Umelá inteligencia

Zadanie č. 1

Eulerov kôň (Knight´s tour)

Norbert Matuška
10-1-2022

Norbert Matuška
UI Streda 15:00-16:40

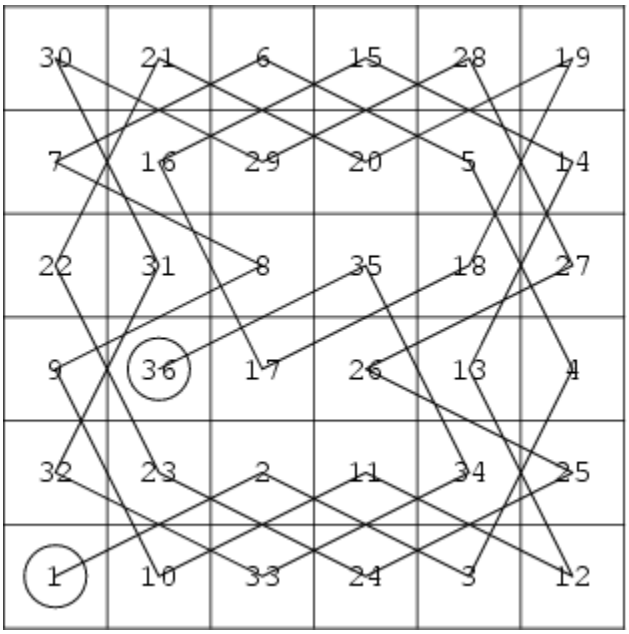
Contents

Norbert Matuška
UI Streda 15:00-16:40

Opis problému

Úlohou je prejsť šachovnicu rôznej veľkosti legálnymi ťahmi šachového koňa tak, aby sme navštívili každé jedno políčko na šachovnici práve jeden krát.

http://www2.fiit.stuba.sk/~kapustik/z2d.html#F



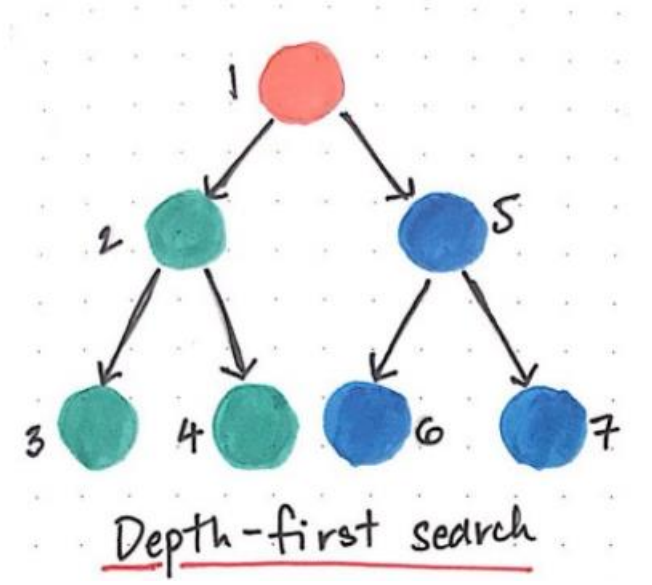
1.

Norbert Matuška
UI Streda 15:00-16:40

Algoritmus použitý na riešenie problému

Na riešenie problému som mal v zadaní použiť algoritmus slepého prehľadávania (Prehľadávanie do hĺbky, ďalej len DFS). DFS algoritmus sa snaží prehľadávať čo najhlbšie ako sa dá. Čiže hľadám cestu až kým nedôjdem na krok, v ktorom sa už nedá nikde posunúť bez toho aby som znova navštívil políčko alebo vystúpil von z šachovnice. V takom prípade sa vrátim naspäť na posledný rozvinutý uzol a prehľadávam možnosti, ktoré ešte neboli navštívené, pokým nenájdem prvé riešenie kedy vráti cestu. Hľadanie do hĺbky nezaručuje nájdenie riešenia a taktiež nezaručuje nájdenie "najlepšej" cesty.

Diagram pre DFS



2.

Zložitosť algoritmu

Pamäťová zložitosť:

-je lineárna: $m * b$ kde m = maximálna hĺbka a b = faktor vetvenia

Časová zložitosť:

-$O(b^m)$ v najhoršom možnom prípade, kedy budeme musieť prezrieť všetky uzly, v najlepšom prípade sa môže nájsť riešenie hneď

Prípustnosť:

-DFS nezaručuje nájdenie najlepšej cesty/riešenia. Je šanca, že sa neoptimálne riešenie nájde skôr ako to najlepšie. Takže tento algoritmus nie je prípustný.

Úplnosť:

-DFS taktiež nezaručuje nájdenie vôbec nejakého riešenia. Ak je strom príliš hlboký, môže sa stať, že algoritmus sa zamotá a riešenie sa nikdy nenájde. Taktiež sa môže stať, že aj z konečnej množiny operátorov sa vygeneruje nekonečne hlboký strom. Preto tento algoritmus nie je úplný.

Norbert Matuška
UI Streda 15:00-16:40

Riešenie problému

Riešenie realizujem pomocou iteratívnej funkcie, ktorá prehľadáva uzly, ukladá cestu a môže dojsť k nasledujúcim záverom:

- Nájde sa riešenie, nedošlo k timeout-u a vypíše sa čas, za ktorý to vyriešilo, počet prejdených uzlov a cesta k riešeniu

```
o  if not goalNotFound and not problem.timeout:
```

- Nenašlo sa riešenie, nedošlo k timeout-u, takže riešenie neexistuje

```
o  elif goalNotFound and not problem.timeout:
```

- Nenašlo sa riešenie za požadovaný čas

```
o  elif goalNotFound and problem.timeout:
```

Môj algoritmus iteratívne prehľadáva všetky možné cesty. Na začiatku máme začiatočný stav, vojdeme do while loop-u, posledný element vo fronte sa pomocou pop() pridelí k uzlu a počet rozšírených uzlov sa zvýši o jedno. Za každý pohyb, ktorý môže rytier vykonať zo svojho aktuálneho miesta sa nájde umiestnenie dlaždice, ktorú táto akcia „navštívi“ a ak je vo vnútri šachovnice, vytvorí sa child uzol a ak tento child uzol ešte nebol navštívený (nie je v node.state) a ak nie je v explored sade a ak dieťa nie vo fronte, potom sa naň použije goalTest. Ak je stav dieťaťa cieľovým stavom, potom je parameter goalNotFound nastavený na hodnotu False, čo znamená, že sa našlo riešenie. Inak bude dieťa umiestnené do fronty.

```
def dfs(problem):
    problem.timer.start()
    start_time = time.time()

    node = Node(problem)
    if problem.goalTest(node.state):
        print("A solution found. ")
        print(node.state)
        print("Execution time is %s seconds" % (time.time() - start_time))
        return

    frontier = []
    frontier.append(node)
    explored = []

    goalNotFound = True
    num_expnded_node = 0
    while not problem.timeout and goalNotFound and frontier:
        node = frontier.pop()
        num_expnded_node += 1
        explored.append(node.state)
        for action in problem.actions:
            temp_x = node.location[0] + action[0]
            temp_y = node.location[1] + action[1]
            if temp_x > 0 and temp_x <= problem.n and temp_y > 0 and temp_y
<= problem.n:
                child = Node(problem, node, action)
                if child.location not in node.state and child.state not in
explored and not isChildInFrontier(frontier,
```

```
child):
                    if problem.goalTest(child.state):
                        goalNotFound = False
                        break
                    frontier.append(child)
    if not goalNotFound and not problem.timeout:
        print("A solution found. ")
        print(child.state)
        print("Execution time is %s seconds" % (time.time() - start_time))
    elif goalNotFound and not problem.timeout:
        print("No solution exists. ")
        print(child.state)
    elif goalNotFound and problem.timeout:
        print("Timeout")
    print("Searched with depth first search method. ")
    print("Number of nodes expanded: " + str(num_expnded_node))
```

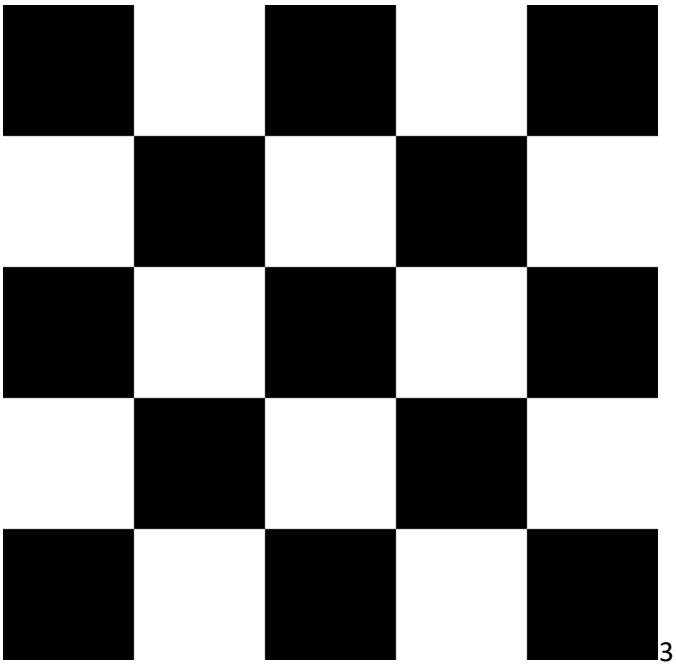
Pohyb po šachovnici

Kôň má 8 možností pohybu a teda 8 operátorov, tie sú nasledovné:

```
self.actions = [[-2, 1], [-2, -1], [-1, 2], [1, 2],
                [2, 1], [2, -1], [1, -2], [-1, -2]]
```

Neriešiteľné prípady

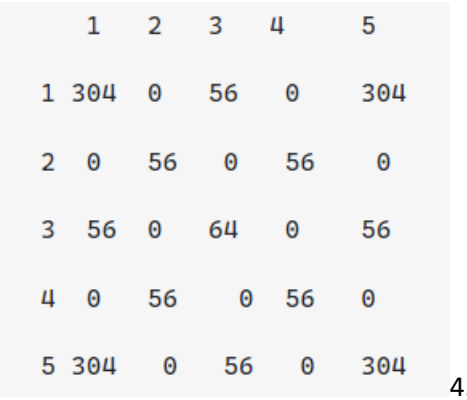
V mojom prípade, keďže sa zaoberám iba šachovnicami veľkosti 5x5 a 6x6, tak pri šachovnici o veľkosti 5x5 nie vždy existuje riešenie. Takýto záver nastáva práve vtedy, keď kôň začína na bielom políčku šachovnice, podľa obrázku nižšie:

 3.

Taktiež to môžeme definovať nasledovne: Ak ľavý horný roh má súradnice (1, 1) tak potom v prípade, že kôň začína na súradniciach, ktoré x-ová je párna a y-ová je nepárna alebo naopak, pre danú začiatočnú súradnicu neexistuje riešenie.

Norbert Matuška
UI Streda 15:00-16:40

Dobre znázornené aj na nasledujúcej šachovnici:



	1	2	3	4	5
1	304	0	56	0	304
2	0	56	0	56	0
3	56	0	64	0	56
4	0	56	0	56	0
5	304	0	56	0	304

4.

Čo znamená, že ak začíname na rohových súradniciach, napríklad (1, 5) tak môže byť 304 rôznych riešení pre euler-ovho koňa.

Imagine that you are coloring the board like a checkerboard, where the center square is white. Then the board has 13 white squares and 12 black squares.

Every time a knight moves, it changes the color of its square. So if you start on a black square, after 1 move you'll be on a white square (with your original square used up), after 2 moves you'll be on a black square (with 1 black and 1 white square used up), after 3 moves you'll be on a white square (with 2 blacks and 1 white square used up).

Continuing this pattern, after 24 moves, you will have used up 12 black squares, 12 white squares, and your knight will be on a white square.

But the only unused square is white!

It's impossible for you to get to that last square, because it's the same color as the square your knight is on.

5.

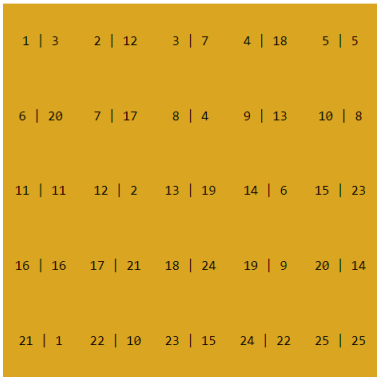
Norbert Matuška
UI Streda 15:00-16:40

Testovanie

5x5 šachovnica:

Vysvetlenie pre obrázky:

Číslo políčka | číslo kroku



Test number 1:

For initial state: [1, 5]

A solution found.

[[1, 5], [2, 3], [1, 1], [3, 2], [5, 1], [4, 3], [3, 1], [5, 2], [4, 4], [2, 5], [1, 3], [2, 1], [4, 2],

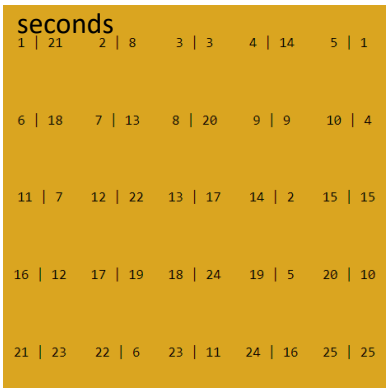
[5, 4], [3, 5], [1, 4], [2, 2], [4, 1], [3, 3], [1, 2], [2, 4], [4, 5], [5, 3], [3, 4], [5, 5]]

Execution time is 0.0010280609130859375

Searched with depth first search method.

Number of nodes expanded: 109

Pomôcka pre vytvorenie obrázku zo stránky: https://courageous-cactus-21740b.netlify.app/



Test number 2:

For initial state: [5, 1]

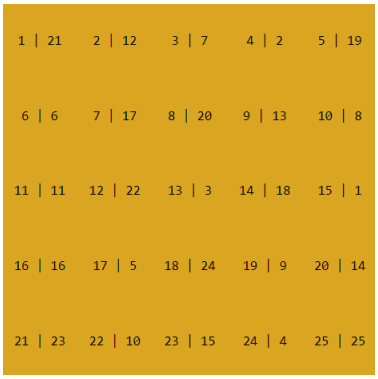
A solution found.

[[5, 1], [4, 3], [3, 1], [5, 2], [4, 4], [2, 5], [1, 3], [2, 1], [4, 2], [5, 4], [3, 5], [1, 4], [2, 2], [4, 1], [5, 3], [4, 5], [3, 3], [1, 2], [2, 4], [3, 2], [1, 1], [2, 3], [1, 5], [3, 4], [5, 5]]

Execution time is 6.667561292648315 seconds

Searched with depth first search method.

Number of nodes expanded: 29651



Test number 3:

For initial state: [5, 3]

A solution found.

[[5, 3], [4, 1], [3, 3], [4, 5], [2, 4], [1, 2], [3, 1], [5, 2], [4, 4], [2, 5], [1, 3], [2, 1], [4, 2], [5, 4], [3, 5], [1, 4], [2, 2], [4, 3], [5, 1], [3, 2], [1, 1], [2, 3], [1, 5], [3, 4], [5, 5]]

Execution time is 30.941256999969482 seconds

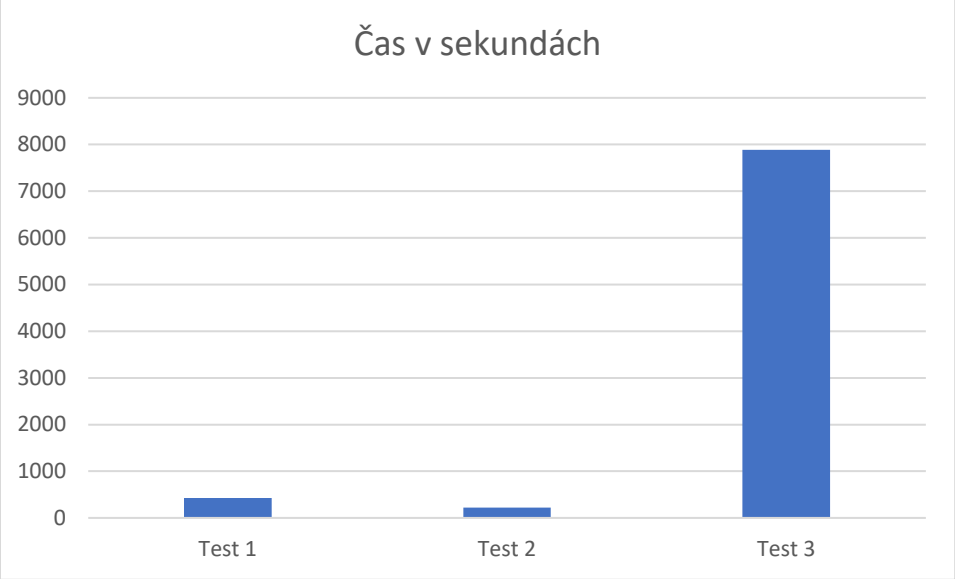
Searched with depth first search method.

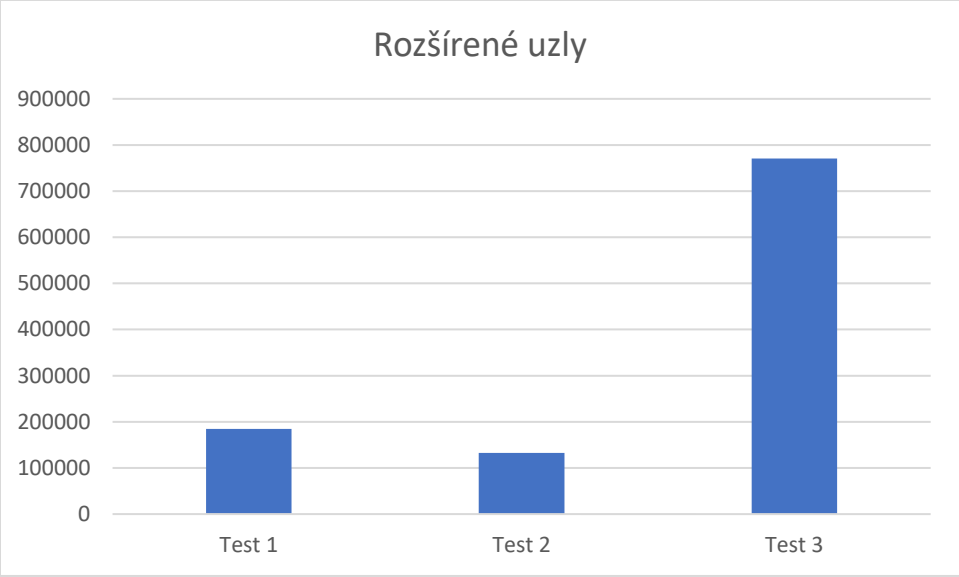
Number of nodes expanded: 59384

Norbert Matuška
UI Streda 15:00-16:40

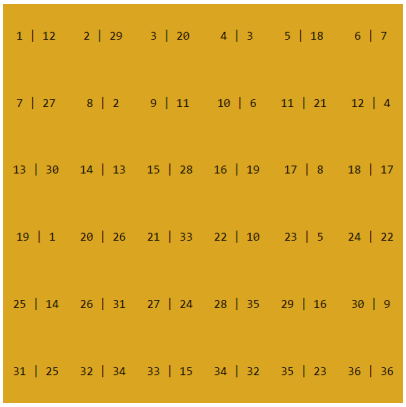
Porovnanie pre 5x5 šachovnice

	Test 1	Test 2	Test 3
Čas v sekundách	0.001028	6.667561	30.94126
Rozšírené uzly	109	29651	59384





Norbert Matuška
UI Streda 15:00-16:40



6x6 šachovnica

Test number 1:

For initial state: [1, 4]

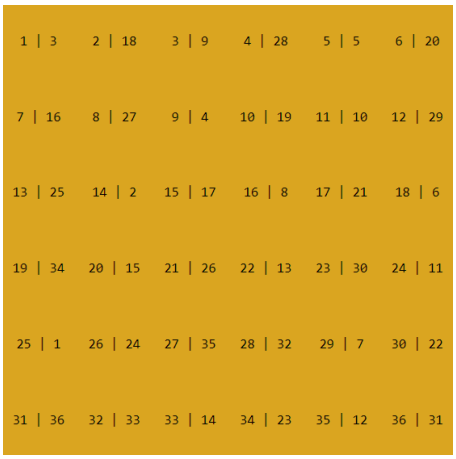
A solution found.

[[1, 4], [2, 2], [4, 1], [6, 2], [5, 4], [4, 2], [6, 1], [5, 3], [6, 5], [4, 4], [3, 2], [1, 1], [2, 3], [1, 5], [3, 6], [5, 5], [6, 3], [5, 1], [4, 3], [3, 1], [5, 2], [6, 4], [5, 6], [3, 5], [1, 6], [2, 4], [1, 2], [3, 3], [2, 1], [1, 3], [2, 5], [4, 6], [3, 4], [2, 6], [4, 5], [6, 6]]

Execution time is 426.6281979084015 seconds

Searched with depth first search method.

Number of nodes expanded: 184529



Test number 2:

For initial state: [1, 5]

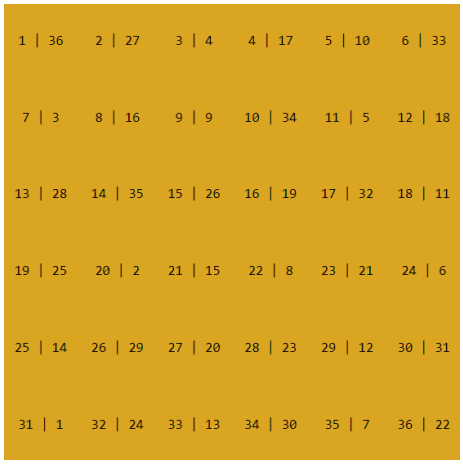
A solution found.

[[1, 5], [2, 3], [1, 1], [3, 2], [5, 1], [6, 3], [5, 5], [4, 3], [3, 1], [5, 2], [6, 4], [5, 6], [4, 4], [3, 6], [2, 4], [1, 2], [3, 3], [2, 1], [4, 2], [6, 1], [5, 3], [6, 5], [4, 6], [2, 5], [1, 3], [3, 4], [2, 2], [4, 1], [6, 2], [5, 4], [6, 6], [4, 5], [2, 6], [1, 4], [3, 5], [1, 6]]

Execution time is 217.8883991241455 seconds

Searched with depth first search method.

Number of nodes expanded: 132813



Test number 3:

For initial state: [1, 6]

A solution found.

[[1, 6], [2, 4], [1, 2], [3, 1], [5, 2], [6, 4], [5, 6], [4, 4], [3, 2], [5, 1], [6, 3], [5, 5], [3, 6], [1, 5], [3, 4], [2, 2], [4, 1], [6, 2], [4, 3], [3, 5], [5, 4], [6, 6], [4, 5], [2, 6], [1, 4], [3, 3], [2, 1], [1, 3], [2, 5], [4, 6], [6, 5], [5, 3], [6, 1], [4, 2], [2, 3], [1, 1]]

Execution time is 7889.918001413345 seconds

Searched with depth first search method.

Number of nodes expanded: 770744

```
Enter board size: 6
Enter timelimit: 36000
Initial starting positions:

[[1, 6], [4, 4], [1, 4], [5, 6], [1, 1]]

Test number 1:
For initial state: [1, 6]
A solution found.
[[1, 6], [2, 4], [1, 2], [3, 1], [5, 2], [6, 4], [5, 6], [4, 4], [3, 2], [5, 1], [6, 3], [5,
Execution time is 7889.918001413345 seconds
Searched with depth first search method.
Number of nodes expanded: 770744
```

Test number 4:

For initial state: [1, 6]

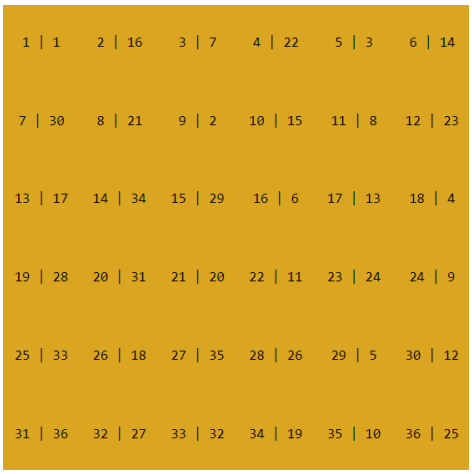
A solution found.

[[1, 1], [3, 2], [5, 1], [6, 3], [5, 5], [4, 3], [3, 1], [5, 2], [6, 4], [5, 6], [4, 4], [6, 5], [5, 3], [6, 1], [4, 2], [2, 1], [1, 3], [2, 5], [4, 6], [3, 4], [2, 2], [4, 1], [6, 2], [5, 4], [6, 6], [4, 5], [2, 6], [1, 4], [3, 3], [1, 2], [2, 4], [3, 6], [1, 5], [2, 3], [3, 5], [1, 6]]

Execution time is 183.55751156806946 seconds

Searched with depth first search method.

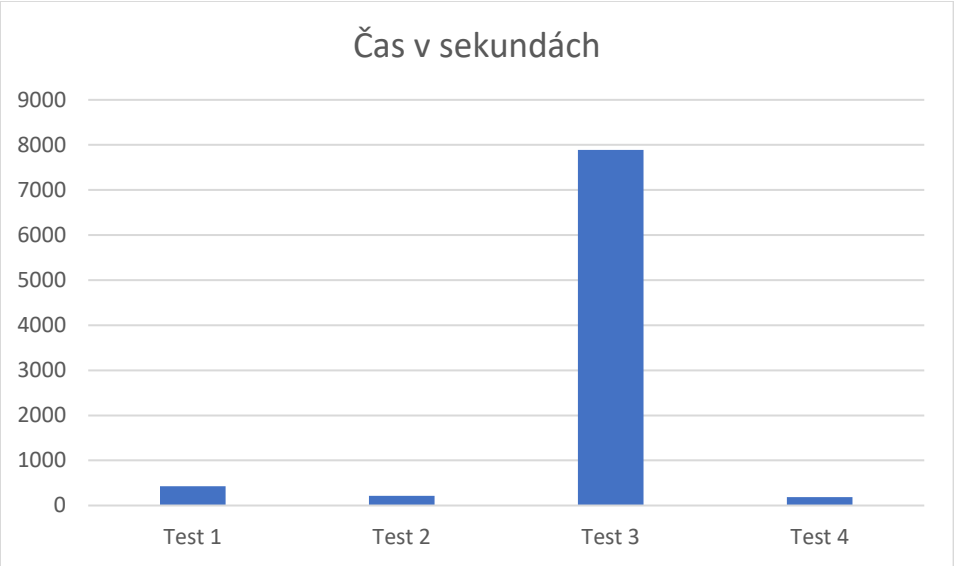
Number of nodes expanded: 121694

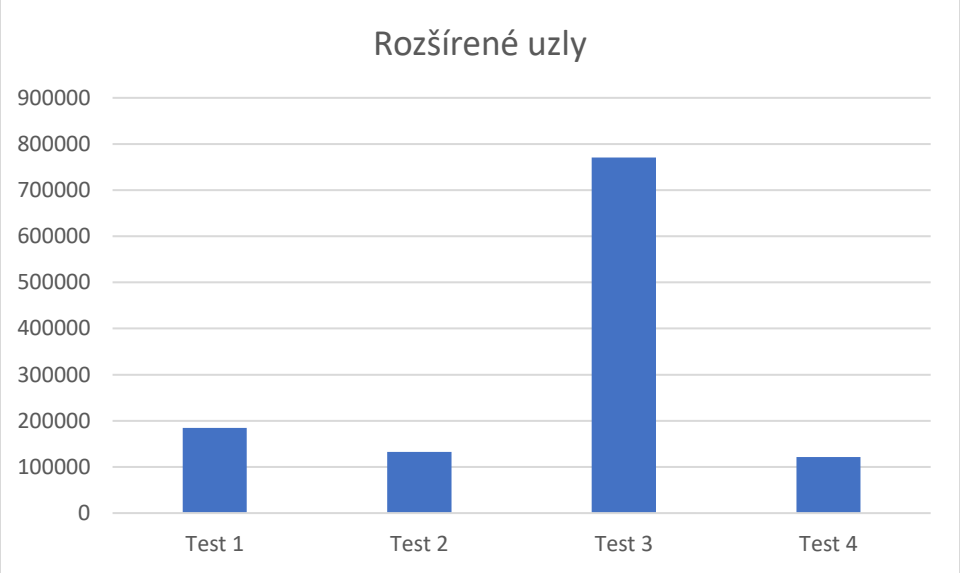


Norbert Matuška
UI Streda 15:00-16:40

Porovnanie pre 6x6 šachovnicu

	Test 1	Test 2	Test 3	Test 4
Čas v sekundách	426.6282	217.8884	7889.918	183.5575
Rozšírené uzly	184529	132813	770744	121694





Norbert Matuška
UI Streda 15:00-16:40

Používateľská príručka

Program v podstate robí všetko za vás, stačí spraviť nasledovné kroky:

1. Otvoriť main.py v nejakom IDE alebo python 3 kompilátore
2. Spustiť program
3. Napísať veľkosť šachovnice
4. Napísať časový limit podľa ktorého má program vedieť kedy skončiť prehľadávanie a vrátiť neúspešné hľadanie ak vyprší čas
5. Nechať program otestovať 4 náhodné a jednu predurčenú začiatočnú pozíciu
6. Na konci testovania sa Vás program opýta či chcete pokračovať, napíšte „y“ pre pokračovanie a „n“ pre skončenie programu

Zdroje obrázkov

1. https://archive.lib.msu.edu/crcmath/math/math/k/k099.htm
2. https://medium.com/basecs/breaking-down-breadth-first-search-cebe696709d9
3. https://commons.wikimedia.org/wiki/File:Checkerboard_pattern.svg
4. https://stackoverflow.com/questions/31411487/knights-tour-on-a-5-x-5-board-start-from-any-square
5. https://stackoverflow.com/questions/31411487/knights-tour-on-a-5-x-5-board-start-from-any-square

Zdroje

1. NÁVRAT, P. et al. Umelá inteligencia. STU Bratislava, 2015. 47 s. ISBN 9788022743440