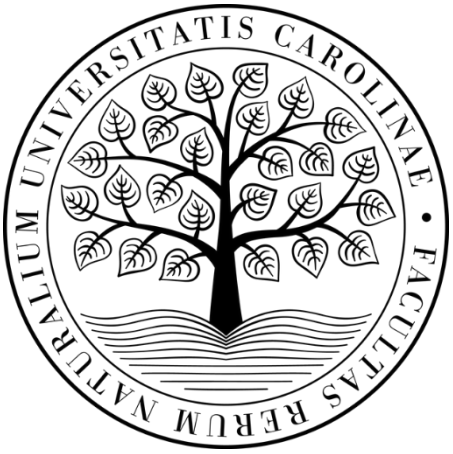
Univerzita Karlova

Přírodovědecká fakulta



Hra piškvorky

Úvod do programování

Dokumentace

Eliška Králová

2. B-GEKA

Protivín 2023

Zadání

Příklad č. 115: Hra piškvorky (Tic-tac-toe) s použitím minimax algoritmu pro hraní proti počítači.

Rozbor problému

Pravidla

Hráči se střídají v umísťování svých kamenů na hrací plán. Hráč, který má nepřerušenou horizontální, vertikální či diagonální řadu daného počtu kamenů, vyhrává. Běžně je potřebné mít v řadě 3 kameny – pro plán 3x3 a 5 kamenů – pro větší plán. S tím souvisí nejvýhodnější umístění kamenů. K tomu se využívá minimax algoritmus.

Použitý algoritmus

Minimax algoritmus se snaží maximalizovat zisk hráče a minimalizovat zisk protihráče. Možný zisk získá z ohodnocení jednotlivých polí v daném stavu hracího plánu dle rozdílu počtu kamenů v řadě mezi hráči. Následně vybere to nejvýhodnější pole. Pomocí hloubky se dá nastavit obtížnost počítače, jelikož s její pomocí vypočítává pravděpodobné budoucí tahy, kdy se protihráč snaží maximalizovat svůj zisk.

Při větším počtu polí v hracím plánu či při velké obtížnosti (hloubce propočtu) dochází ke zdlouhavému ohodnocování polí. Proto bylo za potřebí využít alfa-beta ořezávání, které tuto dobu výrazně snížilo. Alfa-beta ořezávání snižuje počet procházených tahů, nemůže-li nastat změna výsledku. Tyto tahy se již dál neprovádí.

Pseudokód algoritmu[1]

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            if value > β then
                break (* β cutoff *)
            α := max(α, value)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            if value < α then
                break (* α cutoff *)
            β := min(β, value)
        return value
(* Initial call *)
alphabeta(origin, depth, −∞, +∞, TRUE)
```

[1] Zdroj: Wikipedia.org

Struktura programu

Program používá knihovny TkInter, Math, Enum a Random. Z datových struktur jsou využity seznamy.

Třídy

Pole

Datové položky třídy

volne_pole – zápis volného pole v plánu

hrac – zápis hráče v plánu

pocitac – zápis počítače v plánu

Piskvorky

Datové položky třídy

platno – definování a nastavení proměnné pro pozdější využití v inicializátoru

sirka – definování proměnné, která je parametrem při volání funkce

vyska – definování proměnné, která je parametrem při volání funkce

počet_vyhra – definování proměnné, která je parametrem při volání funkce

obtiznost – definování proměnné, která je parametrem při volání funkce

plan – seznam polí, v němž jsou vnořené seznamy polí jednotlivých řádků

tahy – definování proměnné, pomocí které je počítán počet umístěných kamenů

Funkce

tah

Přepočítá souřadnice kliknutí na hrací pole na souřadnice 0, 0 až 2, 2 (hrací pole 3x3). Zavolá funkci **umisteni** s danými souřadnicemi. Jsou-li souřadnice již použity (pole je obsazené), upozorní program hráče do terminálu a čeká, dokud neklikne na volné políčko. Pro tah počítače nejdřív zavolá funkci **minimax**, ze které si vezme souřadnice pole a přidá je jako parametr funkci **umisteni**.

umisteni

Určuje, který hráč je na tahu a po zavolání funkce s daným znakem jej umístí na příslušné pole. V seznamu herního plánu se přepíše znak pro hráče (1 nebo -1) a na pole již nebude možné nic zapsat. Pomocí funkce **počet_v_rade** kontroluje, zda hráč na tahu nevyhrál (počítá počet kamenů v řadě pro umístěný kamen) a zaplnění hracího plánu.

krizek

Funkce krizek nakreslí křížek na dané pole dle přiřazených souřadnic z parametru.

kolecko

Funkce kolecko nakreslí kolečko na dané pole dle přiřazených souřadnic z parametru.

minimax

Dle hráče na tahu, pro kterého byla funkce volána, určí počáteční hodnotu polí a následně je v náhodném pořadí postupně prochází. Je-li hloubka rovna 0, hodnota polí se zjistí pomocí funkce **ohodnoceni_planu**. Při jiné hodnotě se spustí znovu funkce **minimax** pro druhého hráče, než byla předtím spuštěna, dokud není hloubka 0. Z ohodnocení všech polí se pro maximalizujícího hráče vezme pole s největší hodnotou a uloží se i jeho souřadnice pro umístění. V opačném případě (minimalizující hráč) se vybere pole s nejmenší hodnotou. Funkce také kontroluje, zda hráč na tahu nevyhrál.

počet_v_rade

Funkce prochází všechny možné směry (horizontální, vertikální a diagonální) a počítá počet již umístěných kamenů jednoho znaku. Zároveň kontroluje, aby neprocházela i neplatná pole pomocí funkce **je_na_planu**.

ohodnoceni_planu

Po uložení nejvyššího počtu kamenů v řadě ze všech políček na hracím plánu pro oba hráče, se z těchto hodnot vypočítá rozdíl. Výsledná hodnota se poté využije ve funkci **minimax**, kde se s její pomocí určí nejvýhodnější pole pro umístění kamene.

je_na_planu

Při procházení počtu kamenů v řadě zajišťuje tato funkce to, že se neprochází neplatná pole (souřadnice mimo hrací plán).

Alternativní programová řešení

Nejdříve jsem pro grafické znázornění používala Želví grafiku, avšak při dalším rozvíjení programu se stávaly souřadnice zmatečnými, a tak jsem přešla na TkInter.

Spuštění programu

Při svém spuštění program nejprve zkontroluje, zda jsou zadané parametry číselné a kladné. Následně se otevře okno s hracím plánem a již stačí kliknout na požadované políčko. Po kliknutí se na dané pole umístí kolečko (znak hráče) a minimax algoritmus vypočítá pole pro počítač, na které se vykreslí křížek. Takto se hráči střídají a program pokračuje, dokud jeden z hráčů nedosáhl požadovaného počtu kamenů v řadě či se nezaplnil celý hrací plán. Jakmile někdo vyhraje nebo nastane remíza, otevře se nové okno, které hráče informuje o nastalém stavu a po potvrzení se program ukončí.

Možná vylepšení

Vhodné vylepšení je uživatelské rozhraní, kde by bylo tlačítko menu a v něm příslušné možnosti jako volba počtu hráčů (2 proti sobě, hráč a počítač či 2 počítače), začít novou hru a další. Program by mohl být vylepšen o možnost výběru pořadí hráčů a výběr jejich znaku. Díky tomu by hráč neměl výhodu začínajícího hráče a hra proti počítači na vyšší obtížnosti by mohla být o něco těžší.

Zdroje dat

Programujte: Úvod do Tkinter
http://tkinter.programujte.com/, [cit. 28.12.2022]

Python Software Foundation: Python 3.10.2 documentation
https://docs.python.org/3/, [cit. 14.1.2023]

Wikipedia: Minimax
https://en.wikipedia.org/wiki/Minimax, [cit. 7.1.2023]

Wikipedia: Alpha_beta pruning
https://en.wikipedia.org/wiki/Alpha-beta_proning, [cit. 30.1.2023]