Inteligență Artificială

-Documentație Tema 1-

Inundație

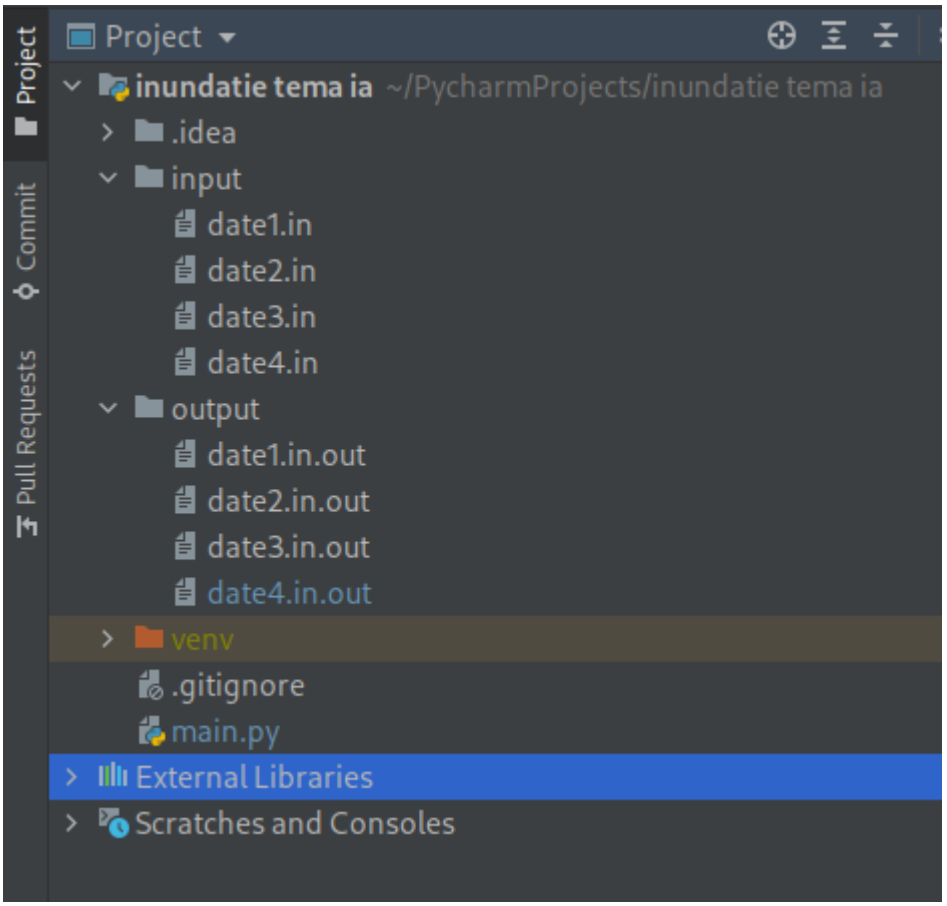
Student: Staicu Octavian-Florin

Grupa 231

Inundație

Introducere

Structura fișierelor

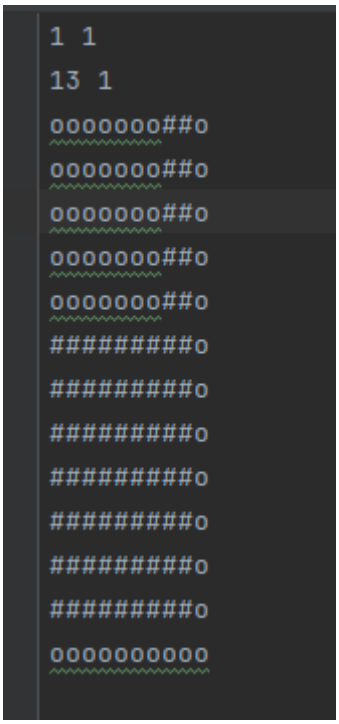


Exemplu de input:

```
cale folder fișier intrare (./folder/): ./input/
cale folder fișier ieșire (./folder/): ./output/
Număr soluții căutate: 5
Timeout: 120
```

Euristici

1. ***Banală - pentru fiecare succesor al unui nod, euristica banală va întoarce doar valoarea de construcție a acelui succesor (`current_state.val`)***
2. ***`"urmatorul_pas"` - Această euristică se va uita la succesorii stării curente. Fiecărui succesor i se va lua costul de construcție, păstrându-se cel mai mic. Excepția se întâmplă dacă sunt într-o stare finală, caz în care se va returna valoarea stării curente. Întrucât pentru a ajunge într-o stare finală trebuie să eliminăm cel puțin un obstacol, costul până la canal va fi cel puțin egal cu spargerea celui mai mic obstacol.***
3. ***`"peste_2_pasi"` - Față de euristica precedentă, acum se vor genera succesorii fiecărui copil, după aceeași logică. Dacă unul dintre copii va fi stare finală, se va întoarce acea valoare și nu se vor calcula nepoții din ea. Deși aceasta ar trebui să ne duc pe un drum mai ok, având în vedere că se uită peste 2 pași, în practică se comportă destul de prost, consumând atât de mult timp pentru precalcularea nepoților încât algoritmii ies dintr-un timp decent (>1 min pentru exemple mici). Această euristică este corectă conform aceluiași argument ca la euristica precedentă.***
4. ***`"distanta_manhattan_neadmisibila"` - Această euristică nu este ok. Pentru un exemplu în care există un obstacol ușor de îndepărtat, dar foarte departe de canal, și altele cu un cost mai mare, dar care sunt aproape, această euristică va alege greșit. Exemplu:***



Euristica manhattan nu va elimina obstacolele din colțul dreapta sus. În loc de un cost de 5, va obține un cost de 9.

```
AStar Optim (euristica banala):
Nr noduri calculate dupa aceasta solutie: 72
Nr de noduri aflate in memorie dupa generarea acestei sol: 15615
1) Configuratie Initiala
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
Lungime intermediara drum: 1
Cost actual: 0
Timp gasire: 0.0s
..........
2) Eliminam obstacolul de pe linia 1, coloana 8
['*', '*', '*', '*', '*', '*', '*', '*', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
Lungime intermediara drum: 2
Cost actual: 3
Timp gasire: 0.00042s
..........
3) Eliminam obstacolul de pe linia 1, coloana 9
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*']
Lungime intermediara drum: 3
Cost actual: 5
Timp gasire: 0.00334s
```

```
AStar Optim (euristica distanta_manhattan_neadmisibila):
Nr noduri calculate dupa aceasta solutie: 338
Nr de noduri aflate in memorie dupa generarea acestei sol: 15615
1) Configuratie Initiala
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
Lungime intermediara drum: 1
Cost actual: 0
Timp gasire: 0.0s
..........
2) Eliminam obstacolul de pe linia 4, coloana 8
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '*', '#', 'o']
['*', '*', '*', '*', '*', '*', '*', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['#', '#', '#', '#', '#', '#', '#', '#', '#', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
Lungime intermediara drum: 2
Cost actual: 5
Timp gasire: 0.0011s
..........
3) Eliminam obstacolul de pe linia 4, coloana 9
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*', '*', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['#', '#', '#', '#', '#', '#', '#', '#', '#', '*']
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*']
Lungime intermediara drum: 3
Cost actual: 9
Timp gasire: 0.03121s
```

Comparare algoritmi

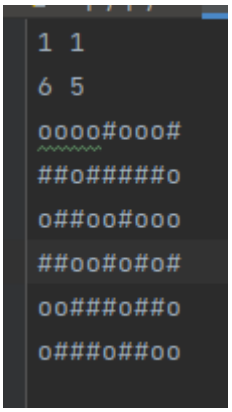
Algoritm\ Euristica	*Banală*	*urmatorul_pas*	*peste_2_pasi*	*dist_manhattan*
UCS	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 72*	*-*	*-*	*-*
*A**	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 72*	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 24*	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 24*	*Lungime drum: 3 Cost: 9 Timp gasire: 0.03121s Nr noduri calc: 1125*
A Optim*	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 72*	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 24*	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 24*	*Lungime drum: 3 Cost: 9 Timp gasire: 0.03121s Nr noduri calc: 338*
*IDA**	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 72*	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 36*	*Lungime drum: 3 Cost: 5 Timp gasire: 0.00334s Nr noduri calc: 24*	*Lungime drum: 3 Cost: 9 Timp gasire: 0.03121s Nr noduri calc: 2954*

Observații:

Se observă că pe acest exemplu cea mai bună alegere este A* optimizat, folosind euristica „următorul pas”. Merită menționat că codul păstrează nodurile precalculate anterior de alți algoritmi, deci este foarte posibil ca „peste_2_pași” să se comporte mai prost când rulează „la rece”.

IDA* calculează foarte mult în plus, datorită modului în care este gândit (repetă calculări)

Exemplul 2



<u>Comparare algoritmi</u>

Algoritm\ Euristica	*Banală*	*urmatorul_pas*	*peste_2_pasi*	*dist_manhattan*
UCS	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 14112*	-	-	-
*A**	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 14112*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 5113*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.13116s Nr noduri calc: 1181*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 62*
A Optim*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 2040*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 1225*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 591*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.1313s Nr noduri calc: 62*
*IDA**	*Lungime drum: 4 Cost: 9 Timp gasire: 0.13116s Nr noduri calc: 17032*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.13116s Nr noduri calc: 6250*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.13116s Nr noduri calc: 1535*	*Lungime drum: 4 Cost: 9 Timp gasire: 0.13116s Nr noduri calc: 79*

Observații:

Se observă că pe acest exemplu cea mai bună alegere este A* optimizat. La o primă vedere, ar părea că euristica „peste_2_pași" ar fi cea mai bună alegere. Totuși, în datele din tabel „nr noducri calc" cuprinde doar nodurile alese de algoritm, nu și precalcularea euristicii. Acea precalculare, care generează toți nepoții, consumă extrem de mult timp și memorie în practică, luând beneficiile alegerii drumul din mai puține mișcări (chiar dacă face după precalculare mai puține mișcări, per total timpul de execuție e mult mai lent - a se observa faptul că la rulare, în memorie -per total- se sare la 348303 noduri). Merită menționat că codul păstrează nodurile precalculate anterior de alți algoritmi, deci este foarte posibil ca „peste_2_pași" să se comporte mai prost când rulează „la rece".

IDA* calculează foarte mult în plus, datorită modului în care este gândit (repetă calculări), făcându-l nerentabil, cu excepția cazului în care nu este disponibilă memorie.

Bibliografie

1. http://irinaciocan.ro/inteligenta_artificiala/lab12.php
2. http://irinaciocan.ro/inteligenta_artificiala/a_star.php
3. https://algorithmsinsight.wordpress.com/graph-theory-2/ida-star-algorithm-in-general/