SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

ZADANIE 2(g) - Eulerov kôň

Martin Nemec
FIIT STU
Cvičenie: Štvrtok 12:00
28.10.2021

1 Zadanie úlohy

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.

Kôň má vo všeobecnosti 8 možností skoku, ak nie je limitovaný okrajom šachovnice alebo už použitým miestom. To znamená, že existuje 8 operátorov, ktoré je možné označiť napríklad (1, 2), (1, -2), (2, 1), (2, -1), (-1, 2), (-1, -2), (-2, 1) a (-2, -1), kde prvé číslo znamená posun od aktuálnej pozície v riadku a druhé v stĺpci.

2 Opis riešenia

Na riešenie tohto problému som využil prehľadávanie do hĺbky pomocou rekurzívnej funkcie a implementoval som Windsor heuristiku.

Na začiatku podľa vstupu z klávesnice pre veľkost šachovnice sa vo funkcii declare() vytvorí šachovnica NxN a všetky políčka sa nastavia na hodnotu 0.

Ďalej na základe vstupu z klávesnice pre začiatočnú pozíciu koňa sa zavolá funkcia `lets_tour_with_heuristic(x, y, counter, board_size, chess_board, start, step_count)`

`Kde x, y sú začiatočné súradnice koňa, counter je počítadlo pre očíslovanie šachovnice koňom, board_size je veľkosť šachovnice, chess_board je 2D pole, ktoré zapĺňa kôň , start je začiatok času a step_count je počítadlo krokov.`

2.1 Algoritmus

```
end = time.time()
if end - start > 15:
    return True
chess_board[x][y] = counter
if counter == board_size ** 2:
    return True
options = get_possible_steps(x, y)
if options:
```

Na začiatku sú overovania. Overuje sa či algoritmus nepreskočil čas 15 sekúnd, či sa counter nerovná druhej mocnine veľkosti šachovnice, ak áno program má hotovo. Overujú sa všetky možné kroky z aktuálnych súradniciach koňa pomocou funkcie get_possible_steps(). Ak nie sú k dispozícii žiadne kroky aktuálna pozícia sa prepíše naspäť na 0 a vráti sa o krok späť, inač pokračuje.

```
if options:
    new = []
    number_of_options = []
    for i in options:
        number_of_options.append(len(get_possible_steps(i[0], i[1])))   # do number_of_options vloži počet dalších krokov
    counter += 1

    while options:     # cyklus do new prida všetky možnosti podla počtu dalších možností od najmenej možnych
        new.append(options[number_of_options.index(min(number_of_options))])
        del options[number_of_options.index(min(number_of_options))]
        del number_of_options[number_of_options.index(min(number_of_options))]

    for i in new:
        tour_value = lets_tour_with_heuristic(i[0], i[1], counter, board_size, chess_board, start, step_count)
        if tour_value:
            return True
```

Ak sú možné kroky z aktuálnej pozície koňa, tak pomocou prvého cyklu for vložím do poľa number_of_options počet dalších možných krokov z aktuálnych možných krokov.

Pomocou while cyklu pridá do poľa new už zoradené možné dalšie kroky podľa počtu dalších krokov.

V ďalšom for cykle prechádza pole new, v ktorom sú zoradené kroky od najmenších ďalších možných a posiela to znovu do funkcie lets_tour_with_heuristic, takto sa prehľadáva do hĺbky s danou heuristikou, keďže ďalšia pozícia koňa bude tá z ktorej je najmenej ďalších možností. A odznova ide táto funkcia s už spomínaním overovaním.

```
if counter == 1:
    chess_board[x][y] = 1
    return False
else:
    chess_board[x][y] = 0
    return False
```

V prípade že options = 0, čiže nemá žiadne ďalšie možné kroky, nastaví svoju pozíciu na 0, vráti sa. Vráti False a ide na ďalší možný krok.

Takto sa to opakuje až kým tour_value = True tým pádom kôň našiel cestu.

3 Testovanie

Testoval som pri rôznych veľkostiach šachovnice a pri rôznych začiatočných pozíciach, meral som priemerný čas pri 5 spusteniach každej možnosti. Čas je udávaný v ms.

Šachovnica 5x5		
[x, y]	Priemer z 5 meraní [ms]	Počet krokov
0x0	<1	25
4x0	<1	25
0x4	<1	25
4x4	<1	25
2x2	<1	25
2x3	4885	1028893

Obr. 1: 5x5 - v 2x3 nemá riešenie

Šachovnica 7x7		
[x, y]	Priemer z 5 meraní [ms]	Počet krokov
0x0	~1	49
7x0	~1	49
0x7	~1	49
7x7	~1	49
3x3	~1	49

Obr. 2: 7x7

Šachovnica 8x8		
[x, y]	Priemer z 5 meraní [ms]	Počet krokov
0x0	~1	64
1x3	~1	64
7x0	~1	64
2x4	~1	64
0x7	~1	64
5x1	~1	64
7x7	~1	64
3x3	~1	64
6x5	~1	64

Obr. 3: 8x8

Šachovnica 9x9		
[x, y]	Priemer z 5 meraní [ms]	Počet krokov
0x0	~1	81
8x0	~1	81
0x8	~1	81
8x8	~1	81
4x4	~1	81

Obr. 4: 9x9

Šachovnica 15x15		
[x, y]	Priemer z 5 meraní [ms]	Počet krokov
0x0	3,592	225
14x0	4,196	225
0x14	3,796	225
14x14	4,002	225
7x7	3,404	225
13x3	3,601	225
4x7	15s	~2 720 152

Obr. 5: 15x15

Šachovnica 30x30		
[x, y]	Priemer z 5 meraní [ms]	Počet krokov
0x0	15,4106	900
29x0	14,996	900
0x29	15,608	900
29x29	15s	~2 706 736
15x15	15,396	900
12x6	14,998	900
21x18	15	~2 690 620

Obr. 6: 30x30

Šachovnica 50x50		
[x, y]	Priemer z 5 meraní [ms]	Počet krokov
0x0	15s	~2 697 152
49x0	43,196	2500
0x49	43,003	2500
49x49	44,406	2 500
25x25	43,998	2500
30x12	43,605	2500

Obr. 7: 50x50

4 Záver

Na základe testovanie som sporozoval že pri zväčšovaní šachovnice sa logicky predlžuje čas na nájdenie cesty. Pri niektorých rozmeroch šachovnice a niektorých možnostiach kôň nestihne nájsť cestu do 15 sekúnd alebo nemá riešenia ako napríklad pri šachovnici 5x5, kde x, y = [2, 3]