Homework 1: pathfinding con A*

L'environment consiste in un grid world delimitato a una regione rettangolare, in cui un certo numero di celle sono "piene" e quindi inaccessibili. L'agente può muoversi nelle 4 direzioni, una cella per mossa. Si vuole programmare un agente che dia l'impressione di vagare senza una meta precisa. Più precisamente, l'agente deve ripetutamente:

1. estrarre una cella (non piena) da raggiungere all'interno della regione;
2. cercare un percorso ottimale che la raggiunga, se esiste;
3. seguire il percorso scelto.

A tal fine si sceglie di usare l'algoritmo A* per implementare il passo 2.

Esercizio 1

Completare il codice Python nel file “studente/esercizio1.py” (vedi la sezione “Struttura del codice”), in maniera che il comportamento dell'agente sia **quello descritto sopra**. In particolare, vanno implementate le seguenti funzioni:

- **l'algoritmo A*** (si può usare la classe Node già presente nel codice per rappresentare i nodi);
- **l'euristica** da utilizzare in A*.

Il significato di *percorso ottimale* è da intendersi come quello di **percorso con numero di mosse minimo**.

Esercizio 2

Completare il codice Python nel file “studente/esercizio2.py” (vedi la sezione “Struttura del codice”), in maniera che, anche se non è possibile arrivare al goal, **venga comunque ritornato** un percorso ottimale che contenga il **minor numero possibile di celle piene**. In particolare, vanno implementate le seguenti funzioni:

- **l'euristica** da utilizzare in A*;
- la funzione che genera i **successori** di uno stato e i **costi** associati a ogni transizione.

Nel caso in cui esista un percorso che arrivi al goal, l’agente si deve comportare come nell’esercizio 1. Per definire l’euristica in questo esercizio si possono utilizzare anche le dimensioni della griglia.

In entrambi gli esercizi, la **scelta dell’euristica va motivata** con un commento sopra la funzione da completare.

Struttura del codice

Il codice fornito contiene:

- una cartella “*studente*” contenente:
    - due file “*esercizio1.py*” e “*esercizio2.py*” **da completare a cura dello studente**

- un file “*main.py*” che contiene il resto dell’implementazione e richiama i due file di cui sopra

Per testare la propria soluzione si può eseguire come script python il file “main.py”, che per ogni esercizio proverà ad estrarre e raggiungere 10 goal, stampando i risultati sul terminale. Per eseguire il codice è necessario aver installato la libreria PyGame. Per ulteriori dettagli si vedano i commenti nel template dei file “esercizio1.py” ed “esercizio2.py” forniti.

Regole di consegna

- Va consegnato un singolo file ZIP di nome **<matricola>.zip (ad es. “123456.zip”)** contenente una singola cartella di nome “*studente*” con i due file “*esercizio1.py*” ed “*esercizio2.py*”, con scadenza **5 giorni prima della data dell’esame** che si intende sostenere**.**
- Gli homework sono validi **solo per gli appelli di esame di Giugno e Luglio** e **sostituiscono la prova orale**.

Valutazione

Ogni homework è valutato da 0 a 30 punti. Il punteggio finale per questa parte del corso verrà calcolato come punteggio (voto_hw1 + voto_hw2 + 2 * voto_scritto) / 4