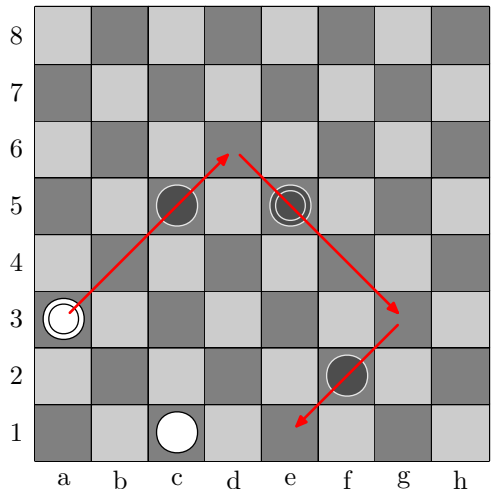
▷ Příklad 1.2. Herní pozice zaznamenaná na obrázku 3. se od té z obrázku 2. liší pouze umístěním bílé dámy na pole **a3**; na tahu zde je opět bílý hráč. Bílá dáma zde dle pravidel musí provést některý z možných skoků **a3-e7**, **a3-f8**, **a3-d6-f4**, **a3-d6-h2** nebo **a3-d6-g3-e1**. Poslední zmíněný skok je na obrázku naznačen červenými šipkami. □

1.3. Herní strom

V této části si ukážeme zobrazení "všech" možností, kterými se může z dané pozice hra ubírat, pomocí tzv. herního stromu. Slovo "všech" je zde záměrně uvedeno v uvozovkách, protože není v možnostech člověka ani počítače zobrazit, projít nebo vzít do úvahy všechny možné pozice, které se mohou vyskytnout v běžných deskových hrách, v rozumném čase. Herní stromy budeme v tomto textu používat pro ilustraci průběhů výpočtů probíraných algoritmů.



Obrázek 2. Příklad pozice s bílým hráčem na tahu, šipky naznačují možné tahy.

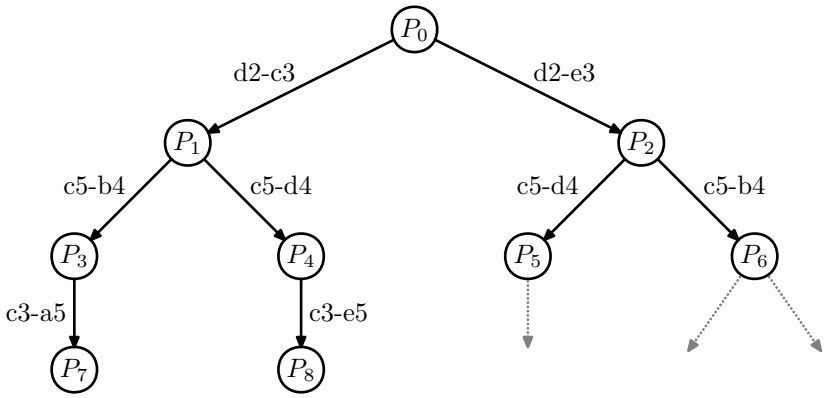


Obrázek 3. Ukázka jednoho z možných skoků bílé dámy.

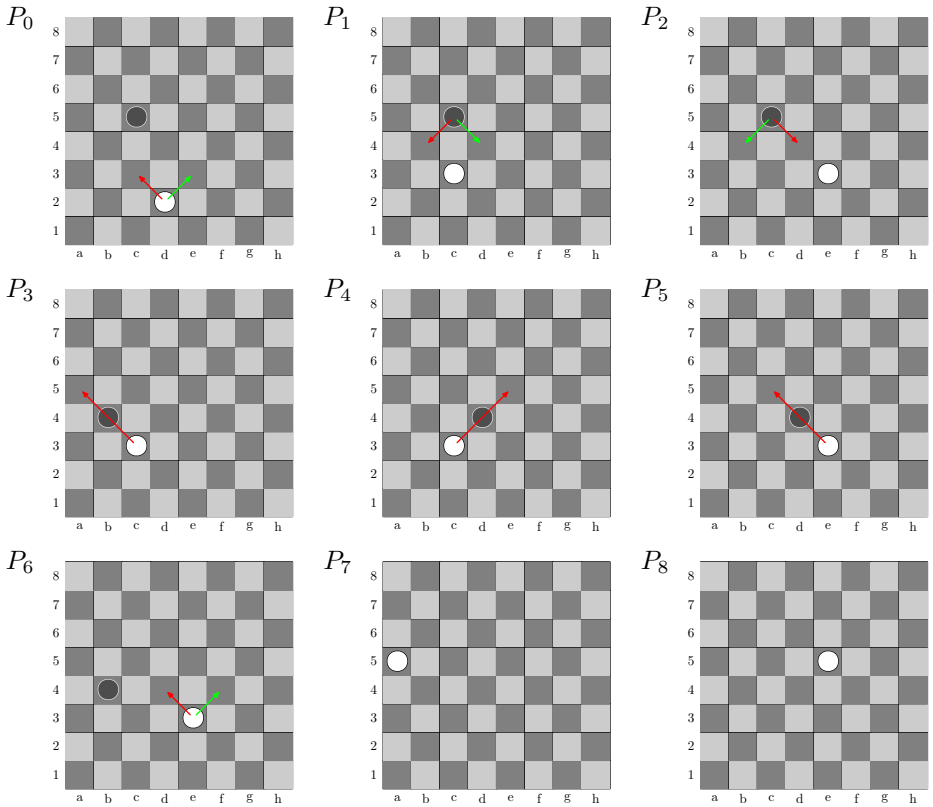
Konstrukce herního stromu je relativně jednoduchá. Kořenem stromu je vždy uzel odpovídající aktuální pozici. Pokud z této pozice existuje nějaký tah (v rámci pravidel dané hry), přidáme do stromu uzel odpovídající pozici vzniklé z aktuální pozice provedením tohoto tahu. Tyto dva uzly pak spojíme orientovanou hranou, přičemž bývá zvykem tuto hranu popsat tahem, který z pozice odpovídající počátečnímu uzlu vede k pozici odpovídající koncovému uzlu této hrany. Takto můžeme do stromu přidat uzly odpovídající všem tahům z aktuální pozice a poté rekurzivně zkoumat také tahy z pozic odpovídajících nově přidaným uzlům.

Z popisu konstrukce herního stromu je zřejmé, že počet uzlů stromu (a tedy i počet pozic, které mohou ve hře nastat) roste exponenciálně s jeho výškou (ta odpovídá počtu tahů, tedy délce hry). Vzhledem k tomu, že partii běžné deskové hry obvykle tvoří desítky až stovky tahů a že z každé pozice může existovat i větší množství tahů, je zobrazení či prozkoumání celého herního stromu prakticky nemožné.

▷ Příklad 1.3. Na obrázku 4. je zobrazena část herního stromu hry Česká dáma. Pozice odpovídající uzlům pak tvoří obrázek 5. □



Obrázek 4. Herní strom.



Obrázek 5. Pozice odpovídající uzlům herního stromu na obrázku 4.

2. Algoritmus Minimax

V této kapitole bude popsán základní algoritmus Minimax, který je v mnoha modifikacích využíván při hledání nejlepších tahů počítačových hráčů v nejrůznějších deskových hrách. Tento algoritmus je dobře popsán jak tištěnou literaturou (například [2]), tak internetovými zdroji [3, 4].

2.1. Princip algoritmu

Minimax je jednoduchý, ale velmi mocný algoritmus, který prozkoumává část herního stromu odpovídajícího aktuální pozici a nalezne "nejlepší" možný tah pro hráče, který je aktuálně na tahu. Jak bylo již naznačeno výše, nelze u běžných deskových her prozkoumávat celý herní strom (v rozumném čase). Minimax tedy postupuje od kořene stromu (aktuální pozice) do předem stanovené maximální **hloubky** výpočtu (odpovídá výšce prozkoumávané části herního stromu).

Minimax nejprve ohodnotí pozice odpovídající listům prozkoumávaného herního stromu pomocí heuristické ohodnocovací funkce. Prozatím předpokládejme, že tuto funkci již máme k dispozici. Ohodnocovací funkce je silně závislá na konkrétní deskové hře, o principech jejího vytváření pojednává 2.3. kapitola.

Pokud již máme ohodnocené pozice odpovídající listům stromu, vypočítáme z nich ohodnocení pozic odpovídající jejich rodičům. Tato část výpočtu algoritmu je založena na principu minimalizace možných ztrát. Algoritmus "předpokládá", že oba z hráčů budou hrát své "nejlepší" možné tahy. Ohodnocení pozice, kde je na tahu aktuální hráč (tj. hráč, který je na tahu také v pozici, z níž hledáme "nejlepší" tah), je tedy vypočteno jako maximum z ohodnocení všech jejích následovníků. Naopak ohodnocení pozice, kde je na tahu soupeř, se vypočítá jako minimum z ohodnocení následovníků této pozice, protože nejlepší tah soupeře je z pohledu aktuálního hráče ten nejhorší. Odtud vznikl i název algoritmu Minimax.

Postup popsaný v předcházejícím odstavci můžeme opakovat, dokud nejsou ohodnoceny všechny pozice odpovídající následovníkům kořene stromu, čili všechny pozice, do kterých se lze dostat z aktuální pozice provedením jednoho tahu. Z těchto pozic vybereme tu s maximálním ohodnocením a tah, kterým se hra může do této pozice dostat, pak vrátíme jako "nejlepší" možný tah.

▷ Příklad 2.1. Uvažujme pozici na obrázku 6. a nastavení hloubky algoritmu Minimax na hodnotu 4. Při výpočtu nejlepšího tahu z této pozice tedy bude postupně prozkoumáván herní strom až do této hloubky. Tuto část herního stromu ilustruje obrázek 7.

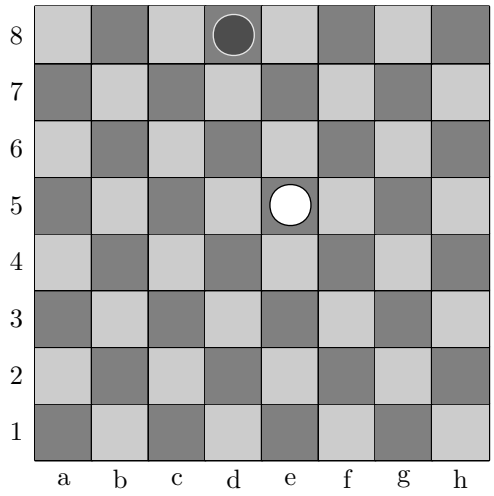
Listové pozice, které jsou na obrázku zvýrazněny červenou barvou, získají svá ohodnocení (zobrazena jako popisky uzlů) prostřednictvím volání heuristické ohodnocovací funkce (viz kapitola 2.3.). Ohodnocení ostatních uzlů pak je pak vypočítáno jako maximum, pokud je v dané pozici na tahu černý hráč (modré zvýraznění ve stromu), nebo jako minimum, pokud je na tahu bílý (zvýrazněno zeleně), z ohodnocení jejich přímých následovníků. Při tomto výpočtu postupujeme od listů směrem ke kořeni stromu.

U kořene stromu nás pak pochopitelně již tolik nezajímá jeho ohodnocení, ale tah, který vede k jeho nejvýše ohodnocenému následovníkovi. Tento tah, který je na obrázku 7. zvýrazněn červenou hranou, je vrácen Minimaxem jako nejlepší možný tah z aktuální pozice. □

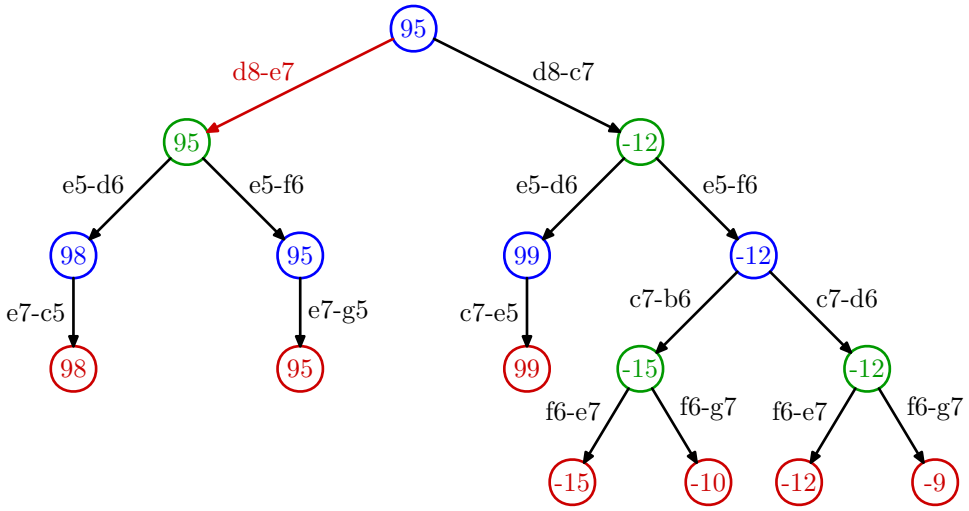
2.2. Implementace algoritmu

Přestože jsme si princip algoritmu Minimax ukázali na vytvořeném herním stromu (dané maximální hloubky s kořenem odpovídající aktuální pozici), nebudeme při jeho implementaci herní strom potřebovat ani sestrojovat. Algoritmus Minimax totiž řeší prozkoumávání pozic elegantně jednoduchou rekurzivní funkcí, jejíž zjednodušenou verzi popisuje algoritmus 2.1.

Rekurzivní funkce *minimax* popsaná v algoritmu 2.1. má dva vstupní parametry – aktuální pozici a hloubku, do které se má herní strom prozkoumávat. Podmínka na 2. řádku je mezní podmínkou rekurze, výpočet zde se "zastaví" ve chvíli, kdy předaná pozice odpovídá konci hry (čili již nelze provádět žádné tahy) nebo když se posloupnost rekurzivních volání funkce dostane



Obrázek 6. Pozice z hry Česká dáma, černý na tahu.



Obrázek 7. Herní strom odpovídající pozici z obrázku 6.

do požadované hloubky herního stromu. Ohodnocování na řádku 3 je vždy prováděno z pohledu hráče, který je v dané pozici na tahu. Následuje stěžejní část algoritmu realizující rekurzivní průchod herního stromu do hloubky. Proměnná *ohodnocení* je na 5. řádku inicializována na nejmenší možnou hodnotu, což je běžně používaný postup při výpočtu maxima z většího množství čísel. Poté jsou vygenerovány veškeré pozice, do kterých se může hra prostřednictvím jednoho tahu z předané pozice dostat. Zde se implicitně předpokládá existence nějakého generátoru možných tahů, ze kterých pak vytvoříme potomky dané pozice. Generátor tahů bude podrobněji popsán v kapitole 2.4. V cyklu začínajícím na řádku 6 jsou pak dle očekávání pomocí rekurzivního volání funkce *minimax* prozkoumáváni potomci dané pozice.

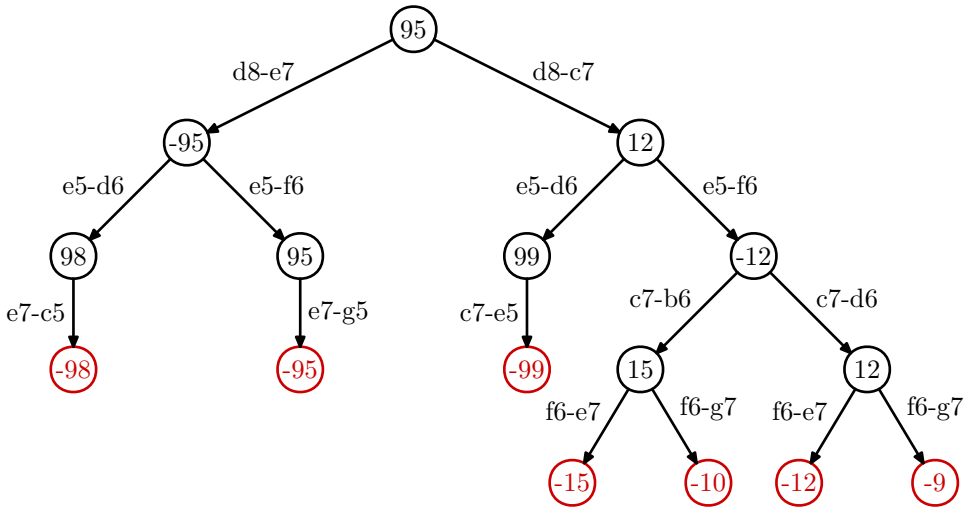
Řádek 7 si ovšem zaslouží podrobnější vysvětlení. Pozorný čtenář si již jistě všiml, že kód algoritmu obsahuje pouze funkci pro výpočet maxima — funkce pro výpočet minima se v něm nikde nevyskytuje, což odporuje tomu, co bylo uváděno výše. Tento rozpor je však pouze zdánlivý. Algoritmus zde využívá fakt, že ohodnocovací funkce je navržena tak, aby hodnota 0 odpovídala zcela vyrovnané pozici, kladná ohodnocení pak odpovídají pozicím výhodnějším pro hráče na tahu, záporná ohodnocení pak nevýhodným pozicím z pohledu hráče na tahu. Změnou znaménka ohodnocení tedy můžeme jednoduše přejít z pohledu jednoho hráče do pohledu druhého hráče. Pokud si

dále uvědomíme, že platí rovnost $\min(a, b) = -\max(-a, -b)$, zjistíme, že výpočet této rekurzivní funkce odpovídá výše uvedenému principu algoritmu s tím rozdílem, že ohodnocení pozic je nyní vypočítáno z pohledu hráče, který je v těchto pozicích na tahu a ne z pohledu hráče, který byl na tahu v kořenové pozici.

Algoritmus 2.1. *Zjednodušený algoritmus Minimax (pseudokód)*

```
1:  function minimax(pozice, hloubka)
2:  if (pozice je koncová or hloubka = 0) then
3:      return heuristické ohodnocení pozice
4:  else
5:      ohodnocení ← −∞
6:      for all potomek pozice do
7:          ohodnocení ← max(ohodnocení, −minimax(potomek, hloubka − 1))
8:      end for
9:      return ohodnocení
10: end if
11: end function
```

▷ Příklad 2.2. Na obrázku 8. je znázorněn herní strom ohodnocený zjednodušenou verzí rekurzivní funkcí *minimax*. Červeně zvýrazněné uzly byly ohodnoceny heuristicky, u ostatních uzlů bylo jejich ohodnocení vypočteno jako maximum z opačných čísel k ohodnocením přímých potomků daného uzlu. Pokud tedy jsou $o_1, o_2, \ldots$ ohodnocení potomků, vypočítáme ohodnocení dané pozice jako $max(-o_1, -o_2, \ldots)$. □



Obrázek 8. Herní strom ilustrující výpočty zjednodušeného Minimaxu.

Představená zjednodušená verze Minimaxu zatím neřeší výběr nejlepšího tahu v kořenové pozici. Dále by tato verze měla problém s ohodnocováním pozic, které vedou ke konci hry. Jednoduše řečeno, Minimax zatím nerozlišuje mezi výhrou (resp. prohrou) prvním tahem a výhrou (resp. prohrou) desátým tahem, což by mohlo vést k situaci, kdy počítačový hráč i na první pohled vyhranou partii zbytečně prodlužuje.

Tyto problémy již řeší následující podrobněji rozpracovaný algoritmus Minimax. Jeho rekurzivní část je zde uvedena jako algoritmus 2.2. Část řešící výběr tahu v kořeni herního stromu pak jako algoritmus 2.3.

V této verzi algoritmu Minimax je také explicitně rozepsáno ohodnocování koncových herních pozic. Pozici, ve které daný hráč prohrál, přiřadíme ohodnocení, které je menší než ohodnocení jakékoli neprohrávající pozice. Symetricky, přiřadíme vyhrávající pozici, z pohledu vítězného

hráče, maximální možné ohodnocení. Ohodnocení výhry a prohry by se vždy měla lišit pouze znaménkem, v našem pseudokódu používáme číselné konstanty MAX a −MAX. Pokud v dané pozici hra skončila remízou, je ohodnocení této pozice rovno nule. U velkého množství deskových her nemůže podle pravidel dojít k vítězství hráče po soupeřově tahu, řádky 5, 6 a 7 je tedy možné většinou vynechat.

Heuristická funkce přiřazující pozici ohodnocení z pohledu hráče na tahu je tak volána pouze v případě nekoncové *pozice* a nulové hodnoty parametru *hloubka* (viz řádky 11 a 12). Na řádcích 14 až 19 se toho oproti zjednodušené variantě příliš nezměnilo, pouze jsme vytváření potomků dané pozice rozdělili do několika příkazů (generování tahů, zahrání tahu), což více odpovídá praktickému řešení tohoto dílčího problému.

Řádky 20 až 25 algoritmu 2.2. upravují ohodnocení pozice takovým způsobem, aby algoritmus preferoval výhru za menší počet tahu před vzdálenější výhrou a naopak vzdálenější prohru před rychlejší prohrou. Podmínky na těchto řádcích využívají konstantu MNOHO pro rozlišení ohodnocení běžných pozic od ohodnocení, která přísluší vyhrávajícím či prohrávajícím pozicím v horizontu několika tahů. Při předávání ohodnocení vyhrávající či prohrávající pozice od listu směrem ke kořeni stromu se pak absolutní hodnota tohoto ohodnocení s každým patrem stromu zmenšuje o 1, což vede k výše popsanému žádoucímu chování algoritmu.

Algoritmus 2.2. *Kompletní pseudokód algoritmu Minimax*

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

2:  if je_prohra(pozice) then
3:     return −MAX
4:  end if

5:  if je_výhra(pozice) then
6:     return MAX
7:  end if

8:  if je_remíza(pozice) then
9:     return 0
10: end if

11: if hloubka = 0 then
12:    return ohodnocovaci_funkce(pozice)
13: else
14:    tahy ← generuj_tahy(pozice)
15:    ohodnoceni ← −MAX

16:    for all tah v kolekci tahy do
17:       potomek ← zahraj(pozice, tah)
18:       ohodnoceni ← max(ohodnoceni, −minimax(potomek, hloubka − 1))
19:    end for

20:    if ohodnoceni > MNOHO then
21:       ohodnoceni ← ohodnoceni − 1
22:    end if

23:    if ohodnoceni < −MNOHO then
24:       ohodnoceni ← ohodnoceni + 1
25:    end if

26:    return ohodnoceni
27: end if

28: end function
```

Jak již bylo uvedeno výše, u pozice, která tvoří kořen stromu nás nezajímá ani tak její ohodnocení jako tah, který vede k nejlépe ohodnocenému potomkovi, což je nejlepší možný tah z dané pozice. Toto rozhodování zajišťuje funkce *nej_tah* popsaná algoritmem 2.3., která je volána s pozicí, z níž potřebujeme zjistit nejlepší možný tah, a hloubkou, do které chceme nechat minimax prohledávat herní strom.

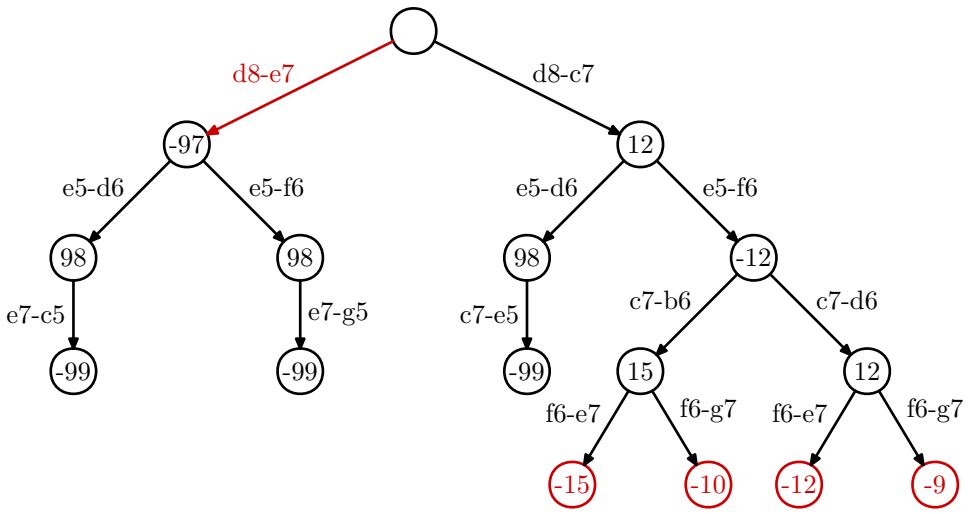
Od funkce *minimax* se tato funkce liší prakticky jen tím, že si kromě nejlepšího nalezeného ohodnocení (uloženého v proměnné *nejlepsi_ohodnoceni*) ukládá (do proměnné *nejlepsi_tah*) také tah, který k tomuto ohodnocení vedl. Nejlepší nalezený tah funkce také vrací jako svou návratovou hodnotu.

U kořene herního stromu není většinou potřeba testovat, zda není předaná pozice koncová, ani zda nebyla zadaná nulová hloubka. Také úpravy ohodnocení odpovídajících výhře či prohře zde nemají žádný význam.

Algoritmus 2.3. *Pseudokód funkce pro zjištění nejlepšího tahu*

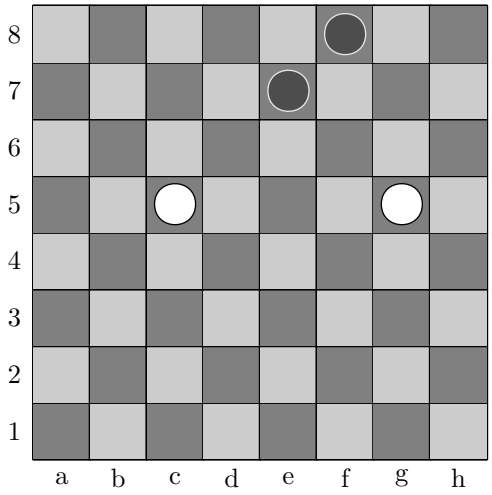
```
1:  function nej_tah(pozice, hloubka)
2:  tahy ← generuj_tahy(pozice)
3:  nejlepsi_ohodnoceni ← −MAX
4:  for all tah v kolekci tahy do
5:     potomek ← zahraj(pozice, tah)
6:     ohodnoceni ← −minimax(potomek, hloubka − 1)
7:     if ohodnoceni > nejlepsi_ohodnoceni then
8:        nejlepsi_ohodnoceni ← ohodnoceni
9:        nejlepsi_tah ← tah
10:    end if
11: end for
12: return nejlepsi_tah
13: end function
```

▷ Příklad 2.3. Na obrázku 9. je ukázáno ohodnocení herního stromu odpovídajícímu herní pozici z obrázku 6. Při ohodnocování 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. □

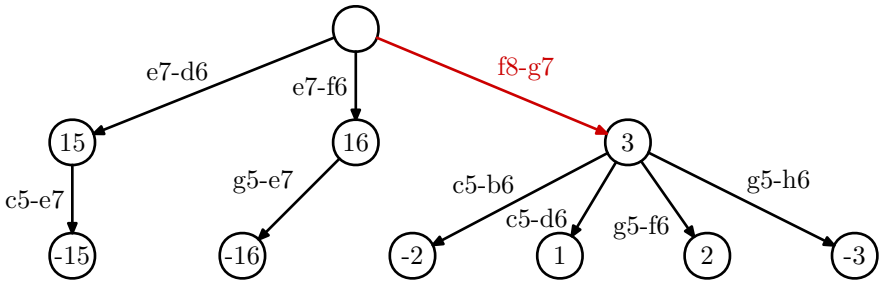


Obrázek 9. Herní strom ilustrující výpočty komplexního Minimaxu.

▷ Příklad 2.4. Prohlédněte si pozici z obrázku 10. a herní strom odpovídající výpočtu Minimaxu do hloubky 2 pro tuto pozici, který je znázorněn na obrázku 11. Nemusíte být přeborníci ve hře Česká dáma, abyste poznali, že Minimax nezvolil právě nejlepší tah. Největší slabinou Minimaxu, která se zde projevila, je jeho omezené "vidění" do předem určené hloubky. V některých situacích, kdy dochází k větším změnám v ohodnocení pozic (např. uprostřed výměny kamenů), není vhodné, aby Minimax ukončil rekurzi. Z tohoto důvodu se v praxi často setkáváme s modifikacemi algoritmu Minimax, které výměnu kamenů vždy dokončí, přestože se již ocitly v "nulové hloubce" a měly by tedy prohledávání stromu ukončit. □



Obrázek 10. Pozice ze hry Česká dáma, na tahu je černý hráč

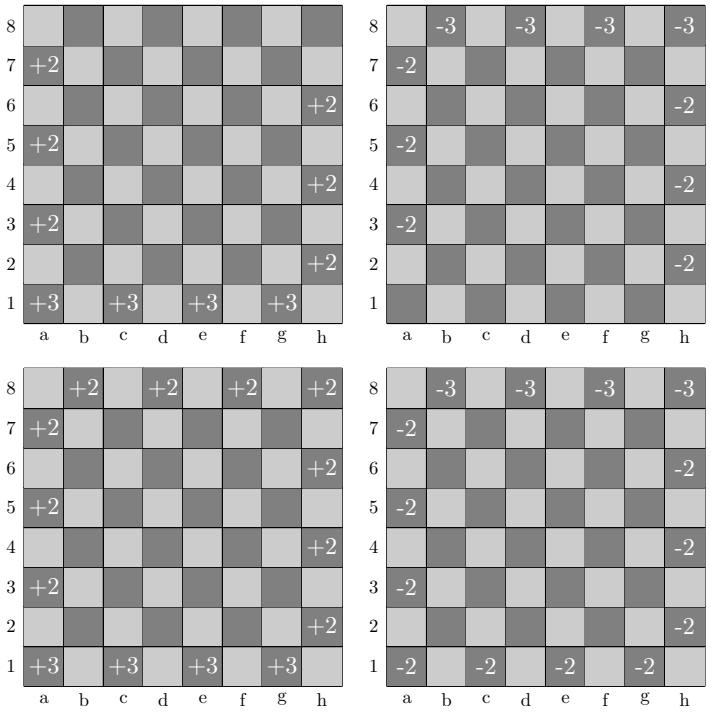


Obrázek 11. Herní strom ilustrující výpočet Minimaxu pro pozici z obrázku 10.

2.3. Ohodnocovací funkce

Nyní se podívejme na detaily týkající se návrhu a implementace ohodnocovací funkce, která byla použita v kódu algoritmu 2.2. Jak již bylo naznačeno výše, jedná se o funkci, jejímž vstupním parametrem je nekoncová herní pozice a výstupem je celočíselné ohodnocení dané pozice z pohledu hráče na tahu. Toto ohodnocení je vždy z intervalu $\langle -\text{MNOHO}, \text{MNOHO}\rangle$; ohodnocení, jejichž absolutní hodnota je větší než MNOHO jsou vyčleněna pro vyhrávající a prohrávající pozice (viz popis algoritmu 2.2. v kapitole 2.2.). Vždy by také mělo být splněno, že ohodnocení jedné pozice z pohledu prvního a druhého hráče se liší pouze znaménkem. V praxi si tedy vystačíme s ohodnocovací funkcí pro jednoho z hráčů, ohodnocení z pohledu druhého hráče získáme změnou znaménka. Funkce by dále měla být pokud možno jednoduchá a rychlá, vesměs je mnohem účinnější použít jednoduché ohodnocení a prozkoumávat herní strom do větší hloubky než nasadit složitou ohodnocovací heuristiku.

Ohodnocovací funkce patří samozřejmě mezi ty části algoritmu, jejichž výpočet je silně závislý na konkrétní deskové hře. Přesto se nyní pokusíme zformulovat několik obecných zásad, kterých se bývá dobré držet při návrhu ohodnocovacích funkcí většiny jednoduchých deskových her. Pokud je daná hra založena na zajímání kamenů soupeře, měly by být počty mých a soupeřových kamenů na desce primárním kritériem při ohodnocování pozice. Dále lze hodnotit umístění jednotlivých kamenů na desce. Toto většinou řešíme statickou tabulkou bonusů a postihů pro jednotlivé typy figur (např. dáma, pěšec). Bonus či postih se jednoduše započítá, pokud daná figura stojí na daném poli. Případně lze také přidávat bonusy a postihy za různá seskupení figur na desce, osamělé figury a podobně. Tyto výpočty už ale mohou být komplikovanější a mohou tak negativně ovlivnit rychlost ohodnocovací funkce.

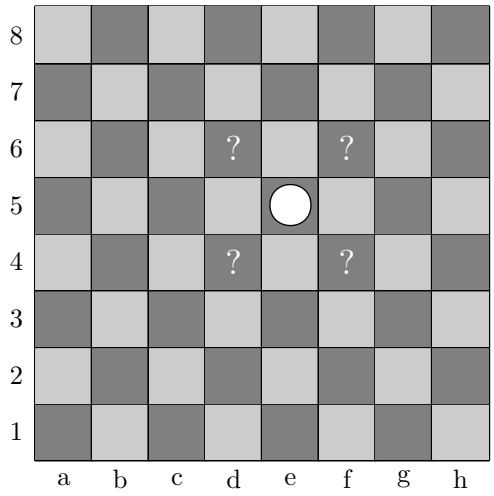


Obrázek 12. Statické "tabulky" pro ohodnocování umístění bílého pěšce (vlevo nahoře), černého pěšce (vpravo nahoře) bílé dámy (vlevo dole) a černé dámy (vpravo dole).

▷ Příklad 2.5. Zkusme si nyní navrhnout ohodnocovací funkci pro hru Česká dáma. Jak již bylo řečeno výše, vystačíme si s funkcí pro hodnocení pozice z pohledu bílého hráče. Ohodnocení z pohledu černého pak získáme změnou znaménka. Níže uvedená ohodnocovací funkce je inspirována bakalářskou prací Ladislava Vitáska [5].

Základem každého hodnocení herní pozice je ohodnocení materiálu. Zde je třeba určit hodnotu, kterou budou mít jednotlivé druhy figur. Pro potřeby tohoto příkladu budeme za každého bílého pěšce na desce přičítat k aktuálnímu ohodnocení 25 bodů, za každou bílou dámu pak 100 bodů. Symetricky za každého černého pěšce odečteme 25 a za každou černou dámu 100 bodů.

K ohodnocení materiálních složek pak přidáme bonusy za umístění figur. Toto je možné realizovat například přidáním bonusů za umístění kamenů na konkrétních polích. Pokud si uvědomíme, že na polích prvního a osmého sloupce a první a osmé řady nelze kámen nikdy přeskočit, můžeme figurám na těchto polích přidat za jejich umístění bonus v hodnotě 2 bodů. Bílé figury v první řadě, stejně jako černé figury v osmé řadě, navíc brání soupeřovým pěšcům v proměně na dámu. Těmto polím tedy přidáme další jednobodový bonus. Bonusy jednotlivých polí pro jednotlivé typy figur ukazuje obrázek 12.

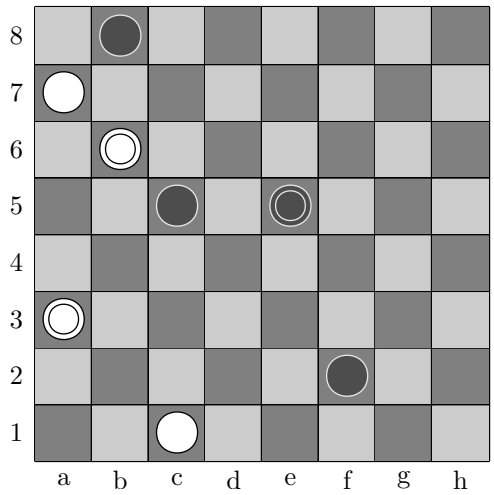


Obrázek 13. Ilustrace přidělování bonusů za umístění v blízkosti jiných figur.

Bonusy za umístění je možné vypočítávat také principiálně zcela jiným způsobem. Můžeme například pro každou figuru určit počet vlastních figur na sousedních polích (viz obrázek 13.) a za každou zjištěnou sousední figuru přidat k ohodnocení jeden bod. Samozřejmě je možné oba principy libovolně kombinovat. Je ovšem potřeba dávat pozor na to, aby jedna věc nebyla do konečného ohodnocení započítána vícekrát! □

▷ Příklad 2.6. Prostudujme si pozici na obrázku 14. a pokusme se ji ohodnotit funkcí zkonstruovanou v příkladu 2.5. Nejprve vypočítejme materiální složku ohodnocení. Bílý má na desce 2 pěšce a 2 dámy, černý pak 3 pěšce a dámu, čili po zvážení materiálu je pozice ohodnocena $2 \times 25 + 2 \times 100 - 3 \times 25 - 100 = 75$ bodů.

Nyní určíme také poziční část ohodnocení. Bílý pěšec na poli **a7** získá navíc 2 body za umístění na poli a 1 bod za sousední bílou dámu. Bílý pěšec na **c1** obdrží 3 body za umístění v první řadě. Bílá dáma na **b6** získá 1 bod za sousedního pěšce, bílá dáma na **a3** pak 2 body za umístění na okraji desky. Celkově tedy bílé figury přidají k ohodnocení 9 bodů. Z černých figur získává poziční bonus pouze pěšec na poli **b8** a to konkrétně 3 body. Poziční složka ohodnocení tedy činí $9 - 3 = 6$ bodů, celkové ohodnocení pozice pak $75 + 6 = 81$ bodů. □



Obrázek 14. Herní pozice pro ilustraci výpočtu ohodnocení.

2.4. Generátor tahů

Poslední zatím podrobně neokomentovanou součástí algoritmu 2.2. je generátor tahů, kterému bude věnovaná tato kapitola. Vstupem generátoru je herní pozice, výstup pak tvoří kolekce všech tahů, které lze z dané pozice podle pravidel dané hry provést. Při vytváření této kolekce se pochopitelně vyplatí postupovat systematicky.

U hry Česká dáma (a mnohých dalších) například pravidla přikazují provést skok, kdykoli je to možné. Je tedy nanejvýš rozumné vygenerovat nejprve všechny možné skoky a až pokud se zjistí, že žádný legální skok neexistuje, generovat ostatní tahy. Při samotném generování tahů obvykle systematicky procházíme desku (případně nějaké pomocné kolekce figur) a pro každou figuru hráče na tahu opět systematicky hledáme všechny legální tahy.

3. Alfa-beta ořezávání

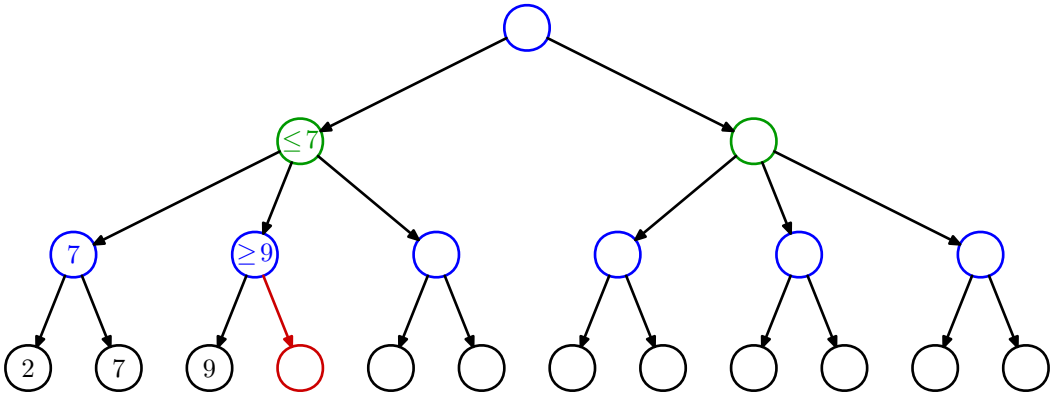
V této části textu bude ukázáno a okomentováno jedno z nejčastěji využívaných vylepšeních algoritmu Minimax. Níže prezentovaný algoritmus dosahuje stejných výsledků jako samotný Minimax. Na druhou stranu je možné použitím tohoto algoritmu značně snížit časovou náročnost výpočtu a tím umožnit výpočet do větší hloubky. Princip Alfa-beta ořezávání spočívá v neprocházení těch částí herního stromu, které již zjevně neovlivní právě vypočítávaný nejlepší tah počítačového hráče. Algoritmus Alfa-beta ořezávání je dobře popsán v mnoha publikacích [2, 4, 6].

3.1. Princip algoritmu

Jak již bylo řečeno výše, základním principem tohoto vylepšení algoritmu Minimax je nevyhodnocování těch větví herního stromu, které již nemohou ovlivnit ohodnocení uzlů nacházejících se nad nimi a tedy ani volbu nejlepšího tahu počítačového hráče z dané pozice.

Pro snadnější objasnění tohoto principu se v této podkapitole vrátíme k intuitivní představě ohodnocování herního stromu, které bylo použito v kapitole 2.1. Připomeňme, že šlo o ohodnocování pozic z pohledu hráče, který je na tahu v pozici odpovídající kořeni stromu. V pozicích, kde je na tahu tento hráč, se vybírá maximum z ohodnocení následovníků dané pozice. V pozici, kde je na tahu soupeř, vybíráme naopak minimum z ohodnocení jejích následovníků. Při zobrazování herních stromů budeme dále předpokládat, že jsou ohodnocení jednotlivých následovníků každé pozice vypočítávána od nejlevějšího směrem doprava.

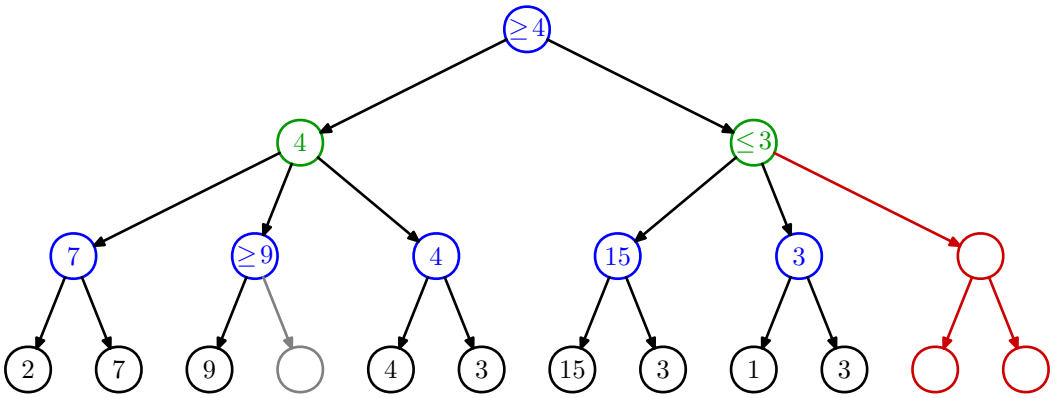
Principiálně rozlišujeme dva druhy ořezávání herního stromu. O alfa-ořezávání mluvíme tehdy, pokud byla mezi ohodnoceními potomků nalezena velmi malá hodnota, která již zaručuje, že daná větev nebude v žádném případě zvolena hráčem na tahu. V této situaci pak již nemá smysl zkoumat další potomky dané pozice. U beta-ořezávání naopak nalezení velmi velkého ohodnocení mezi potomky zaručí, že zkoumaná větev herního stromu nebude zvolena soupeřem. I zde pak můžeme upustit od zkoumání dalších potomků dané pozice. Oba typy ořezávaní herního stromu ilustruje následující příklad.



Obrázek 15. Ilustrace beta-řezu.

▷ Příklad 3.1. Uvažujme abstraktní herní strom znázorněný na obrázcích 15. a 16. Podobně jako v kapitole 2.1. jsou i zde modře označeny uzly, ve kterých se vybírá maximum z ohodnocení potomků, a zeleně uzly, ve kterých vybíráme minimum. Ohodnocení jednotlivých uzlů stromu jsou, jak již bylo zmíněno výše, vypočítávána postupně a to zleva doprava.

V situaci na obrázku 15. již byla část ohodnocení vypočtena. Ve chvíli, kdy je vypočtena hodnota 9 jednoho z listů, je zřejmé, že jeho rodič bude mít také hodnotu 9 nebo dokonce ještě více. Tento poznatek je triviálním důsledkem faktu, že ohodnocení tohoto rodiče je vypočítáváno jako maximum ze všech jeho přímých potomků. Podobně je z dříve vypočtených hodnot již k dispozici odhad výsledného ohodnocení zeleného uzlu s popiskem "≤ 7". Zde se opět využila úvaha, že zelený



Obrázek 16. Příklad alfa-ořezání.

uzel "≤ 7" bude ohodnocen minimem ze všech ohodnocení jeho přímých potomků. Zřejmě tedy modrý uzel "≥ 9" ani jeho potomci nemohou již ovlivnit vypočítávané hodnoty uzlů ve vyšších patrech stromu a tudíž je možné další, zatím nevyhodnocované, následovníky uzlu "≥ 9" ignorovat. Toto odpovídá výše zmíněnému beta-ořezávání.

Uzly daného herního stromu jsou pak dále ohodnocovány standardním způsobem. K dalšímu ořezu dochází až v situaci znázorněné na obrázku 16. Odhad ohodnocení pravého potomka kořene je aktuálně 3 nebo méně (minimum z hodnot 15, 3 a neznámé hodnoty), zatímco hodnota kořene může být odhadnuta na 4 nebo více. Vzhledem k faktu, že v kořeni vybíráme maximum z hodnot jeho potomků, je tedy již zřejmé, že druhý potomek kořene – uzel "≤ 3" výpočet neovlivní a tudíž již není potřeba ani zkoumat další jeho potomky, kteří jsou na obrázku označeni červenou barvou. V této situaci došlo k alfa-ořezání. □

3.2. Implementace algoritmu

V této kapitole se pokusíme proniknout do detailů implementace algoritmu Alfa-beta ořezávání. Podobně jako čistý algoritmus Minimax i toto vylepšení je realizováno pomocí rekurzivní funkce, která vypočítává ohodnocení pozic do určité hloubky herního stromu. Kromě zkoumané pozice a požadované hloubky je však nyní potřeba předávat při volání zmíněné funkce také jakési hodnoty α a β, které určují interval $\langle \alpha, \beta \rangle$ očekávaných ohodnocení. Pokud je nalezeno ohodnocení mimo tento interval, dochází k výše zmíněnému ořezávání.

Podobně jako u algoritmu Minimax, i nyní se budou ohodnocení pozic "blízkých konci hry" pohybovat v intervalech $\langle -\mathrm{MAX}, -\mathrm{MNOHO}\rangle$ a $(\mathrm{MNOHO}, \mathrm{MAX}\rangle$. Při rekurzivních voláních (posunu dále od aktuální pozice) je pak potřeba zvětšovat absolutní hodnotu těchto ohodnocení, čímž se vyjadřuje fakt, že se dostáváme blíže k dané výhře či prohře. Toto je v algoritmu realizováno pomocnou funkcí *dal*. Naopak při návratu z rekurze (přiblížení se aktuální pozici) je potřeba snížit absolutní hodnoty "ohodnocení blízkých ke konci hry", což obstarává funkce *bliz*. Funkce *dal* a *bliz* jsou uvedeny v algoritmu 3.1.

Algoritmus 3.1. *Pseudokód funkcí pro přepočty ohodnocení blízkých konci hry*

```
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
```

Stěžejní část algoritmu Alfa-beta ořezávání (viz algoritmus 3.2.) je realizována rekurzivní funkcí *alfabeta*, která se do značné míry podobá v předchozích kapitolách prezentované funkci *minimax*. Rozdíly mezi těmito funkcemi jsou patrny pouze v části kódu realizující rekurzivní volání na řádku 17 a dále v části algoritmu zajišťující ořezávání na řádcích 21–23.

Při rekurzivním volání funkce *alfabeta* je pochopitelně nutné předávat také interval očekávaných ohodnocení pro danou větev výpočtu. Podobně jako se při rekurzivním volání a návratu z rekurze mění u vypočteného ohodnocení "pohled jednoho hráče" na "pohled druhého hráče" změnou znaménka, je potřeba provést tuto změnu i u intervalu očekávaných hodnot, kde se očekávání $\langle \alpha, \beta \rangle$ mění na $\langle -\beta, -\alpha \rangle$. U případných hodnot blízkých ohodnocení konce hry se navíc provádí také zvětšení jejich absolutních hodnot o 1. Všimněte si také, že aktuální maximální nalezené ohodnocení se na řádku 20 ukládá přímo do proměnné *alfa*, což ihned zpřesňuje interval očekávaných ohodnocení při prozkoumávání dalších potomků dané pozice.

Ořezávání realizované v algoritmu na řádcích 21–23 spočívá v jednoduchém testu, zda právě nalezené maximální ohodnocení již dosahuje či překračuje maximální očekávanou hodnotu *beta*. Pokud ano, je ihned ukončeno vyhodnocování dalších potomků dané pozice, protože již nemohou volbu nejlepšího tahu nijak ovlivnit. Návratová hodnota *beta* nijak neovlivňuje výsledné ohodnocení rodiče dané pozice v herním stromu.

Algoritmus 3.2. *Pseudokód stěžejní části algoritmu Alfa-beta*

```
1:  function alfabeta(pozice, hloubka, alfa, beta)
2:  if je_prohra(pozice) then
3:    return −MAX
4:  end if
5:  if je_výhra(pozice) then
6:    return MAX
7:  end if
8:  if je_remíza(pozice) then
9:    return 0
10: end if
11: if hloubka = 0 then
12:   return ohodnocovaci_funkce(pozice)
13: end if
14: tahy ← generuj_tahy(pozice)
15: for all tah v kolekci tahy do
16:   potomek ← zahraj(pozice, tah)
17:   ohodnoceni ← −alfabeta(potomek, hloubka − 1, dal(−beta), dal(−alfa))
18:   ohodnoceni ← bliz(ohodnoceni)
19:   if ohodnoceni > alfa then
20:     alfa ← ohodnoceni
21:     if ohodnoceni ≧ beta then
22:       return beta
23:     end if
```

```
24:    end if
25: end for
26: return alfa
27: end function
```

U aktuální pozice (kořene herního stromu) nás opět více než ohodnocení této pozice zajímá tah, který vede k jejímu nejlépe ohodnocenému potomkovi, což je právě hledaný nejlepší tah počítačového hráče z této pozice. Výpočet tohoto tahu je realizován funkcí *nej_tah*, jež je popsána v algoritmu 3.3. Principiálně se tato funkce shoduje s funkcí pro výpočet nejlepšího tahu v algoritmu Minimax. Interval očekávaných ohodnocení je na začátku výpočtu nastaven na $\langle -\mathrm{MAX}, \mathrm{MAX}\rangle$, na této úrovni nedochází k žádným ořezům.

Algoritmus 3.3. *Pseudokód realizující výpočet nejlepšího tahu u kořene herního stromu*

```
1: function nej_tah(pozice, hloubka)
2: tahy ← generuj_tahy(pozice)
3: alfa ← −MAX
4: for all tah v kolekci tahy do
5:    potomek ← zahraj(pozice, tah)
6:    ohodnoceni ← −alfabeta(potomek, hloubka − 1, −MAX, dal(−alfa))
7:    ohodnoceni ← bliz(ohodnoceni)
8:    if ohodnoceni > alfa then
9:       alfa ← ohodnoceni
10:      nejlepsi_tah ← tah
11:   end if
12: end for
13: return nejlepsi_tah
14: end function
```

▷ Příklad 3.2. Uvažujme pozici znázorněnou na obrázku 17., v níž je na tahu bílý hráč. Nyní se pokusme pomocí algoritmu Alfa-beta ořezávání s počáteční hloubkou výpočtu nastavenou na 3 tahy vypočítat nejlepší tah v této pozici. Interval očekávaných ohodnocení i nakonec vypočítaná ohodnocení jednotlivých pozic jsou znázorněna v herním stromu na obrázku 18. Uzly herního stromu jsou zde označeny trojicí čísel — hodnota vlevo odpovídá parametru *alfa* předané při rekurzivním volání funkce *alfabeta* pro tuto herní pozici, číslo vpravo pak hodnotě parametru *beta* a číslo uprostřed udává vypočtenou návratovou hodnotu funkce *alfabeta*.

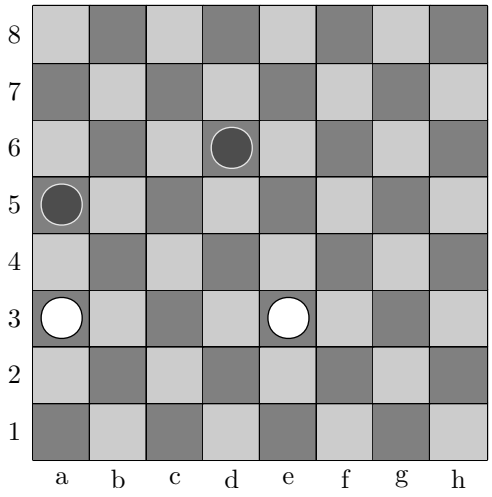
Podobně jako dříve budeme i nyní vyhodnocovat jednotlivé potomky dané pozice směrem zleva doprava. Vzhledem k tomu, že vypočtená ohodnocení i intervaly očekávaných hodnot plně odpovídají popsanému algoritmu, nebudou zde detailně okomentována veškerá volání funkce *alfabeta*, ale pouze některé zajímavé situace.

Po prozkoumání jediného potomka červeně zvýrazněné pozice je vráceno ohodnocení −85, které po změně znaménka překračuje hodnotu *beta* (70) ve volání funkce *alfabeta* odpovídajícím červeně zvýrazněné pozici. Toto způsobí okamžitý návrat z rekurze "do vyššího patra herního stromu". Protože však již byly prozkoumány všechny možné tahy z této pozice, nenastává zde typické ořezání.

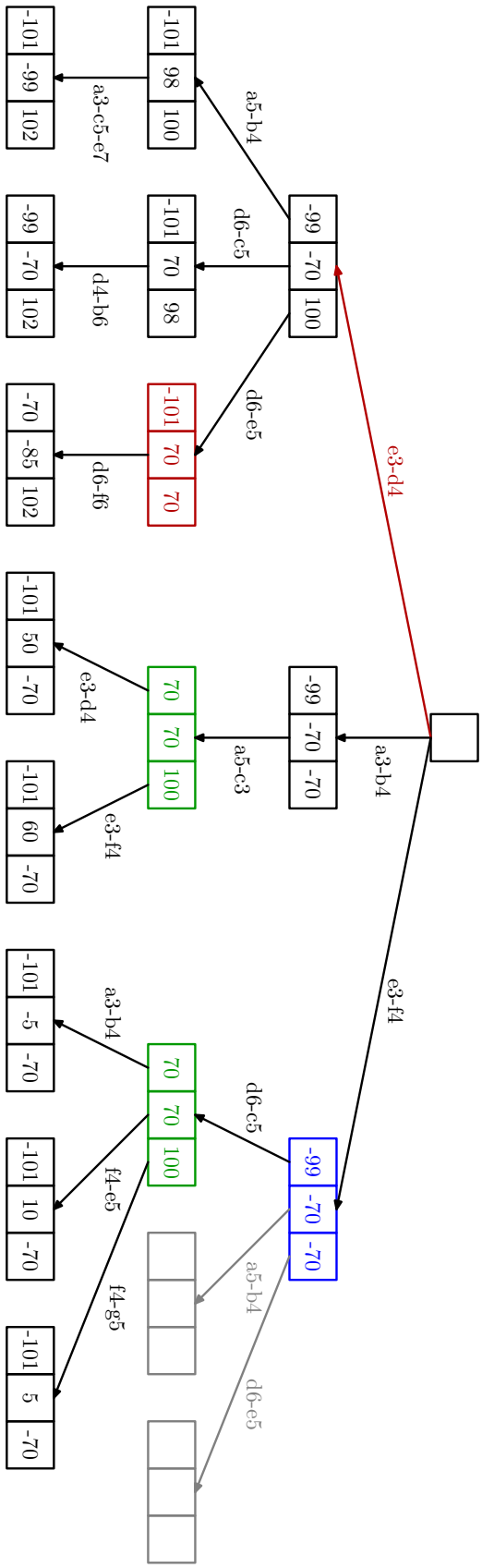
V zeleně zvýrazněných pozicích prozkoumáváme postupně jednotlivé potomky a získáváme jejich ohodnocení 50 a 60 (resp. −5, 10 a 5), která však po změně znaménka nepřekračují ani hodnoty parametrů *alfa* (v obou případech 70) ve voláních funkce *alfabeta* odpovídajících těmto zeleně zvýrazněným pozicím. Z tohoto důvodu je po prozkoumání všech potomků těchto pozic vrácena hodnota 70 také jako jejich výsledné ohodnocení.

Po prozkoumání nejlevějšího potomka modře zvýrazněné pozice získáváme jeho ohodnocení 70, což po změně znaménka dosahuje hodnoty parametru *beta* volání funkce *alfabeta*, které odpovídá modře zvýrazněné pozici. Toto vede k okamžitému návratu z rekurze "do vyššího patra stromu". Z tohoto důvodu už nedochází k prozkoumávání dalších šedě vykreslených potomků modře zvýrazněné pozice, což odpovídá typickému ořezání herního stromu v algoritmu Alfa-beta.

Ve volání funkce *nej_tah*, která prozkoumává aktuální pozici – kořen zobrazeného herního stromu, máme po prozkoumání všech možných tahů z této pozice zapamatováno maximální ohodnocení 70 a tah, který jako první k tomuto ohodnocení vedl **e3-d4**. Tento v obrázku 18. červeně zvýrazněný tah odpovídá nejlepšímu tahu vypočtenému představovaným algoritmem. □



Obrázek 17. Počáteční pozice k příkladu 3.2., na tahu je bílý hráč.



Obrázek 18. Herní strom z příkladu 3.2.

Literatura

[1] Česká federace dámy: *Oficiální stránky České federace dámy*, citováno 2. 9. 2010. Dostupné na adrese http://www.damweb.cz.

[2] Dieter Steinwender, Frederic A. Friedel: *Šachy na PC*. Unis Publishing, Přerov, 1997.

[3] *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).

[4] 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.

[5] Ladislav Vitásek: *Pokročilá implementace deskové hry Dáma, bakalářská práce*. České vysoké učení technické v Praze, 2006.

[6] *Alfa-beta ořezávání (česky, německy, anglicky) – Wikipedie, otevřená encyklopedie* [online], poslední revize 8. 6. 2011 (citováno 27. 10. 2011).
Dostupné na adrese http://cs.wikipedia.org/wiki/Alfa-beta_ořezávání.