Faculty of Informatics and Information Technologies STU, Bratislava Ilkovičova 2, 842 16 Bratislava 4

Zadanie 1 - 8 Hlavolam

Umelá inteligencia

Frederik Duvač
3. semester
7.10. 2023

Obsah

1. Teoretické východiská

1.1. Technické pozadie

Hardvér, na ktorom bol skompilovaný zdrojový kód a testovaný program, je notebook Lenovo Legion S7, 12. generácia Intel® Core™ i5-12500H, 16 GB RAM. Celý projekt je implementovaný v programovacom jazyku Python 3.

1.2. Zadanie

Úlohou je vyriešiť 8 hlavolam, ktorý sa skladá z 8 políčok, každé so svojou číselnou hodnotou a jedno prázdne políčko. Cieľom programu je, nájsť a vypísať postupnosť krokov, s implementáciou algoritmu obojsmerného vyhľadávania, medzi počiatočným a cieľovým stavom, ktoré poskytuje používateľ v nastaveniach programu, za pomoci jednoduchých ťahov - HORE, DOLE, VĽAVO, VPRAVO susedných políčok s prázdnym miestom.

Príklad vstupu a výstupu hlavolamu:

START:	END:
7 ■ 8	1 2 3
3 2 1	4 5 6
6 5 4	7 8 ■

1.3. Algoritmus obojsmerného hľadania

Obojsmerné vyhľadávanie umožňuje rýchlejšie nájdenie riešenia tým, že sa hľadá z počiatočného a zároveň cieľového stavu, vďaka tomu sa znižuje aj počet stavov, ktoré musí prehľadávať v porovnaní s jednosmerným vyhľadávaním.

Kroky algoritmu sa dajú opísať nasledovne:

1. **Inicializácia** - V počiatočnom kroku je potrebné inicializovať začiatočný stav hry a koncový stav hry, ktorý chceme dosiahnuť. Taktiež je potrebné definovať množiny pre nespracované uzly a spracované uzly, každé 2-krát pre obe strany odpredu aj odzadu.

2. **Kontrola prieniku** - Na úvod je potrebné skontrolovať začiatočný stav s cieľovým ci sa už rovnajú, ak áno hlavolam je už vyriešený a program skončí, inak sa začína algoritmus obojstranného hladania, kde je v každej iterácií potrebné skontrolovať aktuálne stavy odpredu aj odzadu riešenia, ak sa rovnajú začína sa rekonštrukcia riešenia.

3. **Expandovanie stavov** - Vytvoria sa nasledovníci, tak že sa vykonajú všetky povolené pohyby z daného stavu aktuálneho uzla z jednej strany, ktorý sa porovnavaju so zoznamom neopracovaných uzlov z druhej strany riešenia. Ak sa riešenie našlo začína sa Rekonštrukcia riešenia. V opačnom prípade sa uzol z konca uzavrie a vytvoria sa nasledovníci od konca, ktorý sa rovnako porovnávajú s nespracovanými uzlami od štartu riešenia.

4. **Opakovanie krokov** - Opakujeme kroky 2 - 3 až kým sa nenájde riešenie alebo nezistí, že riešenie neexistuje (keď sú oba nespracované zoznamy prázdne a nedošlo k prieniku alebo sa prekročil stanovený počet iteráci).
5. **Rekonštrukcia riešenia** - Ak sa našla zhoda medzi niektorým uzlom od začiatku a niektorým od konca, je potrebné vypísať riešenie. Je potrebné dať pozor, že v riešení je nutné jeden zo zhodných krokov vynechat, keďže sú tieto dva stavy zhodné len sa k nim prišlo z iných ťahov. Taktiež riešenie zo štartovej časti sa musí vypísať v opačnom poradí a riešenie z koncovej časti je potrebné vypísať v rovnakom poradí ale opačnými krokmi, aby sme zachovali konzistentnosť výpisu krokov riešenia.

2. Implementácia riešenia

2.1. Reprezentácia údajov

Celý problém 8-hlavolamu je reprezentovaný ako samostatná trieda `Game`. Trieda obsahuje metódy pohybu `up()`, `down()`, `left()`, `right()`, metódy na výpis stavu hry, nájdenie prázdneho miesta a nájdenie možných pohybov.

Každá postupnosť krokov je reprezentovaná uzlom, ktorý obsahuje list potomkov, rodiča a pohyb, ktorý bol vykonaný pre prechod do stavu, ktorý je taktiež uložený v pohybe uzla. Stav je reprezentovaný dvojrozmerným polom napr. `[[7, -1, 8], [3, 2, 1], [6, 5, 4]]`.

```
class Move:
    def __init__(self, value, game_board_state):
        self.value = value
        self.game_board_state = game_board_state


class Node:
    def __init__(self, previous, children: list, move: Move):
        self.previous: Node = previous
        self.children: list[Node] = children
        self.move: Move = move
```

2.2. Pohyby

Pri vytváraní možných stavov do ktorých sa dokážeme dostať z jedného stavu sa volá metóda `get_possible_moves()` nad triedou `Game`, ktorá vráti všetky možné stavy okrem tých, ktoré by posunom vyšli mimo hracej dosky. Ukážka metódy možného posunu políčka hore.

```
def up(self, game_field):
    void = self.find_void(game_field)
    x = void[0]
    y = void[1]

    if x == M - 1:
        return game_field
    else:
        game_field[x][y] = game_field[x + 1][y]
        game_field[x + 1][y] = VOID
        return game_field
```

2.3. Algoritmus

Hlavný algoritmus je implementovaný v main programe a popis krokov je podrobne vysvetlený v kapitole 1.3.

2.4. Rekonštrukcia riešenia

Výpis riešenia je realizovaný pomocou funkcie `show_solution()`. Funkcia prijíma na vstupe aktuálny uzol od štartu a aktuálny uzol od konca riešenia, toto sú naše zhodné uzly. Traverzujeme jednoduchým cyklom od aktuálneho uzla po počiatočný a riešenie vypíšeme reverzne z dôvodu, že jednotlivé kroky riešenia začínajú od počiatočného stavu.

Druhá časť funguje podobne, ale je potrebné odstrániť aktuálny koncový uzol, pretože nechceme vypísať duplicitne tento stav, tiež je potrebné si pamätať pohyb do stavu ob iteráciu cyklu z dovodu udržania konzistentnosti výpisu. Namiesto opačného výpisu ako v prvej časti sa musia vypísať opačné pohyby, ktorými sme sa dostali z jedného stavu do nasledovného.

Nakoniec táto funkcia vypíše štatistické údaje o hlavnom algoritme.

3. Testovanie

Našu implementáciu 8-hlavolam solvera sme dôkladne testovali s cieľom overiť jej presnosť a spoľahlivosť. Testovanie bolo zamerané na porovnávanie výsledkov našej aplikácie s výstupmi z iných dostupných online solverov pre 8-hlavolam.

Testovanie malo nasledovnú postupnosť krokov:

Získanie referenčných výstupov - Na začiatku sme získali referenčné výstupy z niekoľkých online 8-hlavolam solverov. Tieto referenčné výstupy slúžili ako meradlo správnosti pre našu implementáciu.

Testovacie scenáre - Vybrali sme rôzne testovacie scenáre 8-hlavolamu s rôznymi počiatočnými konfiguráciami. Scenáre zahŕňali jednoduché aj komplexné puzzle, aby sme mohli testovať rôzne úrovne ťažkosti.

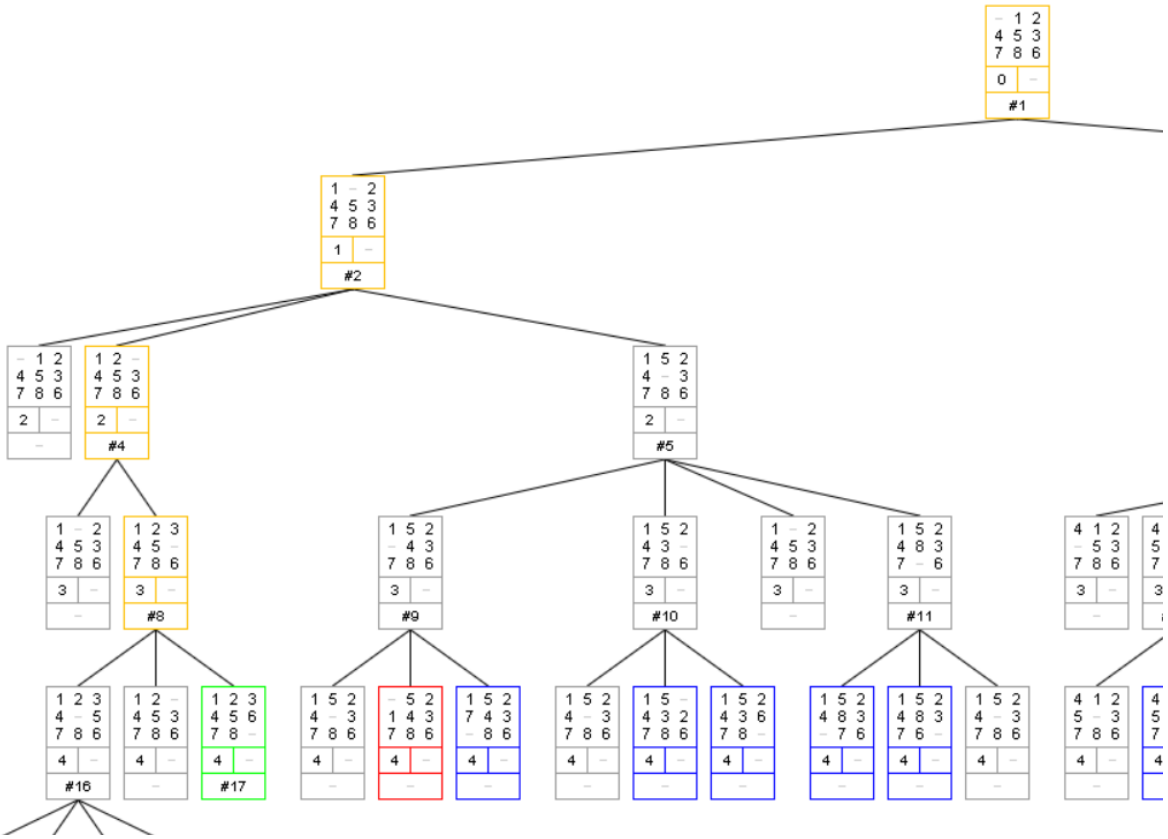
Porovnávanie výsledkov - Spustili sme našu aplikáciu pre každý testovací scenár a získali sme postupnosti krokov na dosiahnutie cieľa. Následne sme tieto postupnosti krokov porovnali s referenčnými výstupmi z online solverov.

Zhodnotenie - Výsledky porovnania ukázali, že naša implementácia poskytovala rovnaké postupnosti krokov ako referenčné online solvery. Toto pozorovanie potvrdilo správnosť a spoľahlivosť našej aplikácie.

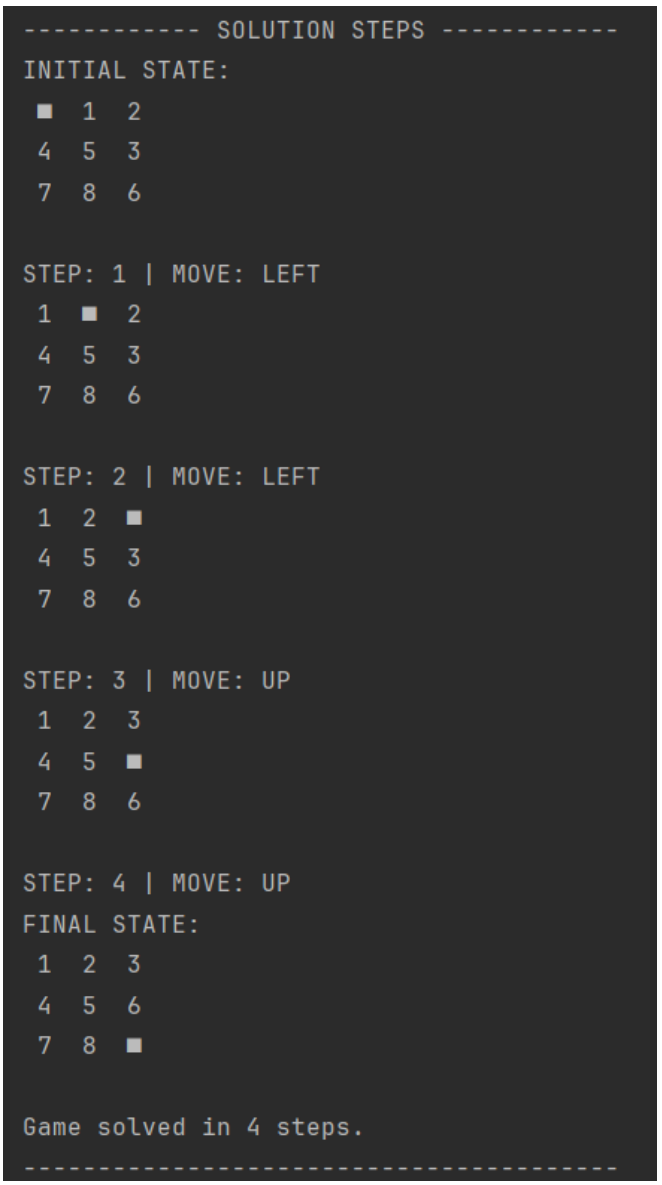
Overenie Efektivity - Okrem porovnania výstupov sme monitorovali aj rýchlosť a výkon našej aplikácie v porovnaní s online solvermi. Tým sme sa presvedčili o efektivite nášho riešenia.

Chybové Stavy - Testovali sme aj správanie nášho solvera pri neriešiteľných vstupoch a overili sme, že program správne ošetruje chybové stavy.

Príklad možného testovania:



Obr. 1. Výsledok online solvera https://tristanpenman.com/demos/n-puzzle/



Obr. 2. Výsledok nášho programu

4. Zhodnotenie

Dosiahnuté výsledky v tejto implementácii 8-hlavolam solvera boli úspešne skontrolované a overené. Výsledky našich riešení boli porovnané s výsledkami online 8-hlavolam solverov a postupnosti krokov, ktoré sme získali, sa zhodovali s tými, ktoré boli prezentované online.

Toto overenie potvrdzuje presnosť a úspešnosť nášho programu pri hľadaní riešení pre rôzne vstupné a výstupné konfigurácie 8-hlavolamu. A taktiež, že implementácia algoritmu bola správne prevedená.

Celkovo sme spokojní s dosiahnutými výsledkami a veríme, že naša implementácia môže byť užitočným nástrojom pre riešenie problémov 8-hlavolamu.

5. Používateľská príručka

Pre nastavenia programu bol zvolený spôsob bez výrazného používateľského rozhrania, a preto pre správne spustenie a výsledok programu je potrebné skontrolovať súbor constants.py.

V tomto súbore sa nachádzajú predpripravené možné vstupy a výstupy hlavne pre 8-hlavolam čiže rozmer 3x3. Ak si prajeme iný DEFAULT_INPUT prípadne DEFAULT_OUTPUT je potrebné zakomentovať a odkomentovať príslušné konfigurácie. V prípade iného vlastného DEFAULT_INPUT alebo DEFAULT_OUTPUT nám pomôže nasledujúca šablóna, v ktorej je potrebné prázdne pole reprezentovať hodnotou -1 `DEFAULT_INPUT/OUTPUT=[[,,],[,,],[,,]]`.

Program je navrhnutý tak, aby fungoval aj pre 15-hlavolam, čiže rozmer 4x4, architektúra programu by mala umožniť aj väčšie rozmery, každopádne na väčších rozmeroch program testovaný nebol.

V prípade záujmu o 15-hlavolam je potrebné zakomentovať velkosti M=3 a N=3 pre riadky a stĺpce a odkomentovať M=4 a N=4. Taktiež je potrebné zakomentovať DEFAULT_INPUT a DEFAULT_OUTPUT pre 8-hlavolam a odkomentovať DEFAULT_INPUT a DEFAULT_OUTPUT pre 15-hlavolam.