Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2

842 16 Bratislava 4

Umelá Inteligencia – zadanie

# Zadanie č. 2

Vypracoval: Tomáš Liščák   
Krúžok: streda 10:00  
Akademický rok: 2014/2015, letný semester

**Stručný opis riešenia a jeho podstatných častí:**

Na riešenie problému je použitý algoritmus slepého prehľadávania do hĺbky. Prehľadáva sa strom ktorého vrcholy sú reprezentácia stavov. Na začiatku je vytvorený vrchol s konfiguráciou šachovnice podľa zadaných koordinátou začiatočnej pozície a dimenzii šachovnice. Tento vrchol sa uloží do zásobníka a následne môže začať hlavná slučka algoritmu. Ak je obsah zásobníka prázdny, znamená to že žiadne ďalšie stavy nie sú na prechádzanie a neexistuje riešenie. Ak nie, vyberie sa z neho prvý vrchol a prebehne test pre jeho ekvivalenciu k cieľu. Ak sa rovná, riešenie sa našlo a ak nie, tak sa súčasný vrchol podrobí operátorom. Ak sa našlo vypíše trasa koordinátov pomocou odkazov na rodiča v každom vrchole a pomocou koordinátov aktuálnej polohy koňa v každom vrchole až po koreň. Operátorov je osem a každý má explicitne definované dve hodnoty, ktoré odpočítava, alebo pripočítava ku koordinátom súčasnej pozície koňa. Sú takisto definované aj hranice šachovnice dané jej rozmermi. Ak sa na políčko žiadané operátorom dá prejsť, vytvorí sa nový vrchol so stavom rovnakým ako predchádzajúci až na jednotku v tomto políčku. Vzniknuté vrcholy sa pridávajú do zásobníka a program pokračuje na začiatok slučky pre výber aktuálneho vrcholu zo zásobníka.

Na porovnanie už navštívených stavov s aktuálnymi kvôli zamedzeniu prechádzaniu rovnakých vetví je použitá hash funkcia, ktorá obsahuje obrovské pole inicializované na null, do ktorého index sa počíta ako násobok decimálnej hodnoty binárneho čísla mapy vrcholu a konštantného čísla určeného súradnicami tabuľky. Inkrement pri kolízii je jedno poľa explicitne určených prvočísel do ktorého index je súčin súradníc aktuálnej pozície koňa.

**Reprezentácia údajov problému:**

Problém Eulerovho koňa zahŕňa stavy ako konfigurácie šachovnice a operátory ako akcie prechádzaní políčkami koňom podľa pravidiel, ktoré tak definujú možné zmeny konfigurácii. Pre riešenie prechádzaním stromu je potrebné reprezentovať stavy vrcholmi. Vrchol má o niečo viac údajov ako samotný stav a ako údajová štruktúra obsahuje osem referencii na možné ďalšie vrcholy, ktoré by obsahovali stavy vzniknuté aplikovaním operátorov na stav daného vrcholu. Stav vrcholu predstavuje binárna mapa šachovnice, ktorej políčko je 1 pokiaľ ho kôň už navštívil a 0 pokiaľ nie a koordináty súčasnej pozície koňa. Mimo toho vrchol obsahuje aj referenciu na rodičovský vrchol. Kôň je reprezentovaný koordinátmi jeho súčasnej polohy vo vrchole a samotná šachovnica je reprezentovaná ako binárna mapa stavu vo vrchole.

**Spôsob testovania:**

Program som testoval pre rôzne rozmery šachovnice od 4x4 po 6x6 funkciou ktorá testuje všetky koordináty. Rovnomernú väčšinu prvých riešení som ručne overil na papieri. Následne som zaznačil päť náhodne zvolených bodov pre obe šachovnice s tým že jeden je vždy v ľavom dolnom rohu a zaznamenal do tabuľky údaje.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Koordináty | Rozmery šachovnice | Počet otestovaných vrcholov | Výsledok | Použitie Hash |
| 4 0 | 5 5 | 8839 | Existuje riešenie | Nie |
| 4 0 | 5 5 | 7212 | Existuje riešenie | Áno |
| 4 2 | 5 5 | 365420 | Existuje riešenie | Nie |
| 4 2 | 5 5 | 228477 | Existuje riešenie | Áno |
| 3 0 | 5 5 | 1829421 | Neexituje riešenie | Nie |
| 3 0 | 5 5 | 500000 | Prekročený počet pokusov | Áno |
| 3 3 | 5 5 | 60160 | Existuje riešenie | Nie |
| 3 3 | 5 5 | 44604 | Existuje riešenie | Áno |
| 1 3 | 5 5 | 86208 | Existuje riešenie | Nie |
| 1 3 | 5 5 | 60327 | Existuje riešenie | Áno |
| 5 0 | 6 6 | 248168 | Existuje riešenie | Nie |
| 5 0 | 6 6 | 134746 | Existuje riešenie | Áno |
| 4 1 | 6 6 | 1243052 | Existuje riešenie | Nie |
| 4 1 | 6 6 | 527446 | Existuje riešenie | Áno |
| 1 4 | 6 6 | 5000000 | Prekročený počet pokusov | Nie |
| 1 4 | 6 6 | 500000 | Prekročený počet pokusov | Áno |
| 1 1 | 6 6 | 2253 | Existuje riešenie | Nie |
| 1 1 | 6 6 | 2161 | Existuje riešenie | Áno |
| 5 5 | 6 6 | 58691 | Existuje riešenie | Nie |
| 5 5 | 6 6 | 43417 | Existuje riešenie | Áno |

Príklad riešenia pre (11 66) : (x: 0 y: 3) – (x: 1 y: 5) – (x: 3 y: 4) – (x: 5 y: 5) – (x: 4 y: 3) – (x: 2 y: 4) – (x: 0 y: 5) – (x: 1 y: 3) – (x: 2 y: 5) – (x: 4 y: 4) – (x: 5 y: 2) – (x: 3 y: 3) – (x: 1 y: 4) – (x: 3 y: 5) – (x: 5 y: 4) – (x: 4 y: 2) – (x: 5 y: 0) – (x: 3 y: 1) – (x: 1 y: 0) – (x: 0 y: 2) –(x: 2 y: 3) – (x: 0 y: 4) – (x: 1 y: 2) – (x: 0 y: 0) – (x: 2 y: 1) – (x: 4 y: 0) –(x: 3 y: 2) – (x: 5 y: 1) – (x: 3 y: 0) – (x: 2 y: 2) – (x: 0 y: 1) – (x: 2 y: 0) – (x: 4 y: 1) – (x: 5 y: 3) – (x: 4 y: 5)

**Zhodnotenie riešenia, možnosti rozšírenia, výhody a nevýhody konkrétnej implementácie:**

Riešenie prehľadávaním do hĺbky funguje, ale je rozhodne horšie než použitím nejakej rozumnej heuristiky. Kvôli hĺbkovému prehľadávaniu pri formovaní hash funkcie ak ako vstupné údaje akokoľvek zoberiem konfiguráciu šachovnice, stavy ktoré sa prehľadávajú za sebou sú príliš podobné a vzniká množstvo kolízii. Na 100tisíc prehľadávaní vrcholov je približne 800tisíc kolízii pri pokuse o nájdenie prázdneho miesta v tabuľke, ktorá ma milión prvkov. Riešenie s hash funkciou tým pádom trvá dlhší čas aj napriek menej preskúmaných vrcholov. Okrem hash funkcie som skúšal aj vyvážený strom s časom log(n) ale pomocou neho vyľadávanie už napr 50tisíc prejdených stavov trvá veľmi dlho. Výhodou tohto riešenia je jednoduchosť implementácie. Programovacie prostredie je java a jej výhoda bola použitie utilit ako zásobník, alebo konverzia typov.

**Porovnanie vlastností použitých metód pre rôznu dĺžku riešenia:**

Prehľadávanie do hĺbky prechádza postupne celým stromom a pre čím väčšiu hĺbku stromu a teda väčšiu šachovnicu, je nájdenie riešenia exponenciálne zložitejšie. Pre nájdenie prvého riešenia je to však lepšia technika ako prehľadávanie do šírky alebo cyklicky sa prehlbujúce hľadanie, pretože sa o dosť skoršie dostane na k listom a otestuje väčšinu listov predtým, než ostatné dva algoritmy otestujú aspoň jeden.