Homework Piano Mover

Leonardo Serilli (mat: 274426)

30/11/2020

1 - Implementation of Game Model module

Il labirinto è implementato dalla classe **Maze**, questa consiste di **9 attributi** (di cui 3 per stimare il costo associato allo stato attuale della tabella) e **12 metodi** (di cui tre 3 controlli relativi all'euristica e per stampare informazioni sull'istanza) :

```
# Attributi
self.father # puntatore all'oggetto padre (ovvero il labirinto il cui spostamento di un oggetto ha portato alla configurazione attuale)
self.board # matrice 10X10 contente il piano da muovere e i vari ostacoli
self.density # parametro indicante la percentuale di celle occupate della board
self.obstacles # contiene la coppia {id, coordinate} di ogni ostacolo nella tabella (compreso il piano, il quale ha id=0)
self.end_pos # contiene le coordinate dell'uscita
self.start_pos # contiene le coordinate iniziali del piano

# Attributi aggiuntivi necessari alla search strategy adottata
self.h # parametro della search strategy
self.g # parametro della search strategy
self.f # parametro della search strategy -> è la somma h+g; valore utilizzato come peso dello stato nell'euristica A* utilizzata
```

```
# Inizializzazione Labirinto
fill_w_obstacles() # riempie casualmente di ostacoli (di dimensione 1/2/3) la board in base al threshold definito dal parametro di densità
put_obstacles(id, dim) # posiziona casualmente un oggetto di dimensione 1 sulla board e lo espande tramite la funzione seguente
fill_2Neighborhood(id, x, y) # espande l'ostacolo riempiendo casualmente, con l'id dell'ostacolo, una cella libera e adiacente ad esso
check_neighborhood(x, y) # controla se il 4 vicinato di una cella è libero

# Spostamento di oggetti
mv(obstacle, direction) # sposta, se possibile, un oggetto nella direzione indicata tramite i seguenti metodi, e aggiorna l'attributo f
mv_obst_r(obstacle) # Right
mv_obst_l(obstacle) # Left
mv_obst_t(obstacle) # Top
mv_obst_b(obstacle) # Bottom

# Controlli e info
check_end() # controlla se il Piano è nella posizione finale (uscita)
eval_state() # computa il valore dell'attributo h
describe(option) # stampa la configurazione della board, e, se il parametro 'option' è "verbose", stampa anche una descrizione dello stato dell'oggeto
```

2 - Implementazione Heuristic module

Il modulo **Heuristic** è composto di due funzioni:

```
astar_search(initial_state) # partendo dallo stato iniziale del Labirinto applica l'euristica fino ad una convergenza
expand_horizon(horizon, index) # genera i figli dello stato visitato e li aggiunge all'orizzonte
```

Il modulo importa **datetime** e **psutil**: la prima libreria è utilizzata per controllare il tempo che impiega l'euristica per giungere a una soluzione, mentre la seconda controlla la **memoria (in bytes)** occupata dal processo durante l'esecuzione.

Inoltre è utilizzata la libreria **ctypes**, la quale è utilizzata per gestire l'attributo **padre** delle istanza del **Game model** tramite **puntatori**, questo fa si che la memoria occupata dal processo diminuisca drasticamente.

```
import ctypes
import psutil
from datetime import datetime
```

L'euristica implementata è l' A Start **A*** , e agisce nel seguente modo:

1. Aggiunge all'**Horizon** uno **stato radice**, ovvero la configurazione iniziale della board;
2. Itera i seguenti steps fino a che non viene aggiunto all'orizzonte lo **stato finale**, ovvero quello la cui board ha il Piano sull'uscita:
    1. Sceglie lo stato **s** nell'orizzonte con **minor peso associato**, tale peso è definito dalla **Search Strategy** descritta in seguito sulla base della configurazione della board ;
    2. Computa l'**Horizon** dello stato **s**: ovvero tutti gli stati figli della configurazione iniziale, quindi tutte le configurazioni risultanti dallo spostamento di un singolo oggetto, e in ognuna imposta lo stato **s come padre**;
    3. Aggiunge **s** alla lista degli stati visitati: **view**;
    4. Rimuove **s** dall'orizzonte **Horizon**;
3. L'algoritmo ritorna il **path della soluzione**, ovvero il path che va dallo stato radice fino allo stato finale (dallo stato finale si può risalire il Path fino allo stato radice tramite l'attributo padre)

3 - Search strategy

A ogni board è associato un costo, computato ogni volta che un oggetto viene spostato, il costo **f** è dato dalla somma di due variabili **h, g**:

- **h** equivale alla differenza tra la **distanza(piano, uscita)** e la **sommatoria** della **distanza(ostacolo, uscita) per ogni ostacolo**
- **g** è invece incrementata di **1** ogni volta che il Piano si avvicina all'uscita

Come detto precedentemente, l'euristica sceglie ogni volta lo stato con **peso minore** nell'orizzone, di conseguenza, questa Search Strategy fa si che gli stati vengano scelti in modo che il **piano si avvicini all'uscita** e che contemporaneamente gli **ostacoli si allontanino dall'uscità** (grazie ad **h**).
Inoltre a pari merito di **h**, viene scelto sempre lo **stato che avvicina di più il piano all'uscita** (grazie al contributo di **g**)

> Il contributo in **h** della "**sommatoria** della **distanza(ostacolo, uscita) per ogni ostacolo**" è stato introdotto poichè senza di esso, l'euristica convergeva troppo lentamente alla soluzione e occupava troppa memoria per una **denstità** di oggetti **maggiore di 0,4**.

4 - Esempi d'esecuzione

Le esecuzioni con diversa densità di celle occupate nella board ha portato ai risultati illustrati nella tabella.

Purtoppo l'**euristica A* con la search strategy utilizzata**, per una densià **superiore a 0.6** porta ad un **eccessivo utilizzo di tempo e memoria**

Densità	Lunghezza del Solution Path	Tempo impiegato	Memoria occupata
0.0	16	0.33451 **s**	13.758464 **Mb**
0.1	16	0.50193 **s**	14.06976 **Mb**
0.2	17	0.102613 **s**	17.38752 **Mb**
0.3	115	3.957681 **s**	143.384576 **Mb**
0.4	95	9.962194 **s**	282.95168 **Mb**
0.5	298	1.04 **m**	1.136381952 **Gb**
0.6	259	1.31 **m**	1.66690816 **Gb**

Nei paragrafi sottostanti, la **prima immagine** rappresenta la **configurazione iniziale** con le relative informazioni, mentre la **seconda** illustra la configurazione finale, con il numero di bytes utilizzati dal processo e il tempo impiegato per convergere a una soluzione.

> L'esecuzione dell'algoritmo per le densità 0.4, 0.5 e 0.6 sono mostrate nei rispettivi video nella cartella **Executions_videos**

Density 0.0

```
initial configuration

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[-1, -1, 0, 0, 0, 0, 0, 0, 0, 0]
[-1, -1, 0, 0, 0, 0, 0, 0, 0, 0]

 Density of obstacles:  0.0

 Obstacles:
{'id': -1, 'coords': [[8, 0], [8, 1], [9, 0], [9, 1]]}

Heuristic Params
h:  128 , g:  0 , f:  128
```

```
Execution...
[0, 0, 0, 0, 0, 0, 0, 0, -1, -1]
[0, 0, 0, 0, 0, 0, 0, 0, -1, -1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

occupied bytes:  13758464
executed in 0:00:00.033451

Lenght of solution path:  16
```

Density 0.1

```
initial configuration

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 2, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[-1, -1, 0, 0, 0, 0, 0, 0, 0, 0]
[-1, -1, 0, 0, 0, 0, 0, 0, 0, 0]

 Density of obstacles:  0.1

 Obstacles:
{'id': -1, 'coords': [[8, 0], [8, 1], [9, 0], [9, 1]]}
{'id': 1, 'coords': [[6, 0], [7, 0], [7, 1]]}
{'id': 2, 'coords': [[1, 7], [2, 7], [2, 6]]}

Heuristic Params
h:  26 , g:  0 , f:  26
```

```
Execution...
[0, 0, 0, 0, 0, 0, 0, 0, -1, -1]
[0, 0, 0, 0, 0, 0, 0, 2, -1, -1]
[0, 0, 0, 0, 0, 0, 2, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

occupied bytes:  14069760
executed in 0:00:00.050193

Lenght of solution path:  16
```

Density 0.2

```
initial configuration

[0, 0, 0, 0, 0, 0, 2, 0, 0, 0]
[0, 0, 0, 0, 0, 3, 2, 0, 0, 0]
[0, 0, 4, 4, 0, 3, 2, 0, 0, 0]
[7, 8, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[6, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 5, 5, 0, 0, 0, 0, 0]
[-1, -1, 1, 5, 0, 0, 0, 0, 0, 0]
[-1, -1, 1, 1, 0, 0, 0, 0, 0, 0]

 Density of obstacles:  0.2

 Obstacles:
{'id': -1, 'coords': [[8, 0], [8, 1], [9, 0], [9, 1]]}
{'id': 1, 'coords': [[9, 3], [9, 2], [8, 2]]}
{'id': 2, 'coords': [[0, 6], [1, 6], [2, 6]]}
{'id': 3, 'coords': [[1, 5], [2, 5]]}
{'id': 4, 'coords': [[2, 2], [2, 3]]}
{'id': 5, 'coords': [[8, 3], [7, 3], [7, 4]]}
{'id': 6, 'coords': [[5, 0]]}
{'id': 7, 'coords': [[3, 0]]}
{'id': 8, 'coords': [[3, 1]]}

Heuristic Params
h:  -341 , g:  0 , f:  -341
```

```
Execution...
[0, 0, 0, 0, 0, 0, 2, 0, -1, -1]
[0, 0, 0, 0, 0, 3, 2, 0, -1, -1]
[0, 0, 4, 4, 0, 3, 2, 0, 0, 0]
[7, 8, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[6, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 5, 5, 0, 0, 0, 0, 0]
[0, 1, 0, 5, 0, 0, 0, 0, 0, 0]
[0, 1, 1, 0, 0, 0, 0, 0, 0, 0]

occupied bytes:  17387520
executed in 0:00:00.102613

Lenght of solution path:  17
```

Density 0.3

```
initial configuration

[0, 0, 0, 0, 0, 0, 0, 0, 2, 0]
[0, 12, 3, 0, 0, 0, 0, 0, 8, 8]
[13, 12, 3, 6, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
[0, 0, 0, 10, 10, 0, 7, 7, 0, 1]
[0, 0, 0, 0, 10, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 4]
[0, 11, 0, 0, 0, 0, 0, 0, 0, 4]
[-1, -1, 0, 9, 9, 9, 0, 0, 0, 5]
[-1, -1, 0, 0, 0, 0, 0, 0, 5, 5]

 Density of obstacles:  0.3

 Obstacles:
{'id': -1, 'coords': [[8, 0], [8, 1], [9, 0], [9, 1]]}
{'id': 1, 'coords': [[3, 8], [3, 9], [4, 9]]}
{'id': 2, 'coords': [[0, 8]]}
{'id': 3, 'coords': [[1, 2], [2, 2]]}
{'id': 4, 'coords': [[7, 9], [6, 9]]}
{'id': 5, 'coords': [[8, 9], [9, 9], [9, 8]]}
{'id': 6, 'coords': [[2, 3]]}
{'id': 7, 'coords': [[4, 6], [4, 7]]}
{'id': 8, 'coords': [[1, 9], [1, 8]]}
{'id': 9, 'coords': [[8, 3], [8, 4], [8, 5]]}
{'id': 10, 'coords': [[5, 4], [4, 4], [4, 3]]}
{'id': 11, 'coords': [[7, 1]]}
{'id': 12, 'coords': [[1, 1], [2, 1]]}
{'id': 13, 'coords': [[2, 0]]}

Heuristic Params
h:  -430 , g:  0 , f:  -430
```

```
Execution...
[0, 0, 0, 0, 0, 0, 0, 0, -1, -1]
[0, 12, 0, 0, 0, 0, 0, 0, -1, -1]
[13, 12, 0, 0, 0, 0, 0, 0, 0, 2]
[0, 0, 0, 0, 0, 0, 0, 0, 8, 8]
[0, 0, 0, 10, 10, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 10, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 4]
[0, 11, 0, 0, 0, 0, 0, 0, 0, 4]
[0, 3, 0, 0, 0, 0, 1, 1, 0, 5]
[6, 3, 7, 7, 9, 9, 9, 1, 5, 5]

occupied bytes:  143384576
executed in 0:00:03.957681

Lenght of solution path:  115
```

Density 0.4

```
initial configuration

[7, 7, 0, 0, 0, 0, 0, 16, 0, 0]
[7, 0, 0, 0, 3, 3, 3, 0, 0, 0]
[0, 0, 0, 0, 11, 11, 0, 0, 0, 0]
[0, 0, 5, 5, 6, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 6, 0, 0, 0, 0, 4]
[12, 12, 0, 0, 6, 8, 8, 0, 0, 0]
[9, 0, 0, 15, 15, 15, 8, 17, 10, 10]
[9, 14, 0, 2, 2, 0, 17, 17, 10, 1]
[-1, -1, 0, 0, 0, 0, 0, 0, 0, 0]
[-1, -1, 13, 13, 13, 0, 0, 0, 0, 0]

 Density of obstacles:  0.4

 Obstacles:
{'id': -1, 'coords': [[8, 0], [8, 1], [9, 0], [9, 1]]}
{'id': 1, 'coords': [[7, 9]]}
{'id': 2, 'coords': [[7, 3], [7, 4]]}
{'id': 3, 'coords': [[1, 6], [1, 5], [1, 4]]}
{'id': 4, 'coords': [[4, 9]]}
{'id': 5, 'coords': [[3, 3], [3, 2]]}
{'id': 6, 'coords': [[5, 4], [4, 4], [3, 4]]}
{'id': 7, 'coords': [[0, 1], [0, 0], [1, 0]]}
{'id': 8, 'coords': [[6, 6], [5, 6], [5, 5]]}
{'id': 9, 'coords': [[7, 0], [6, 0]]}
{'id': 10, 'coords': [[6, 9], [6, 8], [7, 8]]}
{'id': 11, 'coords': [[2, 5], [2, 4]]}
{'id': 12, 'coords': [[5, 0], [5, 1]]}
{'id': 13, 'coords': [[9, 2], [9, 3], [9, 4]]}
{'id': 14, 'coords': [[7, 1]]}
{'id': 15, 'coords': [[6, 3], [6, 4], [6, 5]]}
{'id': 16, 'coords': [[0, 7]]}
{'id': 17, 'coords': [[7, 6], [7, 7], [6, 7]]}

Heuristic Params
h:  -764 , g:  0 , f:  -764
```

```
Execution...
[7, 7, 0, 0, 0, 0, 0, 16, -1, -1]
[7, 0, 0, 3, 3, 3, 0, 0, -1, -1]
[0, 0, 0, 0, 11, 11, 0, 0, 0, 0]
[5, 5, 0, 6, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 6, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 6, 0, 0, 8, 8, 0, 0]
[15, 15, 15, 0, 0, 0, 0, 8, 0, 0]
[12, 12, 0, 0, 0, 0, 0, 17, 0, 4]
[9, 0, 0, 0, 0, 0, 17, 17, 10, 10]
[9, 14, 2, 2, 13, 13, 13, 0, 10, 1]

occupied bytes:  282951680
executed in 0:00:09.962194

Lenght of solution path:  95
```

Density 0.5

```
initial configuration

[25, 0, 0, 11, 5, 5, 0, 1, 3, 3]
[25, 0, 0, 11, 24, 24, 0, 0, 0, 0]
[0, 9, 9, 0, 0, 0, 0, 17, 22, 22]
[0, 19, 0, 0, 10, 18, 18, 17, 0, 22]
[2, 19, 19, 0, 0, 15, 0, 7, 8, 8]
[2, 0, 0, 0, 0, 0, 7, 7, 0, 21]
[2, 13, 0, 0, 14, 4, 12, 12, 0, 0]
[0, 0, 0, 0, 0, 0, 23, 23, 0, 6]
[-1, -1, 0, 0, 0, 0, 0, 0, 6, 6]
[-1, -1, 0, 0, 0, 20, 0, 0, 16, 0]

 Density of obstacles:  0.5

 Obstacles:
{'id': -1, 'coords': [[8, 0], [8, 1], [9, 0], [9, 1]]}
{'id': 1, 'coords': [[0, 7]]}
{'id': 2, 'coords': [[4, 0], [5, 0], [6, 0]]}
{'id': 3, 'coords': [[0, 9], [0, 8]]}
{'id': 4, 'coords': [[6, 5]]}
{'id': 5, 'coords': [[0, 4], [0, 5]]}
{'id': 6, 'coords': [[8, 8], [8, 9], [7, 9]]}
{'id': 7, 'coords': [[5, 6], [5, 7], [4, 7]]}
{'id': 8, 'coords': [[4, 8], [4, 9]]}
{'id': 9, 'coords': [[2, 2], [2, 1]]}
{'id': 10, 'coords': [[3, 4]]}
{'id': 11, 'coords': [[0, 3], [1, 3]]}
{'id': 12, 'coords': [[6, 6], [6, 7]]}
{'id': 13, 'coords': [[6, 1]]}
{'id': 14, 'coords': [[6, 4]]}
{'id': 15, 'coords': [[4, 5]]}
{'id': 16, 'coords': [[9, 8]]}
{'id': 17, 'coords': [[2, 7], [3, 7]]}
{'id': 18, 'coords': [[3, 6], [3, 5]]}
{'id': 19, 'coords': [[4, 2], [4, 1], [3, 1]]}
{'id': 20, 'coords': [[9, 5]]}
{'id': 21, 'coords': [[5, 9]]}
{'id': 22, 'coords': [[3, 9], [2, 9], [2, 8]]}
{'id': 23, 'coords': [[7, 7], [7, 6]]}
{'id': 24, 'coords': [[1, 5], [1, 4]]}
{'id': 25, 'coords': [[1, 0], [0, 0]]}

Heuristic Params
h:  -818 , g:  0 , f:  -818
```

```
Execution...
[0, 0, 0, 0, 0, 0, 3, 3, -1, -1]
[25, 0, 0, 0, 0, 0, 0, 0, -1, -1]
[25, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[11, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[11, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[9, 9, 24, 24, 0, 0, 0, 0, 0, 0]
[2, 5, 5, 8, 8, 18, 18, 0, 22, 22]
[2, 19, 10, 1, 0, 0, 7, 17, 0, 22]
[2, 19, 19, 12, 12, 7, 7, 17, 6, 21]
[13, 4, 14, 15, 20, 23, 23, 6, 6, 16]

occupied bytes:  1136381952
executed in 0:01:04.262697

Lenght of solution path:  298
```

Density 0.6

```
initial configuration

[33, 33, 28, 28, 11, 11, 11, 17, 0, 27]
[10, 0, 19, 19, 19, 0, 29, 17, 6, 13]
[10, 3, 3, 0, 24, 0, 29, 29, 6, 0]
[10, 0, 18, 0, 0, 0, 0, 23, 0, 0]
[4, 22, 22, 12, 0, 25, 0, 0, 0, 7]
[4, 14, 12, 12, 9, 9, 5, 30, 16, 7]
[0, 0, 0, 0, 0, 21, 32, 32, 1, 0]
[8, 0, 31, 0, 0, 0, 32, 1, 1, 0]
[-1, -1, 0, 0, 26, 0, 15, 15, 2, 0]
[-1, -1, 0, 0, 0, 0, 0, 0, 20, 0]

 Density of obstacles:  0.6

 Obstacles:
{'id': -1, 'coords': [[8, 0], [8, 1], [9, 0], [9, 1]]}
{'id': 1, 'coords': [[6, 8], [7, 8], [7, 7]]}
{'id': 2, 'coords': [[8, 8]]}
{'id': 3, 'coords': [[2, 2], [2, 1]]}
{'id': 4, 'coords': [[4, 0], [5, 0]]}
{'id': 5, 'coords': [[5, 6]]}
{'id': 6, 'coords': [[2, 8], [1, 8]]}
{'id': 7, 'coords': [[5, 9], [4, 9]]}
{'id': 8, 'coords': [[7, 0]]}
{'id': 9, 'coords': [[5, 4], [5, 5]]}
{'id': 10, 'coords': [[1, 0], [2, 0], [3, 0]]}
{'id': 11, 'coords': [[0, 6], [0, 5], [0, 4]]}
{'id': 12, 'coords': [[5, 2], [5, 3], [4, 3]]}
{'id': 13, 'coords': [[1, 9]]}
{'id': 14, 'coords': [[5, 1]]}
{'id': 15, 'coords': [[8, 6], [8, 7]]}
{'id': 16, 'coords': [[5, 8]]}
{'id': 17, 'coords': [[1, 7], [0, 7]]}
{'id': 18, 'coords': [[3, 2]]}
{'id': 19, 'coords': [[1, 2], [1, 3], [1, 4]]}
{'id': 20, 'coords': [[9, 8]]}
{'id': 21, 'coords': [[6, 5]]}
{'id': 22, 'coords': [[4, 1], [4, 2]]}
{'id': 23, 'coords': [[3, 7]]}
{'id': 24, 'coords': [[2, 4]]}
{'id': 25, 'coords': [[4, 5]]}
{'id': 26, 'coords': [[8, 4]]}
{'id': 27, 'coords': [[0, 9]]}
{'id': 28, 'coords': [[0, 3], [0, 2]]}
{'id': 29, 'coords': [[2, 7], [2, 6], [1, 6]]}
{'id': 30, 'coords': [[5, 7]]}
{'id': 31, 'coords': [[7, 2]]}
{'id': 32, 'coords': [[6, 7], [6, 6], [7, 6]]}
{'id': 33, 'coords': [[0, 1], [0, 0]]}

Heuristic Params
h:  -1242 , g:  0 , f:  -1242
```

```
Execution...
[33, 33, 28, 28, 0, 0, 0, 0, -1, -1]
[10, 11, 11, 11, 0, 0, 0, 0, -1, -1]
[10, 19, 19, 19, 0, 0, 0, 0, 0, 27]
[10, 24, 0, 0, 0, 0, 0, 0, 0, 13]
[18, 3, 3, 0, 0, 0, 0, 0, 0, 6]
[22, 22, 0, 0, 0, 0, 0, 0, 0, 6]
[9, 9, 0, 29, 0, 0, 0, 30, 16, 1]
[4, 0, 12, 29, 29, 17, 32, 32, 1, 1]
[4, 12, 12, 23, 25, 17, 32, 2, 0, 7]
[8, 14, 31, 5, 21, 26, 15, 15, 20, 7]

occupied bytes:  1666908160
executed in 0:01:31.767260

Lenght of solution path:  259
```