Slovenská technická univerzita v Bratislave
Fakulta elektrotechniky a informatiky

Piškvorky

hranie hier – hľadanie víťaznej stratégie

(Zadanie číslo 3 z predmetu Umelá inteligencia)

autor: Róbert Trebula
rok: 2001

1 Zadanie

Uvažujeme také hry, ktoré:

- hrajú dvaja hráči proti sebe (striedajú sa),
- sú perfektne informované, čo znamená, že každý z hráčov vie, aký ťah urobil protihráč (napríklad tic-tac-toe),
- hra môže skončiť výhrou, prehrou, alebo remízou

Úlohou je určiť, či pri danom počiatočnom rozložení hry (zvoľte si len jeden typ hry) existuje pre Maxa víťazná stratégia a ak existuje, hrať s používateľom hru (a vyhrať nad ním :-).

2 Úvod a analýza

2.1 Implementačné prostredie

Program je implementovaný v jazyku common **lisp**. Bol testovaný s verziou *CLisp* na unixovej platforme.

2.2 Víťazná stratégia

Riešenie tejto úlohy spočíva v nájdení víťaznej stratégie pre danú hru. Ideálna víťazná stratégia nájde postupnosť ťahov, ktorými môžeme vyhrať bez ohľadu na to, ako bude ťahať súper, ak takáto postupnosť existuje.

začiatočný stav → *ťah* → *stav* → ... → *ťah* → *víťazný stav*

V piškvorkách sú na rozdiel od Gundyho hry možné tri výsledky: výhra, prehra a remíza. Preto nie je možné použiť priamo algoritmus popísaný v zadaní úlohy (and/or graf).

Implementovaná víťazná stratégia je založená na algoritme **minimax**. Pre zvolenú hru - piškvorky - som sa rozhodol algoritmom minimax úplne prehľadať celý priestor možných stavov odvodených zo začiatočného stavu. Zložitosť takéhoto prehľadávania bude maximálna, keď začiatočným stavom bude prázdne hracie pole. Vtedy bude počet prehľadávaných stavov z hora ohraničený číslom $9! = 362880$, čo považujem za únosné.

2.3 Minimax

Tento algoritmus pracuje nad stromom možných stavov vygenerovateľných zo začiatočného stavu striedaním ťahov hráčov.

Listom tohoto stromu sú priradené číselné hodnoty udávajúce mieru úspešnosti Maxa. Označím číslom 2 stav pre Maxa víťazný, číslom 1 remízu a číslom 0 stav, keď Max prehrá.

Číselné hodnoty sa propagujú z nasledovníckych uzlov grafu do rodičovských takýmto spôsobom:

- ak v rodičovskom uzle je na ťahu Max, tento uzol dostane priradenú ako hodnotu **maximálnu** hodnotu z hodnôt svojich nasledovníkov
- ak v rodičovskom uzle je na ťahu Min, tento uzol dostane priradenú ako hodnotu **minimálnu** hodnotu z hodnôt svojich nasledovníkov

Takto priradené hodnoty uzlov majú význam najlepšieho možného výsledku, ktorý môže Max dosiahnuť z tohoto stavu. Ako svoj nasledujúci ťah Max vyberie ten ťah z nasledovníkov súčasného stavu, ktorý má najväčšiu hodnotu.

Algoritmus minimax je rekurzívne definovaný a preto som ho aj rekurzívne naprogramoval. O implementácii hovorí časť 3.3.

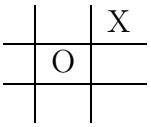
3 Implementácia programu

3.1 Reprezentácia stavov, prechody medzi stavmi

Ako najvhodnejšiu reprezentáciu stavu v hre piškvorky som zvolil reprezentáciu zoznamom, kde prvky reprezentujú riadky hracieho poľa. Riadok hracieho poľa je reprezentovaný zoznamom, kde prvkami sú jednotlivé políčka hracej plochy. Možné stavy políčka sú:

- neobsadené políčko reprezentované atómom `-`
- políčko obsadené Maxom reprezentované atómom `X`
- políčko obsadené Minom reprezentované atómom `O`

Príklad reprezentácie stavu: `((- - X)(- O -)(- - -))` označuje stav:



Nasledujúce stavy sa pre každý stav generujú funkciou `mozne_tahy`. Táto dostane ako argumenty aktuálnu pozíciu a symbol hráča, ktorý má vykonať ťah(`X` alebo `O`). V tejto funkcii sa iteratívne (funkciou `do`) prechádzajú všetky políčka hracieho poľa. Hodnota na políčku sa nastaví na symbol a ak sa takto získaný zoznam líši od pôvodného stavu, tento stav sa zaradí do zoznamu nasledovníkov daného stavu, ktorý sa na konci cyklu vráti z funkcie `mozne_tahy`.

3.2 Detekovanie výhry a prehry

Na určenie, či je nejaký stav stavom výhry alebo prehry Maxa slúžia funkcie `hodnota`, `vyhra`, `prehra`.

Výhra, resp prehra nastáva, keď sú na hracom poli v jednej línii 3 rovnaké symboly. Táto línia môže byť riadok, stĺpec, kladná diagonála alebo záporná diagonála. Preto som implementoval funkcie `riadok`, `stlpec`, `diagonala+`, `diagonala-`. Tu som s výhodou využil reprezentáciu stavu ako zoznamu riadkov. Všetky tieto funkcie sú v podstate aplikáciou funkcie `mapcar` vhodným spôsobom.

3.3 Implementácia hľadania víťaznej stratégie

Jadrom celého programu je funkcia `odpoved`, ktorá určuje nasledujúci ťah Maxa pri stave zadanom ako argument funkcie. Ak je zadaná pozícia stavom výhry alebo prehry, vráti `nil`. Ináč si vytvorí zoznam všetkých možných nasledujúcich stavov po Maxovom ťahu (`mozne_tahy pozicia 'X`). Na každý z nich aplikuje funkciu `ohodnot_max`, ktorá im priradí ohodnotenie. Vyberie tú pozíciu, pre ktorú hodnotenie vyšlo maximálne (funkciou `maxpos`).

Funkcie `ohodnot_max` a `ohodnot_min` implementujú samotný minimax algoritmus.

Funkcia `ohodnot_max` slúži na ohodnotenie stavov po Maxovom ťahu. Ak je zadaná pozícia výhrou, vráti sa stav ohodnotený číslom 2. Ak ide o prehru, vráti sa stav ohodnotený číslom 0. Ak tento stav nie je ani výhrou, ani prehrou, generujú sa možné ťahy Mina (`mozne_tahy pozicia 'O`). Na každý z nich sa aplikuje funkcia `ohodnot_min`, ktorá im priradí hodnotu z pohľadu Mina. Keďže sa jedná o Minov ťah, predpokladáme, že Min bude ťahať tak, aby minimalizoval Maxovu úspešnosť. Preto ako Minov predpokladaný ťah beriemu ťah ohodnotený minimálnym číslom. Toto číslo je hodnota stavu, ktorú funkcia `ohodnot_max` vráti.

Funkcia `ohodnot_min` slúži na ohodnotenie stavov po ťahu Mina. Ak zadaný ťah nie je ani výhrou, ani prehrou, generujú sa možné ťahy Maxa, na každý z nich sa aplikuje funkcia `ohodnot_max`. Tentoraz ide o Maxov ťah, teda vyberáme stav ohodnotený maximálnym číslom. Toto číslo je hodnota stavu, ktorú funkcia `ohodnot_min` vráti.

3.4 Komunikácia s používateľom

Na komunikáciu (hru) s používateľom programu slúži funkcia `hra`. Ako argument dostáva začiatočný stav, po ktorom nasleduje Maxov ťah. Tento sa vypíše na obrazovku a od používateľa v úlohe Mina sa vypýta jeho reakcia – nový stav po Minovom ťahu. Toto sa opakuje až do skončenia hry.

4 Záver

4.1 Funkčnosť programu

Program splnil požiadavky zadania – ak existuje víťazná stratégia, bude nájdená a program v hre s používateľom vyhrá.

Hru je možné začať s ľubovoľným začiatočným stavom, čo umožňuje otestovať správanie programu v rôznych situáciách (nerozhodný stav, istá výhra, istá prehra a pod.).

Na testovacej hardvérovej a softvérovej konfigurácii (Pentium 75, Linux, Clisp) trvalo vygenerovanie odpovede Maxa na začiatočný stav s jedným obsadeným políčkom cca. 1 minútu. To je pomerne veľa, najmä s ohľadom na to, že ak by bol zadaný ako začiatočný stav prázdne pole, generovanie odpovede by bolo zhruba 9 krát pomalšie. Na silnejšom hardvéri, prípadne efektívnejšom lispe by tento problém nemusel nastať.