Problem solving as a search

Salvatore Corvaglia, Franco Savino, Marco Villani

8/10/2017

0.1 Problem description

Assignment (1st part)
Transform the breadth-first Python code into a depth-first implementation.
Assignment (2nd part)
Transform the breadth-first Python code into an A* implementation. Use both h1 and h2.

0.2 Solution

Nella prima parte dell'assignment, dato il codice del puzzle con metodo di risoluzione di tipo breadth first, è stato necessario adattare il codice ad effettuare una ricerca di tipo depth first.
Questi due tipi di algoritmo effettuano una ricerca della soluzione al problema senza alcuna conoscenza relativa al percorso migliore per raggiungere il goal.
L'algoritmo prevede l'utilizzo di due liste, Open e Closed, nelle quasi verranno inseriti rispettivamente i nodi da visitare e quelli già visitati. In particolare la differenza tra Breadth first search e Depth first search risiede nella modalità di estrazione dei nodi dalla lista Open. Nel primo caso viene utilizzata una logica di tipo FIFO (First In First Out) mentre nel secondo caso è di tipo LIFO (Last In First Out).
Finchè la lista Open non è vuota vengono analizzati tutti i nodi prelevati secondo la logica assegnata (LIFO). Per implementare questa logica abbiamo utilizzato il metodo "popleft" perché la lista open è stata riempita utilizzando il metodo "appendleft" quindi l'ultimo arrivato è quello che si trova più a sinistra.
La seconda parte dell'Assignment richiede, invece, l'implementazione dell'algoritmo A*. Questo algoritmo non è più di tipo uninformed. In questo caso la lista Open viene ordinata in funzione di Fn (costo totale) definita come la somma tra g(n) e h(n) che rappresentano rispettivamente la distanza tra il nodo di partenza e il nodo attuale e la distanza euristica (espressa in termini di distanza di manhattan) tra il nodo attuale e il goal (stato finale).
La g(n) l'abbiamo interpretata come il numero di nodi attraversati per raggiungere il nodo attuale a partire dal nodo di partenza, mentre per h(n) abbiamo implementato una funzione (chiamata h2) che permette di definire la somma delle distanze di manhattan sulla base delle coordinate dei tasselli fuori posto rispetto alla posizione da raggiungere per ottenere il goal.
In particolare in questo confronto abbiamo utilizzato la seguente formula:

```
distance += abs(x - x_goal) + abs(y - y_goal)
```

dove x e y sono le coordinate del nodo attuale mentre x_goal e y_goal sono le coordinate del goal (per ogni singolo tassello).