SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

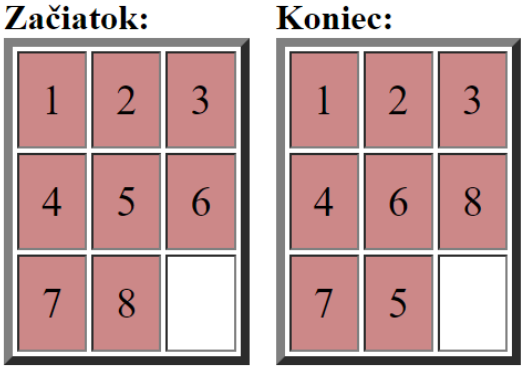
Zadanie 2 - Prehľadávanie stavového priestoru

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Riešený problém – 2. c)

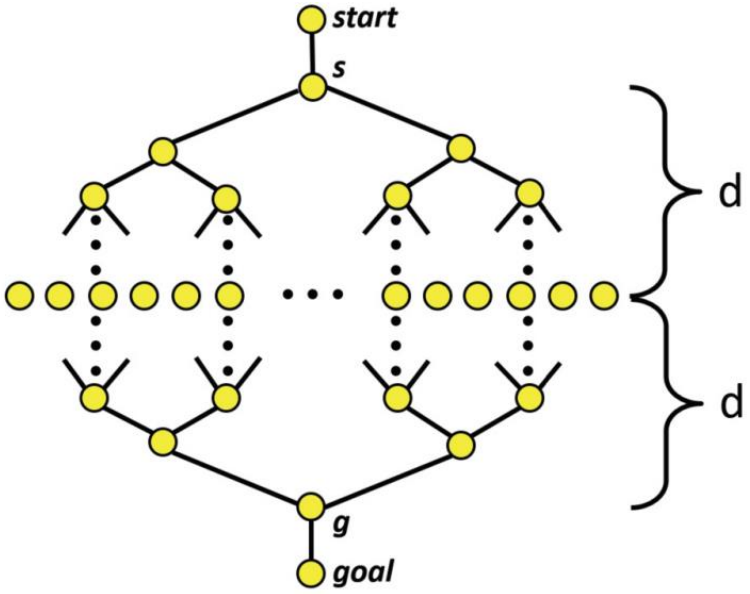
Problém sa týka riešenia (m*n)-hlavolamu, ktorý sa skladá z (m*n) očíslovaných políčok reprezentovaných číslami od 1 po (m*n)-1 a jedného prázdneho políčka, ktoré je reprezentované číslom 0. Jednotlivé políčka je možné presúvať **HORE**, **DOLE**, **VĽAVO** a **VPRAVO**, a to len vtedy, pokiaľ sa v danom smere posunu nachádza prázdne políčko. Vždy je zadané štartovacie a cieľové rozloženie (očíslovanie) políčok, pričom je potrebné nájsť takú postupnosť krokov, pomocou ktorých sa dostaneme zo štartovacej pozície do cieľovej.

Napríklad:



Opis riešenia

Úlohou je použiť algoritmus obojsmerného hľadania. Ten sa realizuje pomocou "dvojitého" prehľadávanie do šírky, kde pri prvom prehľadávaní sa hľadá cieľové rozloženie políčok od štartovacieho rozloženia a v druhom prehľadávaní sa zase hľadá štartovacie rozloženie políčok od cieľového, čiže opačne od konca.



Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Pomocné funkcie nebudem opisovať tie sú dostatočne okomentované v programe, opíšem len tie najhlavnejšie. Hlavná funkcia **find_solution()** pomocou, ktorej sa môže spustiť aj BFS algoritmus prepísaním atribútu bfs na True. Najprv overí či je možne riešiť zadaný vstup ak áno vypíšu sa zadané pozície.

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Reprezentácia údajov a použitý algoritmus

Samotný uzol je reprezentovaný pomocou triedy ***Node,*** v ktorej sa nachádzajú atribúty:

- **state** (Reprezentuje stav políčok a je 1D typu ***touple***)
- **parent** (Ukazovateľ na rodičovský uzol)
- **depth** (HĺBka daného uzla)
- **move** (Vykonaná operácia)
- **nothing** (Pozícia prázdneho políčka)
- **map** (stav ako string)

```
class Node:
    def __init__(self, state, nothing, parent = None, move = 0, depth = 0):
        self.state = state  #dostane 1D typ tuple repreyentujuci stav hru
        self.parent = parent    #referencia na rodica
        self.depth = depth  #hlbka uzla
        self.move = move    #vykonana operacia
        self.nothing = nothing  #pozicia prazdneho policka
            #prevedie tuple na string
        if self.state:
            self.map =  ''.join(str(c) for c in self.state)
```

Algoritmus:

1) Vytvor 2 počiatočné uzly a každý z nich umiestni do zásobníka pre vytvorené a zatiaľ nespracované uzly pre konkrétny strom riešenia
2) Ak sa v zásobníku nenachádza žiadny ďalší uzol na spracovanie, skonči s neúspechom – riešenie neexistuje
3) Vyber nasledujúci uzol z oboch zásobníkov a označ ich ako aktuálne
4) Vytvor nasledovníkov aktuálnych uzlov a zaraď ich medzi spracované uzly
5) Vytrieď nasledovníkov a ulož ich medzi vytvorené a zatiaľ nespracované
6) V aktuálnej hĺbke prvého stromu skontroluj pre každý jeho uzol vzájomnú podobnosť s každým uzlom druhého stromu v rovnakej hĺbke. Pokiaľ sa niektoré 2 uzly rovnajú – našiel si riešenie (prepojenie)
7) Choď na krok **2.**

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Spôsob testovania

Možnosti rozšírenia, prípadné optimalizácie, výhody a nevýhody konkrétnej implementácie (aj či sú závislé alebo nezávislé na programovacom prostredí)

Rôzne príklady vstupov a grafy

Časová a pamäťová zložitosť

Zhodnotenie riešenia

Porovnanie vlastností použitých metód pre rôznu dĺžku riešenia

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia