Dima Vlad 233

Extratereștrii sunt printre noi

Apelare program

Programul se apelează cu următorii parametrii:

1. calea folder-ului unde se află fișierele de input
2. calea folder-ului în care se află fișierele de output (în cazul în care nu există, fișierele vor fi create automat)
3. numărul de soluții de căutat
4. timpul de timeout

Exemple de apel:

- main.py D:\Facultate\An_2_Sem_2\IA\Tema1\input_folder D:\Facultate\An_2_Sem_2\IA\Tema1\output_folder 2 20
- main.py input_folder output_folder 2 20

Euristicile folosite

banală – returnează 1 dacă nodul nu este scop, altfel 0.

```
if nrOameniRapiti < nrOameniDeRapit:
    return 1
return 0
```

admisibilă 1 – calculează distanța Manhattan până la cel mai apropiat om. Cum între cel mai apropiat om și navă nu au cum să existe alți oameni, costul de deplasare minim este atins când pe drumul respectiv nu există obstacole/clădiri. Înmulțind distanța până la nod cu număul de oameni rămași (care se modifică doar în momentul în care nava ajunge la om) și împărțind rezultatul la doi mă asigur că $h'(nod) \leq h(nod)$.

```
dist = min([abs(pozNouaNava[0] - om[0]) + abs(pozNouaNava[1] - om[1]) for om in
oameniRamasi])
return dist * len(oameniRamasi) // 2
```

admisibilă 2 – se calculează distanța Manhattan până la cel mai îndepărtat om pe care trebuie să-l răpească pentru a deveni scop. La acest calcul adăugăm și criteriul ca nava să fi răpit cât mai mulți oameni deja. Împărțind distanța calculată la 2, mă asigur că euristica rămâne admisibilă.

```
dist = sorted([abs(pozNouaNava[0] - om[0]) + abs(pozNouaNava[1] - om[1]) for om in
oameniRamasi])[:min(nrOameniDeRapit - nrOameniRapiti, len(oameniRamasi))]
try:
    return int(dist[-1] - 1 + len(oameniRamasi) / nrOameniDeRapit) * len(oameniRamasi)
// 2
except:    # nu mai sunt oameni de rapit
    return 0
```

neadmisibilă – calculează distanța până la cel mai îndepărtat nod și o înmulțește cu 2 (ca și cum nava ar parcurge drumul doar prin obstacole) .

```
try:
    dist = max([abs(pozNouaNava[0] - om[0]) + abs(pozNouaNava[1] - om[1]) for om in
oameniRamasi])
except:
    dist = 0
return dist * 2 * len(oameniRamasi)
```

Optimizări

- am folosit un obiect Queue la BF
- am folosit un PriorityQueue la A*
- pentru fiecare stare am memorat un tuplu pentru poziția curentă a navei și o listă de liste pentru pozițiile oamenilor rămași

Timpi algoritmi

Am rulat algoritmii cu parametrii NSOL = 3 și timeout = 5 minute

Fișier **input3.txt** (cu soluția mai lungă)

Nume algoritm	Timp total	Soluția 1			Soluția 2			Soluția 3		
		lungime	cost	noduri	lungime	cost	noduri	lungime	cost	noduri
BF	2.8304s	20	275	75364	20	275	78721	21	264	118633
DF	0.6891s	69	1007	7792	69	1007	19969	86	1194	26921
DFI	4.5378s	20	275	200654	20	275	208389	21	264	317152
A* euristica banală	294.65s	21	210	11577750	21	210	11577752	21	210	11577754
A* euristica admisibilă 1	-	-	-	-	-	-	-	-	-	-
A* euristica admisibilă 2	11.7645s	21	210	545888	21	210	545894	21	210	545900
A* euristică neadmisibilă	6.0012s	27	271	331480	27	271	331500	27	271	331532
A* optimizat euristica banală	1.0305s	21	210	9381	-	-	-	-	-	-
A* optimizat euristica admisibilă 1	0.9634s	21	210	8657	-	-	-	-	-	-
A* optimizat euristica admisibilă 2	0.2792s	21	210	4134						
A* optimizat euristica neadmisibilă	0.0309s	27	271	1085	-	-	-	-	-	-
IDA* euristica banală	-	-	-	-	-	-	-	-	-	-
IDA* euristica admisibilă 1	-	-	-	-	-	-	-	-	-	-
IDA* euristica admisibilă 2	114.76s	21	210	7643800	21	210	7643810	21	210	7643826
IDA* euristica neadmisibilă	36.136s	25	244	3089680	31	292	3089711	31	304	3089739

Fișier **input4.txt** (cu soluția mai scurtă)

Nume algoritm	Timp total	Soluția 1			Soluția 2			Soluția 3		
		lungime	cost	noduri	lungime	cost	noduri	lungime	cost	noduri
BF	0.2224s	12	55	16737	12	65	19520	12	65	20316
DF	0.0139s	55	248	110	81	373	180	89	411	197
DFI	0.2632s	12	55	30137	12	65	34933	12	65	36269
A* euristica banală	6.8686s	12	53	483683	12	53	483686	12	53	483689
A* euristica admisibilă 1	5.5429s	12	53	364878	12	53	364881	12	53	364884
A* euristica admisibilă 2	0.3959s	12	53	25280	12	53	25287	12	53	25294
A* euristică neadmisibilă	0.0129s	12	53	844	13	56	847	14	59	850
A* optimizat euristica banală	0.1725s	12	53	4067	-	-	-	-	-	-
A* optimizat euristica admisibilă 1	0.1346s	12	53	3436	-	-	-	-	-	-
A* optimizat euristica admisibilă 2	0.0339s	12	53	1466	-	-	-	-	-	-
A* optimizat euristica neadmisibilă	0.0039s	12	53	323	-	-	-	-	-	-
IDA* euristica banală	15.337s	12	53	2227279	12	53	2239491	12	53	2244391
IDA* euristica admisibilă 1	15.244s	12	53	1897027	12	53	1911535	12	53	1956605
IDA* euristica admisibilă 2	1.1676s	12	53	138092	12	53	138703	12	53	138962
IDA* euristica neadmisibilă	0.005s	21	84	255	21	84	259	23	92	271

Din tabele observăm că cel mai eficient algoritm care dă soluția cu cost minim este A* optimizat (cu euristica 2), iar cel mai ineficient este IDA* (cu euristica banală), care reconstruiește nodurile pentru fiecare iterație.
Cel mai rapid algoritm este DF-ul, care obține cea mai slabă soluție dintre toți algoritmii (cu lungimea drumului și costul foarte mari)