

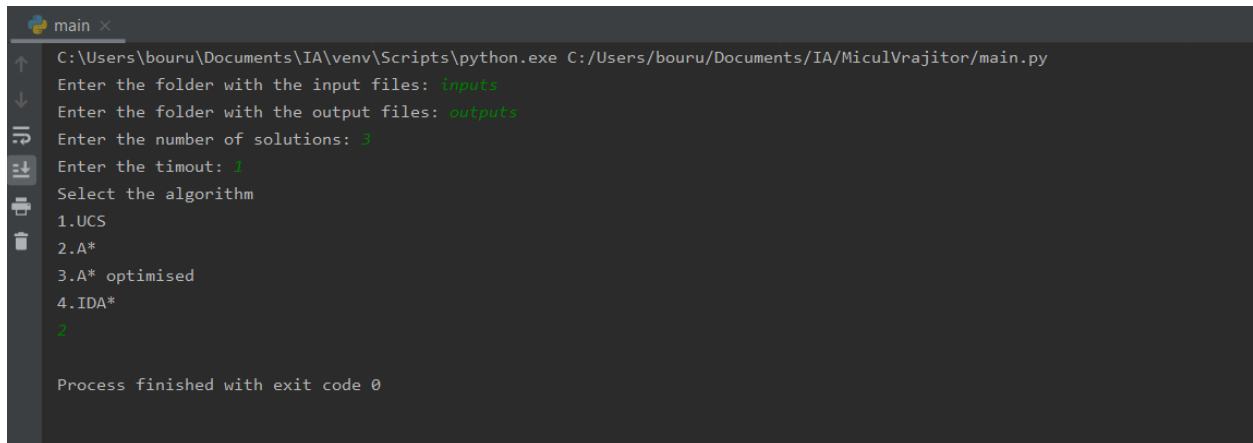
Tema1

-Micul Vrajitor-

1. Rularea proiectului

Programul se ruleaza avand ca entry point "main.py". Dupa, va trebui sa se introduca in terminal datele cerute: folderul cu fisierele de input, folderul unde se vor crea fisierele de output, numarul de solutii pe care sa le genereze un algoritm, timpul dupa care se va lua timeout si algoritmul cu care se vrea a se rula programul.

Exemplu:



```
main x
C:\Users\bouru\Documents\IA\venv\Scripts\python.exe C:/Users/bouru/Documents/IA/MiculVrajitor/main.py
Enter the folder with the input files: inputs
Enter the folder with the output files: outputs
Enter the number of solutions: 3
Enter the timeout: 1
Select the algorithm
1.UCS
2.A*
3.A* optimised
4.IDA*
1

Process finished with exit code 0
```

2. Memorarea unei stari

In memorie, o stare este reprezentata sub forma unei liste, astfel:

```
[
    pozitia_curenta_x,
    pozitia_curenta_y,
    culoarea_cizmelor_din_picioare,
    numarul_de_purtari_ale_cizmelor_din_picioare,
    culoarea_cizmelor_din_rucsac,
    numarul_de_purtari_ale_cizmelor_din_rucsac,
    daca_are_piatra,
    mesaj_de_afisat
```

]

3. Euristicile folosite

- a. Euristica banala: se bazeaza pe distanta Manhattan. Aceasta calculeaza cel mai scurt drum de la pozitia curenta la pozitia de start. Cel mai eficient drum pe care il poate urma vrajitorul este cel mai scurt drum de la punctul de start la cel final (exista posibilitatea ca vrajitorul sa ocoleasca pentru a lua perechile de cizme necesare), astfel distanta Manhattan nu va supraestima valoarea h , euristica este deci una admisibila.
- b. Euristica admisibila 1: se bazeaza pe distanta Manhattan. Aceasta calculeaza cel mai scurt drum de la pozitia curenta la pozitia de start. Cel mai eficient drum pe care il poate urma vrajitorul este cel mai scurt drum de la punctul de start la cel final (exista posibilitatea ca vrajitorul sa ocoleasca pentru a lua perechile de cizme necesare), astfel distanta Manhattan nu va supraestima valoarea h , euristica este deci una admisibila.
- c. Euristica admisibila 2: aceasta euristica calculeaza 2 vectori: unul de la start la piatra, altul de la pozitia curenta la start. Daca acesti 2 vectori sunt aliniati, inseamna ca produsul (cross-product) va fi mai mic si algoritmul va preferenta astfel de noduri in procesul de expansiune. Acel 0.001 este ales astfel incat sa fie mai mic decat (costul pentru a avansa un pas) / (numarul maxim de pasi pe care te astepti sa ii faci)
- d. Euristica inadmisibila: pentru aceasta, luam un numar foarte mare si scadem euristici bune. Astfel costul estimat va fi mult mai mare decat cel real

4. Validari si optimizari

In momentul citirii, datele sunt supuse unei serii de validari pentru a vedea daca acestea iau valorile corespunzatoare.

Pentru aceasta problema, nu se poate determina doar din configuratia unei stari daca dupa ce va fi expandata poate sau nu sa ajunga la final. Pentru a vedea aceasta, trebuie sa iteram prin problema (nu avem de unde sti doar dintr-un punct daca vom gasi pe parcurs cizme pentru a ne putea continua drumul).

5. Tabele cu valori

Valorile din fiecare casuta sunt in aceasta ordine: timpul de executie, lungimea drumului, numarul de noduri generate, numarul maxim de noduri existente in memorie la un moment dat.

- a. UCS

Input 3	0.000999, 12, 95, 35
Input 4	Timeout!

b. A*

	Banala	Admisibila 1	Admisibila 2	Inadmisibila
Input 3	0.001, 12, 38, 9	0.00096, 12, 34, 13	0.0019, 12, 87, 32	0.002, 131, 49
Input 4	Timeout	Timeout	Timeout	Timeout

c. A* optimizat

	Banala	Admisibila 1	Admisibila 2	Inadmisibila
Input 3	0.001, 12, 31, 29	0.001, 12, 29, 27	0.001, 12, 55, 43	0.001, 12, 67, 52
Input 4	0.003, 20, 194, 110	0.004, 20, 170, 100	0.005, 212, 122	0.003, 12, 237, 130

d. IDA*

	Banala	Admisibila 1	Admisibila 2	Inadmisibila
Input 3	0.0039, 12, 36, 26	0.0029, 12, 36, 26	0.002, 12, 36, 26	0.003, 12, 36, 26
Input 4	Timeout	Timeout	Timeout	Timeout

Dupa cum se poate observa, algoritmul A* optimizat este unul foarte bun din punct de vedere a timpului de executie, fiind singurul care a putut sa ruleze inputul 4. Pe de alta parte, IDA* este eficient din punct de vedere al memoriei, fiind bazat pe functionalitatea DFS.

Bouruc Petru-Liviu

Grupa 234

Pe de alta parte, euristicele alese ofera solutii optime, doar ca, atunci cand se ruleaza dand ca input un NSOL mai crescut, euristica inadmisibila va genera solutii nu neaparat in ordine crescatoare dupa cost.