Esercitazione 2: Ricerca non informata

Fondamenti di Intelligenza Artificiale

SAPIENZA
UNIVERSITÀ DI ROMA

A.A. 2022/2023

DIAG, Dipartimento di Ingegneria Informatica, Automatica, e Gestionale “Antonio Ruberti”.

Installazione pygame

Installazione con pip:

```
pip3 install pygame
```

Installazione con anaconda:

```
conda install pygame
```

Si consigliano le versioni 3.7 di python e 2.3.0 di pygame

Repository codice esercitazioni

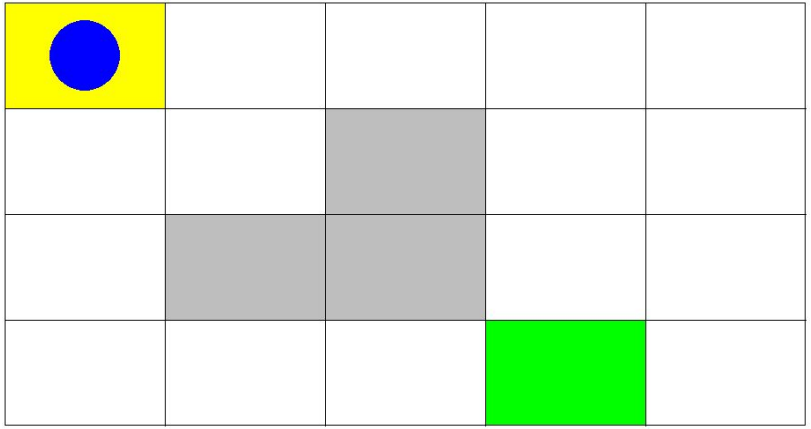
link: https://github.com/KRLGroup/FondamentiIA-2223

Download della repo con git (opzionale):

```
git clone https://github.com/KRLGroup/FondamentiIA-2223.git
```

Il codice per questa esercitazione è in “`esercitazione02.py`”

Esercizio 1: problema



Grid world dove:

- azioni: “N”, “S”, “W”, “E” (tutte a costo 1)
- le celle grigie sono inaccessibili
- l’agente deve raggiungere la cella verde

Esercizio 1

A) Quali sono il maximum branching factor e la profondità della soluzione? Calcolare le risultanti complessità di spazio e tempo (worst case) di breadth-first search

B) La soluzione trovata da breadth-first search è sempre ottima per ogni configurazione di start, goal e celle grigie (assumendo che la soluzione esista)? Motivare la risposta

Esercizio 1

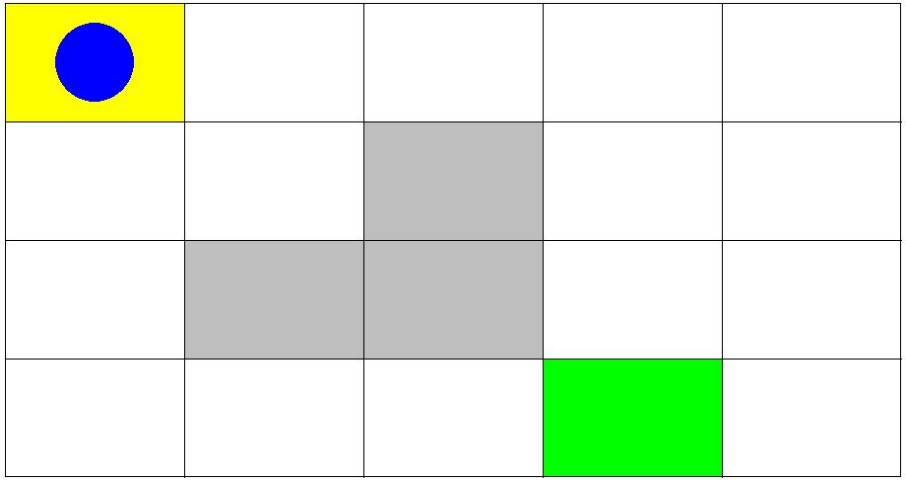
C) Se si aggiungessero le azioni diagonali, ognuna di costo $c > 2$, sarebbe ancora garantita l'ottimalità della soluzione ritornata da breadth-first search? Motivare la risposta, e in caso negativo, fornire un algoritmo di ricerca non informata che garantisca l'ottimalità della soluzione trovata.

Esercizio 1

D) Implementare la variante appropriata (su grafi o su alberi) di breadth-first search (completare codice)

```
def breadth_first_search(
    initial_state,
    goal_test,
    successor_fn,
    cost_fn
):
    pass
```

Esercizio 2: problema



Come es 1, ma disponibili anche azioni diagonali (costo $c > 2$)

Esercizio 2

A) Qual è il limite inferiore al costo di ogni azione e il costo della soluzione ottima per la configurazione data? Calcolare le risultanti complessità di spazio e tempo (worst case) di uniform cost search

B) La completezza di uniform cost search su **alberi** è garantita per ogni configurazione di start, goal e celle grigie (assumendo esista una soluzione)? Motivare

Esercizio 2

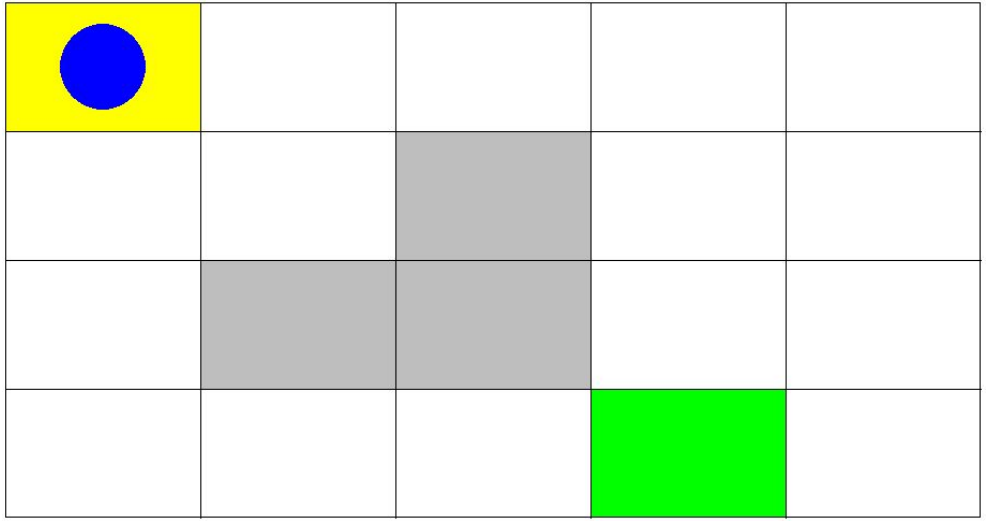
C) Se si aggiungesse l’azione None (nessun movimento, costo 0), sarebbe ancora garantita la completezza di uniform-cost search su **alberi**? Motivare

Esercizio 2

D) Implementare la variante appropriata di uniform cost search (completare codice)

```
def uniform_cost_search(
    initial_state,
    goal_test,
    successor_fn,
    cost_fn,
):
    pass
```

Esercizio 3: problema



Stesso problema dell’esercizio 1

Esercizio 3

A) Implementare la variante appropriata di depth-first search (completare codice)

```
def depth_first_search(
    initial_state,
    goal_test,
    successor_fn,
    cost_fn
):
    pass
```

Esercizio 3

B) Nel caso in cui la griglia non abbia confini, è ancora garantita la completezza con depth-first search? In caso negativo, fornire un algoritmo di ricerca alternativo che sia completo e non abbia complessità di spazio esponenziale (rispetto alla profondità)

Esercizio 4

Nel caso in cui le dimensioni della griglia siano molto grandi (e.g. > 1000x1000), esiste un algoritmo di ricerca non informata che possa nella pratica trovare una soluzione ottima (se esiste) per ogni configurazione possibile di start, goal e ostacoli? Motivare e, in caso negativo, fornire una possibile classe di algoritmi alternativa