# Algorítmo de Q-Learning

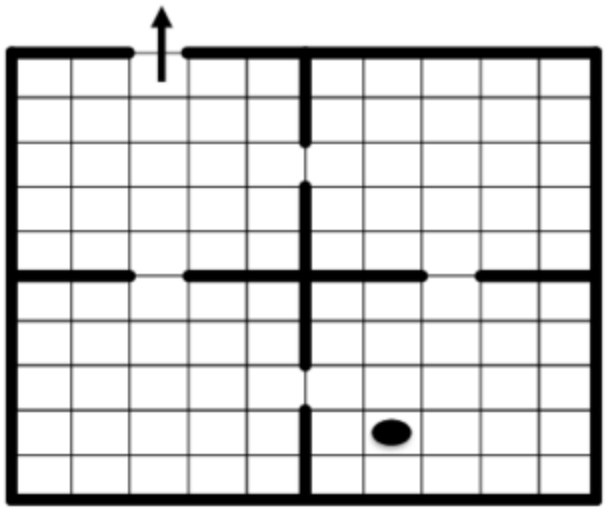
En este ejercicio vamos a probar el algorítmo de Q-learning como un representatnte de los métodos off-policy. Nuestro objetivo, es evaluar el algoritmo sobre distintos ambientes. Para cada uno de los ambientes deben ejecutar un agente de Q-learning en el ambiente, evaluar su ejecución y validar la efectividad del aprendizaje del agente entrenado sobre el ambiente.

## Gridworld

Sobre el ambiente de Gridworld que hemos venido utilizando, ejecute el algoritmo de Q-learning. Debe ejecutar el algoritmo hasta su convergencia y entregar tanto la política resultado y la Q-tabla.

## Laberinto de cuartos

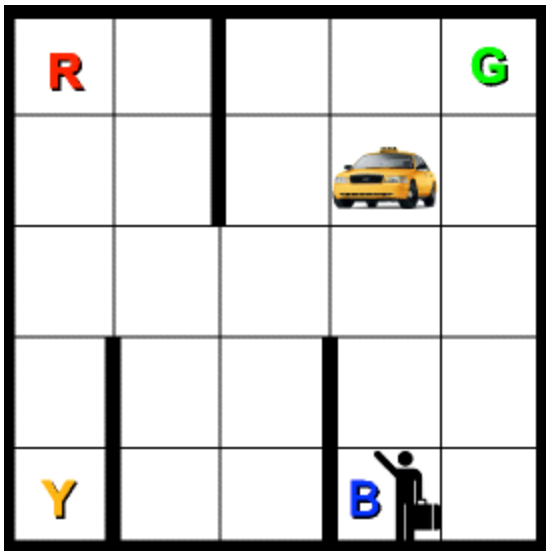
El ambiente del laberinto de cuartos consiste en una cuadricula con 4 cuartos como se muestra a continuación.



Para este ambiente queremos que el agente aprenda a salir por el cuarto superior izquierdo en la menor cantidad de pasos posible. La única restricción de este ambiente es que al final de cada episodio el agente comienza nuevamente en cualquier posicón valida del laberinto. Usted debe definir los parametros ($\alpha$, $\gamma$, $\epsilon$, recompensa) para asegurar el comportamiento del agente

## Taxi

El ambiente de taxi consiste en una cuadrícula de `5x5`, con 4 estaciones (`R`, `G`, `Y`, `B`), como se muestra en la figura. El taxi puede moverse libremente entre las casillas de la cuadrícula. sin embargo, no puede atravesar por los separadores (las lines más gruesas en la figura).




El taxi (i.e., el agente) se mueve por el ambiente buscando recoger un pasajero. Los pasajeros aparecen aleatoriamente en alguno de los paraderos (uno a la vez) y deben llegar a su destino (algún otro paradero).

Las acciones del agente corresponden a los movimientos del agente en el tablero y las acciones para recoger y dejar pasajeros. Tratar de recoger o dejar un pasajero en un lugar indebido o cuando no hay pasajero, son consideradas malas accciones del agente y deben ser penalizadas (tienen una recompenza de -10). Para asegurar que el agente efectivamente recoge pasajeros, debemos darle una recompensa de 1 a la acción. Efectivamente dejar al pasajero tiene una recompensa de 5.

Implemente el algoritmo de Q-learning (defina sus propios parámetros) para el aprendizaje del agente.

In [1]:
```
import pandas as pd

from assignment_q_learning.q_learning_agent import QLearningAgent
from assignment_q_learning.taxi_environment import TaxiEnvironmentWorld, Sta
from assignment_q_learning.walled_environment import WalledEnvironmentWorld
```

In [1]:
```
walls = [((0, 4), (0, 5)), ((1, 4), (1, 5)), ((3, 4), (3, 5)), ((4, 4), (4,
         ((4, 3), (5, 3)), ((4, 4), (5, 4)), ((4, 5), (5, 5)), ((4, 6), (5,
         ((5, 4), (5, 5)), ((6, 4), (6, 5)), ((8, 4), (8, 5)), ((9, 4), (9,

board = [[' ' for i in range(10)] for j in range(10)]
board[0][2] = '+100'

wall_environment = WalledEnvironmentWorld(board=board, walls=walls, terminal
```

```
print(wall_environment)
```

```
   0  1     2  3  4  5  6  7  8  9
0        +100
1
2   C
3
4
5
6
7
8
9
```

In [2]:
```
q_agent_walls = QLearningAgent(wall_environment, learning_rate=0.8, discount
```

In [3]:
```
q_agent_walls.iterate_learning(1000000)
q_agent_walls.print_policy()
```

```
100%|██████████| 1000000/1000000 [00:08<00:00, 122358.01it/s]
       0      1     2     3     4      5      6     7     8     9
0  right  right  None  left  left   down   down  down  down  down
1     up     up    up    up    up   down   down  down  down  down
2     up     up    up    up    up   left   left  left  left  left
3     up     up    up    up    up     up     up    up    up    up
4     up     up    up    up    up     up     up    up    up    up
5  right  right    up  left  left  right  right    up  left  left
6     up     up    up    up    up   down     up    up    up    up
7     up     up    up    up    up   left   left    up    up    up
8     up     up    up    up    up     up     up    up    up    up
9     up     up    up    up    up     up     up    up    up    up
```

In [2]:
```
taxi_walls = [
    ((1, 0), (2, 0)), ((1, 1), (2, 1)),
    ((0, 3), (1, 3)), ((0, 4), (1, 4)),
    ((2, 3), (3, 3)), ((2, 4), (3, 4)),
]

r, g, y, b = (
    Station((0, 0), 'R'),
    Station((4, 0), 'G'),
    Station((0, 4), 'Y'),
    Station((3, 4), 'B')
)

stations = [r, g, y, b]

taxi_world = TaxiEnvironmentWorld(board=[[' '] * 5] * 5, walls=taxi_walls, s
```

In [10]:
```
q_agent_taxi = QLearningAgent(taxi_world, learning_rate=0.8, discount_factor
                              learning_rate_decay=0.1 ** (1 / 2500000))
```

In [11]:
```
q_agent_taxi.iterate_learning(5000000)
```

```
100%|██████████| 5000000/5000000 [00:38<00:00, 131578.68it/s]
```

In [12]:
```
def print_taxi_policy(q_agent, passenger_station, on_passenger=False):
    world = q_agent.world
    policy_matrix = [[None for y_ in range(world.num_rows)] for x in range(w
    for y in range(world.num_rows):
        for x in range(world.num_cols):
            state = TaxiState((x, y), passenger_station, on_passenger)
            if state in q_agent_taxi.Q:
                action = q_agent.get_opt_action(state)
                if action is not None:
                    policy_matrix[x][y] = action.value
            else:
                policy_matrix[x][y] = None

    print(pd.DataFrame(policy_matrix).transpose())
```

In [13]:
```
print_taxi_policy(q_agent_taxi, None, False)
```

```
      0      1      2      3      4
0  down   left  right  right  right
1    up   left  right     up     up
2    up   left  right   down   left
3  down  right  right   down   left
4    up     up     up   down   left
```

In [14]:
```
print_taxi_policy(q_agent_taxi, None, True)
```

```
       0     1     2       3       4
0  pickup  None  None    None  pickup
1    None  None  None    None    None
2    None  None  None    None    None
3    None  None  None    None    None
4  pickup  None  None  pickup    None
```

In [15]:
```
for station in stations:
    print(f'STATION: {station}_____________')
    print_taxi_policy(q_agent_taxi, station, False)
```

```
STATION: Station(position=(0, 0), name='R')_____________
         0     1     2     3     4
0  dropoff  left  down  down  left
1       up  left  down  down    up
2       up    up  left  left  left
3       up    up    up    up    up
4       up    up    up    up  left
STATION: Station(position=(4, 0), name='G')_____________
       0      1      2      3        4
0   down   down  right  right  dropoff
1  right   down  right  right       up
2  right  right  right     up       up
3     up     up     up  right       up
4     up     up     up     up       up
STATION: Station(position=(0, 4), name='Y')_____________
         0     1     2     3     4
0     down  left  down  down  down
1     down  left  down  down  left
2     down  left  left  left  left
3     down    up    up    up  left
4  dropoff    up    up    up  left
STATION: Station(position=(3, 4), name='B')_____________
       0      1      2        3     4
0   down   down  right    right  down
1  right   down  right     down  down
2  right  right  right     down  down
3     up     up     up     down  down
4     up     up     up  dropoff  left
```