Martin Čajka
AIS ID: 116158

Cvičenie: Streda 15:00

Prehľadávanie stavového priestoru

Problém 3. - Eulerov Ťah

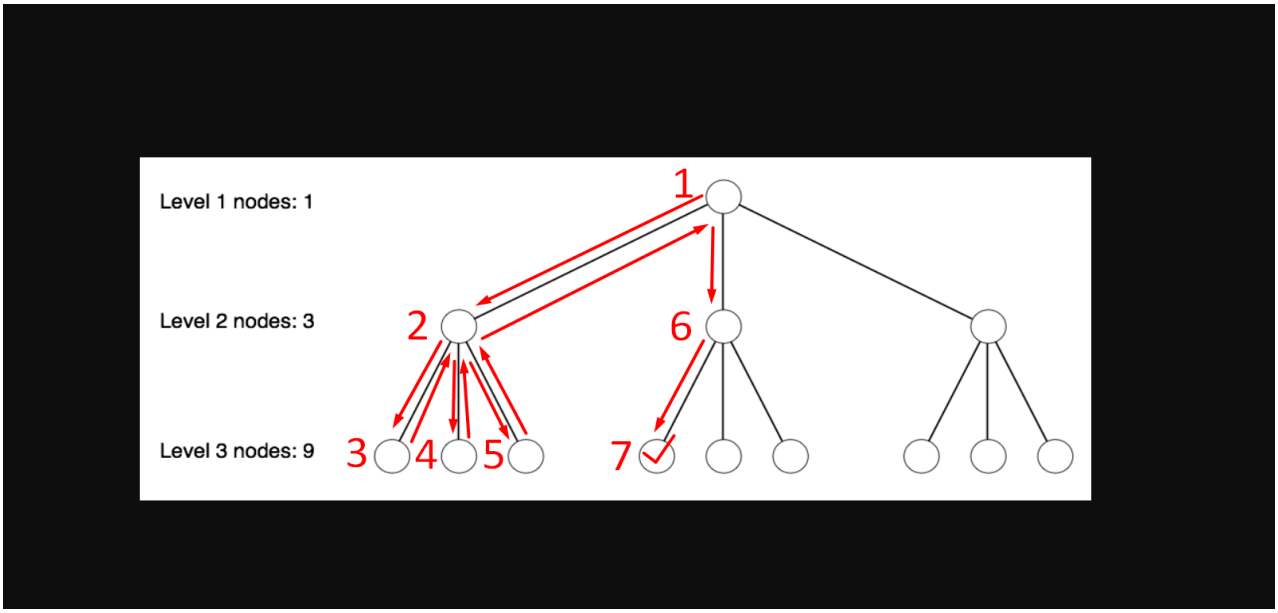
Úloha (f)

Opis zadania

Úlohou zadania je implementovať algoritmus prehľadávania do hĺbky (DFS) na riešenie Eulerovho ťahu na šachovnici - hlavolame známom aj ako Jazdcova cesta (Knight's Tour). Eulerov ťah na šachovnici predstavuje prejdenie každého poľa šachovnice legálnymi ťahmi jazdca práve raz. Algoritmus musí byť schopný nájsť cieľový stav a cestu k nemu z ľubovoľnej štartovnej pozície, ak je to možné, na šachovniciach o veľkosti 5x5 a 6x6. Ak sa do stanoveného limitu riešenie nenájde, algoritmus informuje o neúspešnom hľadaní a efektívnosti. Na účely testovania je potrebné si vybrať 5 náhodných štartovných pozícii pre šachovnice oboch rozmerov, pričom jedna z nich bude dolný ľavý roh.

Algoritmus prehľadávania do hĺbky (DFS)

Prehľadávanie do hĺbky je algoritmus využívaný pri prehľadávaní stromových a grafových dátových štruktúr. Počiatok algoritmu je na nejakom koreňovom uzle (v našom prípade štartovná pozícia jazdca) a odtiaľ začne následne prehľadávať uzly po vetve stromu. Akonáhle narazí na prekážku, vráti sa o uzol späť a prehľadáva susedov neúspešného uzla . V kontexte nášho problému to znamená, že algoritmus začne vyhľadávať pozície/stavy šachovnice postupnými ťahmi jazdca pokiaľ to je možné. Ak nie, vráti sa o ťah spať a volí inú cestu.



Obrázok 1 - Prehľadávanie do hĺbky najprv prehľadá ľavú vetvu kompletne, až následne začne prehľadávať druhú (bod 6. na grafe)

Martin Čajka
AIS ID: 116158

Cvičenie: Streda 15:00

Riešenie

Na začiatok riešenia si bolo potrebné zadefinovať množinu operátorov a reprezentáciu hracej plochy. Pre hraciu plochu bolo vytvorené pole o veľkosti n*n, pričom jednotlivé indexy sú očíslované od 0 až po n-1. Množina operátorov preto pozostáva s možných pohybov jazdca na šachovnici. Tie sa pripočítavajú k aktuálnej pozícii jazdca pre účel pohybu.

```
OPERATORS = [(-2,1),(-1,2),(1,2),(2,1),(2,-1),(1,-2),(-1,-2),(-2,-1)]
```

Program pri spustení od užívateľa požiada vstup pre veľkosť šachovnice a 2 vstupy pre vybratie štartovnej pozície jazdca (jeden vstup pre x-ovú os a druhy pre y-ovú). Následne prejde kontrolou danej pozície či sa nachádza na šachovnici pre dané rozmery. Ak áno, program pokračuje inicializáciou objektu z triedy ktorá reprezentuje šachovnicu – jej atribúty obsahujú všetky informácie o danom stave:

```
24    BOARD = SimBoard(startingPosition)          #inicializacia plochy
```

```
68  class SimBoard:
69
70      def __init__(self,startingPosition):
71          self.markedFields = []
72          self.path = []
73          self.position = startingPosition
74          self.depth = 0
75          self.success = False
76          self.efficiencyLimit = False
77
```

Obrázok 2 – Medzi atribúty patrí cesta (v poradí doposiaľ prejdené uzly), označené polia (už prejdené polia), aktuálna pozícia jazda na šachovnici, hĺbka (v kóde označuje aktuálne poradie ťahu) a bool success/efficiencyLimit ktoré zodpovedajú či už bol dosiahnutý cieľový stav/maximálny počet krokov.

Ďalej sa spustí funkcia startKnightsTour, štartovacia funkcia na spustenie rekurzívneho prehľadávanie do hĺbky a záverečný výpis podľa dosiahnutia cieľových podmienok pre nájdenie cesty.

```
27  def startKnightsTour(startingPosition):         #startovacia funkcia na spustenie prehladavania stavov a finalny vypis podla dosiahnutia cielovehc
28      print("Starting Knight's Tour at position: ("+str(startingPosition[0])+","+str(startingPosition[1])+")")
29      moveKnight()                    #rekurzivna funkcia prhladavania do hlbky
30
31      if BOARD.efficiencyLimit == True:               #dosiahnuty maximalny limit prehladaných uzlov
32          print("The algorithm was killed because the maximum number of moves has been reached.")
33
34      elif BOARD.success == False:                #kontrola ci sa nasiel cielovy stav - ak nie z danej pozicie nie je mozne vykonať eulerov tah
35          print("The Knight's Tour cannot be executed from the given starting position.")
36
37      else:                   #BOARD.success == True - uspesne najdeny cielovy stav, zobrazenie sachovnice a vypis cesty
38          printBoard()
39          print()
40          print("*****-Final Path-*****")
41          print(BOARD.path)
42
43      stop = time.time()      #zastavenie merania trvania algoritmu
44
45      print()
46      print("*****-Results-*****")
47      print("Number of executed moves: "+ str(NUM_OF_MOVES))
48      print("Time to complete algorithm: "+"{:.2f}".format(stop-start)+" seconds.")
```

Obrázok 3 – Funkcia spúšťa rekurzívnu funkciu moveKnight(), po vrátení sa z rekurzie sa skontroluje či bol dosiahnutý maximálny počet krokov (algoritmus bol vypnutý kvôli neefektivite) alebo nedosiahnutý cieľový stav (Program nebol schopný z danej štartovacej pozície cieľový stav). V prípade že ani jedna z týchto podmienok nenastala, cieľový stav bo dosiahnutý a program vypíše plochu, cestu a namerané časové a pamäťové výsledky.

Hlavná funkcia ktorá je základom algoritmu, je funkcia moveKnight(). Reprezentuje pohyby jazdca a prehľadávanie stavového diagramu za pomoci rekurzie. Funkcia po zavolaní začne pracovať s atribútmi objektu šachovnice. Ako prvé pridá súčasnú pozíciu do cesty a označených polí (v prvej rekurzií to je štartovná pozícia). Taktiež inkrementuje globálnu premennú počítajúcu počet prehľadaných uzlov a hĺbku.

```
111     BOARD.addToPath()
```

```
83      def addToPath(self):        #prida danu poziciu do zozn
84          global NUM_OF_MOVES
85          self.path.append(self.position)
86          self.depth += 1
87          self.markedFields.append(self.position)
88          NUM_OF_MOVES += 1
```

Nasledujú kontrolné podmienky pre prípad prestavenia kontrolných boolean premenných, ak už bol dosiahnutý maximálny počet krokov alebo dosiahnutý cieľový stav.

```
115     if NUM_OF_MOVES == MAX_NUM_OF_MOVES:
116             BOARD.efficiencyLimit = True
117             return
118
119     if BOARD.depth == SIZE**2:
120         print("The knight's tour is complete!")
121         BOARD.success = True
122         return
```

Obrázok 4 – Prestavenie atribútov efficiencyLimit a success objektu stavového diagramu na True .

Martin Čajka
AIS ID: 116158

Cvičenie: Streda 15:00

Po prejdení kontrolných podmienok sa vytvorí premenná moves ktorá nadobúda zoznam možných ťahov z danej pozície zavolaním funkcie currentLegalMoves() s parametrom súčasnej pozície. Spomenutá funkcia postupne prepočíta možné nasledujúce pozície pripočítaním operátorov a overením či možné pozície sú naozaj možné – nachádzajú sa na šachovnici a ešte neboli prejdené. V kladnom prípade sa pripoja do zoznamu ďalších možných pozícii a funkcia vráti tento zoznam.

```
124         moves = currentLegalMoves(BOARD.position)
```

```
51  def currentLegalMoves(position):
52      global BOARD
53      nextPossiblePositions = []
54
55      for move in OPERATORS:
56          nextPosition = (position[0]+move[0],position[1]+move[1])
57
58          if 0 <= nextPosition[0] < SIZE and 0 <= nextPosition[1] < SIZE:
59              if nextPosition not in BOARD.markedFields:
60                  nextPossiblePositions.append(nextPosition)
61
62      return nextPossiblePositions
63
```

Obrázok 5 – Funkcia precyklí operátormi, v každom cykle počíta súčet x-ových a y-ových hodnôt usporiadaných dvojíc operátorov a súčasného ťahu. Ak je výsledná pozícia v rámci šachovnici a nie je označená, pridá sa do zoznamu ktorý funkcia vracia.

Následne sa podobne prechádza redukovaným zoznamom možných ťahov aj vo funkcií moveKnight(), tento krát v snahe dosiahnutia rekurzívneho prehľadávania plochy pre danú pozíciu. Pri každej iterácií sa kontrolujú podmienky dosiahnutia cieľového stavu alebo maximálneho počtu krokov. V tejto časti programu slúžia na zastavenie rekurzie. Ak ešte nie sú dosiahnuté podmienky, funkcia ukladá práve iterovanú pozíciu ako súčasnú a volá samú seba. Na záver ak funkcia prešla všetkými ťahmi pre aktuálnu pozíciu a vrátila sa z rekurzie, do nej ani nešla a ani nedosiahla cieľový stav, odstraňuje sa súčasná pozícia z cesty a označených polí a pokračuje sa v rekurzií prechádzajúceho uzla. Po skončení rekurzie vo funkcii program dočíta funkciu startKnightsTour ktorá sa postará a finálny výpis.

```
128     for move in moves:                                          #
129
130         if BOARD.success or BOARD.efficiencyLimit:              #
131             break
132         ##print("Trying move: "+str(move))
133
134         BOARD.position = move                                   #
135
136         moveKnight()                                            #
137
138     if (BOARD.success == False):                                #
139         ##print("Removing from path")
140         BOARD.removeFromPath()
```

Obrázok 6 - Break statement ruší rekurziu po dosiahnutý niektorej z podmienok

Martin Čajka
AIS ID: 116158

Cvičenie: Streda 15:00

Testovanie a Analýza

V zadaní úlohy je jednou z požiadaviek program otestovať na sade 10 testovacích príkladov, rozdelených pre šachovnice 5x5 a 6x6, pričom jedno zo štartovacích polí pre obe šachovnice musí byť ľavý dolný roh šachovnice (0,4). Zvolené štartovacie pozície sú nasledovné.

Šachovnica 5x5 - (0,4) , (1,0) , (4,2) , (4,4) , (3,2)

Šachovnica 6x6 - (0,5) , (2,2) , (4,3) , (5,1) , (5,5)

Na meranie časovej zložitosti bola použitá knižnica time, konkrétne funkcia time.time() na zaznamenanie aktuálneho času. Počítanie času začalo po zadaní vstupov užívateľom a ukončilo sa pred výpisom zložitostí. Na meranie pamäťovej zložitosti bol stanovený limit 1 500 000 prehľadaných uzlov a pri pridávaní uzlov do cesty na prehľadávanie sa inkrementovalo premennou na počítanie prehľadaných uzlov.

```
10  def main():
11      global SIZE,BOARD,start
12      print()
13      SIZE = int(input("Enter size of board: "))
14
15      x = int(input("Enter x coordinate of the starting position: "))
16      y = int(input("Enter y coordinate of the starting position: "))
17
18      start = time.time()                          #spustenie casovaca
```

```
45      stop = time.time()     #zastavenie merania trvania algoritmu
46
47      print()
48      print("*****-Results-*****")
49      print("Number of executed moves: "+ str(NUM_OF_MOVES))
50      print("Time to complete algorithm: "+"{:.2f}".format(stop-start)+" seconds.")
51
```

Vo výpise sa zobrazuje rozdiel dvoch zaznamenaných časov.

Testovacie príklady:

Test 5x5, pozícia (0,4)

```
Enter size of board: 5
Enter x coordinate of the starting position: 0
Enter y coordinate of the starting position: 4

Starting Knight's Tour at position: (0,4)
The knight's tour is complete!

*****-The Final Board-*****
[23, 14, 21, 8, 25]
[20, 9, 24, 13, 18]
[15, 22, 19, 4, 7]
[10, 5, 2, 17, 12]
[1, 16, 11, 6, 3]

*****-Final Path-*****
[(0, 4), (2, 3), (4, 4), (3, 2), (1, 3), (3, 4), (4, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (0, 2), (1, 4), (3, 3), (4, 1), (2, 2),
(0, 1), (2, 0), (1, 2), (0, 0), (2, 1), (4, 0)]

*****-Results-*****
Number of executed moves: 288
Time to complete algorithm: 0.00 seconds.
```

Obrázok 7 - Algoritmus z pozície (0,4) našiel cestu, prehľadal 288 uzlov a trvalo mu to 0,00 sekúnd.

Martin Čajka
AIS ID: 116158

Cvičenie: Streda 15:00

Test 5x5, pozícia (1,0)

```
Enter size of board: 5
Enter x coordinate of the starting position: 1
Enter y coordinate of the starting position: 0

Starting Knight's Tour at position: (1,0)
The algorithm was killed because the maximum number of moves has been reached.

*****-Results-*****
Number of executed moves: 1500000
Time to complete algorithm: 4.83 seconds.
```

Obrázok 8 - Algoritmus cestu nenašiel, dosiahol maximálny počet prehľadaných uzlov 1 500 000 a trvalo mu to 4,83 sekúnd.

Test 5x5, pozícia (4,2)

```
Enter size of board: 5
Enter x coordinate of the starting position: 4
Enter y coordinate of the starting position: 2

Starting Knight's Tour at position: (4,2)
The knight's tour is complete!

*****-The Final Board-*****
[23, 10, 15, 4, 25]
[16, 5, 24, 9, 14]
[11, 22, 3, 18, 1]
[6, 17, 20, 13, 8]
[21, 12, 7, 2, 19]

*****-Final Path-*****
[(4, 2), (3, 4), (2, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (0, 2), (1, 4), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 2), (4, 4),
(2, 3), (0, 4), (1, 2), (0, 0), (2, 1), (4, 0)]

*****-Results-*****
Number of executed moves: 365421
Time to complete algorithm: 1.16 seconds.
```

Obrázok 9 - Algoritmus z pozície (4,2) našiel cestu, prehľadal 365421 uzlov a trvalo mu to 1,15 sekúnd.

Test 5x5, pozícia (4,4)

```
Enter size of board: 5
Enter x coordinate of the starting position: 4
Enter y coordinate of the starting position: 4

Starting Knight's Tour at position: (4,4)
The knight's tour is complete!

*****-The Final Board-*****
[23, 12, 15, 6, 25]
[14, 7, 24, 11, 16]
[19, 22, 13, 2, 5]
[8, 3, 20, 17, 10]
[21, 18, 9, 4, 1]

*****-Final Path-*****
[(4, 4), (3, 2), (1, 3), (3, 4), (4, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (2, 2), (0, 1), (2, 0), (4, 1), (3, 3), (1, 4), (0, 2),
(2, 3), (0, 4), (1, 2), (0, 0), (2, 1), (4, 0)]

*****-Results-*****
Number of executed moves: 14009
Time to complete algorithm: 0.06 seconds.
```

Obrázok 10 - Algoritmus z pozície (4,4) našiel cestu, prehľadal 14009 uzlov a trvalo mu to 0,06 sekúnd.

Martin Čajka
AIS ID: 116158

Cvičenie: Streda 15:00

Test 5x5, pozícia (3,2)

```
Enter size of board: 5
Enter x coordinate of the starting position: 3
Enter y coordinate of the starting position: 2

Starting Knight's Tour at position: (3,2)
The Knight's Tour cannot be executed from the given starting position.

*****-Results-*****
Number of executed moves: 1028893
Time to complete algorithm: 3.20 seconds.
```

Obrázok 11 - Algoritmus z pozície (3,2) cestu nenašiel a zistil, že z danej pozície nie je možné nájsť cestu. (nedosiahol max. počet krokov a ukončil sa prirodzene).

Test 6x6, pozícia (0,5)

```
Enter size of board: 6
Enter x coordinate of the starting position: 0
Enter y coordinate of the starting position: 5

Starting Knight's Tour at position: (0,5)
The knight's tour is complete!

*****-The Final Board-*****
[20, 29, 14, 33, 18, 23]
[15, 32, 19, 22, 13, 34]
[28, 21, 30, 17, 24, 11]
[31, 16, 7, 12, 35, 4]
[8, 27, 2, 5, 10, 25]
[1, 6, 9, 26, 3, 36]

*****-Final Path-*****
[(0, 5), (2, 4), (4, 5), (5, 3), (3, 4), (1, 5), (2, 3), (0, 4), (2, 5), (4, 4), (5, 2), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 2), (4, 0), (2, 1),
(0, 0), (1, 2), (3, 1), (5, 0), (4, 2), (5, 4), (3, 5), (1, 4), (0, 2), (1, 0), (2, 2), (0, 3), (1, 1), (3, 0), (5, 1), (4, 3), (5, 5)]

*****-Results-*****
Number of executed moves: 177048
Time to complete algorithm: 0.70 seconds.
```

Obrázok 12 - Algoritmus z pozície (0,5) našiel cestu, prehľadal 177048 uzlov a trvalo mu to 0,70 sekúnd.

Test 6x6, pozícia (2,2)

```
Starting Knight's Tour at position: (2,2)
The knight's tour is complete!

*****-The Final Board-*****
[15, 26, 35, 22, 13, 24]
[36, 19, 14, 25, 34, 21]
[27, 16, 1, 20, 23, 12]
[2, 9, 18, 31, 6, 33]
[17, 28, 7, 4, 11, 30]
[8, 3, 10, 29, 32, 5]

*****-Final Path-*****
[(2, 2), (0, 3), (1, 5), (3, 4), (5, 5), (4, 3), (2, 4), (0, 5), (1, 3), (2, 5), (4, 4), (5, 2), (4, 0), (2, 1), (0, 0), (1, 2), (0, 4), (2, 3), (1, 1),
(3, 2), (5, 1), (3, 0), (4, 2), (5, 0), (3, 1), (1, 0), (0, 2), (1, 4), (3, 5), (5, 4), (3, 3), (4, 5), (5, 3), (4, 1), (2, 0), (0, 1)]

*****-Results-*****
Number of executed moves: 83112
Time to complete algorithm: 0.33 seconds.
```

Obrázok 13 - Algoritmus z pozície (2,2) našiel cestu, prehľadal 83112 uzlov a trvalo mu to 0,33 sekúnd.

Martin Čajka
AIS ID: 116158

Test 6x6, pozícia (4,3)

```
Enter size of board: 6
Enter x coordinate of the starting position: 4
Enter y coordinate of the starting position: 3

Starting Knight's Tour at position: (4,3)
The knight's tour is complete!

*****-The Final Board-*****
[26, 21, 12, 31, 28, 19]
[13, 32, 27, 20, 11, 30]
[22, 25, 14, 29, 18, 7]
[33, 4, 23, 8, 1, 10]
[24, 15, 2, 35, 6, 17]
[3, 34, 5, 16, 9, 36]

*****-Final Path-*****
[(4, 3), (2, 4), (0, 5), (1, 3), (2, 5), (4, 4), (5, 2), (3, 3), (4, 5), (5, 3), (4, 1), (2, 0), (0, 1), (2, 2), (1, 4), (3, 5), (5, 4), (4, 2), (5, 0),
(3, 1), (1, 0), (0, 2), (2, 3), (0, 4), (1, 2), (0, 0), (2, 1), (4, 0), (3, 2), (5, 1), (3, 0), (1, 1), (0, 3), (1, 5), (3, 4), (5, 5)]

*****-Results-*****
Number of executed moves: 897231
Time to complete algorithm: 3.35 seconds.
```

Obrázok 14 - Algoritmus z pozície (4,3) našiel cestu, prehľadal 897231 uzlov a trvalo mu to 3,35 sekúnd.

Test 6x6, pozícia (5,1)

```
Enter size of board: 6
Enter x coordinate of the starting position: 5
Enter y coordinate of the starting position: 1

Starting Knight's Tour at position: (5,1)
The algorithm was killed because the maximum number of moves has been reached.

*****-Results-*****
Number of executed moves: 1500000
Time to complete algorithm: 5.59 seconds.
```

Obrázok 15 - Algoritmus cestu nenašiel, dosiahol maximálny počet prehľadaných uzlov 1 500 000 a trvalo mu to 5,59 sekúnd.

Test 6x6, pozícia (5,5)

```
Starting Knight's Tour at position: (5,5)
The knight's tour is complete!

*****-The Final Board-*****
[21, 16, 33, 26, 23, 14]
[32, 27, 22, 15, 34, 25]
[17, 20, 31, 24, 13, 8]
[28, 5, 18, 9, 2, 35]
[19, 10, 3, 30, 7, 12]
[4, 29, 6, 11, 36, 1]

*****-Final Path-*****
[(5, 5), (4, 3), (2, 4), (0, 5), (1, 3), (2, 5), (4, 4), (5, 2), (3, 3), (1, 4), (3, 5), (5, 4), (4, 2), (5, 0), (3, 1), (1, 0), (0, 2), (2, 3), (0, 4),
(1, 2), (0, 0), (2, 1), (4, 0), (3, 2), (5, 1), (3, 0), (1, 1), (0, 3), (1, 5), (3, 4), (2, 2), (0, 1), (2, 0), (4, 1), (5, 3), (4, 5)]

*****-Results-*****
Number of executed moves: 58692
Time to complete algorithm: 0.23 seconds.
```

Obrázok 16 - Algoritmus z pozície (5,5) našiel cestu, prehľadal 58692 uzlov a trvalo mu to 0,23 sekúnd.

Zhrnutie:

Na základe výsledkov z testovania môžeme usúdiť že v algoritme je vidieť prvky prehľadávania do hĺbky – časové a pamäťové zložitosti sú výrazne rozdielne pre rôzne pozície. Na šachovniciach o veľkosti 5x5 to môžeme vidieť pri pozíciách (0,4) – (288 prehľadaných uzlov a čas 0,00 sekúnd) a (4,2) – (365421 uzlov a čas 1,15 sekúnd). Rozdiely sú spôsobené úplným prehľadávaním vetiev ktoré nevedú k cieľovému stavu pred prehľadaním vetiev ktoré k nemu vedú. Pozícia (0,4) sa podarí nájsť cieľový stav veľmi rýchlo, nakoľko sa na začiatku vyberie ťahmi ktoré

Martin Čajka
AIS ID: 116158

vedú k jeho dosiahnutiu. Tieto rozdiely by sa dali redukovať uplatnením nejakej heuristiky, ktorá by uprednostňovala prehľadávanie takých pozícii, ktoré majú väčšiu perspektívu dosiahnuť cieľový stav.

Na šachovnici 5x5 je nemožné nájsť cestu z pozícii ktoré nie sú na diagonálne, kvôli nepárnemu počtu polí. Je možné si povšimnúť že pre niektoré z týchto pozícii sa dá určiť či sa dá z nej dosiahnuť cieľový stav do limitu 1 500 000 uzlov alebo nie – vonkajšie pozície ako napríklad (1,0) majú viac možných kombinácii ťahov a z dôvodu maximálneho počtu prehľadaných uzlov bol ukončený program predtým ako dokázal prehľadať celú šachovnicu, zatiaľ čo vnútorné pozície ako (3,2) boli ukončené pri 1 028 893 prehľadaných uzlov bez nájdenie možnej cesty. Pre šachovnice 6x6 sú zaznamenané podobné časové a pamäťové rozdiely na základe štartovnej pozície. Pri šachovniciach 6x6 sa do stanoveného limitu 1 500 000 uzlov nedá zistiť, či pre štartovnú pozíciu existuje cesta. Algoritmus pôsobí rýchlo ale stanovený maximálny počet prehľadaných uzlov sa môže zvýšiť, aby bolo možné prehľadať viac na šachovniciach 6x6.