SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

ZADANIE 2 – RIEŠENIE 8-HLAVOLAMU

Úloha d)

Dávid Kromka
Cvičenie: utorok 16:00
25.10.2021

Meno: Dávid Kromka
Ič: 110834
Cvičenie: Slíž, utorok 16:00
Úloha d)

Obsah

Meno: Dávid Kromka
Ič: 110834
Cvičenie: Slíž, utorok 16:00
Úloha d)

1. Riešený problém

Zadaním úlohy je nájsť riešenie 8-hlavolamu pomocou algoritmu lačného vyhľadávania. 8-hlavolam pozostáva z 8 očíslovaných políčok a jedného prázdneho miesta. Políčka sa môžu posúvať v smere voľného miesta vpravo, vľavo, hore alebo dole. Je daná východisková a cieľová pozícia a je potrebné nájsť postupnosť krokov na prejdenie z jednej do druhej.

Pri riešení je potrebné použiť 2 heuristiky a porovnať ich. Prvou z nich je počet políčok, ktoré nie sú na svojom mieste a druhou je súčet vzdialeností jednotlivých políčok od ich cieľovej pozície.

2. Opis riešenia

Základným prvkom riešenia problému je reprezentácia uzla, ktorý sa skladá z hlavolamu s usporiadanými číslami reprezentovaný dvojrozmerným polom, v ktorom je prázdne políčko reprezentované číslom 0. Ďalším elementom uzla je odkaz na predchádzajúci uzol, z ktorého premiestnením políčka vznikol aktuálny uzol. Takáto štruktúra vytvára strom.

```
class Node:
    def __init__(self, data, next_node):
        self.data = data
        self.next = next_node
```

Funkcia **start(**self**)** slúži na spúšťanie funkciu na hľadanie postupnosti uzlov k cieľovému rozloženiu, meria čas hľadania v milisekundách a vypisuje výsledky hľadania.

Funkcia **search(**self, step_list**)** prechádza pole, v ktorom sú uzly, ktoré je potrebné ďalej rozvíjať a ktoré boli vrátené funckiou **move**(self) reprezentujúcou lačné hľadanie. Vytvára strom postupným pridávaním nových uzlov, ktoré ukazujú na ich predchodcov. Ak je počet preskúmaných uzlov väčší ako 25000, hľadanie sa ukončí, aby sme predišli priveľkej časovej a pamäťovej náročnosti.

```
while len(self.to_do) != 0:
    self.input = self.to_do.pop()
    self.node = Node(self.input, self.last_node)
    self.last_node = self.node
    self.all_nodes.append(self.input)
```

Funkcia **move(**self**)** zistí pozíciu prázdneho políčka a všetky pozície po pohyboch hore, dole, vpravo, vľavo. V ďalšom kroku sa podľa zvolenej heuristiky vypočíta, ktoré rozloženia sú najlepšie, vložia sa do poľa a funkcia vracia toto pole.

```
def up():
    if y == 0:
        return 0
    cp = [row[:] for row in self.input]
    cp[y][x], cp[y - 1][x] = cp[y - 1][x], cp[y][x]
    return cp
```

Meno: Dávid Kromka
Ič: 110834
Cvičenie: Slíž, utorok 16:00
Úloha d)

Pri prvej heuristike sa v cykle prechádza každé políčko hlavolamu v aktuálnom rozložení a porovnáva sa s políčkom na rovnakej pozícii v cieľovom rozložení. Sčítava sa počet nesúhlasných políčok bez prázdneho políčka, ktoré sa do heuristiky nezapočítava. Čím je tento počet vyšší, tým je počet zle umiestnených políčok väčší. Ak je tento počet rovný 0, je dosiahnutý cieľový stav.

Pri druhej heuristike sa počíta vzdialenosť políčka v aktuálnom rozložení od políčka s rovnakým číslom v cieľovom rozložení. V cykle sa prechádzajú políčka v aktuálnom rozložení, zapamätajú sa súradnice a v cieľovom rozložení sa nájdu súradnice políčka s rovnakou hodnotou. Následne sa sčítaním absolútnych hodnôt rozdielov x-ových a y-ových súradníc zistí ich vzdialenosť. Výsledkom heuristiky je súčet všetkých takýchto vzdialeností, prázdne políčko sa nezapočítava.

Program sa spúšťa v terminály a ovláda sa vstupmi z klávesnice. Zadávajú sa rozmery hlavolamu, vstup čísel na políčkach po riadkoch z ľavej strany do pravej a prázdne políčko je interpretované číslom 0. Čísla sa na vstupe oddeľujú čiarkou. Nasleduje výber heuristiky číslo 1 alebo 2. Na výstupe je počiatočný a cieľový stav a počet krokov spolu s trvaním hľadania cesty v milisekundách. Po tomto výpise si môže používateľ zvoliť výpis všetkých krokov, pokračovanie ďalším rozložením alebo ukončiť program.

```
Rozmer hlavolamu
Zadajte horizontálnu dĺžku: 3
Zadajte vertikálnu dĺžku: 3
Zadajte 9 hodnôt hlavolamu oddelených čiarkou, 0 na prázdne políčko: 1,2,3,4,5,6,7,8,0
Zadajte 9 hodnôt cieľového rozloženia oddelených čiarkou, 0 na prázdne políčko: 7,3,8,2,0,5,6,1,4
Heuristika '1' alebo '2': 2
Počiatočné rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Koncové rozloženie:
[7, 3, 8]
[2, 0, 5]
[6, 1, 4]
Čas hľadania: 148.377 ms
Počet krokov: 1442
Na výpis krokov stlačte 'v' na ukončenie stlačte 'e', na pokračovanie 'p': ■
```

3. Použitý algoritmus

V zadaní je využitý algoritmus lačného hľadania, kde je hlavným cieľom minimalizácia ceny cesty do cieľa. Odhad ceny najlacnejšej cesty z uzla do cieľového uzla sa určí na základe heuristiky. Táto hodnota nezávisí od stromu, ale len od aktuálneho a cieľového uzla. V prípade, že je pri pohybe z jedného uzla viac rovnako dobrých najlacnejších ohodnotení, strom sa rozvíja všetkými takýmito smermi. Lačné hľadanie nie je úplné a neoptimalizuje riešenie.

Meno: Dávid Kromka
Ič: 110834
Cvičenie: Slíž, utorok 16:00
Úloha d)

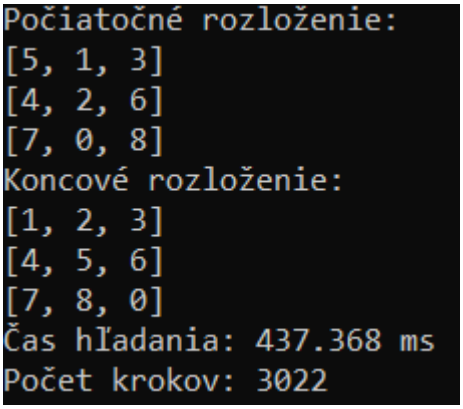
4. Testovanie

4.1. Spôsob testovania

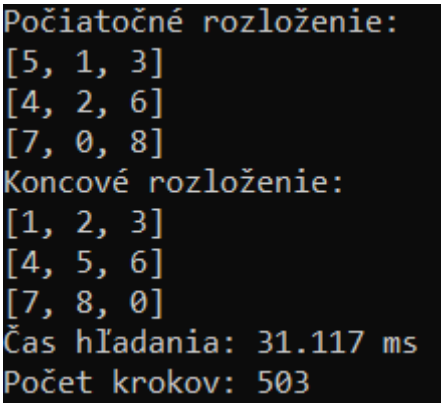
Pri testovaní hľadania krokov z východiskového rozloženia do cieľového rozloženia sa počíta čas hľadania v milisekundách a počet posunov prázdneho políčka.

4.2. Výsledky testovania

Test 1

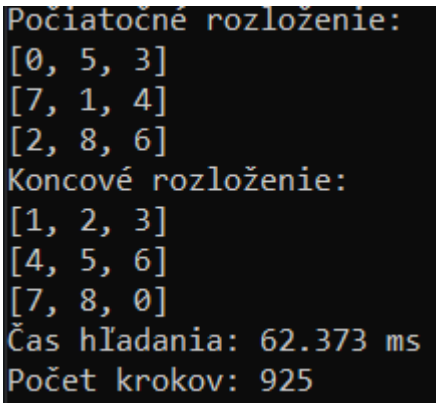


Heuristika 1

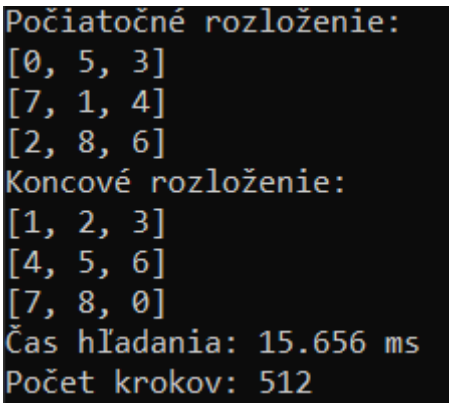


Heuristika 2

Test2



Heuristika 1



Heuristika 2

Meno: Dávid Kromka
Ič: 110834
Cvičenie: Slíž, utorok 16:00
Úloha d)

Test 3

```
Počiatočné rozloženie:
[7, 8, 5]
[0, 1, 3]
[2, 6, 4]
Koncové rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Čas hľadania: 46.742 ms
Počet krokov: 758
```

Heuristika 1

```
Počiatočné rozloženie:
[7, 8, 5]
[0, 1, 3]
[2, 6, 4]
Koncové rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Čas hľadania: 15.46 ms
Počet krokov: 452
```

Heuristika 2

Test 4

```
Počiatočné rozloženie:
[7, 3, 8]
[2, 0, 5]
[6, 1, 4]
Koncové rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Čas hľadania: 15.57 ms
Počet krokov: 410
```

Heuristika 1

```
Počiatočné rozloženie:
[7, 3, 8]
[2, 0, 5]
[6, 1, 4]
Koncové rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Čas hľadania: 46.897 ms
Počet krokov: 731
```

Heuristika 2

Test 5

```
Počiatočné rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Koncové rozloženie:
[7, 3, 8]
[2, 0, 5]
[6, 1, 4]
Čas hľadania: 31.117 ms
Počet krokov: 675
```

Heuristika 1

```
Počiatočné rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Koncové rozloženie:
[7, 3, 8]
[2, 0, 5]
[6, 1, 4]
Čas hľadania: 148.377 ms
Počet krokov: 1442
```

Heuristika 2

Meno: Dávid Kromka
Ič: 110834
Cvičenie: Slíž, utorok 16:00
Úloha d)

Test 6

```
Počiatočné rozloženie:
[6, 1, 8, 3, 5]
[0, 2, 7, 4, 9]
Koncové rozloženie:
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 0]
Čas hľadania: 0.0 ms
Počet krokov: 8
```

Heuristika 1

```
Počiatočné rozloženie:
[6, 1, 8, 3, 5]
[0, 2, 7, 4, 9]
Koncové rozloženie:
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 0]
Čas hľadania: 0.0 ms
Počet krokov: 8
```

Heuristika 2

Test 7

```
Počiatočné rozloženie:
[6, 8, 7, 5, 9]
[2, 1, 4, 3, 0]
Koncové rozloženie:
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 0]
Čas hľadania: 10348.31 ms
Počet krokov: 14486
```

Heuristika 1

```
Počiatočné rozloženie:
[6, 8, 7, 5, 9]
[2, 1, 4, 3, 0]
Koncové rozloženie:
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 0]
Čas hľadania: 1.011 ms
Počet krokov: 16
```

Heuristika 2

Test 8

```
Počiatočné rozloženie:
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]
Koncové rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Čas hľadania: 7.008 ms
Počet krokov: 312
```

Heuristika 1

```
Počiatočné rozloženie:
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]
Koncové rozloženie:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]
Čas hľadania: 77.452 ms
Počet krokov: 982
```

Heuristika 2

Pri veľmi jednoduchých problémoch s minimálnym riešením v jednotkách krokov je vo väčšine prípadov rozdiel medzi heuristikou 1 a 2 minimálny alebo žiadny v počte krokov aj v čase vykonania funkcie, ako v prípade testu 5. Toto tvrdenie však nie je platné pre všetky rozloženia, ako je možné vidieť v teste číslo 6, kde pri heuristike 1 je riešenie vykonané v oveľa dlhšom čase a s oveľa väčším počtom krokov než pri riešení s heuristikou 2.

Z testov sa nedá jednoznačne určiť riešenie s ktorou heuristikou je lepšie a nájdená cesta z východiskového uzla do cieľového je lacnejšia a časovo efektívnejšia. Pri problémoch, kde je minimálny počet krokov na vyriešenie niekoľko desiatok krokov je s využitím lačného vyhľadávania a zvolených heuristík potrebné vyriešiť veľké množstvo uzlov v tisíckach až desať tisíckach, čo má za následok veľkú pamäťovú a časovú náročnosť, nájdenie cieľového stavu môže trvať aj viac ako 10 minút.

V teste číslo 4 a 5 je východiskové a cieľové rozloženie presne opačné, no kroky a ich počet sú rozdielne.

5. Zhodnotenie

Riešenie je funkčné a pre rozloženia s minimálnym počtom ťahov na vyriešenie v jednotkách a pre väčšinu rozložení v rozmere 3x3 a nižšom realizovateľné v desiatkach milisekúnd. Pri väčších rozmeroch a rozloženiach s minimálnym počtom krokov na vyriešenie v desiatkach je počet prehľadaných uzlov veľmi vysoký. Všetky preskúmané uzly sa ukladajú do poľa, aby sa rozloženia hlavolamu neopakovali a tak sa nedostali do nekonečného cyklu, v ktorom by sa algoritmus opakoval donekonečna a nedokázal sa z neho dostať. Z toho dôvodu sa väčší počet preskúmaných uzlov rovná vyššiemu času na analýzu jedného uzla a teda aj nárast času vykonávania programu a zároveň sa zvyšuje aj využitie pamäte.

Riešenie je veľmi ľahko rozšíriteľné s využitím rôznych heuristík, prípadne ich kombináciami. Komplexnejšie heuristiky, ktoré by dokázali nájsť skutočne najlepší krok z aktuálneho rozloženia k cieľovému by mohli pomôcť zredukovať počet skúmaných uzlov a tým zrýchliť riešenie.

Riešenie je realizované v programovacom jazyku Python, no nie je závislé na programovacom prostredí a veľmi ľahko by sa dalo prepísať do iného programovacieho jazyka, či už objektovo alebo procedurálne orientovaného. Riešenie v jazyku Python môže byť o niečo pomalšie ako riešenie na rovnakom princípe v jazykoch s nižšou úrovňou, napríklad C alebo C++.