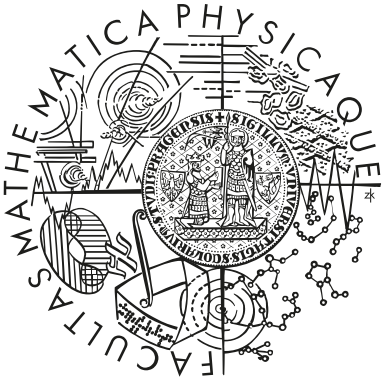
Univerzita Karlova

Matematicko-fyzikální fakulta

Studijní program: Informatika



Dokumentace k zápočtovému programu
Minimax ve hře Tic-Tac-Toe

2022/2023 **Jiří Daněk**

1. Anotace:

Návrh známé hry Tic-Tac-Toe v anglickém jazyce, která se odehrává v příkazovém řádku počítače za použití programovacího jazyka Python3. Vyskytují se zde různé módy, tedy verze pro jednoho hráče proti algoritmu Minimax, proti generátoru náhodných tahů a dále i multiplayer. Celý kód hry je napsán pod třídou Game, která obsahuje potřebné funkce pro exekuci.

Annotation:

Design of the well-known game Tic-Tac-Toe in English language, which takes place in the command line of a computer using programming language Python3. There are different modes, i.e. single player version against Minimax algorithm, against random move generator and also multiplayer. The entire game code is written under the Game class, which contains the necessary functions for execution.

2. Zadání:

Návrh a implementace algoritmu Minimax do hry Tic-Tac-Toe. Hra se odehrává v terminálu zařízení, kde vstupem jsou souřadnice jednotlivých polí (řádek, sloupec). Výstupem je stav, kterým hra byla ukončena (výhra, prohra, remíza).

3. Zvolený algoritmus a datové struktury:

Zvolen byl zmiňovaný algoritmus Minimax. Jako datové struktury v programu byly použity Python List, dále jen jednotlivé proměnné.

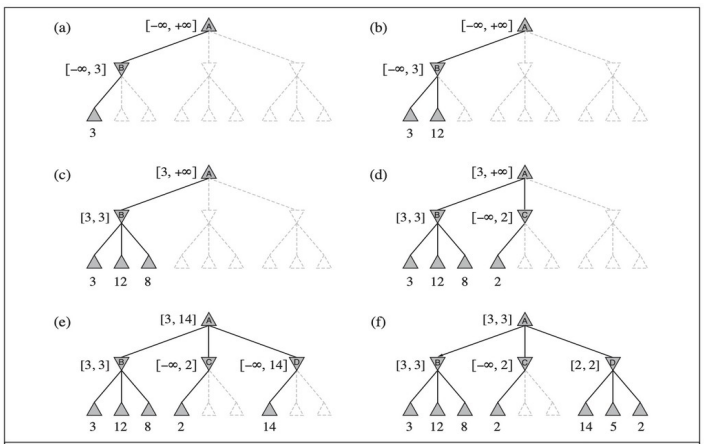
Minimax

Popis: Rozhodovací algoritmus Minimax funguje na principu DFS – Deep First Search, tedy prohledávání stromu hry do hloubky. Slouží k nalezení nejlepšího možného tahu pomocí vybírání minimálních a maximálních hodnot, což odpovídá nejlepší strategii hry. Uplatnění nalezne ve vícehráčových hrách s nulovým součtem – jeden hráč něco získá, druhý musí ztratit, např.: šachy, piškvorky, nebo hra Go.

Složitost: Algotirmus má paměťovou složitost lineární O(m) pro m, které představuje hloubku stromu. Časová složitost je naopak exponenciální, což přináší velký problém pro mohutnější hry, než je Tic-Tac-Toe – (3x3).

Optimalizace: V případech, kdy je stavový prostor ohromně velký, se strom hry prochází pouze do určité hloubky, kdy je použita ohodnocovací funkce na principu heuristik. Další možností je použití Alfa-Beta prořezávání, které dále nevyhodnocuje větve stromu, které přinesou jistě horší výsledek.

Implementace: Algoritmus byl v programu implementován a rozdělen pro přehlednost a jednoduchost do dvou podfunkcí, tedy minimalizační (min) a maximalizační (max). Pořadí, ve kterém budou volány, záleží na volbě prioritního tahu. Tento problém obstarává jim předcházející funkce chooseMoveMinimax, která navíc vytvoří kopii dosavadní hrací plochy, kde bude algoritmem vyhodnocováno hledání optimálního tahu.




```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

Figure 5.3 An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation $\text{argmax}_{a \in S} f(a)$ computes the element a of set S that has the maximum value of $f(a)$.

Python List

Popis: List je v programovacím jazyce Python3 základní datová struktura, která je měnitelná, rozšiřitelná a uspořádaná. Uvnitř se mohou nacházet různé opakující se hodnoty i datové typy. Definuje se hranatými závorkami – [ ], v nichž se nacházejí jednotlivé prvky indexované od 0. Umožňuje základní operace jako přidání, vyhledání, odebrání, řazení, sjednocení, atd.

Složitost: Většina operací na listech má linearní složitost, výjimkou je například operace pop(), nebo append(), která má v průměrném případě složitost konstatní.

Implementace: V programu byla datová struktura použita především pro reprezentaci hrací plochy, která je ve formě dvoudimenzionálního pole (pole polí), v němž se indexuje pomocí dvou hranatých závorek. Dále byl list využit pro kontrolu zadaných souřadnic. Využití mimo jiné našel pro vracení nejvhodnější kombinace v algoritmu Minimax a pro vracení funkce rozhodující jaká kombinace znaků je ta výherní.

4. Diskuze výběru algoritmu:

Alternativou k algorimu Minimax je MCTS – Monte Carlo Tree Search, který narozdíl od Minimaxu u mohutných stromů her nepotřebuje heuristiku (ohodnocovací funkci) a opírá se o náhodný přístupu k tahům, což v omezeném čase může být mnohem výhodnější, než postupovat do stejné hloubky všech větví a to i za použití Alfa-Beta prořezávání. V každé další hloubce je více uzlů, než bylo doposud prohledáno.

V tomto programu byl napříč této výhodě použit algoritmus Minimax. Jelikož se jedná o malý stavový prostor, samotný strom hry také nebude příliš velký, narozdíl od her jakými jsou šachy či Go. Algorimus Minimax dokáže v rozumném čase, řád sekund, prohledat celý stavový prostor a vyhodnotit nejlepší následující tah a to dokonce bez použití Alfa-Beta prořezávání. Zde by se jednalo o čas menší než je jedna sekunda.

5. Program:

Knihovny: Pro příjemnější uživatelský zážitek byly do programu importovány tři knihovny – random, os, time. Knihovna random slouží pro mód, kdy uživatel soupeří s počítačem, který vybírá tahy pseudonáhodně. Knihovna os byla použita pro postupné mazání obsahu terminálu, tak aby uživatel viděl jen potřebné údaje k obsluze hry. Knihovna time slouží k využití zpoždění při animaci představení hry a přecházení mezi jednolivými stavy. Žádná z těchto knihoven tedy není nutná pro správný běh softwaru hry, slouží pro estetickou stránku.

Struktura: Program je rozdělen na části:

- import knihoven – random/os/time,
- třídu – Game obsahující potřebné exekutivní funkce,
- funkce hlavní cyklus hry – playTicTacToe(),
- opakující globální se proměnné,
- vytvoření třídy a zavolání hlavního cyklu hry.

Pro začátek hry je nezbytná tvorba třídy Game, která je uložena v proměnné g. Při tomto procesu se také provede inicializační funkce __init__(), která vytvoří hrací plochu, provede počáteční animaci – funkce introduction(), která nastaví mód a symboly hráčů na hodnotu None. Dále pouze proběhne zavolání funkce playTicTacToe(), která se postará o zbytek běhu programu.

Funkce playTicTacToe() obshauje nekonečný cyklus hry, který ukončí vstup – quit, v počátečním menu hry. V tomto nekonečném cyklu se opakuje výběr módu – funkce menu(), inicializace reprezentace tahů hráčů – funkce chooseSymbols(), následně dle vybraného módu se spustí fuknce starající se o samotnou hru – multiGameStart(), nebo minimax_randomGameStart(). Po konečném stavu hry se pouze zavolá funkce emptyBoard(), která uvede hrací plochu do základního stavu a cyklus hry se opakuje, dokud uživatel nepředá na vstup hodnotu quit.

Celý přiložený kód je opatřen množstvím komentářů v českém jazyce, které slouží k bližšímu pochopení jednotlivých příkazů. Jednotlivé funkce jsou výstižně pojmenovány v anglické analogii a rozděleny do co nejmenších logických bloků tak, aby byla umožněna jednoduchá srozumitelnost a čtení kódu. Dále také pro opakované volání stejné částí programu, což kód uchovává DRY – Do not Repeat Yourself.

6. Alternativní řešení programu:

Zvažována byla implementace algoritmu Minimax pomocí jedné stejnojmenné funkce, která nakonec byla zavhrnuta pro menší přehlednost. Tato funkce by obsahovala hlavní časti funkce min a max, navíc by byl přidán bool příznak is_maximazing, který by upřestňoval jaký stav je na vstupu funkce a poté i její chování.

7. Reprezentace vstupních dat:

Vstupními daty jsou:

- [enter] pro posunutí do dalšího stavu hry
- [rules] pro případné vypsání pravidel v introduction()
- [minimax/multi/random] pro výběr stejnějmenného módu hry v menu()
- [quit] pro opuštění hry v menu()
- jakýkoliv znak/skupina znaků pro reprezentaci tahů hráčů v chooseSymbols()
- [F] pro počáteční tah v minimax_randomGameStart()
- vybrané znaky pro tahy hráčů rozhodující začínajícího v multiGameStart()
- [0-2] celé číslo reprezentující index řádku v chooseMove()
- [0-2] celé číslo reprezentující index sloupce v chooseMove()

8. Reprezentace výstupních dat:

Výstupními daty jsou:

- animace uvítaní uživatele, název hry
- pravidla hry, v případě vstupu [rules]
- výběr herního módu dle zadaného vstupu [minimax/multi/random]
- konec hry v případě vstupu [quit]
- zobrazení dosavadního stavu hry na hrací ploše
- zobrazení reprezentace jednotlivých hráčů, kdo je na tahu
- zobrazení umístěného znaku dle zadaných souřadnic
- chybové hlášky a jejich náprava při špatných vstupech
- výsledný stav hry – výhra hráče/remíza

9. Co nebylo doděláno:

Alfa-Beta ořezávání, jelikož by to, vzhledem k rychlosti vnímání uživatele, nemělo velký vliv na zážitek ze samotné hry.

Cheaty, které by napověděli uživateli nejlepší možný tah dle algoritmu Minimax, ale z důvodu malého hracího prostoru je témě jednoznačné, kam je tah vhodné umístit.

Průběžná tabule s pamětí jakým stavem předešlé hry skončili. Možnost jednoduché implementace, ale vzhledem k malému počtu her to bylo vyhodnoceno jako zbytečné.

10. Závěr:

Jednak rozmýšlení nad implementací algoritmu a psaní samotného kódu a jednak tvorba této dokumentace mně přišla velice přínosná a bezpochyby jsem si z této práce odnesl spoustu zkušeností. Jak už z konzultace, tak ze čtení a rozmýšlení jednotlivých aplikací a implementací algoritmu. Také mi přišlo velice přínosné psaní dokumentace pomocí komentářů i této konečné, kvůli jednoduššímu představení projektu přátelům.