Adam Miškove

Umelá Inteligencia

Zadanie 2: Prehľadávanie stavového priestoru, Problém 3, f)
FIIT STU

Cvičenie: Utorok 14:00-15:40, Kapustík

Opis zadania:

Problém 3, Eulerov kôň.
Zadanie f)
Základom tohto problému je šachová doska a jej konkrétny panáčik - kôň.
Úlohou je postaviť koňa na jeho začiatočnú pozíciu na doske, a v rámci jeho legálnych šachových ťahov prejsť všetky políčka dosky tak, aby na žiadnom nebol dvakrát.
Počas tohto procesu je potrebné označiť každý krok figúrky číslom jeho poradia.
Príklad konečného výstupu(šachovnica 5x5):

1	16	21	10	7
22	11	8	15	20
17	2	25	6	9
12	23	4	19	14
3	18	13	24	5

V tomto príklade je začiatok v ľavom hornom rohu.
Koniec sa vždy nachádza v políčku s číslom riadky*stĺpce
(v tomto prípade 25)

Je nám prístupných 8 rôznych operátorov, ktoré symbolizujú 8 konkrétnych možných krokov šachového koňa, v podobe ich impaktu na pozíciu našej figúrky.

Opis riešenia:

Využívam algoritmus DFS(Depth First Search), alebo aj prehľadávanie do hĺbky, ktorý sa snaží v širšom zmysle v stromovej štruktúre dostať čo najhlbšie, predtým ako narazí na prekážku alebo koniec. Potom sa vracia po svojich stopách do čo najbližšieho bodu, z ktorého sa dá ísť inou cestou, kde znova opakuje svoje poslanie dostať sa čo najhlbšie.

V mojej implementácii je jeden uzol reprezentovaný v našej šachovnici a svojou rekurzívne volanou funkciou.

Jeho jediný "atribút" je key, ktorý je implementovaný ako argument danej funkcie.

Srdcom tohto programu je rekurzívne volanie funkcie iterate(), ktorá riadi všetko spoločné s našim hĺbkovým prehľadávaním.

Tento algoritmus teoreticky rozvíja 8^x uzlov, pričom x je počet políčok na našej doske.

No v skutočnosti mnohé z týchto políčok majú menej ako 8 možných legálnych krokov pre šachového koňa, tým pádom je toto číslo omnoho menšie.

Algoritmus rekurzie v použitej funkcii:

Funkcia rekurzívne volá sama seba pre ďalšie kroky v poradí, pričom vždy zistí, aké kroky sú z konkrétneho políčka v prípade našej aktuálnej šachovnice možné, a pre všetky nájdené sa rozvinie, v poradí danom v zadaní(do štatistík som zahrnul aj opačné poradie).

V momente, kde narazí na miesto, z ktorého sa nedá nikam ísť, a nie je to finálny krok(správne riešenie), rekurzia sa vracia späť na posledné mieto, z ktorého sa rozvinul iný uzol.

Týmto postupne vyčerpá všetky možnosti a zastavuje sa(kompletne opúšťa rekurziu) pri nájdení správneho riešenia.

Kroky sú postupne zapísané do šachovnice, a výstup v podobe šachovnice s konkrétnym postupom krokov na políčkach sa taktiež vypisuje.

Príklady na hľadanie:

Tieto príklady sú pre rozmery šachovnice 5x5.
Začiatočné súradnice sú vypísané nad šachovnicou.
Šachovnica sa v tomto prípade skladá z vypísaných čísel, ktoré symbolizujú poradie krokov koňa.
Riadok s vypísaným poradím vybratých operátorov vyjadruje konkrétne poradie operátorov, ktoré bolo vybraté. Tieto čísla reprezentujú ich poradie, presne ako sú spomenuté v zadaní, pričom 0 znamená prvý operátor, 1 znamená druhý, a tak podobne.

```
start: 0, 0
1  12 3  18 21
4  17 20 13 8
11 2  7  22 19
16 5  24 9  14
25 10 15 6  23
success, --- 0.13912653923034668 seconds ---
poradie vybratých operátorov: 261207431760215642543251
```

```
start: 3, 1
21 8  3  12 23
2  13 22 17 4
7  20 9  24 11
14 1  18 5  16
19 6  15 10 25
success, --- 0.4353940486907959 seconds ---
poradie vybratých operátorov: 740317622671304731670432
```

Štatistiky testovania fungovania programu:

Šachová doska s rozmermi 5x5:

Označenie políčka (riadok, stĺpec)	Čas -- Operátory zľava (s)	Čas -- Operátory zprava (s)
0,0	0.1501	0.15113
0,1		
0,2	0.167	1.5717
0,3		
0,4	0.1529	0.1481
1,0		
1,1	1.21269	0.228
1,2		
1,3	0.6545	0.2602
1,4		
2,0	0.8207	0.1421
2,1		
2,2	0.02702	0.046
2,3		
2,4	1.02092	0.1671
3,0		
3,1	0.47943	0.4714
3,2		
3,3	0.1251	1.05
3,4		
4,0	0.007	0.001
4,1		
4,2	0.028	1.264
4,3		
4,4	0.013	0.1491

Ako môžeme vidieť, je značný rozdiel medzi hľadaním odpovede pri zmene poradia operátorov.

Prázdne políčka symbolizujú neexistujúcu odpoveď (doska má nepárny počet riadkov/stĺpcov).

Vzhľadom na prechádzanie operátorov v statickom poradí sa čas nájdenia riešenia pre konkrétny vstup takmer vôbec nemení, a počet vykonaných operácií zostáva rovnaký.

Rozdiel v poradí prechádzania operátorov má oveľa väčší impakt na výsledky pri väčšej šachovnici, demonštrované nižšie.

Takisto sa dá pozorovať, že políčka v rohoch(vyfarbené na oranžovo) majú priemerne kratší čas pre nájdenie riešenia.

Šachová doska s rozmermi 6x6:

Označenie políčka (riadok, stĺpec)	Čas -- Operátory zľava (s)	Čas -- Operátory zprava (s)
0,0	2.414	64.159
0,1	-----	0.864
0,2	59.684	9.3685
0,3	12.332	2.504
0,4	-----	0.1821
0,5	35.226	65.775
1,0	49.21	2.08889
1,1	30.726	1.182
1,2	-----	4.852
1,3	36.05	1.258
1,4	41.76	1.386
1,5	0.189	0.142
2,0	92.72	14.741
2,1	12.75	0.404
2,2	233.006	5.8272
2,3	29.59	4.9264
2,4	42.35	68.578
2,5	28.865	16.52
3,0	-----	0.7747
3,1	132.0009	2.3891
3,2	2.916	0.4383
3,3	5.967	18.303
3,4	-----	1175.52*
3,5	1.149	2.860
4,0	1.397	0.4043
4,1	0.139	3.5141
4,2	1.145	6.3617
4,3	176.627	1.1870
4,4	1.357	2.7845
4,5	0.137	-----
5,0	0.1961	0.02502
5,1	10.209	0.52747
5,2	2.847	-----
5,3	5.8252	4.596
5,4	0.1441	10.07
5,5	34.213	3.256
Priemer:	34.5	9.3

Ako môžeme pozorovať, zhodou okolností je čítanie operátorov z pravej strany pri tejto verzii šachovnice oveľa efektívnejšie. (priemer neberie do úvahy políčka označené hviezdičkou alebo pomlčkami.)

Taktiež si môžeme všimnúť oveľa väčšie rozdiely v čase potrebnom na získanie správneho riešenia odlišných vstupov.

Vzhľadom na rôznu efektivitu rôznych spôsobov prechádzania operátorov je možné, že pre niektoré vstupy na šachovniciach s vyšším počtom riadkov/stĺpcov trvá nájdenie riešenia viac, ako je nami zvolený časový limit.

Políčka v tabuľke s hodnotou "-----" majú dobu riešenia presahujúcu časový limit. Toto je zapríčinené prirodzenou vlastnosťou prechádzania operátorov v istom smere, teda šancou, že hľadané riešenie má svoje vyžadované operátory pri konci postupnosti operátorov.

Tento fakt má pomerne veľký impakt na niektoré riešenia, vzhľadom na to, že niektoré vstupy majú čas riešenia okolo 0,1 sekundy, a niektoré viac ako 200 sekúnd.

Pri políčkach (0,1), (0,4), (3,0), a (4,5) si môžeme povšimnúť obrovský rozdiel medzi efektivitou spomenutých rôznych prechádzaní operátorov.

Medzi výhody tejto implementácie patrí predvídavosť a pravidelnosť našich výsledkov. Algoritmus bude pre rovnaký vstup s rovnakým poradím operátorov vždy rovnako postupovať a hľadať svoje riešenie.

Medzi nevýhody patrí vlastne tento rovnaký jav, keďže môže byť v istých prípadoch vnímaný ako dvojsečná čepeľ. Pre niektore vstupy s istým poradím operátorov je možné nájsť veľmi dlhú dobu hľadania riešenia, a opakovaná iterácia tomu nijak nepomôže, dá sa to vyriešiť iba zmenením poradia operátorov.

Tieto výhody/nevýhody nie sú závislé na programovacom prostredí.

**Rýchlosť a efektivita programu bola od merania hodnôt zlepšená o 10-15%. Namerané hodnoty však ostávajú perfektne reprezentatívne pre vyjadrenie dynamiky tohto zadania a jeho riešenia.

Bonusová tabuľka pre šachovnicu 7x7:

Označenie políčka (riadok, stĺpec)	Čas (s)
0,0	12.698
0,1	-----
0,2	14.1003
0,3	-----
0,4	24.549
0,5	-----
0,6	35.821
1,0	-----
1,1	18.075
1,2	-----
1,3	*
1,4	-----
1,5	*
1,6	-----
2,0	*
2,1	----
2,2	40.293
2,3	-----
2,4	64.78
2,5	------
2,6	*
3,0	-----
3,1	2.926
3,2	-----
3,3	*
3,4	-----
3,5	3.0318
3,6	-----
4,0	*
4,1	-----
4,2	*
4,3	-----
4,4	*
4,5	-----
4,6	*
5,0	-----
5,1	36.104
5,2	-----
5,3	*
5,4	-----
5,5	35.997
5,6	------
6,0	0.583

6,1	-----
6,2	*
6,3	------
6,4	*
6,5	------
6,6	*

*nájsť riešenie pre tieto políčka trvalo dlhšie ako bol časový limit(70 sekúnd)

Záver

V rámci práce na tomto zadaní som sa oboznámil s problematikou Eulerovho koňa, a prehĺbil svoje znalosti ohľadne prehľadávania do hĺbky z predošlých predmetov. V priebehu testovania a tvorenia štatistík som zistil, že celkom záleží na poradí prechádzania operátorov, keďže to ovplyvňuje čas riešenia políčok.