Herní algoritmus a grafické prostředí hry PIŠKVORKY na neomezeném 2D hracím poli

Mgr. Jaromír Matas

ČVUT-FIT

mail@jaromir.net

https://github.com/isCzech/Unlimited-Tic-Tac-Toe-Python

1. ledna 2023

1 Úkol

Práce sledovala následující cíle:

1. navrhnout hrací algoritmus alespoň na úrovni průměrného hráče,

2. vytvořit minimalistické, ale uživatelsky příjemné grafické prostředí pro lidského hráče,

3. umožnit import (pythonovské) implementace AI hráče třetí strany splňující velmi jednoduché API,

4. porovnat rychlost herního algoritmu s podobnou implementací ve Smalltalku.

2 Pravidla hry, API

Hru hrají dva hráči na neomezeném dvoudimenzionálním čtverečkovaném hracím poli, hráči se střídají v tazích, kdy v každém tahu hráč obsadí jeden libovolný volný čtvereček svým symbolem (standardně se používají symboly *křížek* a *kolečko*) s cílem obsadit svými symboly pět po sobě následujících čtverečků v libovolném směru - svisle, vodorovně nebo diagonálně. Hru zahajuje hráč s křížkem.

Pro implementaci AI hráče platí velmi jednoduché API: Herní prostředí z modulu hráče importuje funkci pojmenovanou `play`, jíž předá celočíselné souřadnice tahu protihráče a očekává celočíselné souřadnice hráčova tahu ve formě uspořádané dvojice (řádek, sloupec). Jméno modulu se zadává jako argument při spuštění hry.

3 Algoritmus

Hrací algoritmus v modulu `bot.py` vychází z reprezentace herní situace pomocí množiny pozic obsazených daným hráčem (proměnná `claimed`) a reprezentace všech potenciálně výherních sekvencí pěti po sobě následujících pozic pomocí množiny všech takových sekvencí daného hráče (proměnná `open_lines`), kde každá sekvence je reprezentována množinou jejích pěti pozic. Kolem každé pozice na prázdném hracím poli je 20 potenciálně výherních sekvencí. Příklad všech takových sekvencí kolem pozice (-1, 1) najdete v testu `test_bot`. Jakmile protihráč obsadí nějaké pole, počet jeho potenciálních výherních sekvencí se snižuje.

Algoritmus po každém tahu protihráče ohodnotí všechny možné "smysluplné" protitahy (proměnná `next_move_candidates`) a vybere tah s nejvyšším ohodnocením. Hodnocení tahu vychází z pozorování, že určité kombinace obsazených pozic mají pro hráče významnou hodnotu, např. sekvence tří symbolů v řadě s volnými pozicemi na obou koncích má vyšší hodnotu než podobná sekvence ohraničená na jednom konci symbolem protihráče a např. sekvence čtyř neohraničených symbolů již vede v dalším tahu k dosažení výherní sekvence pěti symbolů v řadě, samozřejmě pokud protihráč nemá na své straně již připravenou čtveřici symbolů v řadě a dalším tahu vyhraje.

Pro další výklad *herními vzorci* nazveme potenciálně výherní sekvence s některými pozicemi obsazenými výhradně jedním hráčem.

Ohodnocovací funkce (funkce `evaluate_board`) převádí herní vzorce a jejich četnost na index do ohodnocovací tabulky. Algoritmus pro každý zvažovaný protitah zjistí jeho hodnotu pro hráče a současně i jeho hodnotu pro protihráče a vrátí rozdíl obou hodnot. Hodnotu tahu neurčuje pouze to, jaké herní vzorce přinese hráči, ale také to, jaké herní vzorce zkazí protihráči.

Pro ohodnocovací tabulku byla použita Fibonacciho posloupnost, protože dobře aproximuje narůstající hodnotu herních vzorců a jejich četnost a dále proto, že se umožňuje velmi snadno generovat ohodnocovací tabulky pro varianty hry s vyšším počtem po sobě následujících obsazených pozic k dosažení vítězství. Tato jednoduchost je však současně cenou za to, že hrací algoritmus ve své základní podobě dosahuje pouze úrovně průměrného hráče piškvorek.

4 Grafické prostředí

Pro grafické zobrazení a interaktivní vstup byl zvolen modul `curses`, jež poskytuje jednoduché textově orientované grafické prostředí pro zobrazení na různých platformách nezávislé na žádné grafické nadstavbě.

5 Výsledek

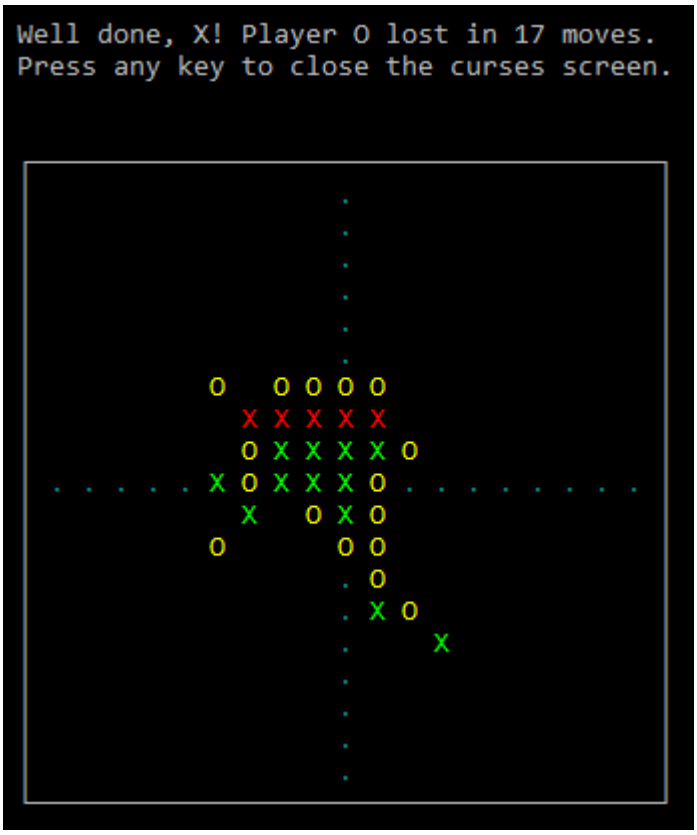
Výsledná aplikace obsahuje složku `pyskvorky` s moduly potřebnými pro hru a složku `tests` s unit testy. Celá struktura projektu vypadá takto:

```
├── pyskvorky
│   ├── __init__.py
│   ├── __main__.py
│   ├── bot.py
│   ├── cli.py
│   ├── helper.py
│   ├── pyskvorky.py
│   └── rob.py
├── tests
│   ├── __init__.py
│   ├── test_bot.py
│   └── test_helper_py
├── report_czech.pdf
└── README.md
```

Modul `pyskvorky.py` implementuje inicializaci, řídící mechanismus střídání hráčů, zobrazení průběhu hry a rozhraní pro zadávání tahů lidským hráčem. Modul `bot.py` implementuje herní algoritmus a modul

`rob.py` je vložen jako ukázka způsobu integrace externího hráče do hry. Modul tak poskytuje možnost vyzkoušet spuštění dvou herních algoritmů proti sobě. Pro jednoduchost jde pouze o kopii modulu `bot.py` s použitím Tribonacciho posloupnosti v ohodnocovací funkci a s příslušným komentářem.

Grafické provedení vypadá následovně:



6 Spuštění a ovládání

Hra se spouští z CLI a ovládá se z klávesnice pomocí šipek a Enter, případně dalšími tlačítky.

V defaultním nastavení spustíte partii mezi AI hráčem a člověkem příkazem

```
python pyskvorky
```

Pokud chcete pořadí hráčů otočit, použijte příkaz

```
python pyskvorky -r
```

Pokud chcete spustit externí implementaci hráče v modulu pojmenovaném např. `rob.py` proti defaultnímu AI hráči, nakopírujete modul rob.py do složky pyskvorky a spustíte hru příkazem

```
python pyskvorky -o rob
```

Pro další instrukce a parametry použijte příkaz

```
python pyskvorky -h
```

7 Testování

Unit testy ve složce tests lze spustit z kořenové složky příkazem

```
pytest tests
```

případně ze složky `tests` příkazem

```
pytest
```

nebo ze složky `pyskvorky` příkazem

```
pytest ../tests
```

Pro spuštění pylint testů ze složky `pyskvorky` lze použít příkaz

```
pylint pyskvorky.py bot.py cli.py helper.py --disable=C0301,C0103,R0801,R0903
--extension-pkg-allow-list=_curses
```

Unit testy lze zkontrolovat pomocí pylint ze složky `tests` příkazem

```
pylint test_bot.py test_helper.py --disable=C0301,C0103,R0801
```

Některá varování pylint byla vypnuta:

C0301 line too long (připouštíme řádky přes 80 znaků)
C0103 variables name (připouštíme krátká, i jednoznaková jména)
R0801 duplicate-code (připouštíme, protože rob.py je úmyslně kopií bot.py)
R0903 too few public methods (připouštíme třídu bez metod umyslně použitou jako datovou strukturu)
W0143: Comparing against a callable (připouštíme porovnání dvou funkcí jako workaround úmyslně použitý v kódu)

Navíc, modul `curses` musí byt explicitně zadán pomocí `--extension-pkg-allow-list=_curses`, jinak není rozpoznán.

8 Co není

pylint vrací dvě varování:

R0912: Too many branches (14/12) – v implementaci ovládání pro lidského hráče je `if/elif` konstrukt pro 14 možných situací, které se dynamicky přepočítávají. Nepodařilo se vymyslet efektivnější způsob, aniž by se úplně zatemnila čitelnost kódu.

W0603: Using the global statement – vzhledem k provázanosti mezi mechanismem řízení hráčů a zadávání tahu lidským hráčem v grafickém zobrazení (funkce `enter_move`) se nepodařilo jednoduše odstranit globální proměnné v hlavním programu bez ztížení čitelnosti. Koncepčním řešením nežádoucí provázanosti bude implementace lidského hráče do samostatného modulu nebo ještě lépe vytvořením společné abstraktní třídy pro AI i lidské hráče. Zbývající globální proměnné se pak již snadno předají jako argumenty.

Program se vyvíjel v prostředí Windows 10 a PowerShell, kde vše funguje uspokojivě. V prostředí Linuxu se mohou projevit některé nedostatky, např. barevné pruhy při zobrazení herního pole spuštěním hry ve vývojovém prostředí PyCharm.

9 Další možnosti rozvoje

Aplikace může posloužit jako základ pro další rozvíjení a experimentování. Uveďme některé možnosti:

Randomizovat výběr z množiny stejně hodnocených protitahů

Logovat partie do souboru

Implementovat hráče pomocí třídy

Zobecnit pro libovolný počet obsazených pozic v řadě k dosažení vítězství, nejen pro pět

Implementovat řídící mechanismus pomocí vláken nebo announcements

Zvýšit úroveň hráče vyhodnocením více tahů dopředu

Hra na omezený počet tahů s možnou remízou

Hra na omezeném hracím poli

Organizace turnaje

Hra na 3D hracím poli

Zobecnění pro více než dva hráče

Závěr

Navržený hrací algoritmus dosahuje zamýšlené úrovně průměrného hráče, a to při použití relativně jednoduché heuristiky, díky čemuž je snadné tento algoritmus zobecnit např. pro jiný počet po sobě následujících obsazených pozic k dosažení výhry. Příklad herní situace pro 6 pozic v řadě je v příloze.

Textové prostředí modulu `curses` poskytuje přiměřený komfort pro hru při zachování přenositelnosti mezi různými platformami (např. Windows, Linux).

Balíček s aplikací obsahuje vzorový modul rob.py jako ukázku jak jednoduše integrovat a spustit jakýkoli hrací algoritmus vytvořený v Pythonu a splňující podmínky API.

Jedním z cílů bylo porovnání rychlosti výsledného programu s podobnou implementací ve Smalltalku, případně i zkušeností při návrhu a vývoji ve Smalltalku (Squeak 6.0, https://squeak.org). Autor vycházel z hracího algoritmu, který realizoval ve Smalltalku, ale při práci v Pythonu došlo k některým vylepšením, které mohly ovlivnit výslednou rychlost. Bez ohledu na to se ukázalo, že hrací algoritmus v Pythonu je rychlejší (partie o 20 tazích trvá ve Smalltalku několik sekund, zatímco v Pythonu méně než sekundu), což může být podle názoru autora způsobeno tím, že běžící program ve Smalltalku obsahuje kód napsaný v C pouze v nezbytné minimální míře a zbytek je kompilovaný bytekód, zatímco podíl kódu napsaného v C v běžícím programu v Pythonu je obecně výrazně vyšší.

Co se týče vývoje aplikace, Smalltalk poskytuje propracovanější vývojové prostředí, zejména debugger, a čistší a předvídatelnější jazykové prostředky, což velmi usnadňuje vývoj aplikace, ale ohromnou výhodou Pythonu je rozsáhlá komunitní podpora, dostupnost detailní dokumentace a nesrovnatelně více doplňkových balíčků.

Reference, zdroje

Práce nevychází z žádných externích zdrojů kromě veřejně dostupné dokumentace, přednášek a cvičení předmětu BI-PYT na FIT ČVUT a vlastní předchozí práce autora zveřejněné na https://github.com/isCzech.

Příloha: Ukázka herní situace pro 6 pozic v řadě po úpravě algoritmu AI hráče