Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Implementace hry šachy

Bakalářská práce

Autor: Chaloupka, Jan
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. RNDr. Petra Poulová, Ph.D.

Hradec Králové duben 2021

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 30.4.2021 Jan Chaloupka

Poděkování:
Děkuji vedoucí bakalářské práce doc. RNDr. Petře Poulové, Ph.D za vedení práce a konzultace při psaní práce. V neposlední řadě také děkuji rodině a svým blízkým za podporu a motivaci při tvorbě této bakalářské práce.

Anotace

Bakalářská práce se zabývá problematikou implementace hry šachy a metodikami s tím spojenými. První část práce se zabývá stručným popisem historie šachů. Od základů objasňuje pravidla a vysvětluje postupy, které se využívají při programování šachových enginů, včetně popisu jednotlivých algoritmů. Dále představuje základní funkce, které musí mít každý šachový program, aby byla zaručena jeho správná funkčnost. V kapitole Šachy z pohledu počítače jsou popsány techniky, které musí počítač ovládat pro správné zahrání šachové partie. Kapitola Implementace představuje proces vytvoření výsledného šachového programu. Poslední kapitola se zabývá testováním vytvořeného programu, kterému byly mimo jiné předkládány různé šachové úlohy a testovalo se, zda dokáže najít jejich vhodné řešení.

Annotation

The bachelor's thesis deals with the implementation of the game of chess and the methodologies associated with it. The first part deals with a brief description of the history of chess. It basically explains the rules and explains the procedures used in programming chess engines, including a description of individual algorithms. It also introduces the basic functions that every chess program must have in order to guarantee its correct functionality. The chapter Chess from the Computer's Perspective describes the techniques that the computer must control in order to play the chess game correctly. The Implementation chapter presents the process of creating the resulting chess program.

Obsah

Seznam obrázků

1 Úvod

Šachy jsou deskovou hrou pro dva hráče. Jedním z důvodů oblíbenosti šachů je nepochybně obrovská variabilita tahů, kterou tato hra nabízí. V součtu první čtyři tahy každého hráče obsahují přes 318 miliard možných kombinací jak je zahrát. (1) Celkový počet všech možných tahů zatím není znám, nicméně se odhaduje na 10^{120} možností. Toto číslo je větší než počet atomů v pozorovatelném vesmíru, který se odhaduje na 10^{80} atomů. (2) Pokud vezmeme v úvahu, že teoreticky nejdelší partie má 5 949 tahů, je zřejmé, že úlohou šachových enginů není najít takový tah, který by vedl k jisté výhře, ale tah, který se jeví jako nejlepší pro několik tahů dopředu.

Cílem mé bakalářské práce je seznámit čtenáře se základními principy programování šachových enginů a naimplementovat šachový program, který bude umět hrát proti člověku.

Kapitola Obecné aspekty šachů popisuje historii a pravidla šachů. Následuje kapitola Šachové algoritmy, ve které jsou vysvětleny základní algoritmy, které šachové enginy používají. Součástí této kapitoly je praktický popis daného algoritmu spolu s naznačením průchodu tímto algoritmem. Kapitola Šachy z pohledu počítače objasňuje důležité aspekty programování šachových enginů. Je zde popsána ohodnocovací funkce, reprezentace šachovnice a generování tahů. V kapitole Implementace je popsán vývoj výsledného programu. Poslední kapitola je nazvána Testování. V této kapitole jsou výslednému programu předkládány testovací úlohy. Kontroluje se nejen správnost vyhodnocení, ale i rychlost, s jakou dokáže program reagovat. Součástí této kapitoly jsou také úvahy nad možnými vylepšeními současné verze aplikace.

2 Obecné aspekty šachů

2.1 Historie šachů

Šachy pravděpodobně mají původ v nějaké deskové hře, která byla známa již mnoho let před naším letopočtem. Předpokládat lze také to, že šachy nemohly vzniknout najednou a nevytvořila je jediná osoba. Pravděpodobně překonaly dlouhou vývojovou cestu, během které byla pravidla či pohyby figur upřesňovány, jak ve své knize *Učebnice šachu pro samouky* popisuje Karel Pliska. (3) První doložené zmínky o šachu jsou kolem 6. století v Indii, kde lze vývoj hry pozorovat na základě písemných záznamů a archeologických památek. Původní šachy se značně lišily od šachů dnešních. V Indii byly nazývány „čaturanga“, to v překladu znamená „čtyři součásti vojska“. Tento překlad symbolizuje čtyři druhy vojska: slony, válečné vozy, jezdectvo a pěchotu. Tyto základní figury měl každý z hráčů. Cílem hry bylo vzít všechny soupeřovy figury. Bylo také možné vzít soupeřova či spojencova krále. Házením kostky se určovalo, kterou figurou se bude táhnout, z toho důvodu hrála náhoda hlavní roli. Z Indie se hra rozšířila do sousední Persie a odsud do arabských zemí. Ve Střední Asii kolem 7. století došlo ke změně pravidel, hra byla přejmenována na „šatrandž“ a stala se hrou pro dva hráče. Pravidla se od dnešního šachu lišila v pohybech figur. Tehdejší střelec se nazýval „alfil“, dáma (vezír) byla nejslabší figurou ze všech, pěšec nemohl využít dnešního dvojkroku. Ze Střední Asie se hra rozšířila do Evropy a kolem 11. století již byla známá po celé Evropě. Ve hře již nehrála žádnou roli náhoda. Hra postupně získala vysokou popularitu. Do Čech se šachy dostaly na začátku 11. století. (4) Kolem 15. století došlo k významné změně pohybu dámy, která se stala nejsilnější figurou ve hře. Střelec mohl táhnout po celé diagonále a byl umožněn dvojkrok pěšce z výchozího postavení. (5) Král získal právo jednou za hru přeskočit figuru, čímž vznikla rošáda, její podoba byla ustálena v 19. století. V této době bylo také upřesněno pravidlo proměny pěšce v jinou figuru, dojde-li na poslední řadu. V 18. století došlo k zavedení algebraické šachové notace, která se uchovala dodnes. (6)

2.2 Pravidla šachů

2.2.1 Výchozí postavení

Šachy se hrají figurami na hrací desce, která je tvořena osmi řadami a osmi sloupci, celkem je na ní tedy 64 čtverců střídajících se světlých (bílých) a tmavých (černých) barev. Vodorovné řady jsou označeny čísly 1 až 8, svislé písmeny „a“ až „h“, čímž je zajištěna jednoznačná identifikace každého pole. Na začátku šachové partie se deska natočí tak, aby každý hráč měl v pravém dolním rohu světlé pole. 32 figur se dělí na 6 typů (pěšci, střelci, jezdci, věže, dáma a král). V první řadě zleva stojí figury v tomto pořadí: věž, jezdec, střelec, dáma, král, střelec, jezdec a věž, přičemž dáma je na poli, které náleží její barvě, tzn. bílá dáma bude na bílém poli a naopak. Druhá řada je obsazená pěšci na každém poli. (7)

Obr. 1 Výchozí postavení šachů

Zdroj: https://chessboardimage.com/

2.2.2 Pohyby figur

Každá z šesti druhů figur se pohybuje jinak. Jednotlivé figury nemohou pohybovat jinými figurami vyjma braní soupeřovy figury. Popisy všech figur, které jsou uvedeny níže, byly převzaty z učebnice šachu pro samouky od Karla Plisky. (3)

Král je nejdůležitější figura ze všech, pokud padne, partie je pro daného hráče prohraná. Král se může pohybovat o jedno pole všemi směry vyjma rošády, kdy může přejít o dvě či tři pole směrem ke své věži, pokud daná cesta není krytá soupeřovou figurou. Oproti ostatním figurám se král nemůže postavit na pole, na kterých by mu hrozilo sebrání soupeřovou figurou, tím pádem nesmí ani vzít soupeřovu figuru, která je krytá nějakou jeho figurou. Oba hráči se snaží dostat soupeřova krále do bezvýchodné situace tak, aby neměl možnost zahrát žádný legální tah a vyhnul se tak šachu. Tomu se říká mat a znamená konec hry. Šachu se lze vyhnout tak, že buď uhneme králem na pole, které není pod soupeřovou kontrolou, nebo do šachové linie postavíme vlastní jinou figuru, nebo můžeme figuru, která dává králi šach, vzít figurou vlastní.

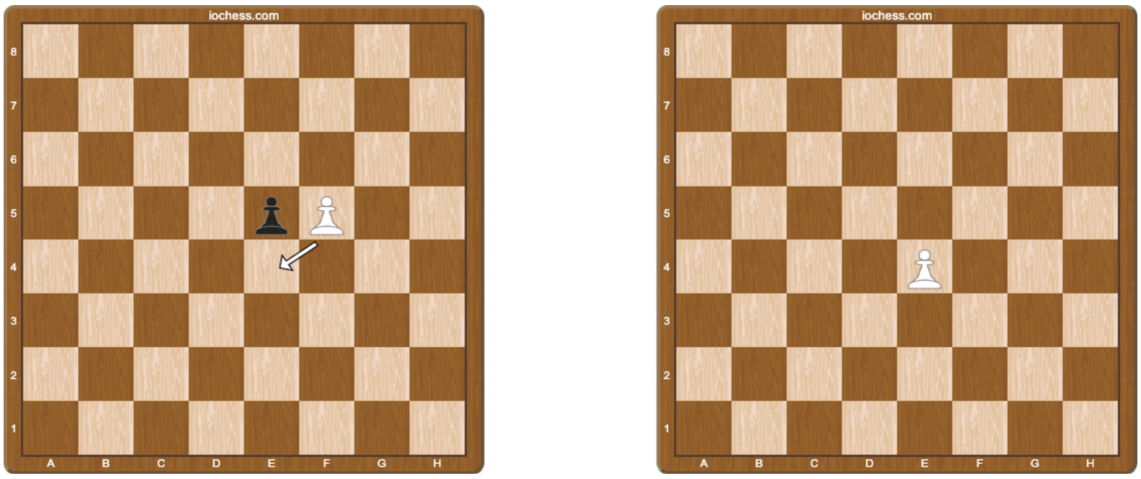
Dáma je nejsilnější figurou ve hře, jelikož se může pohybovat všemi směry tzn. na sloupcích, řadách i obou diagonálách. Její pohyb si lze představit jako kombinaci pohybu věže a střelce. Dámou je možné pohybovat o libovolný počet polí za předpokladu, že má volnou cestu.

Věž, jejíž pohyb je nejjednodušší, se může pohybovat pouze rovně tj. po sloupcích a po řadách o neomezený počet polí. Její pohyb končí, narazí-li na figuru vlastní nebo soupeřovu, kterou může vzít.

Střelec je další figurou, kterou lze pohybovat o neomezený počet polí, ale pouze po diagonálách. Jako u všech figur i pro něj platí, že jeho pohyb končí na poli, na kterém vezme soupeřovu figuru. Každý střelec začíná buď na bílé nebo černé barvě a musí na ní zůstat až do konce hry. Střelec, který ovládá bílá pole, se také označuje jako „bělopolný“, zatímco střelec, který ovládá černá pole, je označován jako „černopolný“. Každá ze stran má oba tyto střelce.

Jezdec je velice odlišnou figurou oproti ostatním, co se jeho pohybu týče. Jezdec se pohybuje o jedno pole jedním směrem a poté o dvě pole v úhlu 90 stupňů nebo naopak o dvě pole v jednom směru a o jedno pole v úhlu 90 stupňů. Jeho pohyb si lze představit jako písmeno „L“ nebo jako číslici 7. Jezdec je také jediná figura, kterou lze táhnout přes jiné figury, ať už vlastní, nebo cizí. Jezdec může brát pouze ty figury, které leží na poli, kde jeho pohyb končí. Barva jeho pole se po každém tahu mění.

Pěšec je nejslabší figurou ze všech. Může se pohybovat pouze dopředu o jedno pole vyjma případu, kdy může táhnout o dvě pole dopředu ze své zahajovací pozice. Pěšec může brát soupeřovy figury pouze diagonálně. Vzhledem k tomuto druhu pohybu, nastane-li situace, kdy je před pěšcem jakákoli figura, nemůže kolem ní projít, ani ji vzít. (3) Mezi další vlastnosti figury pěšce patří tzv. promotion (česky *povýšení*), což znamená, že pokud pěšec dojde na druhou stranu šachovnice, může se z něj stát jakákoli jiná figura vyjma krále. Většinou se pěšec vyměňuje za dámu, jelikož je to nejsilnější figura ve hře. (7) Další pravidlo, které se úzce váže na pěšce, je pravidlo braní mimochodem (*z francouzského „en passant“*), které znamená, že pokud jsou dva pěšci odlišných barev vedle sebe, tak hráč, který je na tahu, může vzít pěšce tím, že bude táhnout na jeho stranu a o jedno pole dopředu tzn. diagonálně o jedno. Tento tah musí proběhnout okamžitě poté, co se pěšec přesunul kolem, jinak toto pravidlo nelze uplatnit. (8) Na následujícím příkladu (Obrázek č.2) bílý pěšec sebral černého pomocí pravidla „en passant“.

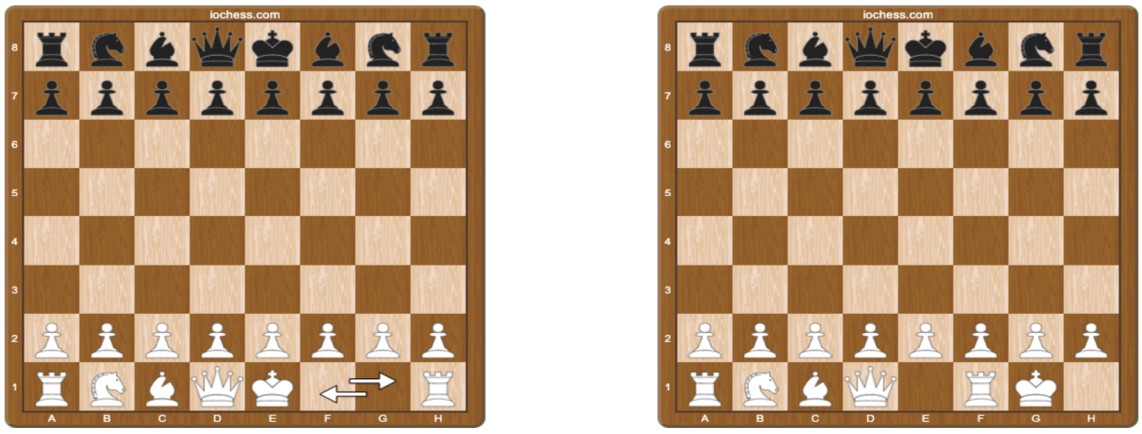


Obr. 2 Braní mimochodem

Zdroj: https://chessboardimage.com/

2.2.3 Další pravidla / pojmy

Rošáda je jedna z možností, jak dostat krále do bezpečí a vtáhnout věž do hry v úvodních fázích duelu. Rošáda se provádí tak, že hráč táhne králem na jednu stranu o dvě pole a věží z dané strany na opačnou stranu vedle krále. Existují ale určitá omezení, za kterých lze tento tah provést. V první řadě rošádu lze provést pouze v případě, kdy se jedná o první tah tohoto krále a první tah dané věže. Mezi králem a věží nesmí být žádné jiné figury a při pohybu nesmí dostat šach. Pokud je král umístěn blíže ke straně hrací desky a udělá rošádu tímto směrem, tato rošáda se nazývá „kingside“. Pokud udělá rošádu na druhou stranu, nazývá se tato rošáda „queenside“. (7)



Obr. 3 Rošáda „kingside“

Zdroj: https://chessboardimage.com/

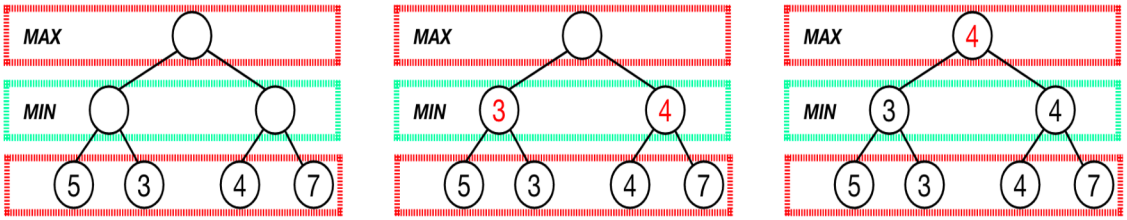
První tah začíná vždy hráč bílé strany. O tom, který z hráčů bude hrát za bílé, rozhoduje náhoda. Hráč s bílými figurami má malou výhodu oproti hráči s černými figurami, protože zahajuje partii a má tedy možnost rozhodnout o způsobu zahájení hry.
Pat neboli remíza je situace, kdy je na řadě hráč, který nedostává šach a jeho každý tah, který může provést, by vedl k tomu, že by jeho král dostal šach.
Mat je prohra pro hráče, jehož král je v šachu a nemůže se z něj dostat žádným legálním pohybem. (7)

3 Šachové algoritmy

3.1 Algoritmus MinMax

MinMax je algoritmus, který se používá v tahových hrách pro dva hráče. Jedná se o rozhodovací algoritmus, jehož cílem je najít optimální další krok. V algoritmu figurují dva hráči, jeden „Max“ a druhý „Min“. Po vyhodnocení pozic na desce ohodnocovací funkcí[1] vybírá jeden hráč stav hry s maximálním skóre a druhý hráč s minimálním skóre. Max se snaží získat nejvyšší skóre, zatímco Min se snaží docílit nejnižšího skóre, aby tak předešel lepšímu tahu ze strany Max. MinMax je založen na konceptu hry s nulovým součtem, ve které je celkové skóre rozděleno mezi hráče, přičemž zvýšení jednoho skóre má za následek snížení druhého. Algoritmus nám umožňuje prohledat dopředu možné tahy na základě výpočetního výkonu. (9)

Princip algoritmu je znázorněn na obrázku č.4. Každý řádek reprezentuje počet možností zahrání daného hráče, přičemž hodnoty každého listu vrátí ohodnocovací[1] funkce. Poté si vybírá hodnotu na základě nižších úrovní stromu a doplňuje jednotlivé větve na vyšších úrovních. V našem případě Min vybere hodnoty 3 a 4 a v dalším kroku vybere Max hodnotu 4.

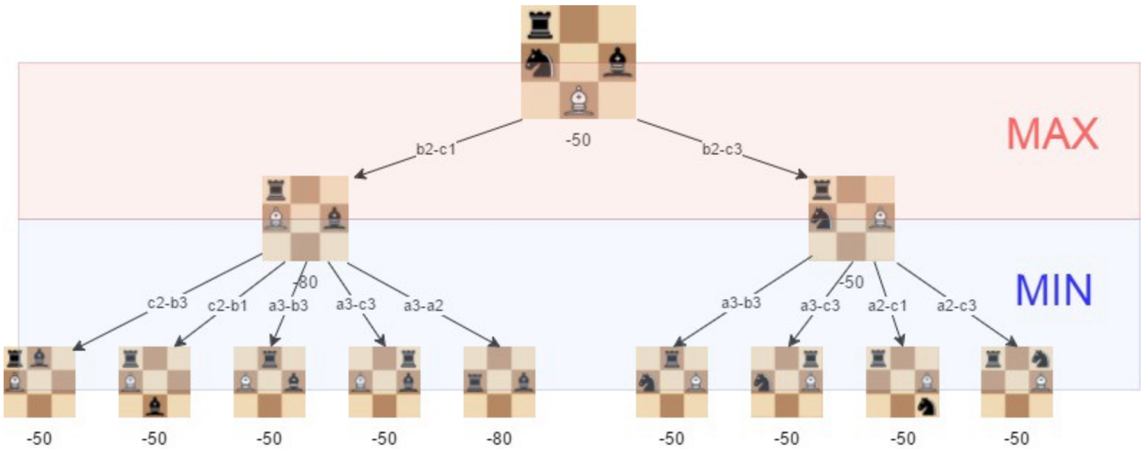


Obr. 4 Algoritmus MinMax

Zdroj: vlastní zpracování

[1] Ohodnocovací funkce je popsána v kapitole 4.2

Časová složitost algoritmu je $O(b^m)$, kde b je větvící faktor herního stromu (počet možných tahů z každé pozice) a m je hloubka prohledávání. (10) Ukázkou MinMax algoritmu v souvislosti s šachy je obrázek č.5. Šachovnice je zjednodušena na velikost 3x3. Bílý je na tahu. Z této pozice má na výběr 2 tahy. Může vzít buď jezdce nebo střelce, to znamená 2 možné kombinace, jak zahrát. Hráč hrající s černými figurami může na vzetí jezdce zareagovat pěti tahy. V případě, že bílý hráč vezme střelce, má na výběr 4 tahy. Předpokládáme-li, že v ohodnocovací funkci mají bílé figury střelec a jezdec hodnotu 30 a věž hodnotu 50 (černé figury mají tyto hodnoty negativní), tak sečtením hodnot figur na šachovnici lze získat ohodnocení této pozice. (11) Na nejnižší úrovni jsou jednotlivé pozice ohodnoceny ohodnocovcí funkcí. O úroveň výš hráč Min vybírá nejmenší hodnoty, v levé větvi vybere tah s hodnotou -80 a v pravé větvi tah s hodnotu -50. V nejvyšší úrovni je v tomto případě Max hráč, který vybírá nejvyšší hodnotu, z těchto dvou tahů má vyšší ohodnocení tah -50. Jak je vidět na obrázku, nejlepším tahem je tedy sebrání střelce. V opačném případě by černý hráč mohl v následujícím tahu střelce sebrat věží.



Obr. 5 Ukázka algoritmu MinMax
Zdroj: (11)

Pseudokód algoritmu je naznačen níže.

```
max(int depth) {
    if(depth == 0) return evaulatePosition();
    generateMoves();
    int maxValue = -infinity;
    for(int i = 0; i < moves.length; i++) {
      makeMove(i);
      int value = min(depth-1);
      if(value > bestValue) {
        bestValue = value;
      }
      undoMove();
    }
  }

min(int depth) {
    if(depth == 0) return evaulatePosition();
    generateMoves();
    int minValue = infinity;
    for(int i = 0; i < moves.length; i++) {
      makeMove(i);
      int value = max(depth-1);
      if(value < bestValue) {
        bestValue = value;
      }
      undoMove();
    }
  }
```

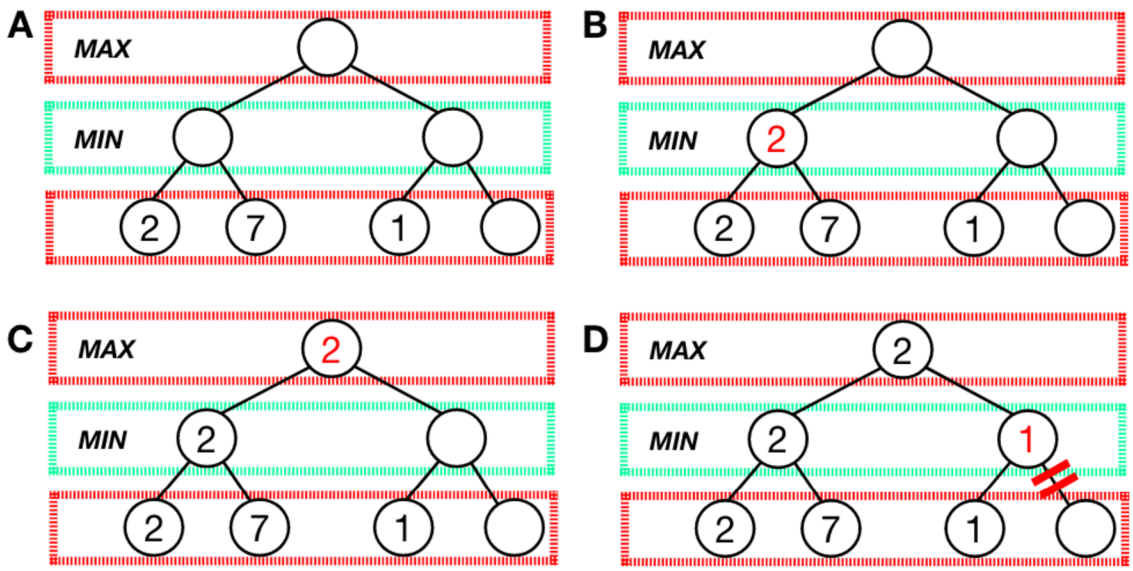
Obr. 6 Pseudokód MinMax
Zdroj: Vlastní zpracování

3.2 Algoritmus Alfa-beta ořezávání

Algoritmus Alfa-beta ořezávání (Alpha-beta pruning) je vylepšení algoritmu MinMax. Nazývá se Alfa-beta, jelikož přidává oproti MinMax algoritmu 2 parametry navíc, alfa a beta. Alfa je nejlepší hodnota, kterou Max v současné době může zaručit na této nebo vyšší úrovni. Beta je nejlepší hodnota, kterou Min v současné době může zaručit na této nebo vyšší úrovni. Algoritmus Alfa-beta ořezávání optimalizuje způsob prohledávání algoritmu MinMax. Snižuje dobu výpočtu stromu ořezáním větví stromu hry, které už nemusí být prohledávány, protože existuje lepší tah. To umožní prohledávat strom

rychleji a jít do hlubších úrovní. (12) Časová složitost algoritmu může dosáhnout ideálního stavu, když jsou jednotlivé větve vhodně seřazeny a v levé části jsou nejlepší tahy. Takový stav lze vyjádřit vzorcem $O(b^{m/2})$, kde b je větvící faktor a m je hloubka, do které algoritmus prohledává. V opačném případě, kdy jsou větve v nejhorším možném řazení, nedojde k žádnému ořezání a algoritmus se chová stejně jako MinMax, jeho časová složitost se nemění a zůstává $O(b^m)$. (13)

Průchod algoritmu je znázorněn na obrázku 7.

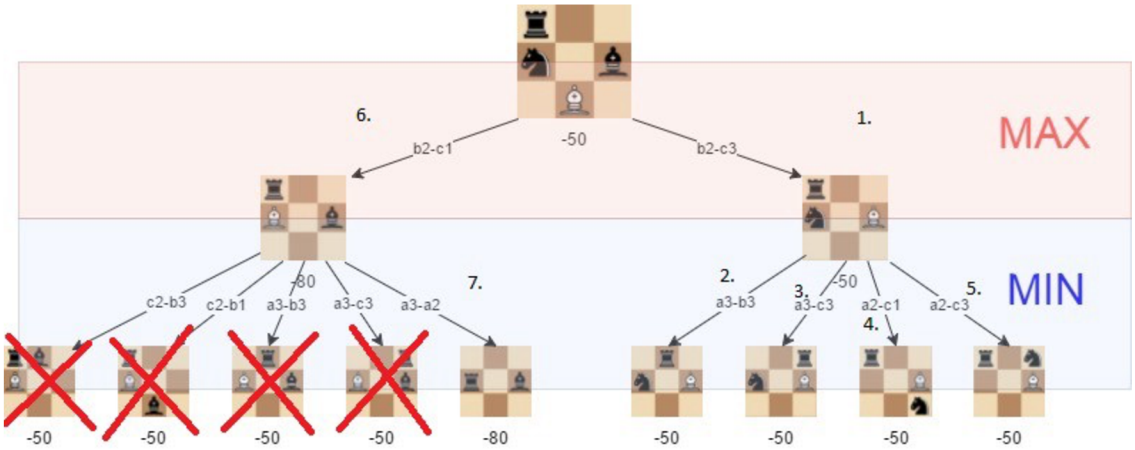


Obr. 7 Alfa-beta ořezávání

Zdroj: vlastní zpracování

Algoritmus začíná jako klasický MinMax, na obrázku 7A jde do větve umístěné nejvíc vlevo a Min vybírá nejmenší číslo, tím pádem vybere 2 - Obrázek 7B. Poté přesune vybrané číslo do nejvyšší větve algoritmu viz. obrázek 7C a jde do pravé větve stromu. V pravé části stromu jde opět do levé části a vybere 1, v tuto chvíli v této části stromu obsahuje Min hodnotu 1, nyní už není potřeba zkoumat hodnotu pravého listu, jelikož Max na nejvyšší úrovní vidí hodnotu 1, a protože vybírá maximální číslo, nikdy se pro tuto větev nerozhodne a není potřeba dále prozkoumávat pravou stranu stromu. Algoritmus našel nejlepší možný tah a vydá se levou částí stromu, kde vybere prvky 2 2 2.

Ukázkou algoritmu Alfa-beta ořezávání v souvislosti s šachy je obrázek č.8. Šachovnice je zjednodušena na velikost 3x3. Bílý je na tahu a má dvě možnosti, může vzít střelce nebo jezdce. V případě sebrání jezdce se vygenerují tahy pro hráče s černými figurami a jsou ohodnoceny. Na obrázku č. 8 v nejnižší úrovni jsou všechny tahy v levé větvi s hodnocením -50 ořezány, jelikož v této úrovni stromu se vybírá minimální hodnota a v tuto chvíli algoritmus ví, že vybere tah s ohodnocením -80. V případě, že by byla nastavena vyšší hloubka prohledávání, ořezané větve by neprocházel do nižších úrovní, jelikož ví, že tuto cestu nikdy nevybere, protože na této úrovni se nachází lepší cesta.

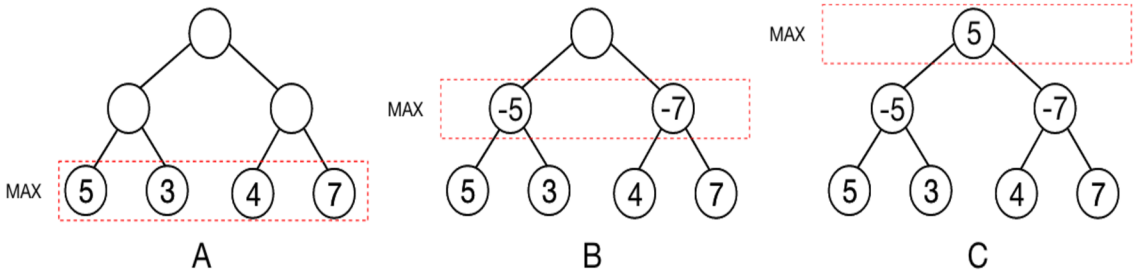


Obr. 8 Ukázka algoritmu Alfa-beta ořezávání
Zdroj: (11)

3.3 NegaMax

Algoritmus NegaMax je jistou analogií k algoritmu MinMax. Zatímco v algoritmu MinMax, kde se každý hráč snažil najít nejlepší hodnotu tak, že hráč Max vybíral maximální hodnotu a hráč Min vybíral minimální hodnotu, v případě algoritmu NegaMax se oba hráči snaží maximalizovat hodnotu, která je pokaždé vynásobena číslem -1 pro každou úroveň stromu vyjma kořene, kde je tato multiplikace zbytečná. V MinMax algoritmu bylo potřeba na algoritmus pohlížet z pohledu hráče, který je právě na tahu, ať už bílý nebo černý, a vyhodnotit jeho hodnotu pro každou větev na základě toho, zda se jednalo o Min hráče a hledalo se minimum, nebo se jednalo o Max hráče a hledalo se maximum. V algoritmu NegaMax toto odpadá, jelikož každý hráč

hledá maximum z dolních větví stromu a tato hodnota je následně negována. NegaMax přinese vždy stejný výsledek jako MinMax (pokud bude nastavena stejná hloubka prohledávání), nicméně v NegaMax algoritmu se nevolají dvě funkce pro nalezení hodnot Min a Max, ale volá se pouze jedna pro hodnoty Max. (14)



Obr. 9 Algoritmus NegaMax

Zdroj: vlastní zpracování

Na obrázku 9 je znázorněna posloupnost algoritmu, čísla odpovídají obrázku 4, kde se ukazovala posloupnost algoritmu MinMax. Bude tedy vidět, že algoritmus NegaMax vybral stejnou větev jako MinMax. Na obrázku A jsou zvýrazněné větve, případně listy herního stromu. Přecházíme-li na obrázek B, kde algoritmus postoupil v herním stromu o úroveň výš a vybírá z předchozí úrovně maximum, tudíž vybere hodnoty 5 a 7, poté hodnoty zneguje a zapíše do větve, tudíž do stromu přibydou hodnoty -5 a -7. Přecházíme na obrázek C, kde algoritmus opět vybírá nejvyšší hodnotu z čísel -5 a -7, což je -5, a tuto hodnotu následně zneguje, to nám dává hodnotu 5, jelikož se jedná o kořen celého stromu, není nutné tuto hodnotu negovat. Ze zpětného protrasování algoritmu je patrné, že výsledný nejlepší tah bude v našem případě celá levá větev a algoritmus půjde větví 5, -5, 5, jelikož v NegaMax se vybírá maximální hodnota. Výsledek je stejný jako u algoritmu MinMax s tím rozdílem, že v implementaci nebude nutné volat funkci Min a Max.

4 Šachy z pohledu počítače

4.1 ELO

Systém hodnocení ELO[2] je metoda měření relativní síly hráče v porovnání s ostatními hráči. Tento systém počítá pravděpodobný výsledek daného hráče proti ostatním hráčům. ELO představuje číslo, které reprezentuje výsledky dané osoby nebo stroje v hodnocených hrách. Po každé odehrané hře se ELO upraví v závislosti na tom, jestli hráč vyhrál, nebo prohrál. Obecně lze předpokládat, že hráč, který má ELO vyšší o 100 bodů než jeho soupeř, vyhraje pět z osmi her (64 %) proti soupeřovi, který má ELO o 100 bodů nižší. Hráč, který má ELO o 200 bodů vyšší než jeho protivník, zpravidla vyhraje tři hry ze čtyř (75 %). Výpočet ELO dokáže brát v úvahu výkyvy ve výkonu daného hráče. Každý má slabší a silnější herní dny. Hráči, kteří mají nižší hodnocení než jejich soupeř, mohou přesto porazit protivníka, který má hodnocení ELO vyšší než oni, a systém ELO vypočítává pravděpodobnost, zda k tomu dojde. Mezi vlastnosti tohoto systému hodnocení patří také skutečnost, že rozdíl ELO bodů mezi jednotlivými účastníky partií určuje, kolik bodů mohou vyhrát nebo naopak ztratit. Předpokládá se, že hráč s výrazně vyšším ELO vyhraje, ale za tuto výhru nezíská mnoho bodů. Očekávána výhra nad hráčem, který má výrazně nižší ELO, přinese vítězi jen pár nebo žádné body. Naopak, pokud vyhraje hráč s nižším hodnocením, je tato skutečnost považována za mnohem významnější a tomuto hráči je přiděleno více bodů. Prohra s hráčem, který má výrazně nižší ELO, značně sníží ELO hráče, který prohrál. (15)

4.2 Ohodnocovací funkce

Šachovnice je reprezentována dvourozměrným polem 8x8, tudíž 64 poli rozdělenými do osmi řad a osmi sloupců. Po nalezení všech možných legálních tahů je třeba každý z těchto tahů ohodnotit ohodnocovací funkci, abychom mohli určit, který z nalezených tahů je v dané situaci nejlepší. Výsledek této funkce je číslo, které nám udává, jak je který tah dobrý. Ohodnocení pozic je jednou ze zásadních funkcí šachových strojů.

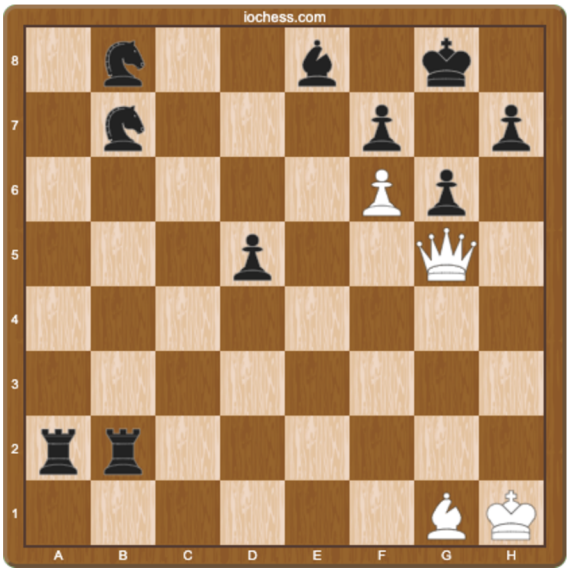
[2] Nejedná se o zkratku, ale o příjmení tvůrce tohoto systému Arpada Ela (1903-1992),

Pokud je ohodnocení nesprávné nebo nepřesné, pak nezáleží na tom, zda algoritmus dokáže hledat ve vysoké hloubce herního stromu, jelikož výsledný tah nebude relevantní. (16)

4.2.1 Hodnocení materiálu

Hodnocením materiálu je myšlen součet hodnot zbývajících figur na šachovnici. Každé figuře je přiřazena hodnota, která reprezentuje její sílu. Pěšcům bývá přidělena hodnota „1“, střelci a koně mívají hodnotu „3“, věž hodnotu „5“, dáma hodnotu „9“ a král hodnotu „10“. Králi je přiřazována nejvyšší hodnota z toho důvodu, že jeho ztráta by znamenala prohru a konec hry. Samotné materiálové ohodnocení ovšem nestačí a vedlo by k chybám. (16)

Na Obrázku 10. je vidět, že samotné ohodnocení materiálu by nestačilo. Na první pohled je zřejmé, že černý hráč má vysokou materiální výhodu i přesto, že ztratil věž. Bílý hráč, který je na tahu, však zahraje dámou na H6, černý hráč nemá, jak tuto hrozbu odvrátit a do pár tahů prohraje na mat dámou.



Obr. 10 Ukázka hodnocení materiálu

Zdroj: https://chessboardimage.com/

4.2.2 Hodnocení pozic

Pro každou figuru je výhodné, nachází-li se v určité fázi hry na určitém místě. Např. pro pěšce je výhodné v úvodních fázích hry zabrat střed desky, zatímco pro krále by v úvodní fázi hry pohyb do středu desky znamenal rychlý mat. Z toho důvodu se využívají tzn. „Piece-square tables“. Hodnoty jsou převzaty z (17).

Pěšec

```
8 { 0,  0,  0,  0,  0,  0,  0,  0},
7 {50, 50, 50, 50, 50, 50, 50, 50},
6 {10, 10, 20, 30, 30, 20, 10, 10},
5 { 5,  5, 10, 25, 25, 10,  5,  5},
4 { 0,  0,  0, 20, 20,  0,  0,  0},
3 { 5, -5,-10,  0,  0,-10, -5,  5},
2 { 5, 10, 10,-20,-20, 10, 10,  5},
1 { 0,  0,  0,  0,  0,  0,  0,  0}
   a   b   c   d   e   f   g   h
```

Obr. 11 Hodnoty pěšců

Zdroj: INTELLIJ, vlastní zpracování

Každý dobrý hráč šachu ví, že začátky partie otevírají pěšci buď na *d4* nebo *e4*, jelikož zabrání středu je v úvodní fázi hry nejvýhodnější, pozice *d5* a *e5* jsou ještě lepší. Pozice *f3* a *g3* mají nižší hodnotu, jelikož dělají „díry“ a odhalují krále spolu s dámou. Řady 6 a 7 mají vyšší hodnoty, aby byli pěšci „povzbuzeni“ tlačit nepřítele. A stále je tu možnost výměny pěšce za dámu v případě, že pěšec dojde na opačný konec šachovnice.

Jezdec

```
8{-50,-40,-30,-30,-30,-30,-40,-50},
7{-40,-20,  0,  0,  0,  0,-20,-40},
6{-30,  0, 10, 15, 15, 10,  0,-30},
5{-30,  5, 15, 20, 20, 15,  5,-30},
4{-30,  0, 15, 20, 20, 15,  0,-30},
3{-30,  5, 10, 15, 15, 10,  5,-30},
2{-40,-20,  0,  5,  5,  0,-20,-40},
1{-50,-40,-30,-30,-30,-30,-40,-50}
   a   b   c   d   e   f   g   h
```

Obr. 12 Hodnoty jezdců

Zdroj: INTELLIJ, vlastní zpracování

U jezdců je nejdůležitější, aby nestáli na straně šachovnice vzhledem k jejich omezené mobilitě. Nejhorší pozice pro jezdce je v rohu, jelikož má pouze dva legální tahy. Z toho důvodu jsou v rozích největší záporné hodnoty. Jezdec naopak dostává bonus, pokud se pohybuje ve středu šachovnice, odkud může kontrolovat mnoho pozic.

Střelec

```
8{-20,-10,-10,-10,-10,-10,-10,-20},
7{-10,  0,  0,  0,  0,  0,  0,-10},
6{-10,  0,  5, 10, 10,  5,  0,-10},
5{-10,  5,  5, 10, 10,  5,  5,-10},
4{-10,  0, 10, 10, 10, 10,  0,-10},
3{-10, 10, 10, 10, 10, 10, 10,-10},
2{-10,  5,  0,  0,  0,  0,  5,-10},
1{-20,-10,-10,-10,-10,-10,-10,-20}
   a   b   c   d   e   f   g   h
```

Obr. 13 Hodnoty střelců

Zdroj: INTELLIJ, vlastní zpracování

Střelci, stejně jako jezdci, mají omezenou účinnost z toho důvodu, že mají omezenou mobilitu, tudíž hodnoty po stranách a primárně v rozích šachovnice mají větší negativní hodnotu. Preferují se střední pole pro dobrou kontrolu nad hrou a čtverce *b3*, *c3*, *f3* a *g3* pro krytí figur v úvodní fázi hry.

Věž

```
8{ 0,  0,  0,  0,  0,  0,  0,  0},
7{ 5, 10, 10, 10, 10, 10, 10,  5},
6{-5,  0,  0,  0,  0,  0,  0, -5},
5{-5,  0,  0,  0,  0,  0,  0, -5},
4{-5,  0,  0,  0,  0,  0,  0, -5},
3{-5,  0,  0,  0,  0,  0,  0, -5},
2{-5,  0,  0,  0,  0,  0,  0, -5},
1{ 0,  0,  0,  5,  5,  0,  0,  0}
   a   b   c   d   e   f   g   h
```

Obr. 14 Hodnoty věží

Zdroj: INTELLIJ, vlastní zpracování

Pro věž je důležitá řada č. 7, jejímž obsazením může pomoci v koncovkách. Čtverce d1 a e1 jsou výhodné ve střední fázi hry ke kontrole středu šachovnice. I u věží platí, že držet je po stranách šachovnice není ve většině případů výhodné.

Dáma

```
8{-20,-10,-10, -5, -5,-10,-10,-20},
7{-10,  0,  0,  0,  0,  0,  0,-10},
6{-10,  0,  5,  5,  5,  5,  0,-10},
5{ -5,  0,  5,  5,  5,  5,  0, -5},
4{  0,  0,  5,  5,  5,  5,  0, -5},
3{-10,  5,  5,  5,  5,  5,  0,-10},
2{-10,  0,  5,  0,  0,  0,  0,-10},
1{-20,-10,-10, -5, -5,-10,-10,-20}
   a   b   c   d   e   f   g   h
```

Obr. 15 Hodnoty dámy

Zdroj: INTELLIJ, vlastní zpracování

Pro dámu je nejvýhodnější střed šachovnice, pokud vezeme v úvahu její vysokou mobilitu a dlouhý dosah. Hrát s dámou po stranách desky nemá ve většině případů smysl, jelikož se nevyužije naplno její potenciál.

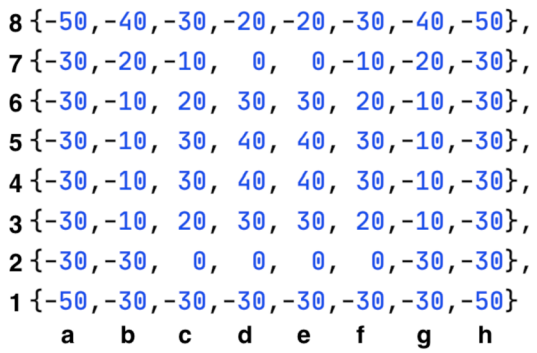
Král

```
8{-30,-40,-40,-50,-50,-40,-40,-30},
7{-30,-40,-40,-50,-50,-40,-40,-30},
6{-30,-40,-40,-50,-50,-40,-40,-30},
5{-30,-40,-40,-50,-50,-40,-40,-30},
4{-20,-30,-30,-40,-40,-30,-30,-20},
3{-10,-20,-20,-20,-20,-20,-20,-10},
2{ 20, 20,  0,  0,  0,  0, 20, 20},
1{ 20, 30, 10,  0,  0, 10, 30, 20}
   a   b   c   d   e   f   g   h
```

Obr. 16 Hodnoty krále ve střední fázi hry

Zdroj: INTELLIJ, vlastní zpracování

V úvodní a střední fázi hry by měl král zůstat v bezpečí za svojí pěšcovou linií, z tohoto důvodu jsou hodnoty od řady 3 vysoce záporné.



Obr. 17 Hodnoty krále v konečné fázi hry

Zdroj: INTELLIJ, vlastní zpracování

Naopak ke konci hry, kdy je na šachovnici málo figur, je lepší udržovat krále ve středu a ztížit soupeřovi možnost dát mat díky volnému prostoru kolem krále.

4.3 Reprezentace šachovnice

Výběr způsobu reprezentace šachovnice má zásadní vliv na rychlost výsledného enginu. S touto datovou strukturou se pracuje opakovaně v průběhu celé hry, ať už se jedná o generování možných tahů, vyhodnocování pozic, provádění a rušení pohybů či držení pozic. Z těchto důvodů je vyžadováno, aby operace probíhaly co nejrychleji.

Některé možnosti reprezentace šachovnice jsou popsány níže podle (18).

Reprezentace šachovnice založené na poli

Seznamy figur používaly jedny z prvních šachových enginů, které musely pracovat s omezeným množstvím paměti. Tyto enginy ukládaly seznamy figur v dobře prohledávatelném pořadí. Pro každou figuru byla uložena informace o pozici a legálních tazích. Seznamy byly rozděleny pro bílého a černého hráče, dále na pěšce a ostatní figury.

Čtvercový seznam nebo také dvourozměrné pole 8x8 je jedním z nejjednodušších způsobů, jak reprezentovat šachovnici. Ekvivalentem by bylo jednorozměrné pole o velikosti 64 prvků. Každý prvek obsahuje informaci o tom, jaká figura se na něm nachází a jaké je příslušnosti (bílá/černá), popřípadě informaci o tom, že dané místo je prázdné.

Bitboard

Implementace šachovnice pomocí bitboardů je oproti implementacím založeným na polích výrazně rychlejší. Bitboard je 64-bitová sekvence bitů, která nabývá hodnot nula nebo jedna. Tyto hodnoty obvykle znamenají přítomnost figury na desce, mluvíme-li o bitboardech v souvislosti s šachy. Příkladem může být vytvořený bitboard pro každý typ figury každé strany, což by znamenalo vytvoření dvanácti bitboardů. Důvod, proč jsou bitboardy značně rychlejší než reprezentace založené na poli je ten, že umožňují paralelní operace na 64-bitových procesorech. Tím se maximalizuje využití používaného hardwaru, jelikož dnes jsou 64-bitové procesory běžným standardem takřka ve všech počítačích.

4.4 Generování tahů

Při generování nových tahů je třeba brát ohled na to, jestli se zahráním nějakého tahu nedostane náš král do šachu. Generování nových tahů můžeme rozdělit na generování tahů legálních a pseudolegálních. Generování pseudolegálních tahů vygeneruje všechny možné tahy každé figury bez ohledu na to, zda tento pohyb uvede krále v šach. Testování toho, zda je král v šachu, se řeší mimo algoritmus generování tahů. Zatímco generování legálních tahů, jak už z názvu napovídá, značí to, že se vygenerují pouze legální tahy. Znamená to, že všechny vygenerované tahy jsou legální a neuvádí krále v šach. Aktuálně generované tahy jsou generované pouze pro jednoho hráče, který je právě na tahu. Tahy je nutné vygenerovat pro počítač, ale také pro hráče, jelikož je třeba ověřit, že hráčem zahraný tah je legální. (19)

5 Implementace

Aplikace je napsaná v jazyce Java. Jako vývojové prostředí byla použita INTELLIJ IDEA 2020.1 Ultimate.

Aplikace se ovládá pomocí myši. Po spuštění uživatel vybírá barvu, za kterou si přeje hrát. Pokud zvolí černé, aplikace automaticky táhne, v případě volby bílé barvy uživatel zahajuje partii.

Zdrojový kód aplikace je členěn do balíčku **chess** a do balíčku **pieces**. Struktura aplikace je znázorněna na obrázku níže.

Obr. 18 Struktura aplikace
Zdroj: INTELLIJ, vlastní zpracování

5.1 Popisy tříd

Chess

Tato třída je spustitelná. Obsahuje algoritmus Alfa-beta ořezávání, metodu pro ověření, zda je král v šachu, a metodu pro generování tahů. V této třídě také probíhá rozhodování o tom, který hráč dostane bílé figury.

Chessboard

Třída, která reprezentuje šachovnici, obsahuje její výchozí nastavení a metodu *mirrorBoard,* která šachovnici zrcadlově otočí.

Evaluation

Tato třída ohodnocuje pozice pomocí metody *getEvaluation*, která využívá doplňující metody k celkovému ohodnocení jako je hodnocení pozic, hodnocení útoku a hodnocení materiální. Tato třída také obsahuje piece-squares-tables, které jsou popsány v kapitole 4.2.2.

UI

Třída, která zodpovídá za události spojené s myší. Mezi zachytávané události patří stisk myši (mouse pressed), které nám v případě kliknutí na naši figurku zvýrazní pole, na která můžeme táhnout. Další událostí je puštění tlačítka myši (mouse released). Tato událost nám slouží k umístění figury na pole, ve kterém jsme myš pustili. V reakci na to hraje počítač svůj vlastní tah. Je zde také algoritmus pro vykreslení šachovnice.

Balíček **pieces** obsahuje 6 tříd, pro každou figuru jednu. Každá z těchto tříd v sobě uchovává implementaci pro pohyb dané figury.

5.2 Šachovnice

V této práci je šachovnice reprezentována dvourozměrným polem 8x8, tudíž 64 poli. Každé z polí může obsahovat buď figuru v podobě Stringu[3] nebo mezeru v podobě Stringu. Mezera značí, že v danou chvíli se na šachovnici nic nenachází. Velká písmena jsou pro hráče s bílými figurami a malá pro hráče s černými figurami. Algoritmus, který hledá nejlepší možné tahy, prochází šachovnici zdola nahoru. Po nalezení nejlepšího

[3] String se používá pro uchování textových řetězců.

možného tahu se šachovnice zrcadlově překlopí a algoritmus hledá nejlepší tah pro druhého hráče. Zrcadlové překlápění šachovnice umožňuje implementovat většinu postupů pouze jednou a není třeba implementovat např. pohyb figur pro černého a bílého hráče zvlášť, po zahrání bílou stranou se šachovnice zrcadlově přetočí. Velkou výhodou této reprezentace je přehlednost a snadné rozmístění figur v případě testování.

```
{{"r","n","b","q","K","b","n","r"},
 {"p","p","p","p","p","p","p","p"},
 {" "," "," "," "," "," "," "," "},
 {" "," "," "," "," "," "," "," "},
 {" "," "," "," "," "," "," "," "},
 {" "," "," "," "," "," "," "," "},
 {"P","P","P","P","P","P","P","P"},
 {"R","N","B","Q","K","B","N","R"}}
```

Obr. 19 Ukázka šachovnice

Zdroj: INTELLIJ, vlastní zpracování

Písmena jsou převzata z anglických názvů. V tabulce níže jsou uvedeny jejich české překlady.

anglický název	český překlad	bílý hráč	černý hráč
Pawn	Pěšec	P	p
Rook	Věž	R	r
Knight	Jezdec	N	n
Bishop	Střelec	B	b
Queen	Dáma	Q	q
King	Král	K	k

S šachovnicí úzce souvisí metoda *mirrorBoard*, která má za úkol figury na šachovnici zrcadlově převrátit a změnit jejich barvu. Výhodou používání této metody je to, že stačí naimplementovat pohyb pouze pro jednu barvu figur a poté pouze zrcadlově přetočit šachovnici.

5.3 Algoritmus Alfa-beta ořezávání

Algoritmus Alfa-beta ořezávání je volán s parametry hloubka, alfa, beta, move a player. Parametr hloubky nám udává, do jaké úrovně, resp. kam až bude algoritmus hledat. Při každé iteraci algoritmu se parametr hloubky zmenší o jedna, čímž se zajistí, že algoritmus půjde od nejnižší úrovně nahoru. Parametry alfa a beta se inicializují na nejnižší, resp. nejvyšší hodnotu. V algoritmu se alfa snaží najít co nejvyšší hodnotu, proto se inicializuje na nejnižší hodnotu, zatímco parametr beta se snaží najít nejnižší hodnotu, a proto se počáteční hodnota nastaví na co nejvyšší. Parametr player značí, pro kterou barvu je daná hodnota vrácena, a parametr move obsahuje String pohybů, do kterého se postupně přidávají další.

5.4 Generování tahů

V této implementaci se generují tahy legální. To znamená, že všechny vygenerované tahy jsou validní a není nutné ověřovat, zda je král po zahrání v šachu. Tato podmínka se provádí při generování tahů. Algoritmus prochází všechny pole na šachovnici. V případě, že narazí na figuru, která má barvu, která je na tahu, zjistí všechny legální tahy dané figury z dané pozice a uloží je do proměnné typu String. Příkladem takového zápisu by mohlo být *„7151“*, který reprezentuje tah ve formátu *„původní řádek, původní sloupec, nový řádek, nový sloupec“*, tudíž tento konkrétní zápis by znamenal pohyb figurou z „*a1*“ na „*a2*“.

6 Testování

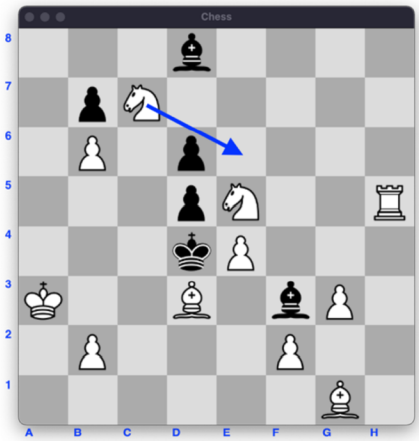
Aplikace byla testována na zařízení MacBook Pro 2019 s čtyřjádrovým procesorem Intel Core i5 1,4 GHz.

6.1 Testování tahů

Testování tahů ověřuje, zda výsledná aplikace dokáže najít správnou posloupnost legálních tahů. Testovací úlohy jsou převzaty z chesspuzzlesonline.com (20)

6.1.1 Test č. 1

Test č.1 ověřuje, zda aplikace dokáže najít tah, kterým lze dosáhnout matu. Na řadě je bílý hráč. Aplikace nachází správný tah jezdcem na *e6* za *618ms*.

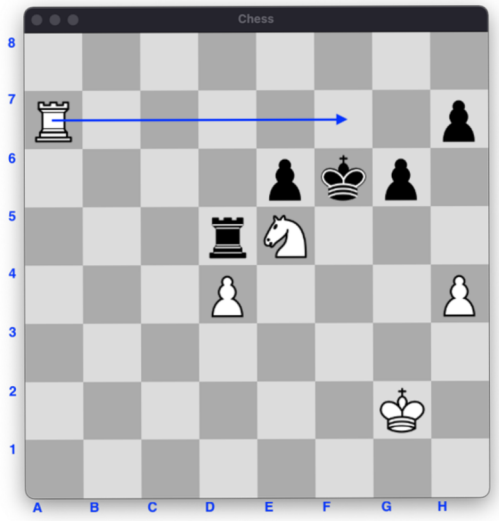


Obr. 20 Testovací pozice č. 1

Zdroj: vlastní zpracování

6.1.2 Test č. 2

Test č. 2 testuje, zda aplikace dokáže najít správný tah. Na řadě je bílý hráč. Správný tah věží na *f7* nachází za *773ms*.

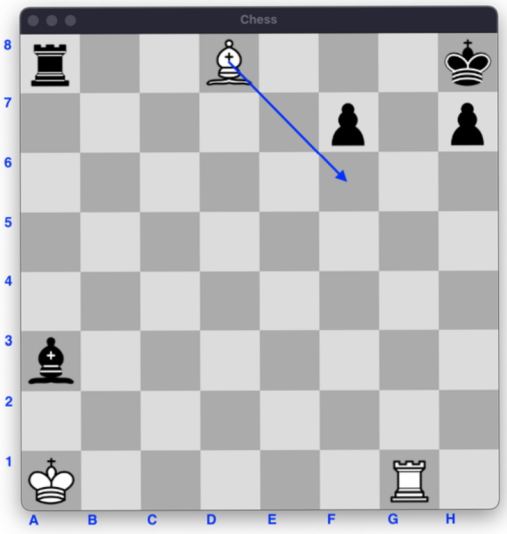


Obr. 21 Testovací pozice č. 2

Zdroj: vlastní zpracování

6.1.3 Test č. 3

Test č. 3 testuje, zda aplikace dokáže najít správný tah. Na řadě je bílý hráč. Správný tah střelcem na *f6* nachází za *688ms*.

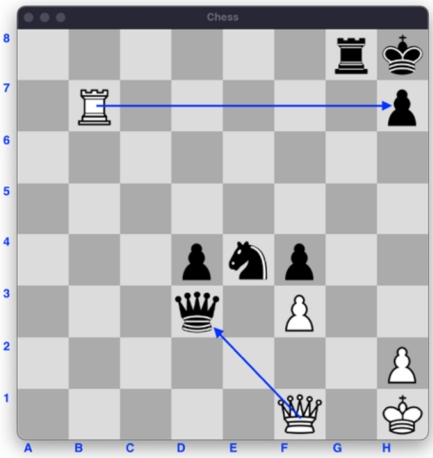


Obr. 22 Testovací pozice č. 3

Zdroj: vlastní zpracování

6.1.4 Test č. 4

Test č.4 se zaměřuje na materiální výhodu. Na tahu je bílý, správný tah věží na *h7* nachází aplikace za *995ms*. Černý nemá jinou možnost než sebrat věž králem, následně bílý hráč bere dámu tahem na *d3*, tento tah nachází aplikace za *74ms*.



Obr. 23 Testovací pozice č. 4

Zdroj: vlastní zpracování

6.1.5 Test č. 5

Test č.5 se zaměřuje na materiální výhodu. Na tahu je bílý, aplikace nachází správný tah střelcem na *d3* za *1050ms*. Černý může buď sebrat střelce dámou, uhnout králem nebo předsunout pěšce. Aplikace zareagovala sebráním střelce dámou. Bílý reaguje pěšcem na *d3* za *112ms* a bere černou dámu, tímto tahem získává bílý hráč materiální výhodu.

Obr. 24 Testovací pozice č. 5

Zdroj: vlastní zpracování

6.2 Testování úvodní fáze hry

Při opětovném hraní stejného zahájení si lze povšimnout, že aplikace reaguje stejně. Je to způsobeno tím, že při stejném zahájení aplikace prochází možné tahy stejným způsobem, což má za následek to, že se najde vždy stejný tah a ten je zahrán. Tento problém by se dal vyřešit přidanou implementací databáze šachových zahájení, díky kterému by hra v úvodních fázích byla rozmanitější.

6.3 Hloubka střední fáze hry

Ve střední fázi hry se obvykle vyskytují rozsáhlé výměny figur. Vzhledem k relativně nízké hloubce prohledávání může dojít k chybám, jelikož výměny nejsou dopočítány do konce. Aplikace může například vyměnit hodnotnější figuru za méně hodnotnou, protože tah, kterým bychom figuru vzali, již neležel v dané hloubce prohledávání.

6.4 Hloubka algoritmu Alfa-beta

Značným vylepšením aplikace by bylo prohledávání do větší hloubky, pokud by se ovšem hloubka zvýšila bez úpravy algoritmu, vyhledávání by trvalo příliš dlouho. Aktuální verze aplikace má nastavenou hloubku prohledávání na 5 tahů.

7 Závěr

První část bakalářské práce obsahuje stručnou historii šachů a popisuje pravidla této hry. Další část se zabývá algoritmy, jejich výhodami, nevýhodami a jejich názorným použitím v kontextu šachů. Následuje kapitola Šachy z pohledu počítače, ve které byly zmíněny klíčové rysy šachových enginů pro správné zahrání partie. Kapitola Implementace popisuje vývoj aplikace. Je zde popsána struktura aplikace a vybrané pasáže jsou zde detailněji popsány. Poslední kapitola se zabývá testováním výsledné aplikace. Obsahuje poziční testy, které testují, zda aplikace dokáže najít správný tah na předložené šachové úlohy. Také popisuje možná vylepšení aktuální verze hry.

Cílem bakalářské práce bylo objasnit zásadní aspekty programování šachových enginů, ty jsou popsány v úvodních kapitolách. Dalším cílem bylo naimplementovat počítačovou hru šachy. Aplikace, která vznikla v rámci této práce, se svojí sílou nemůže rovnat profesionálním šachovým aplikacím, to ale nebyl hlavní účel této práce. Výsledná aplikace napsaná v jazyce Java umožňuje hrát šachovou partii proti člověku, který ji ovládá pomocí myši. Okno aplikace je dostatečně velké, aby bylo pro uživatele srozumitelné a přehledné. Při spuštění je uživatel dotázán, za jakou barvu si přeje hrát. Po zahrání tahu uživatelem aplikace automaticky reaguje tahem vlastním. Použití algoritmu Alfa-beta ořezávání zajišťuje, že algoritmus pro hledání nejlepších tahů zbytečně neprochází všechny větve herního stromu. Testování aplikace odhalilo několik problémů, které jsou popsané v kapitole 6. Ke každému problému bylo naznačeno jeho možné řešení. Dalším rozšířením aplikace by mohla být detailnější ohodnocovací funkce, která by mimo jiné zohledňovala různé fáze hry. V případě vyřešení těchto problémů by došlo k urychlení vyhledávání nových tahů, což by přispělo ke zkvalitnění hry.

8 Bibliografie

1. **chesstantra.com.** Chess Facts. *chesstantra.com.* [Online] [Citace: 28. duben 2021.] www.chesstantra.com.

2. **Avilés, Alfonso Ochoa.** Are there really more possible Chess games than atoms in the Universe? *Alfonso Ochoa Avilés.* [Online] 22. červenec 2020. [Citace: 27. duben 2021.] https://alfonso-ochoa.medium.com/are-there-really-more-possible-chess-games-

3. **Pliska, Karel.** *Učebnice šachu pro samouky: začátečníci s historií a pravidly hry.* Frýdek Místek 2006. ISBN 80-85232-57-x.

4. **wikipedia.org.** Dějiny šachové hry. *wikipedia.org.* [Online] 14. říjen 2020. [Citace: 27. 4 2021.] https://cs.wikipedia.org/wiki/Dějiny_šachové_hry

5. **chess.com.** The 10 Most Important Moments In Chess History. *chess.com.* [Online] . srpen 2018. [Citace: 27. duben 2021.] https://www.chess.com/article/view/the-10-most-important-moments-in-chess-history#mad-queen.

6. **wikipedia.org.** Rules of chess. *wikipedia.* [Online] 10. duben 2021. [Citace: 28. duben 2021.] https://en.wikipedia.org/wiki/Rules_of_chess.

7. **chess.com.** How to Play Chess | Rules + 7 Step to begin. *chess.com.* [Online] 15. duben 2020. [Citace: 30. 11 2020.] https://www.chess.com/learn-how-to-play-chess.

8. **chess.com.** En Passant. *chess.com.* [Online] [Citace: 27. duben 2021.] https://www.chess.com/terms/en-passant.

9. **baeldung.com.** Introduction to Minimax Algorithm with a Java Implementation. *baeldung.* [Online] 3. říjen 2020. [Citace: 23. srpen 2020.] https://www.baeldung.com/java-minimax-algorithm.

10. **javatpoint.com.** Mini-Max Algorithm in Artificial Intelligence. *javatpoint.com.* [Online] [Citace: 28. duben 2021.] https://www.javatpoint.com/mini-max-algorithm-in-ai.

11. **Hartikka, Lauri.** A step-by-step guide to building a simple chess AI. *freeCodeCamp.* [Online] 30. březen 2017. [Citace: 29. duben 2021.] https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/.

12. **geeksforgeeks.org.** Minimax Algorithm in Game Theory | Set 4 (Alpha-Beta Pruning). *geeksforgeeks.org.* [Online] 12. květen 2019. [Citace: 23. srpen 2020.]

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/.

13. **javatpoint.com.** Alpha-Beta Pruning. *javatpoint.com.* [Online] [Citace: 28. duben 2021.] https://www.javatpoint.com/ai-alpha-beta-pruning.

14. **Heineman, George T., Pollice, Gary a Selkow, Stanley.** *Algorithms in a Nutshell a practial guide.* Sebastopol : O'Reilly Media, 2016. Sv. second edition.

15. **chess.com.** Elo Rating System. *chess.com.* [Online] [Citace: 28. 11 2020.] https://www.chess.com/terms/elo-rating-chess.

16. **Cuenca, Pepe.** How do chess engines think? *chess24.* [Online] 24. duben 2017. [Citace: 30. leden 2021.] https://chess24.com/en/read/news/how-do-chess-engines-think.

17. **chessprogramming.org.** Simplified Evaluation Function. *Chess programming wiki.* [Online] 16. duben 2018. [Citace: 30. leden 2021.] https://www.chessprogramming.org/Simplified_Evaluation_Function.

18. **wikipedia.org.** Board representation (computer chess). *wikipedia.org.* [Online] 29. leden 2021. [Citace: 20. 4 2021.] https://en.wikipedia.org/wiki/Board_representation_(computer_chess).

19. **chessprogramming.org.** Move Generation. *Chess Programming Wiki.* [Online] 12. březen 2021. [Citace: 24. duben 2021.] https://www.chessprogramming.org/Move_Generation.

20. **chesspuzzlesonline.com.** Solution to Chess Puzzles. *Chess Puzzles and Tactics.* [Online] [Citace: 13. duben 2021.] https://chesspuzzlesonline.com/solution/.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Jan Chaloupka**
Studium:	I1800173
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Implementace hry šachy**
Název bakalářské práce AJ:	Chess game implementation

Cíl, metody, literatura, předpoklady:

Cíl: Seznámit se s problematikou šachových algoritmů a jejich následující implementací ve vlastním programu v jazyce Java.

Osnova:

1. Úvod
2. Šachové algoritmy
3. Šachy z pohledi počítače/AI
4. Vývoj/implementace
5. Popis výsledného programu
6. Seznam použité literatury

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	doc. RNDr. Petra Poulová, Ph.D.
Datum zadání závěrečné práce:	14.1.2018