Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Umelá inteligencia

Prehľadávanie stavového prestoru

Tibor Dulovec

Meno cvičiaceho: Ing. Ivan Kapustík
Čas cvičení: Streda 13:00
Dátum vytvorenia: 24. 10. 2021

Obsah

Zadanie úlohy

Úlohou je nájsť riešenie **N**-hlavolamu. Hlavolam je zložený z **n** očíslovaných políčok a jedného prázdneho miesta. Políčka je možné presúvať hore, dole, vľavo alebo vpravo, ale len ak je tým smerom medzera. Je vždy daná nejaká východisková a nejaká cieľová pozícia a je potrebné nájsť postupnosť krokov, ktoré vedú z jednej pozície do druhej.

Implementačné prostredie

Program je vytvorený v programovacom jazyku Python vo verzií 3.9. Pre správne fungovanie sa využíva knižnica datetime, ktorá slúži pre vypočítanie času a efektívnosti funkcie a knižnicu random, pre testovanie a vytvorenie náhodných hlavolamov.

Priebeh programu a možnosti testovania

Po spustení programu kód hneď vykoná vypočítanie kroky k výsledku hlavolamu.

V prípade, že v kóde nie je staticky napísaný vstup a výstup sa tieto hodnoty vypočítajú náhodne, podľa poľa „size_of_generated_puzzle“, ktoré určuje rozmery výsledného hlavolamu.

Po tom ako sa hlavolam náhodne vygeneruje, vytvorí sa prvý uzol nazvaný root, ktorý reťazec vstupu premení na dvojrozmernú mapu a vloží do nej všetky hodnoty pre jednoduchšiu manipuláciu.

Vstup a cieľ sa vypíšu a po nájdení postupu sa vypíše aj posledný finálny uzol, pre overenie, či je správny.

Medzera je reprezentovaná znakom M

```
== START ==
M 5 8
7 1 6
2 4 3
=== Goal ===
4 7 M
2 6 1
8 5 3
============
```

Počas pracovania programu sa aktualizuje riadok, ktorý vypisuje aktuálny čas, hĺbku a počet uzlov

```
> Node Num. 517 | Depth: 22 | Time: 0:00:00.1| Average: 0:00:00.000296/node
```

Po spracovaní sa k nájdenému uzlu vypíše aj celkový čas, počet vykonaných krokov, vytvorených uzlov, priemerný čas pre jeden uzol a samozrejme postupnosť krokov pre docielenie riešenia.

```
======= RESULT =======
VPRAVO > DOLE > DOLE > VPRAVO > HORE > HORE > VLAVO > DOLE > VPRAVO > DOLE > VLAVO
Steps: 22
Nodes: 517
Duration: 0:00:00.153841
Avery time per Node: 0:00:00.000298
```

Príklad prebehnutého programu

```
== START ==
4 5 8
3 M 7
1 6 2
=== Goal ===
4 8 7
6 5 M
1 2 3
============
 > Node Num. 297 | Depth: 17 | Time: 0:00:00.0| Average: 0:00:00.000254/node
== FINISH ==
4 8 7
6 5 M
1 2 3
======= RESULT =======
DOLE > VPRAVO > HORE > VLAVO > VLAVO > DOLE > VPRAVO > HORE > VPRAVO > DOLE > VLAVO > VLA
Steps: 17
Nodes: 297
Duration: 0:00:00.075336
Avery time per Node: 0:00:00.000254

Process finished with exit code 0
```

Algoritmus

Vzhľadom na moje zadanie som použil algoritmus A*. Tento algoritmus rozvíja uzly podľa možností, ktoré môžeme v určitých stavoch vykonávať.

Postupne vykonáva všetky uzly, až kým nenájde uzol s heuristickou funkciou rovnej 0. Vtedy sme došli do finálneho a žiadaného stavu a žiaden prvok nie je na mieste, na akom by nemal byť.

Maximálny počet uzlov sa odvíja od veľkosti hlavolamu. Je to faktoriál násobku oboch strán.
V prípade 3x2 to je teda 6!. Čo je v prepočte 720 uzlov.

Dĺžka trvania výpočtu nepriamo úmerne rastie od počtu vypočítaných a otvorených uzlov. To z dôvodu, že počas vykonávania programu je stále viac otvorených uzlov, ktoré sa musia prehľadávať a hľadať ten s najnižšou rozhodovacou funkciou.

Po logickej stránke program funguje tak, že sa v cykle neustále vytvárajú nove uzly z toho s najnižšou heuristikou. Nové uzly sa vytvárajú podľa toho, aké možnosti posunu majú.

Možnosti posunu sa vypočítavajú podľa pozície prázdneho prvku a toho, kde sa môže posunúť. Následne sa pre každú jednu možnosť vytvorí uzol a presunie sa do otvorených uzlov.

Možnosti heuristických funkcií

V súbore node.py je v globalnej premennej definovaná premenná, ktorá určuje typ používanej heurestiky. *„heur_type“*

1) Počet políčok, ktoré nie sú na svojom mieste
2) Súčet vzdialeností jednotlivých políčok od ich cieľovej pozície
3) Kombinácia oboch

Najefektívnejšia je tretia varianta, práve kvôli tomu že používa kombináciu oboch. Druhá varianta je výpočtovo náročnejšia, ale vďaka presnejšiemu odhadu k potrebnému stav sa to časovo viac vyplatí a dosiahneme skôr výsledku.

Výsledky

Náhodné vstupy pre 3x3 puzzle

```
== START ==
8 2 4
3 1 5
7 M 6
=== Goal ===
1 5 6
4 3 7
M 2 8
============
 > Node Num. 197 | Depth: 23 | Time: 0:00:00.0| Average: 0:00:00.000243/node
== FINISH ==
1 5 6
4 3 7
M 2 8
======= RESULT =======
VLAVO > HORE > HORE > VPRAVO > DOLE > VLAVO > HORE > VPRAVO > VPRAVO > DOLE > DOLE > VLAV
Steps: 23
Nodes: 197
Duration: 0:00:00.047826
Avery time per Node: 0:00:00.000243
```

```
== START ==
4 5 8
3 M 7
1 6 2
=== Goal ===
4 8 7
6 5 M
1 2 3
============
 > Node Num. 297 | Depth: 17 | Time: 0:00:00.0| Average: 0:00:00.000254/node
== FINISH ==
4 8 7
6 5 M
1 2 3
======= RESULT =======
DOLE > VPRAVO > HORE > VLAVO > VLAVO > DOLE > VPRAVO > HORE > VPRAVO > DOLE > VLAVO > VLA
Steps: 17
Nodes: 297
Duration: 0:00:00.075336
Avery time per Node: 0:00:00.000254

Process finished with exit code 0
```

Náhodne vstupy pre 3x2

```
== START ==
1 4
3 2
5 M
=== Goal ===
2 4
3 M
1 5
============
 > Node Num. 38 | Depth: 13 | Time: 0:00:00.0| Average: 0:00:00.000026/node
== FINISH ==
2 4
3 M
1 5
======= RESULT =======
HORE > VLAVO > HORE > VPRAVO > DOLE > DOLE > VLAVO > HORE > VPRAVO > HORE > VLAVO > DOLE
Steps: 13
Nodes: 38
Duration: 0:00:00.000997
Avery time per Node: 0:00:00.000026
```

Vstup 2x2 a menšie

Pre rýchle overenie správnosti môžeme vyskúšať aj postupnosti ako tieto, ale tam narásta šanca, že riešenie nebude riešiteľné.

```
== START ==
M 2
3 1
=== Goal ===
2 M
3 1
============
 > Node Num. 1 | Depth: 1 | Time: 0:| Average: 0:00:00/node
== FINISH ==
2 M
3 1
======= RESULT =======
VPRAVO
Steps: 1
Nodes: 1
Duration: 0:00:00
Avery time per Node: 0:00:00
```