Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Zadanie 2/G – Eulerov kôň

Peter Šípoš

Študijný program: Informatika

Ročník: 2

Krúžok: Štv 10:00

Predmet: Umelá inteligencia

Cvičiaci: Ing. Ivan Kapustík

Ak. rok: 2018/2019

**Zadanie problému**

Pre riešenie problému Eulerovho koňa existuje veľmi dobrá a pritom jednoduchá heuristika, skúste na ňu prísť sami. Ak sa vám to do týždňa nepodarí, pohľadajte na dostupných informačných zdrojoch heuristiku (z roku 1823!), prípadne konzultujte na najbližšom cvičení cvičiaceho. Implementujte túto heuristiku do algoritmu prehľadávania stromu do hĺbky a pre šachovnicu 8x8 nájdite pre 10 rôznych východzích bodov jedno (prvé) správne riešenie (pre každý východzí bod). Algoritmus s heuristikou treba navrhnúť a implementovať tak, aby bol spustiteľný aj pre šachovnice iných rozmerov než 8x8. Treba pritom zohľadniť upozornenie v Poznámke 1. Je preto odporúčané otestovať implementovaný algoritmus aj na šachovnici rozmerov 7x7, 9x9, prípadne 20x20 (máme úspešne odskúšaný aj rozmer 255x255) a prípadné zistené rozdiely v úspešnosti heuristiky analyzovať a diskutovať.

Pre problém Eulerovho koňa treba v oboch úlohách uvažovať s tým, že pre niektoré východzie políčka a niektoré veľkosti šachovnice riešenie neexistuje. Program preto treba navrhnúť a implementovať tak, aby sa v prípade, že do určitého času, resp. počtu krokov riešenie nenájde, zastavil a signalizoval neúspešné hľadanie. Maximálny počet krokov, resp. maximálny čas hľadania by preto mal byť ako jeden zo vstupných (voliteľných) parametrov programu. Pre toto zadanie a testovacie príklady je odporúčaný maximálny počet krokov milión, resp. maximálny čas 15 sekúnd.

**Stručný opis riešenia a jeho podstatných častí**

Riešenie problému sa opiera o dve základné myšlienky. Warnsdorffovu heuristiku a prehľadávanie do hĺbky. V spomenutej heuristike ide o informovaný výber nasledujúceho ťahu koňa. Kôň sa vždy pohne na políčko, z ktorého má v ďalšom kole najmenší možný počet ťahov po šachovnici vrátane obmedzenia, že na každé políčko môže vstúpiť iba raz.

Celé riešenie problému je rozdelené do šiestich hlavný tried: Main, MoznyTah, Uzol, Sachovnica, NajdiRiesenie a Riesenie. Z toho najpodstatnejšie sú triedy Uzol a NajdiRiesenie. Uzol predstavuje jeden stav šachovnice, teda aktuálnu polohu koňa na nej, spolu s vedomosťou o jeho rodičovi či ako sa na aktuálne políčko dostal. NajdiRiesenie reprezentuje samotný algoritmus nájdenia riešenia problému.

**Reprezentácia údajov problému**

Trieda Main, ako býva zvykom, slúži iba na spustenie celého programu.

Trieda MoznyTah reprezentuje jeden možný ťah koňom po šachovnici. Tomu zodpovedajú aj jeho atribúty *posunRiadok* a *posunStlpec*, ktoré nadobúdajú hodnoty 1,2,-1 a -2. Možných ťahov je dokopy osem a tvoria ich inštancie danej triedy s hodnotami spomínaných atribútov v pároch 1,2; 1,-2; 2,1; 2,-1; -1,2; -1,-2; -2,1; -2,-1. Posun v riadku v kladnom smere znamená posun dole (v zápornom posun hore) a posun stĺpca v kladnom smere znamená posun doprava (v zápornom posun doľava).

Jednou z najdôležitejších tried je Uzol. Jej inštancie sú konkrétne uzly generované hlavným algoritmom a uchovávajú informácie o konkrétnych stavoch. Každý uzol obsahuje odkaz na svojho rodiča, zoznam svojich nasledovníkov (detí), údaj o aktuálnom riadku a stĺpci (pozícii koňa) na šachovnici, poradové číslo ťahu koňa, stupeň uzla a operátor (posun riadku a stĺpca), ktorým sa kôň dostal na aktuálnu pozíciu. Stupeň uzla je počet možných platných ťahov, ktoré môže kôň z aktuálneho miesta vykonať. Platný (valídny) ťah je taký, pri ktorom kôň nevyjde zo šachovnice a nestúpi na políčko, na ktorom už v minulosti stál.

Šachovnica slúži na vytvorenie jednotlivých šachovníc, v rámci ktorých sa kôň pohybuje. Samotná šachovnica je reprezentovaná dvojrozmerných poľom celých čísel. Hodnota políčka šachovnice obsahuje číslo ťahu, v ktorom naň kôň stúpil.

Riešenie vyberie na základe požiadavky používateľa, či sa bude vykonávať riešenie vzorové alebo dynamické. Ak dynamické, tak si program od používateľa vyžiada počet riešení problému, ktoré chce používateľ dostať, počet riadkov a stĺpcov šachovnice a časový limit, po ktorého dosiahnutí hľadanie riešenia z konkrétneho počiatočného bodu zastaví. Podľa rozmerov šachovnice náhodne vyberie súradnice počiatočného bodu. Tieto údaje následne pošle do hlavného algoritmu hľadania riešenia.

**Konkrétny algoritmus**

Trieda NajdiRiesenie a jej metóda najdiRiesenie predstavuje gro algoritmu. Kombinuje sa tu Warnsdorffova heuristika s algoritmom prehľadávania stromu do hĺbky. Algoritmus využíva všeobecne známy fakt, že problém Eulerovho koňa pre šachovnicu s nepárnym počtom políčok a počiatočným políčkom bielym nemá riešenie. Algoritmus vie takýto prípad detegovať a ak sa tak stane, ani sa nepokúša riešenie problému nájsť, pretože je zaručené, že riešenie neexistuje. Šetrí sa tak výpočtový čas.

Na začiatku sa nastavia údaje štartovacieho uzla, hodnota finálneho (cieľového) ťahu a štartovací uzol sa pridá na začiatok zásobníka. Následne sa spustí hlavná slučka algoritmu, ktorá beží až kým nebude zásobník prázdny. Zo zásobníka sa vyberie uzol (rozvinie sa) a porovná sa hodnota jeho ťahu s hodnotou finálneho ťahu. Ak sa rovnajú (šachovnica je úspešne celá prejdená), tak algoritmus končí a vráti postupnosť operátorov, ktorými sa dostal do koncového uzla. Ak nie, tak sa algoritmus pokúsi vygenerovať deti uzla (dieťa = valídny ťah koňa). Ak sa kôň už nemôže nikam ďalej pohnúť, algoritmus sa vracia k ďalšiemu najlepšiemu ťahu – nastáva backtracking.

Daná pozícia (uzol) sa vymaže z detí rodiča daného uzlu. Ak má uzol nejakých súrodencov (rodič uzla má ešte nejaké deti), hlavný algoritmus pokračuje a teda sa zo zásobníka vyberie ďalšie najvhodnejšie dieťa rodiča. V prípade, že uzol už nemá žiadnych súrodencov, tak sa rodič uzla odoberie z navštívených pozícii na šachovnici a vymaže sa z detí jeho rodiča. Toto sa opakuje, až kým sa nepríde na uzol, ktorý by mal ešte nejakých súrodencov.

Ak má uzol aspoň jedno dieťa, tak ak ešte nebol nikdy predtým navštívený (kôň ešte nestál na danom políčku), tak sa uzol pridá do navštívených, na šachovnicu sa pridá hodnota ťahu, deti uzla sa usporiadajú od najväčšieho stupňa a pridajú do zásobníku. Tým, že sa deti najprv usporiadajú od najväčšieho sa docieli to, že na vrchu zásobníka je dieťa, ktoré má najmenej ďalších možných ťahov – Warnsdorffova heuristika. Ak cyklus prebehne celý, algoritmus vráti *null*, inak povedané, nenašiel možné riešenie problému z danej štartovacej pozície.

Veľkosť stupňa sa vypočíta jednoducho tak, že sa preiteruje cez všetky možné ťahy koňa a ak je niektorý z nich valídny, stupeň sa navýši o jedna.

**Spôsob testovania a jeho výsledky**

Fungovanie algoritmu sa testovalo viacnásobným hľadaním riešenia problému pre šachovnice rôznych rozmerov z náhodne generovaných štartovacích pozícií. Používateľ zadá do konzoly požadovaný počet riešení, rozmery šachovnice a časový limit v sekundách. Ten bol nastavený na 15sekúnd, ako je popísané v zadaní úlohy.

Pozície sú chápané ako dvojice riadkov a stĺpcov v poradí riadok - stĺpec. Tie sú počítané od 0 a nie od 1.

Pre šachovnicu 5x5 a počet pokusov 20 vrátil algoritmus nasledovné výsledky:

8krát sa nepodarilo nájsť riešenie z dôvodu jeho neexistencie (začiatok na bielom políčku pri nepárnom počte políčok). Pri všetkých ostatných východzích pozíciách (3-1, 2-0, 2-4, 0-0, 4-4, 3-3, 1-3, 2-2, 0-4) bol počet vygenerovaných uzlov 48 a počet rozvinutých 25, teda algoritmus našiel optimálnu cestu (prejdenie šachovnice na prvý pokus). Pozícia 4-4 a 1-3 bola vygenerovaná 2x. Pri počiatočnej pozícii 4-2 bol počet vygenerovaných uzlov 1756 a počet rozvinutých uzlov 1734. Algoritmus tu teda nenašiel optimálne riešenie a musel sa vracať a skúšať riešenie iné, ktoré sa mu podarilo nájsť.

Pre šachovnicu 8x8 a počet pokusov 20 dokázal algoritmus nájsť pre každú štartovaciu pozíciu optimálne riešenie. To pozostávalo zo 168 vygenerovaných a 64 rozvitých uzlov. Východzie pozície boli: 5-7, 5-2, 3-7, 6-3, 0-7, 6-7, 7-2, 4-0, 0-2, 5-4, 4-4, 7-4, 2-0, 4-3, 1-1, 1-7, 0-0. Pozície 3-7, 0-7 a 7-2 boli vygenerované 2x.

Pre šachovnicu 20x20 a počet pokusov 10 dokázal algoritmus nájsť 10 optimálnych riešení problému s 1368 vygenerovanými a 400 rozvinutými uzlami. Štartovacie pozície boli: 13-5, 18-7, 0-13, 11-19, 13-9, 14-10, 9-1, 14-13, a 15-16. Pre pozíciu 6-18 sa algoritmus potreboval viackrát v hľadaní vrátiť a vygenerovať 5665 a rozviť 4698 uzlov.

Pre šachovnicu 100x100 s počtom pokusov 10 už výsledky neboli až tak priaznivé. Podarilo sa nájsť síce sedem optimálnych riešení (z pozícií 65-42, 2-70, 20-35, 47-22, 75-22, 75-20, 27-20 ), kde sa vygenerovalo 38808 uzlov a z čoho sa rozvinulo 10000. No pri zvyšných troch pokusoch o riešenie sa ho nepodarilo nájsť do časového limitu 15sekúnd.Východzie pozície boli 12-61, 93-13, 2-70.

**Zhodnotenie riešenia**

Na základe výsledkov testovania môžeme skonštatovať, že vo väčšine prípadov s rastúcimi rozmermi šachovnice rastie aj počet vygenerovaných a rozvinutých uzlov. No nemusí to tak byť vždy, ako dokazuje prípad z pozície 4-2 na šachovnici 5x5. Vtedy sa vygenerovalo a rozvilo viac uzlov, než v optimálnom prípade na šachovnici 8x8 a dokonca aj na 20x20. No pre rozumne veľké rozmery šachovnice (do 100x100) môžeme skonštatovať, že ak riešenie existuje, tak náš algoritmus vo väčšine prípadov vyberie to optimálne.

Možnosti zlepšenia algoritmu spočívajú napríklad v zlepšení backtrackingu a následného zmenšenia objemu vygenerovaných a rozvitých uzlov, čím by sa mohlo docieliť nájdenie riešenia aj pre väčšie rozmery v požadovanom čase 15sekúnd.