Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Prehľadávanie stavového priestoru

Adam Šípka

Študijný program: Informatika
Ročník: druhý
Krúžok: Po 14:00
Predmet: Umelá inteligencia
Cvičiaci: doc. Ing. Peter Lacko, PhD.
Ak. rok: 2019/2020

Zadanie

Úlohou je prejsť šachovnicu legálnymi ťahmi šachového koňa tak, aby každé políčko šachovnice bolo prejdené (navštívené) práve raz. Riešenie treba navrhnúť tak, aby bolo možné problém riešiť pre štvorcové šachovnice rôznych veľkostí (minimálne od veľkosti 5 x 5 do 20 x 20) a aby cestu po šachovnici bolo možné začať na ľubovoľnom východziom políčku.

Algoritmom slepého prehľadávania (do hĺbky) je možné nájsť (všetky) riešenia (v bežných výpočtových – čas a pamäť – podmienkach PC) iba pri šachovniciach do veľkosti 6x6, max 7x7. Implementujte tento algoritmus pre šachovnice s rozmermi 5x5 a 6x6 a skúste nájsť prvých 5 riešení pre každú šachovnicu tak, že pre šachovnicu 5x5 aj 6x6 si vyberte náhodne 5 východzích bodov (spolu teda 10 východzích bodov) s tým, že jeden z týchto bodov je (pre každú šachovnicu) ľavý dolný roh a pre každý z týchto bodov nájdite (skúste nájsť) prvé riešenie. V prípade, že ho v stanovenom limite nenájdete, signalizujte neúspešné hľadanie. V diskusii potom analyzujte pozorované výsledky.

Riešenie problému, algoritmus

Cestu Eulerovho koňa na šachovnici som riešil štandardným prehľadávaním do hĺbky, pri ktorom sa kôň bude pohybovať v jednom smere do maximálnej možnej vzdialenosti a ak sa dostane do bodu, v ktorom sa už ďalší pohyb nebude dať uskutočniť, tak sa bude postupne presúvať na predchládzajúce pozície, aby bol schopný vykonať pohyb na predtým nenavštívené miesto. Tieto akcie sa budú vykonávať dovtedy, pokým nebude nájdená výsledná cesta, alebo neuplynie časový limit, ktorý bol zadaný používateľom.

Na začiatku nám funkcia ***create_board()*** vytvorí šachovnicu nami zadanej veľkosti. Šachovnica je reprezentovaná cez ***nested dictionary*** a medzi jeho údaje, ktoré sa doň zapisujú patrí číslo aktuálneho políčka, poradové číslo, ktoré symbolizuje koľké v poradí bolo navštívené, zoznam všetkých susediacich políčok, na ktoré sa dá z daného políčka dostať a stav, ktorý zaznačuje, či je políčko navštívené, alebo nie. Pre každé jednotlivé políčko vytvoríme jeden dictionary a postupne ho naplníme počiatočnými údajmi. Na začiatku nastavíme všetky políčka na nenavštívené a na ich poradové číslo uložíme 0, údaje, ktoré ukladáme už teraz je aktuálne číslo políčka a zoznam možných susedov.

Pomocná funkcia ***create_possible_moves()*** nám pre každé políčko vytvorí zoznam súradníc na ktoré je možné presunúť sa. Obsahuje cyklus, ktorý sa môže vykonať až 8 krát, kde každé vykonanie označuje jeden možný smer v ktorom sa môže kôň pohnúť. Všetky možné pohyby sú uložené v dvoch zoznamoch, jeden je určený pre pohyb v smere osi x (riadkoch) a ďalší v smere osi y (stĺpcoch). K dispozícii je viacej párov zoznamov, pred každým hľadaním máme možnosť vybrať si jeden z nich. Vyberie prvý pár súradníc a zistí, či je možné presunúť sa na dané pole (či sa nachádza na hracej ploche). Ak daný presun nie je možný, tak skúsi nasledujúci pár súradníc. Všetky možné presuny ukladá do dvoch zoznamov, jeden pre os x, druhý pre os y a tie potom vracia. Tie budú vo funkcii ***create_board()*** premenené na čísla políčok a uložené do zoznamu, ktorý priradíme k danému políčku. Po prejdení všetkých políčok nám funkcia vráti šachovnicu, ktorá je už pripravená na používanie.

Vo funkcii ***set_starting_point()*** zadáme počiatočné súradnice políčka, z ktorého sa bude začínať hľadanie Eulerovej cesty. Tieto súradnice, kvôli väčšej prehľadnosti, prerátame na číslo políčka na ktorom sa nachádzajú a vrátime ho. Do funkcie, ktorá nám má výslednú cestu nájsť dávame ako

argumenty samotnú šachovnicu, jej rozmery, počet navštívených políčok, zoznam ***path_final***, ktorý bude obsahovať postupnosť súradníc výslednej cesty, číslo aktuálneho políčka(z predchádzajúcej funkcie) a nami zadaný časový limit, v ktorom máme cestu nájsť.

Funkcia ***find_path()*** najprv nastaví aktuálne políčko na navštívené a priradí mu poradové číslo, čiže počet doteraz navštívených políčok. Potom do zoznamu obsahujúceho súradnice výslednej cesty vloží aktuálne súradnice daného políčka, ktoré sú vyrátané vo funkcii ***num_to_coordinates()***, keďže k dispozícii máme len číslo daného políčka a nie jeho presnú polohu na šachovnici. Následne porovná, či sa počet navštívených políčok, ktorý dostáva ako svoj argument, nerovná ich celkovému počtu. Ak áno, tak vieme, že Eulerova cesta bola nájdená a funkciu môžeme úspešne ukončiť. Potom sa funkcia presunie k cyklu, ktorý sa môže vykonať maximálne toľko krát, koľko možných presunov z daného políčka existuje. Zoberie prvého suseda zo zoznamu, a pomocou funkcie ***unvisited()***, ktorá nám povie, či bolo dané políčko už navštívené, zisťuje nasledujúce políčko na ktoré sa presunie. Keď nájde prvého nenavštíveného suseda na ktorého sa môže pohnúť, znova sa spustí funkcia na vyhľadávanie a všetko začne prebiehať odznovu. Ak sa náhodou dostaneme do situácie, v ktorej nie je možné vykonať žiaden pohyb, bez toho, aby sme sa presunuli na už navštívené políčko, tak sa poradové číslo aktuálneho políčka nastaví na 0, jeho stav sa zmení na nenavštívený a zo zoznamu ***path_final*** odstránime naposledy pridaný pár súradníc a začneme sa postupne vynárať z funkcie až dovtedy, pokým sa nenájde políčko pri ktorom môžeme vyskúšať presun na nového, ešte nenavštíveného suseda. Keď nájdeme výslednú cestu, funkciu úspešne ukončíme a vypíšeme šachovnicu reprezentovanú maticou, kde každé jej pole symbolizuje poradie, v ktorom bolo navštívené. Okrem toho vypíšeme aj zoznam ***path_final***, ktorý po úspešnom hľadaní bude obsahovať všetky páry súradníc zoradené v takom poradí, v akom boli navštevované.

Testovanie a porovnanie výsledkov

Na testovanie programu som použil pomocnú funkciu ***time_counter()***, merajúcu čas, za ktorý sa stihne vyhľadať Eulerova cesta v šachovnici. Meranie začína pred prvým spustením funkcie ***find_path()*** a končí sa buď úspešným nájdením cesty, alebo vypršaním časového limitu, ktorý je zadaný na začiatku programu. Časový limit som nastavoval na 5 sekúnd pri šachovnici o veľkosti 5 x 5, a 10 sekúnd pri šachovnici s rozmermi 6 x 6. Okrem merania času som zaznamenával aj počet vykonaných krokov potrebných na nájdenie cesty. Samozrejme, čas a počet vykonaných krokov záleží aj od poradia v akom kôň dané políčka navštevuje. K dispozícii sú dve poradia v ktorých kôň skúša navštíviteľnosť políčok a pred každým hľadaním si môžeme jedno z nich vybrať. Ak preskočíme možnosť výberu, tak bude použitá možnosť B, ktorú som nastavil ako predvolenú.

Zoznam možností presunov v poradí v akom sú testované:
A: (1, 2) (1, -2) (2, 1) (2, -1) (-1, 2) (-1, -2) (-2, 1) (-2, -1)
B: (-1, -2) (-1, 2) (-2, -1) (-2, 1) (1, -2) (1, 2) (2, -1) (2, 1)

Pri šachovnici o veľkosti 5 x 5 som si zvolil 5 párov počiatočných súradníc z ktorých sa začalo vyhľadávanie Eulerovej cesty. Pre každý pár som otestoval oba spôsoby možných presunov a výsledky som zaznamenal do nižšie uvedenej tabuľky. Rovnakým spôsobom som postupoval aj pri šachovnici s veľkosťou 6 x 6. Keďže sa jednalo o šachovnicu s veľkosťou 5 x 5, tak všetky vyhľadania stihli do zadaného časového limitu úspešne prebehnúť, aj keď výsledky pre jednotlivé možnosti presunov na susedné polia sa navzájom odlišovali a medzi nameranými hodnotami, či už sa jedná o čas, alebo počet

Počiatočné súradnice X	Y	Čas vyhľadania	Počet vykonaných krokov	Poradie pohybov A	B	Hľadanie úspešné
0	0	0,21	74357	X		ÁNO
		0,02	6215		X	ÁNO
0	5	0,17	75098	X		ÁNO
		0,01	3440		X	ÁNO
2	2	0,04	13647	X		ÁNO
		0,04	13647		X	ÁNO
4	2	0,03	13647	X		ÁNO
		0,22	82577		X	ÁNO
1	1	1,54	601407	X		ÁNO
		0,16	61596		X	ÁNO

vykonaných krokov, boli niekoľkonásobné rozdiely. Len pre jeden pár súradníc bolo poradie pohybov A rýchlejšie ako poradie B. V 3 prípadoch z 5 bolo však poradie B až niekoľkonásobne rýchlejšie, čo značí, že správne zvolená postupnosť presunov je pre rýchle a efektívne fungovanie programu veľmi podstatná a môže značne ovplyvniť rýchlosť vyhľadania výslednej cesty. V prípade, pri ktorom sa hľadanie začína zo stredu šachovnice, výber zvolenej postupnosti samozrejme nezohráva žiadnu úlohu, keďže postupnosť A je len obrátená postupnosť B a naopak, čiže čas trvania a aj počet krokov boli v oboch rovnako efektívne.

Pri šachovnici s rozmermi 6 x 6 boli výsledky testovania podstatne rozdielne, takmer v polovici prípadov sa v 10 sekundovom časovom limite, napriek veľkému počtu vykonaných krokov, nepodarilo nájsť výslednú cestu (6 úspešných vyhľadaní z 10). Podľa toho som usúdil, že obe postupnosti presunov neboli úplne ideálne, keďže úspešnosť postupnosti A, rovnako ako aj B, bola 60 % (3 úspešné vyhľadania z 5), ale pre každý pár súradníc sa podarilo nájsť výslednú cestu aspoň jedným spôsobom. Aj pri niektorých úspešných pokusoch počet vykonaných krokov presiahol pol milióna a čas vyhľadania bol viacej ako sekunda, preto treba dať dôraz na správny výber poradia v akom sa budú políčka navštevovať. Okrem šachovníc o veľkosti 5 x 5 a 6 x 6 som skúsil vyhľadať cestu aj na šachovniciach s rozmermi 7 x 7 a 8 x 8, no tam bola úspešnosť vyhľadania nízka a aj úspešné nájdenie cesty trvalo niekoľko sekúnd a počet vykonaných krokov sa približoval k 3 miliónom, takže tieto výsledky som nezaznamenával.

Počiatočné súradnice X	Y	Čas vyhľadania	Počet vykonaných krokov	Poradie pohybov A	B	Hľadanie úspešné
0	5	10	3657931	X		NIE
		0,25	95083		X	ÁNO
5	4	0,19	67559	X		ÁNO
		10	3593683		X	NIE
4	2	1,53	554557	X		ÁNO
		10	3675039		X	NIE
1	5	0,26	90301	X		ÁNO
		1,93	677788		X	ÁNO
1	1	10	3717150	X		NIE
		1,87	677788		X	ÁNO

Celkové výsledky testovania dokázali, že pre čo najväčšiu efektivitu a najrýchlejší priebeh programu je potrebné pre každý pár počiatočných súradníc zvoliť správnu postupnosť presunov. Na šachovnici s rozmermi 5 x 5 to nie je až tak podstatné, keďže v tom prípade prebehne vyhľadanie stále relatívne rýchlo a úspešne, no pri šachovniciach väčších rozmerov môže efektivita a rýchlosť rapídne klesnúť.

Zhodnotenie riešenia

Moje riešenie bolo zväčša jednoduché na implementáciu, čo sa zaraďuje medzi jeho výhody, no efektivita pri šachovniciach väčších ako 6 x 6 rapídne klesala, čo je celkom prirodzené, lebo som použil prehľadávanie do hĺbky, ktoré je skôr vhodné na šachovnicu menších rozmerov, kde je aj menší počet stavov. Použité prehľadávanie do hĺbky je greedy algoritmus, keďže k výslednej ceste sa nedostávame najrýchlejším možným postupom, no to je kvôli tomu, že pri vyhľadávaní výslednej cesty záleží od poradia v ktorom budeme z daného políčka navštevovať jeho susedov. Vďaka tomu sa nám nemusí vždy podariť nájsť cestu ani v niekoľkosekundovom limite. Ideálne by bolo, keby sme mali k dispozícii viacej možností postupností presunov, vďaka čomu by sa mohla celková efektivita programu zvýšiť. Toto je jedna z možností ako program rozšíriť, no keďže nevieme, ktorá z postupností je pre daný prípad ideálna, tak by sa stále celková efektivita nemusela zvýšiť. Ale ak by bolo možné podľa pozície, z ktorej hľadanie začína, vybrať, alebo vytvoriť vhodnú postupnosť, bolo by možné program dostatočne zefektívniť. Vďaka tomu by sa dalo v relatívne rýchlom čase nájsť správne riešenie aj na šachovniciach s väčšími rozmermi.