Trușcă Luca-Marian
grupa 252

Documentație

Șoareci și pisici

Linia de comandă

Algoritm:

```
> python3 -m tema1.src.main --help
```

```
usage: main.py [-h] -i PATH -o PATH -n SOLUTIONS -t S

Rezolva problema 'Șoareci și pisici'.

optional arguments:
  -h, --help            show this help message and exit
  -i PATH, --input PATH
                        calea absoluta catre folderul de input
  -o PATH, --output PATH
                        calea absoluta catre folderul de output
  -n SOLUTIONS          numarul de solutii de calculat
  -t S, --timeout S     timpul de timeout per solutie, in secunde
```

Exemple:

```
> python3 -m tema1.src.main --input
/Users/luca1152/Desktop/IA/Labs/tema1/input --output
/Users/luca1152/Desktop/IA/Labs/tema1/output -n 5 --timeout 2.5
```

```
> python3 -m tema1.src.main -i /Users/luca1152/input -o
/Users/luca1152/output/ -n 3 -t 1
```

Vizualizare:

```
> python3 -m tema1.src.view --help
```

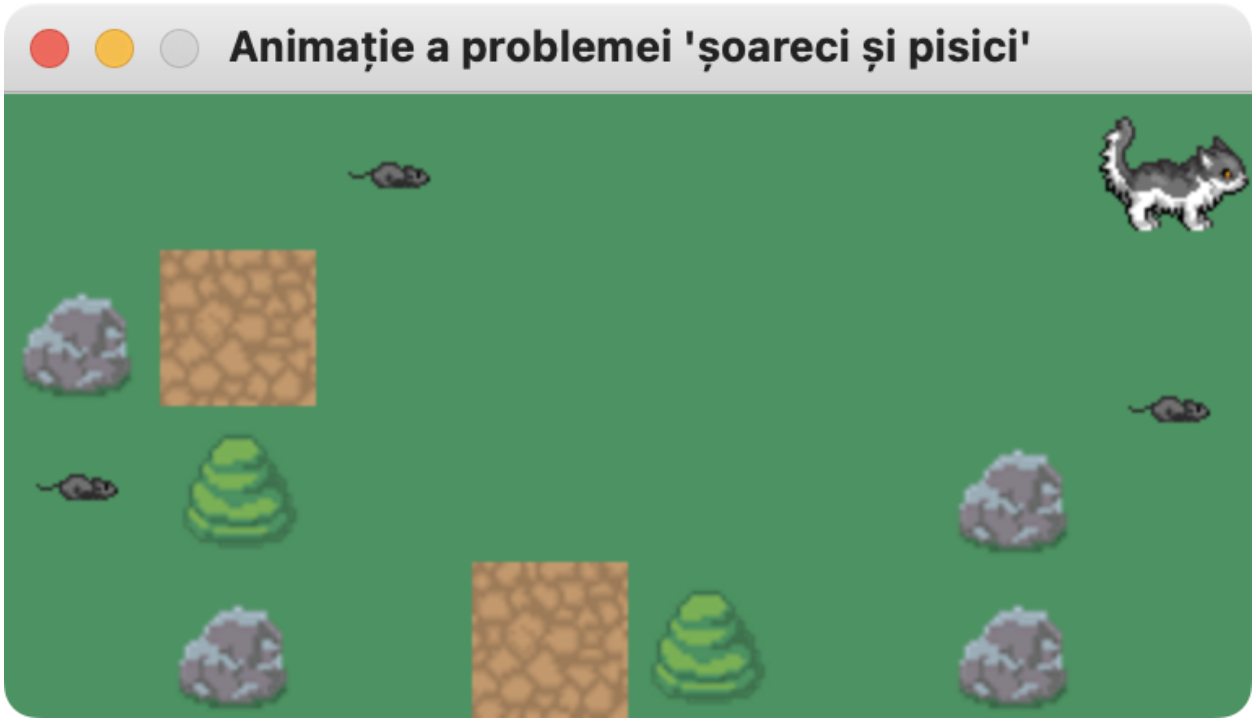
```
usage: view.py [-h] -i PATH

Vizualizeaza rezultatul problemei 'Șoareci și pisici'.
```

```
optional arguments:
  -h, --help            show this help message and exit
  -i PATH, --input PATH
                        calea absoluta catre fisierul de input
```

Exemplu:

```
python3 -m tema1.src.view -i
/Users/luca1152/Desktop/IA/Labs/tema1/output/d-a*-opt-admisibila2-1.o
ut
```



Euristici

Euristica banală:

- Estimăm că h-ul este 1
- Justificare: conform enunțului problemei, costul unei mutări este cel puțin 1; dacă estimăm întotdeauna că distanța către cel mai apropiat nod scop este 1, niciodată nu vom depăși valoarea reală

Euristica admisibilă #1:

- Estimăm că h-ul este suma primelor [număr de șoareci ce mai trebuie să iasă de

pe hartă] cele mai mici **distanțe Manhattan** de la șoareci la cea mai apropiată ieșire de aceștia
- Justificare: pentru a câștiga, k șoareci trebuie să iasă de pe hartă; în cel mai bun caz, vor ieși acei k șoareci ce sunt cei mai apropiați de o ieșire; astfel, nu vom supraestima niciodată distanța către cel mai apropiat nod scop

Euristica admisibilă #2:
- Estimăm că h-ul este suma primelor [număr de șoareci ce mai trebuie să iasă de pe hartă] cele mai mici **distanțe reale** de la șoareci la cea mai apropiată ieșire de aceștia
- Justificare: pentru a câștiga, k șoareci trebuie să iasă de pe hartă; în cel mai bun caz, vor ieși acei k șoareci ce sunt cei mai apropiați de o ieșire; astfel, nu vom supraestima niciodată distanța către cel mai apropiat nod scop

Euristica neadmisibilă:
- Estimăm că h-ul este suma distanțelor soarecilor față de cele mai apropiate pisici; ideea euristicii este să evităm pisicile, sperând să ajungem la un nod scop
- Justificare că e neadmisibiliă: exemplul c)

Optimizări:
- Se verifică fișierul de input pentru diverse erori. De exemplu, fișierul va fi detectat ca fiind invalid dacă numărul de celule de pe linii diferă, dacă sunt mai puțini șoareci decât numărul k, dacă indicii șoarecilor/pisicilor nu sunt în ordine crescătoare, etc.
- Se verifică la început, înainte de a expanda orice nod, dacă există vreo soluție pentru harta dată. Pentru asta, se verifică dacă pot ieși (cel puțin) k șoareci de pe hartă (dacă nu au cumva toate căile spre ieșiri blocate de obstacole).
- E utilizat un deque în loc de liste atunci când e nevoie să se elimine noduri de la început și să se adauge noduri la final
- Algoritmul de expandare a unui nod este un backtracking nerecursiv (implementat similar cu un BFS), pentru a evita stack overflow

Statistici

Fișierul de input c:

Algoritm	Lungime	Cost	Max noduri	Total noduri
BFS	3	7	289	344
DFS	ST OVFLW	ST OVFLW	ST OVFLW	ST OVFLW
DFI	3	7	4	420

Algoritm	Lungime	Cost	Max noduri	Total noduri
A*, banală	3	6	45	134
A*, admisibilă 1	3	6	37	73
A*, admisibilă 2	3	6	37	73
A*, neadmisibilă	5	8	33	93
A* opt, banală	3	6	64	134
A* opt, admisibilă 1	3	6	48	73
A* opt, admisibilă 2	3	6	48	73
A* opt, neadmisibilă	5	8	52	93
IDA*, banală	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
IDA*, admisibilă 1	3	6	5	50
IDA*, admisibilă 2	3	6	5	50
IDA*, neadmisibilă	5	10	6	42

Fișierul de input d:

Algoritm	Lungime	Cost	Max noduri	Total noduri
BFS	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
DFS	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
DFI	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
A*, banală	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
A*, admisibilă 1	6	16	871	2254
A*, admisibilă 2	6	16	727	1930
A*, neadmisibilă	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
A* opt, banală	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
A* opt, admisibilă 1	6	16	1043	2254

Trușcă Luca-Marian
grupa 252

Algoritm	Lungime	Cost	Max noduri	Total noduri
A* opt, admisibilă 2	6	16	872	1930
A* opt, neadmisibilă	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
IDA*, banală	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
IDA*, admisibilă 1	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
IDA*, admisibilă 2	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT
IDA*, neadmisibilă	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT

Concluzii:

- Algoritmul BFS este mai bun decat DFS (DFS-ul ajungand la STACK OVERFLOW foarte usor)
- O alternativa buna pentru BFS si DFS este algoritmul DFI. Deși necesită mai multe noduri în total, numărul maxim de noduri la un moment dat este foarte mic
- Euristica banală pentru A* ajunge la soluția corectă, dar destul de încet
- Cea de-a doua euristică admisibilă este mai bună decât prima
- Euristica neadmisibilă ne duce la un rezultat greșit
- Algoritmul IDA* ajunge mai rapid la TIMEOUT decât celelalte două variante de A*
- Algoritmul IDA* este mult mai eficient din punct de vedere al spațiului decât celelalte două variante de A*