FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

ALGORITMY PRO VYTVÁŘENÍ HER NA POČÍTAČI

Západočeská Univerzita V Plzni
Katedra Kybernetiky
Semestrální práce z HKUI

Andrea Kadlecová
2. ročník 1. semestr
20. října 2020

1 Algoritmy pro hry na počítač

Hraní her je oblíbená volnočasová aktivita. Bohužel deskové hry jsou velmi neskladné, ale takové mobily či notebooky máme v posledních letech pořád u sebe. Takové klasické hry pro dva jako jsou např. šachy, dáma, piškvorky se dají lehce naprogramovat pomocí algoritmů, které dokáží prohledat celý stavový prostor, či alespoň nějakou jeho část do určité hloubky.

Proč se používá metoda pro prohledání stavového prostoru? Člověk právě takovýmto způsobem přemýšlí. Jde vlastně o promýšlení všech možných tahů, které může hráč, který je právě na tahu, udělat, aby se přiblížil k vítězství.

Nejznámějšími funkcemi pro prohledávání stavového prostoru je **metoda Minimax**, která prohledává celý stavový prostor, nebo **Alfa-beta prořezávání**, která právě prohledává jen část stavového prostoru.

1.1 Trocha historie

John Von Neumann v roce 1928 napsal článek s názvem Theory of Parlor Games. Sám byl vášnivý hráč pokeru a chtěl nějak formalizovat strategii blafování. V tomto článku poukázal na teorii her a algoritmus Minimax. Byl si vědom toho, že tato teorie pomůže v ekonomické sféře, a proto s ekonomem Oskarem Morgensternem napsal knihu Theory of Games and Economic Behavior. Kniha se dotkla nejen ekonomiky, ale také třeba sociologie a politiky, také samozřejmě i teorie her.

Další osobností, která významně pomohla k vývoji algoritmů pro hraní her byl **Alan Turing**. Zaměřil se o naprogramování hry šachy. Napsal rozsáhlý algoritmus na papír, který se však neúspěšně přepsal do počítače Tudíž Alan Turing si nikdy nevyzkoušel svůj program na počítači, ale pouze jen na papíru. Algoritmus byl schopný hrát proti člověku, ale nikdy ne vyhrát.

Nesmím opomenout ani **Johna Nashe**, který dokázal, že každá konečná hra má alespoň jedno řešení, kdy ani jeden hráč není schopen táhnout ve svůj vlastní prospěch (= remíza).

Posledním člověkem, na kterého nesmím zapomenout, je **John McCarthy**. Zasloužil se o vytvoření algoritmu alfa-beta prořezávání.

1.2 Stavový prostor

Nejdřív bych ráda přiblížila spojení "stavový prostor". Hra je znázorněna stromem, kdy jednotlivé uzly stromu představují jednotlivé pozice hry a hrany představují přípustné tahy podle pravidel hry. Listy stromu odpovídají koncovým stavům hry. Takový konečný stav může dopadnout výhrou jednoho a prohrou druhého hráče, nebo remízou.

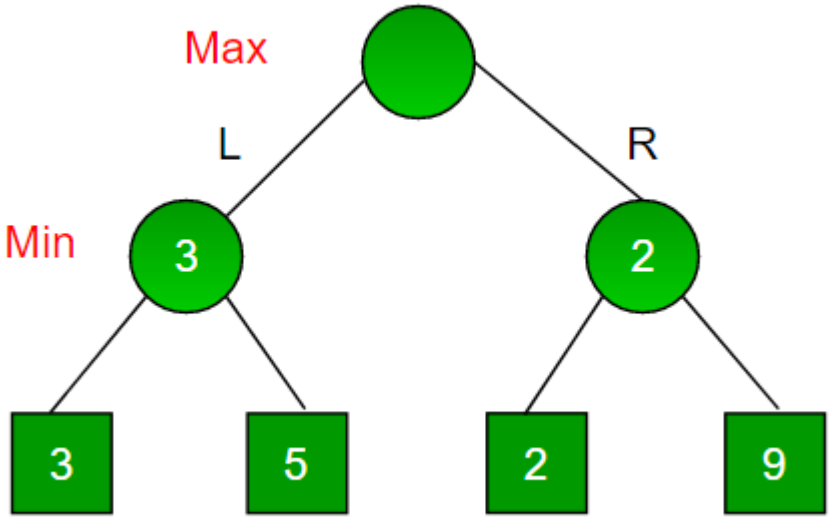
1.3 Minimax

Jak jsem psala výše algoritmus Minimax prohledává celý stavový prostor. Používá tzv. backtracking, což je metoda, která si pamatuje vždy jen jedinou cestu stromu vedoucí od kořene k poslednímu vygenerovanému uzlu. Backtracking je nevhodný pro náročné úkoly, kdy se "může vydat na nekonečnou cestu", a tedy nemusí nikdy dojít k řešení. Proto se pomocí algoritmu Minimax řeší jen jednodušší hry, aby nedošlo k nekonečnému procházení stromu řešení.

Jedná se o hru se dvěma hráči, které si nazveme **maximátor (= hráč)** a **minimátor (= protihráč)**. Maximátor se snaží maximalizovat svoji šanci na vítězství a minimátor se snaží minimalizovat maximátorovo šanci na skóre.

Stavový prostor má takovou velikost, že ho zvládneme prohledat úplně celý a dojít k řešení. Hra končí ve chvíli, kdy vyhrává maximátor, či minimátor, pokud maximátor udělal během hry nějakou chybu, nebo remízou.

Pak se může stát, že při komplikovanějších hrách je stavový prostor až moc velký a prohledat ho celý by bylo velmi neefektivní a zbytečné. V tomto případě je stavový prostor prohledáván do předem definované hloubky, ale nedostaneme vítěze, proto se vítěz určuje pomocí hodnot, které jsou každému uzlu přiřazeny. Na základě těchto hodnot lze určit, zda je ten daný stav výhodnější pro maximátora, minimátra, nebo jde o remízu.



Obr. 1 *Příklad algoritmu Minimax*

1.4 Alfa-beta prořezávání

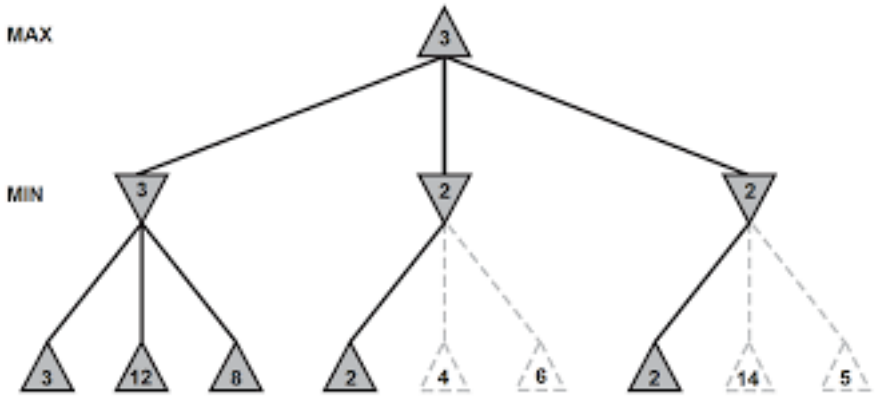
Alfa-beta prořezávání není není nějaký nový algoritmus. Spíš se jedná o drobnou modifikaci Minimaxu. Někdy se může stát, že určitou část stromu vůbec nepotřebujeme, jelikož víme, že řešení, které se tam budou vyskytovat nebudou ideální. Díky tomu můžeme řešení najít mnohem rychleji i v nějaké větší hloubce, kam bychom se za použití Minimaxu dostali za velmi dlouhou dobu.

V grafu pohybujeme se dvěma hodnotami. Hodnota **alfa**, která udává ohodnocení maximátoru, a hodnota **beta**, což je ohodnocení minimátoru. Program se ukončí, na jakémkoliv uzlu za splnění jedné ze dvou podmínek, které jsou:

- Když je beta menší, nebo rovna alfě jakéhokoliv maximátorova předchozího uzlu.
- Když je alfa větší, nebo rovna betě jakéhokoliv minimátorova předchozího uzlu.

V nejhorším případě se může stát, že neořežeme žádnou část stromu.

V nejlepším případě ořežeme strom tak, že stihneme projít zbytek stromu dvakrát tak rychleji. Závisí to na pořadí prováděných tahů.



Obr. 2 *Příklad Alfa-beta prořezávání*

2 Zdroje

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduct
https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-bet
?ref=lbp
https://www.kiv.zcu.cz/studies/predmety/uir/predn/P2/FThema2_
hry.pdf
https://docplayer.cz/5007482-Algoritmy-pro-hrani-tahovych-her.
html
https://www.yumpu.com/xx/document/read/18046437/algoritmus-minimax
https://towardsdatascience.com/how-a-chess-playing-computer-thinks-about-its-n
https://cs.stanford.edu/people/eroberts/courses/soco/projects/
1998-99/game-theory/neumann.html
https://www.history.com/news/in-1950-alan-turing-created-a-chess-computer-prog
http://www.simulace.info/index.php/Nash_equilibrium/cs
https://www.chessprogramming.org/John_McCarthy