**RESULTS WHEN REVISITED AFTER READING SUTTON SECTION I**

**1.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= last\_best\_state + LEVEL\_GRANULARITY:

return 1

else:

return -1

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

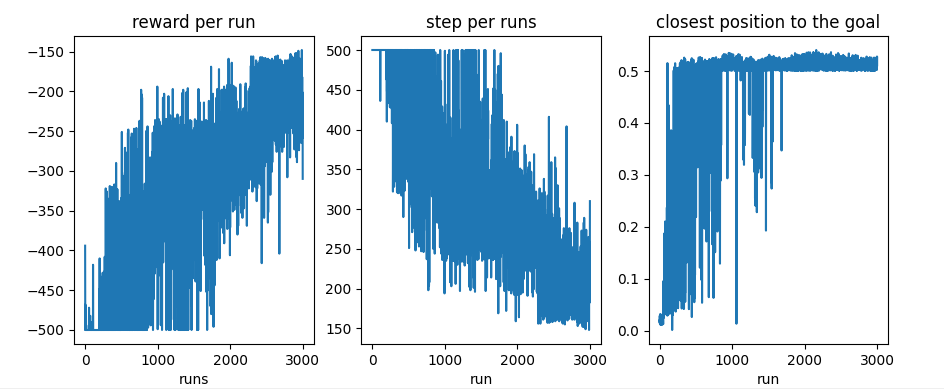
LEVEL\_GRANULARITY=0.001

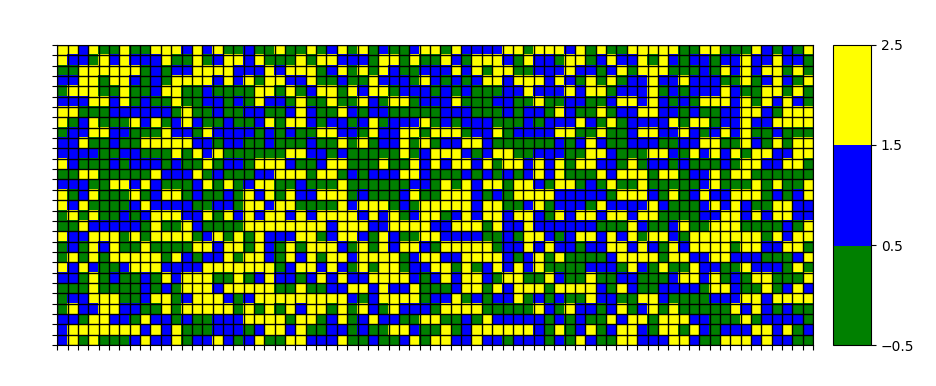
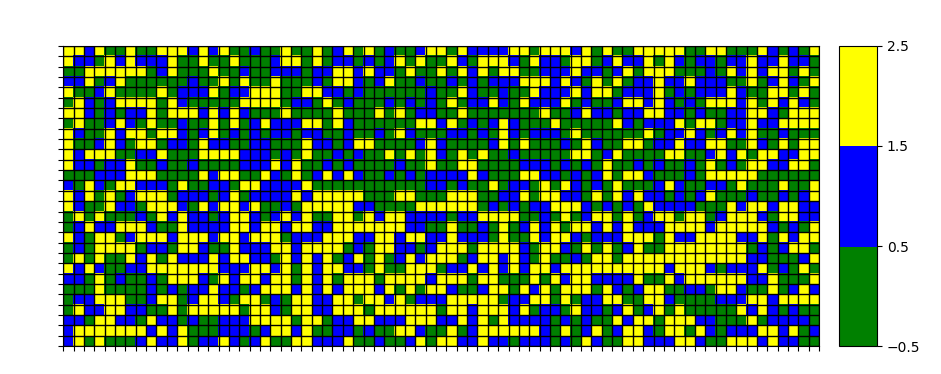
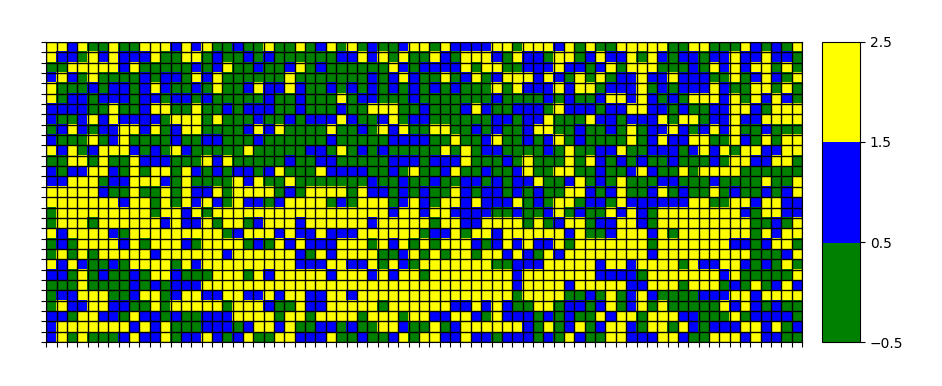
**INIT**

self.q\_values = np.random.uniform(low = -1, high = 1,

size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))



**2.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= last\_best\_state + LEVEL\_GRANULARITY:

return 1

else:

return -1

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.99

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

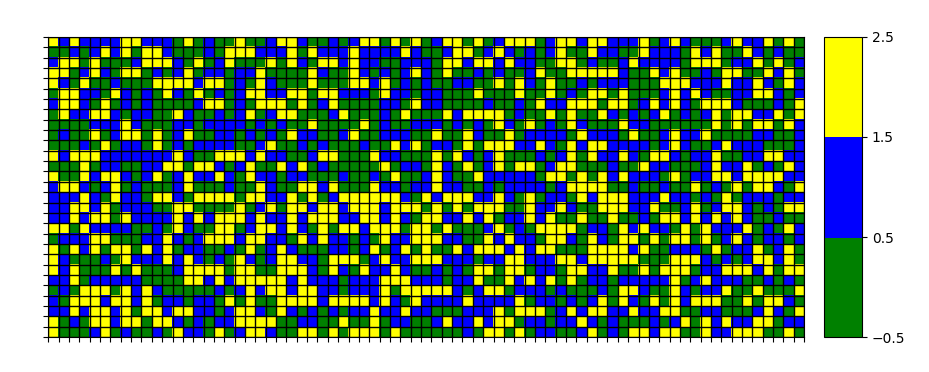
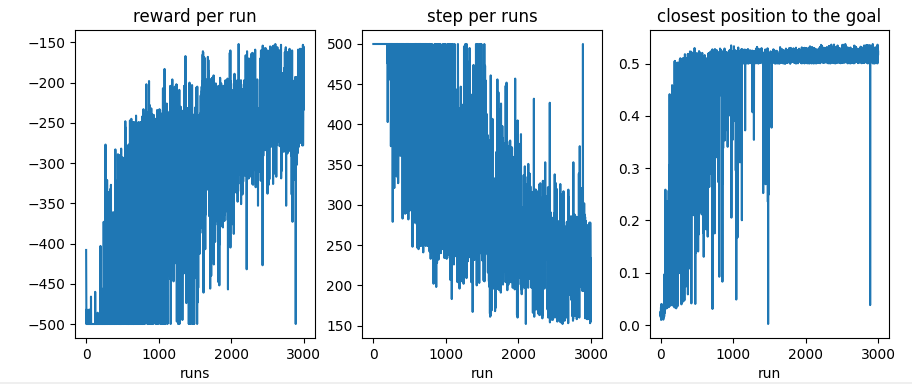
LEVEL\_GRANULARITY=0.001

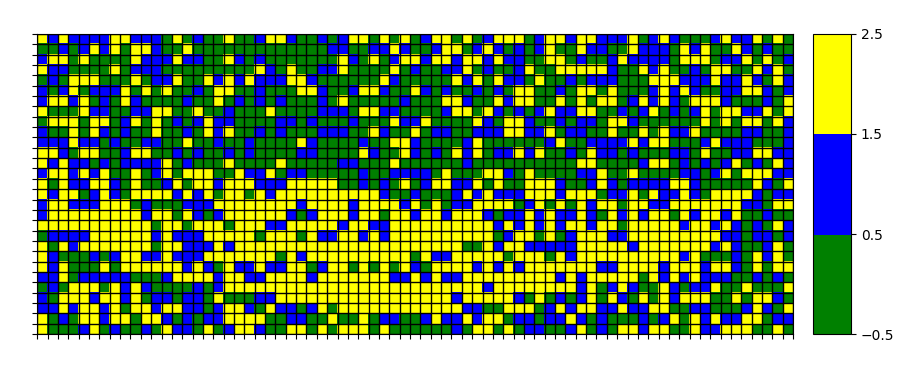
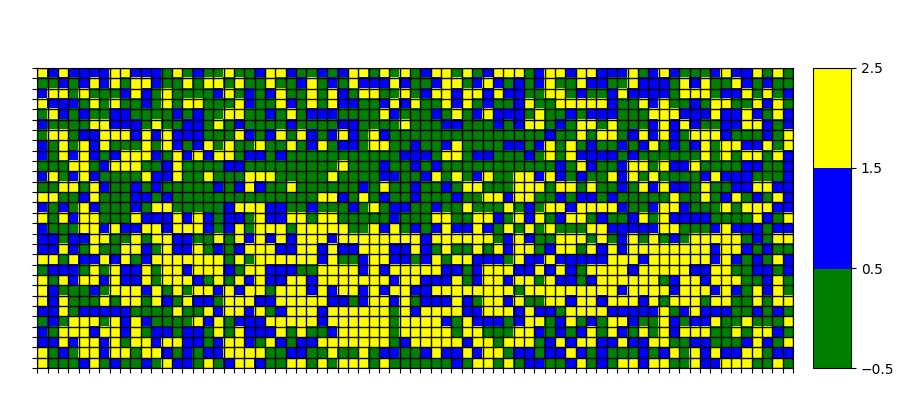
**INIT**

self.q\_values = np.random.uniform(low = -1, high = 1,

size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))

****

****

**3.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= last\_best\_state + LEVEL\_GRANULARITY:

return 1

else:

return -1

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.4

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

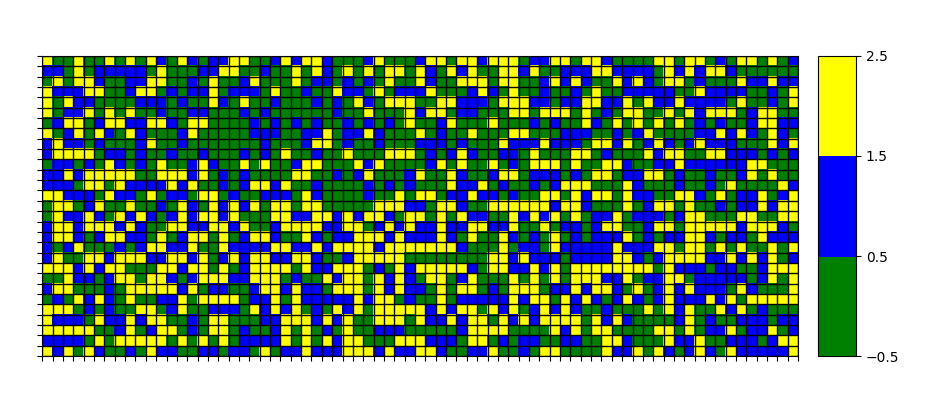
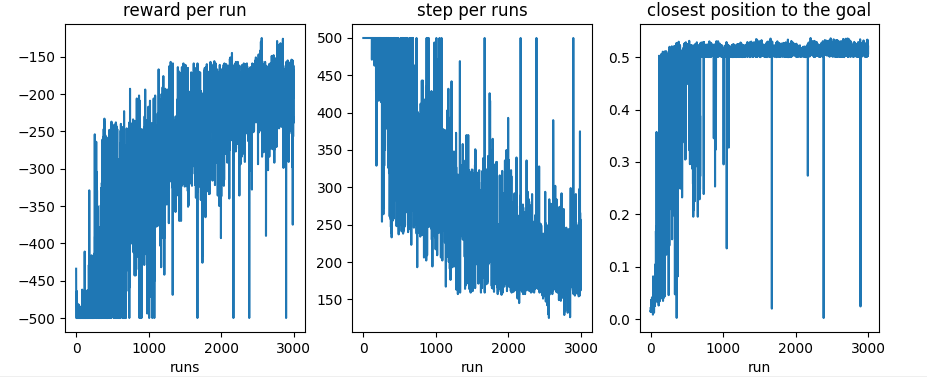
LEVEL\_GRANULARITY=0.001

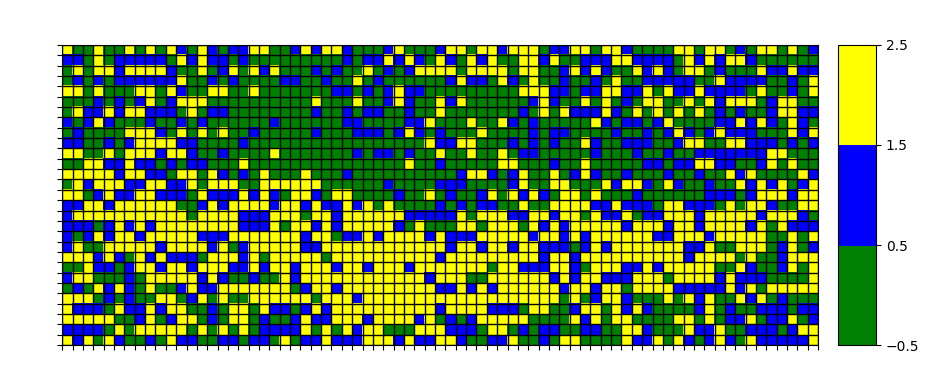
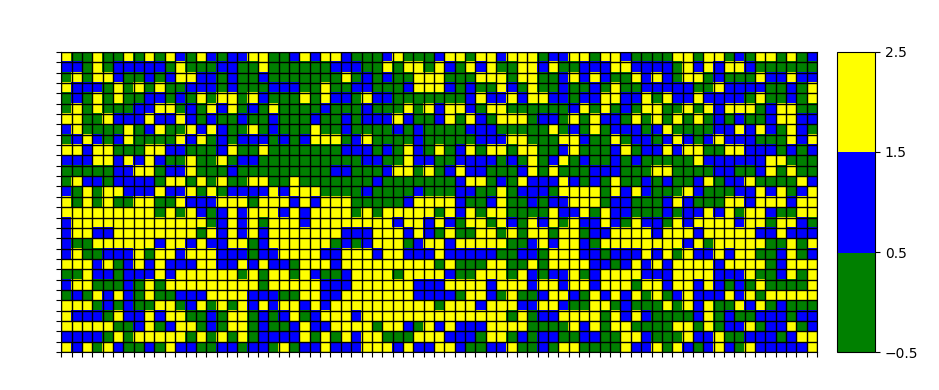
**INIT**

self.q\_values = np.random.uniform(low = -1, high = 1,

size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))

****

****

**4.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= last\_best\_state + LEVEL\_GRANULARITY:

return 1

else:

return -1

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

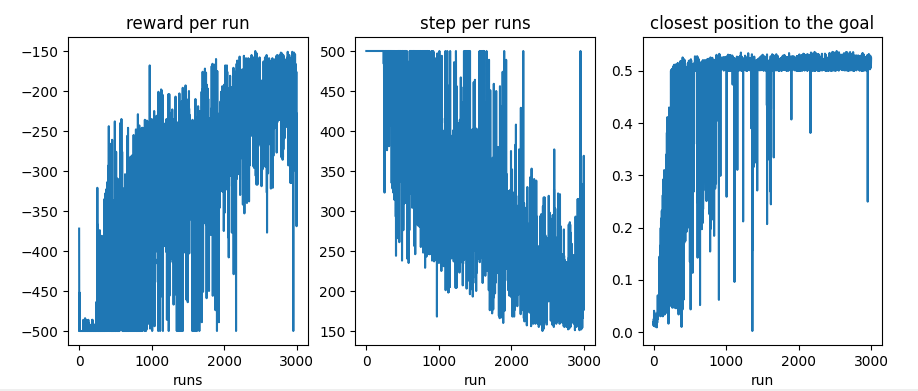
LEVEL\_GRANULARITY=0.001

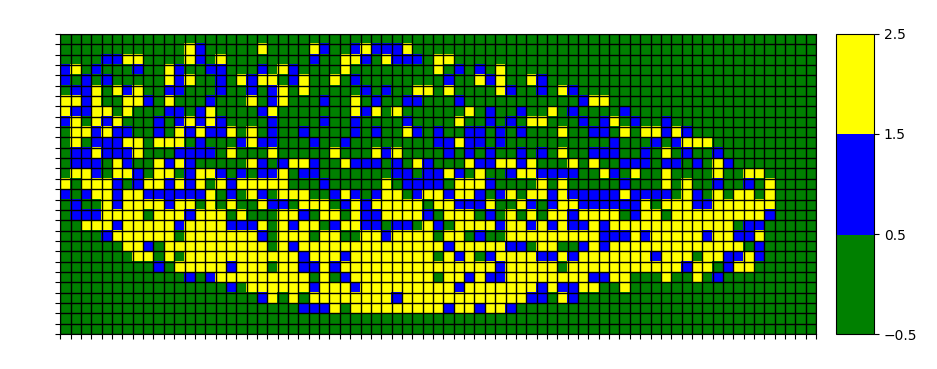
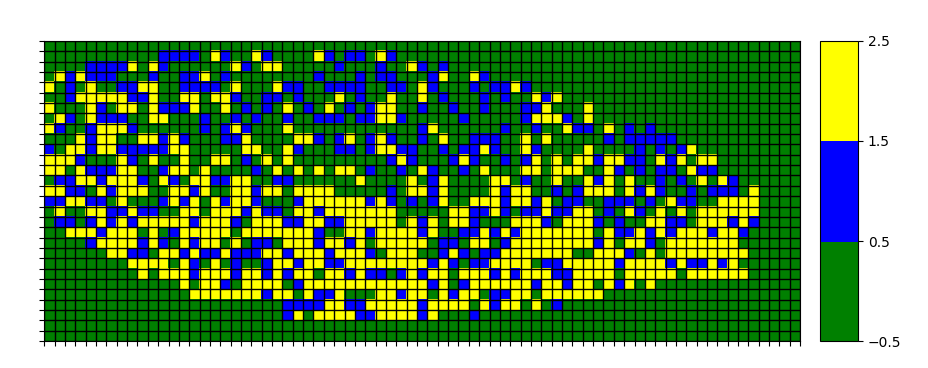
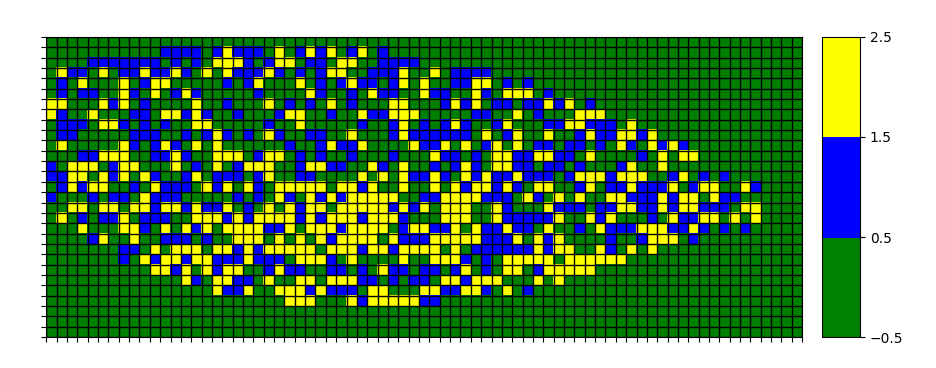
**INIT**

self.q\_values = np.random.uniform(low = 0, high = 0,

size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))





**5.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= 0.5

return 0

else:

return -1

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

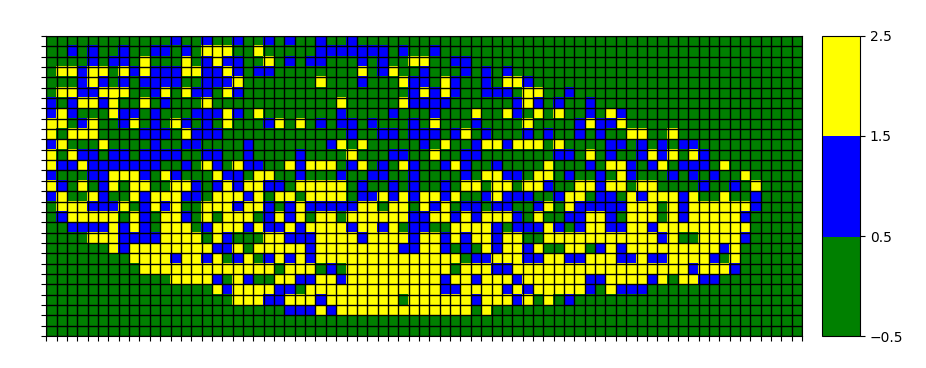
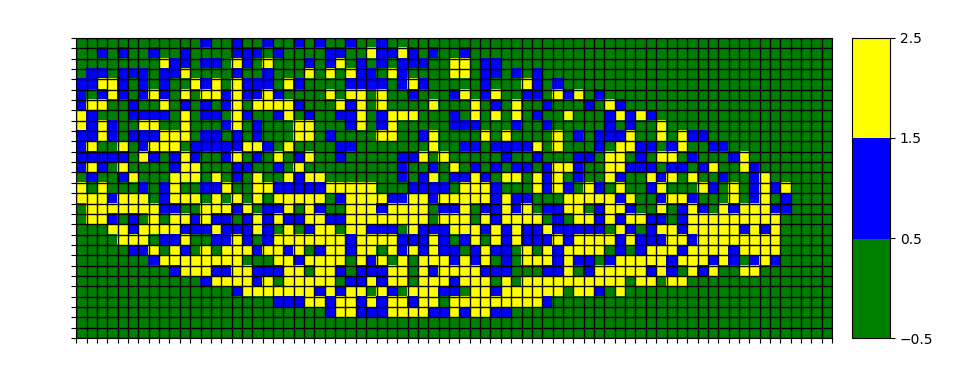
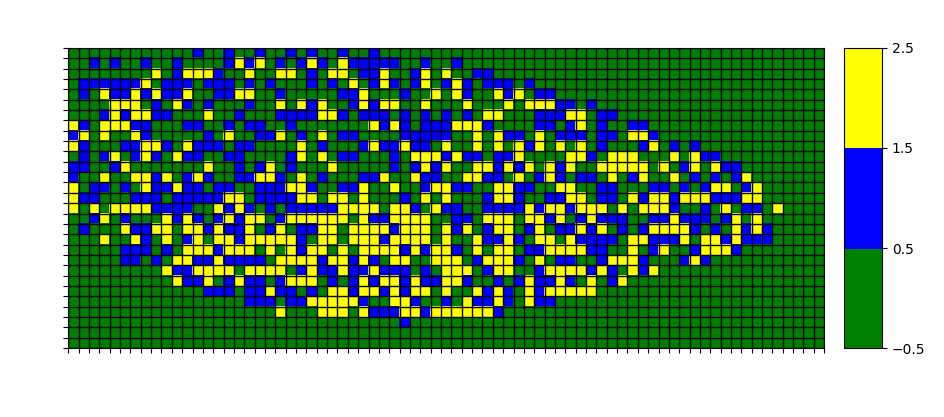
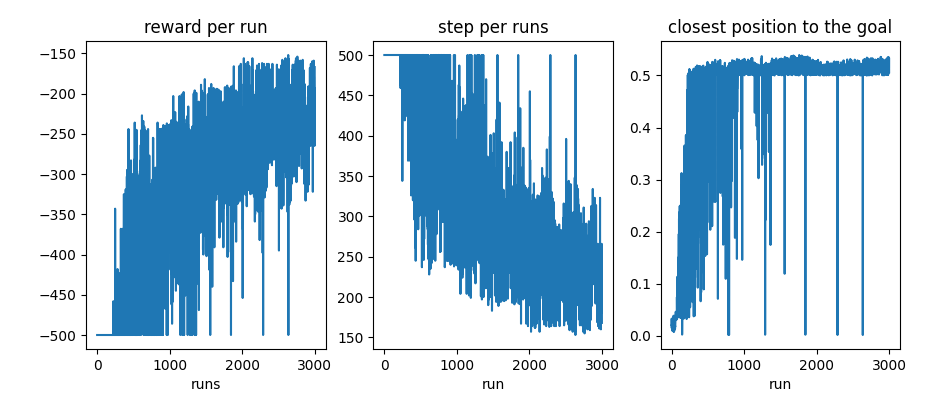
LEVEL\_GRANULARITY=0.001

**INIT**

self.q\_values = np.random.uniform(low = 0, high = 0,

size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))

****

**6.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= last\_best\_state + LEVEL\_GRANULARITY:

return 0

else:

return -1

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

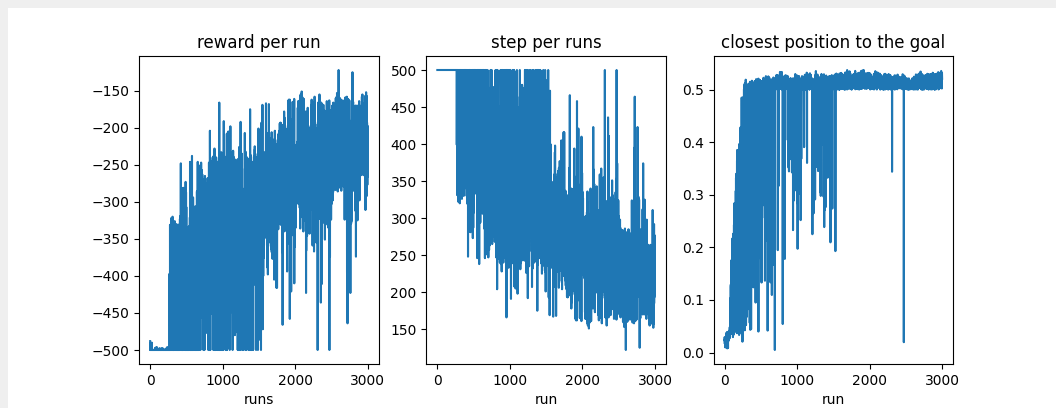
LEVEL\_GRANULARITY=0.01

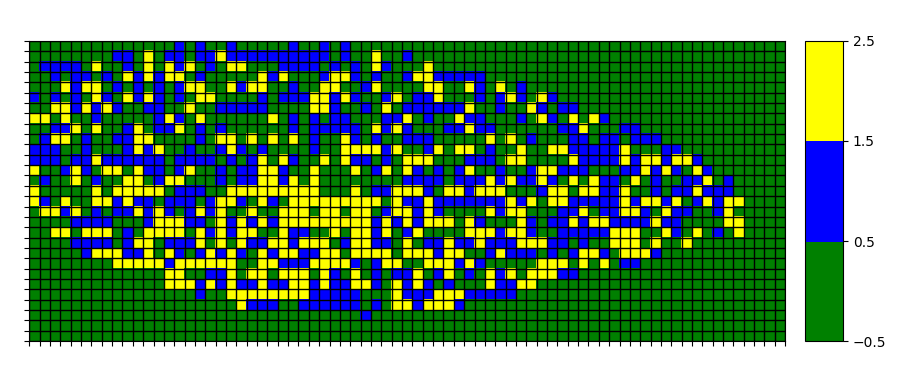
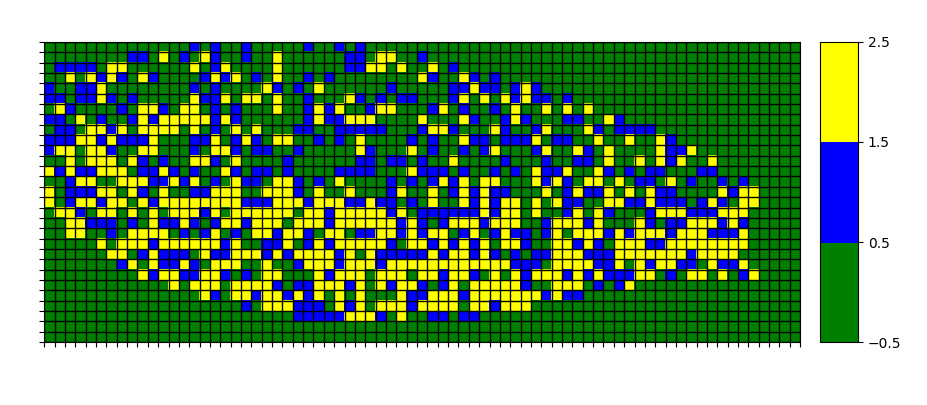
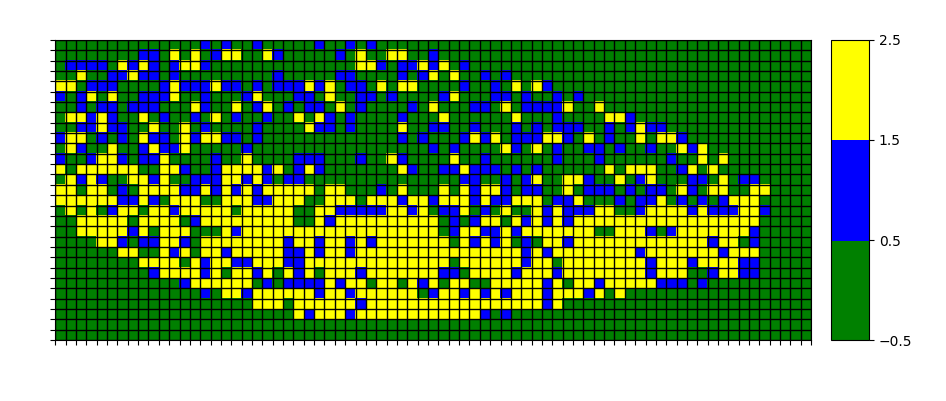
**INIT**

self.q\_values = np.random.uniform(low = 0, high = 0,

size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))



**7.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= 0.5

return 0

else:

return -1

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

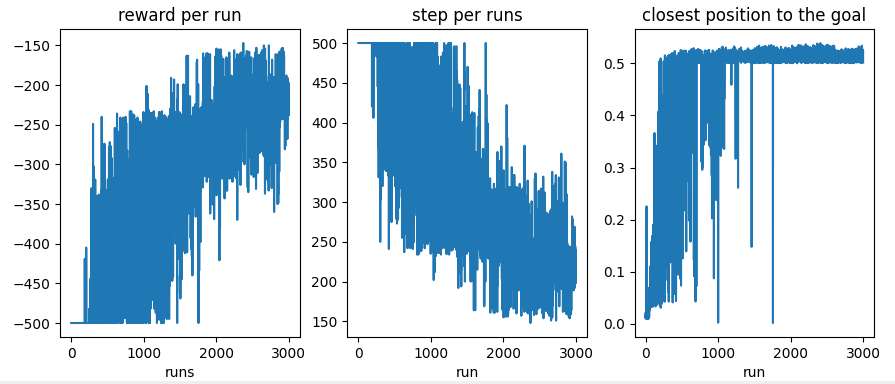
LEVEL\_GRANULARITY=0.01

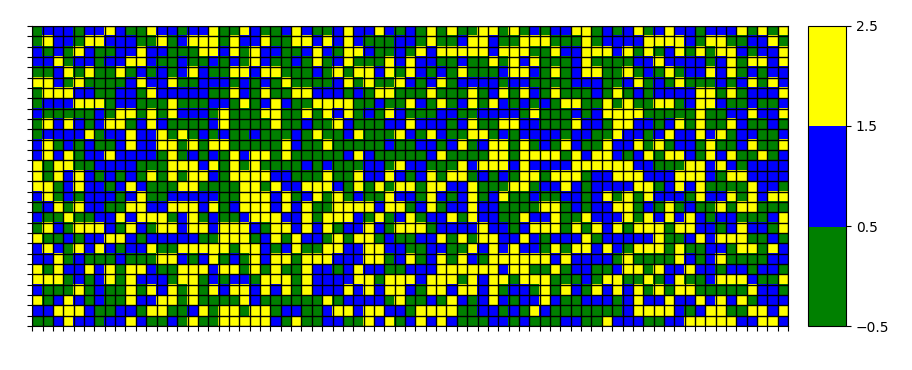
**INIT**

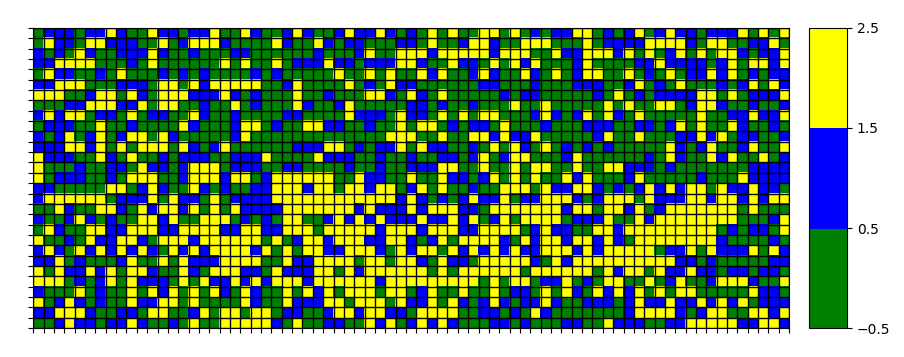
self.q\_values = np.random.uniform(low = -1, high = 1,

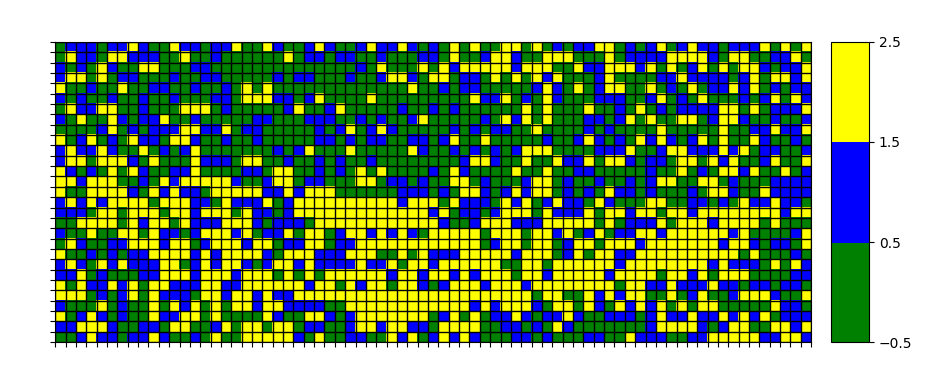
size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))









**8.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= 0.5:

return 1

else:

return 0

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

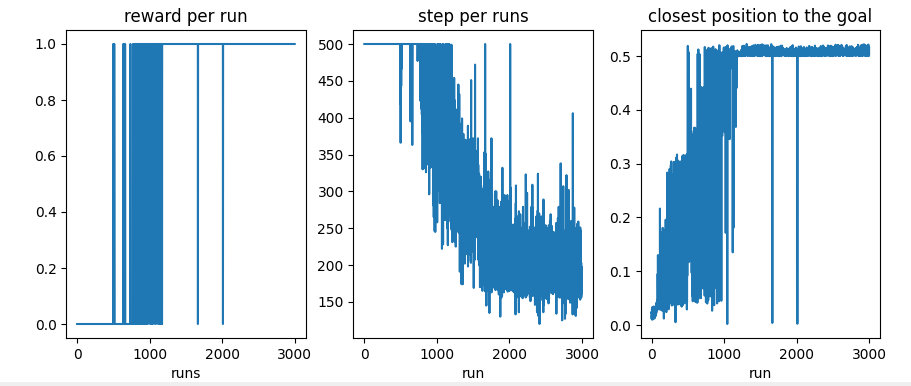
LEVEL\_GRANULARITY=0.01

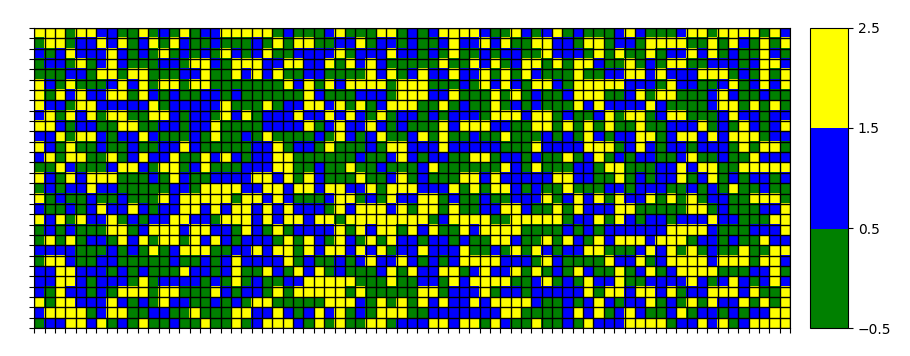
**INIT**

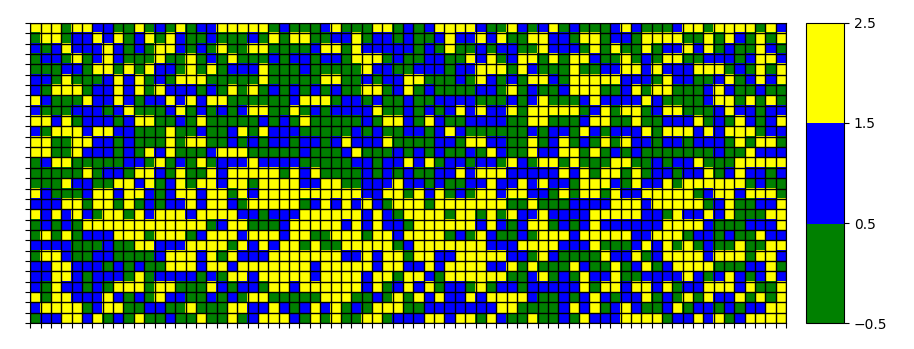
self.q\_values = np.random.uniform(low = -1, high = 1,

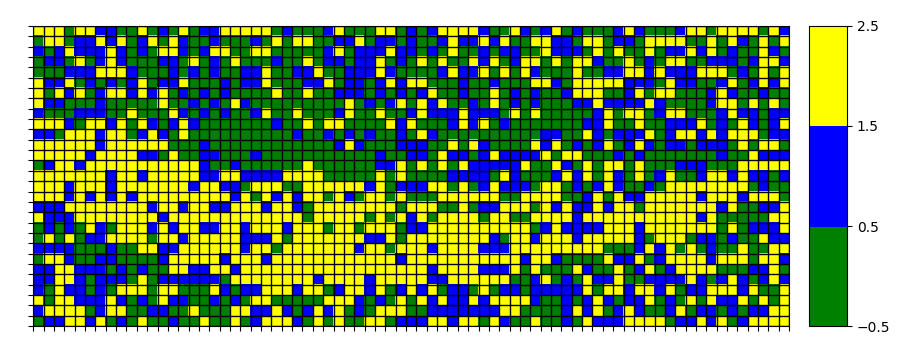
size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))

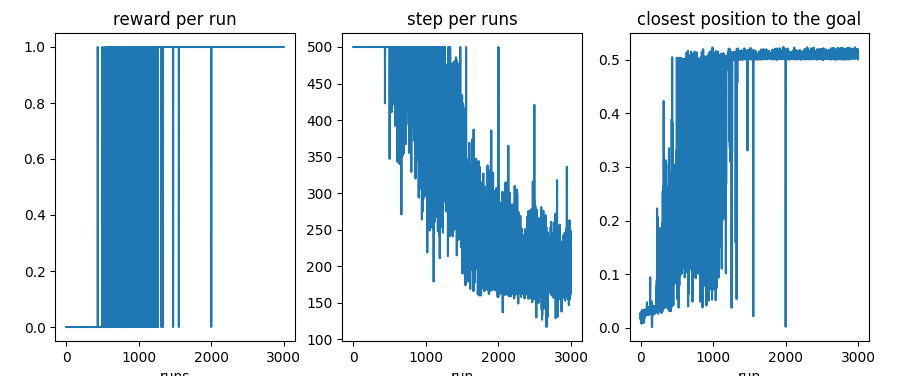


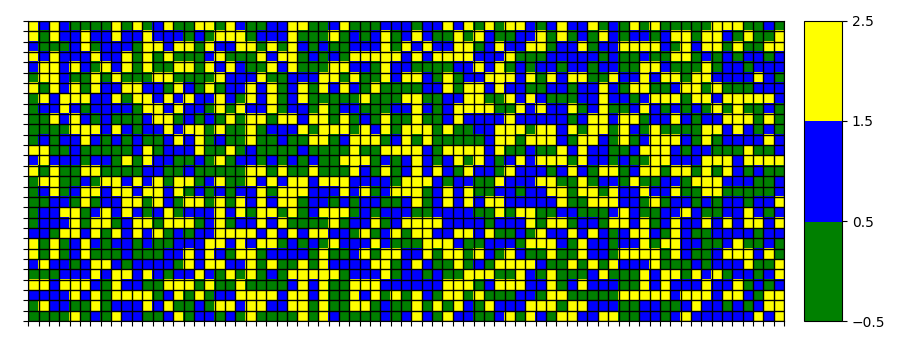


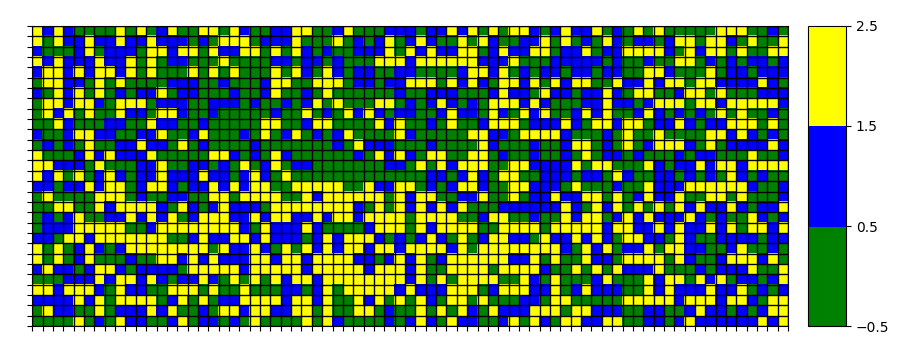


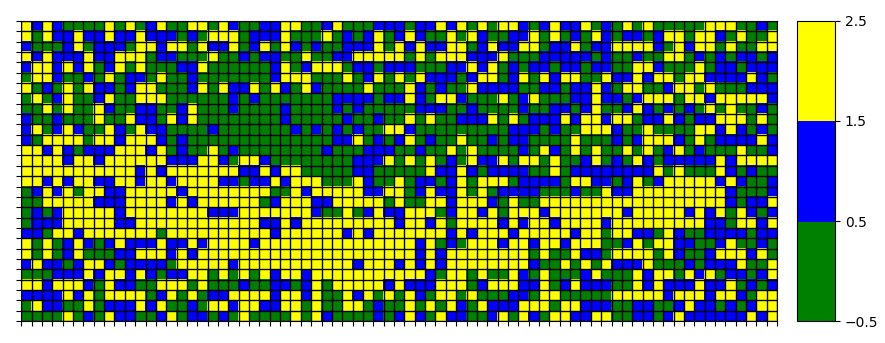


**This configuration has been executed twice:**









**IMPORTANCE OF INITIALIZATION VALUES IN SOME ALGORITHM CONFIGURATIONS**

**9.**

**REWARD**

def get\_reward(state, step, last\_best\_state):

if state[0] >= 0.5:

return 1

else:

return 0

return 0

**HYPERPARAMETERS**

MAX\_RUNS=3000

MAXIMUM\_STEPS=500

EXPLORATION\_STEPS\_PER\_STATE=100

INTERPOLATION=MAX\_RUNS/10

ENV\_NAME = "MountainCar-v0"

GAMMA = 0.95

LEARNING\_RATE = 0.2

EXPLORATION\_MAX = 1.0

EXPLORATION\_MIN = 0.1

EXPLORATION\_DECAY = 0.99995

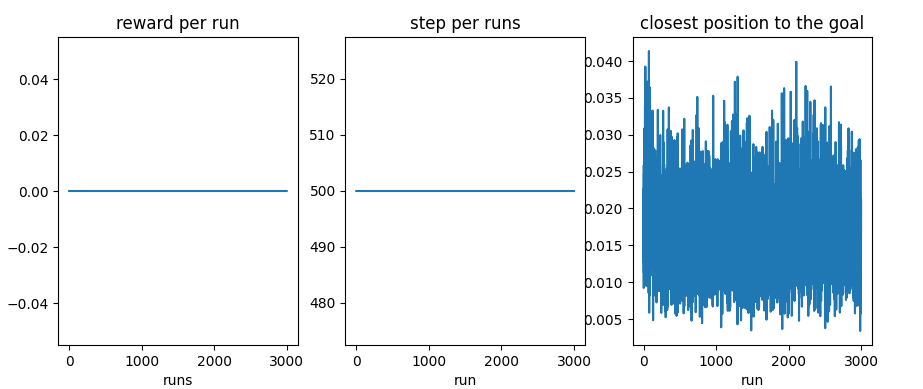
LEVEL\_GRANULARITY=0.01

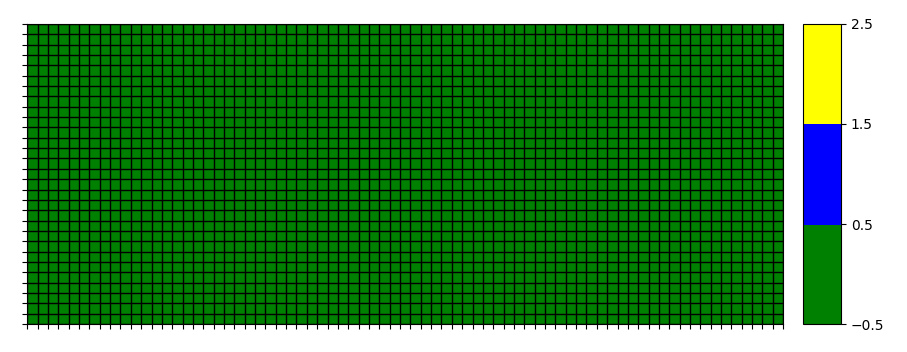
**INIT**

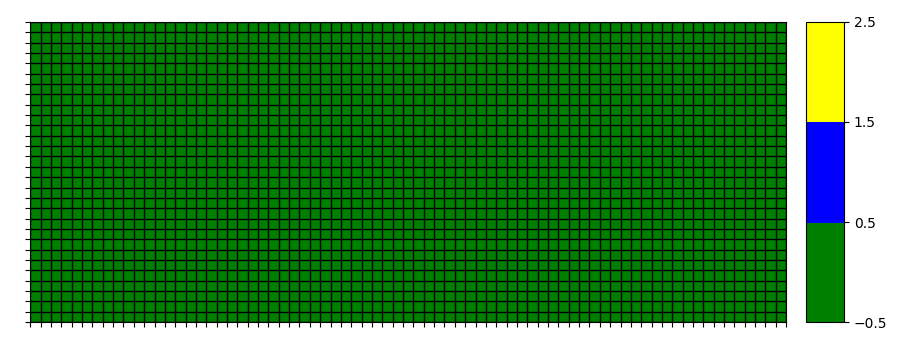
self.q\_values = np.random.uniform(low = 0, high = 0,

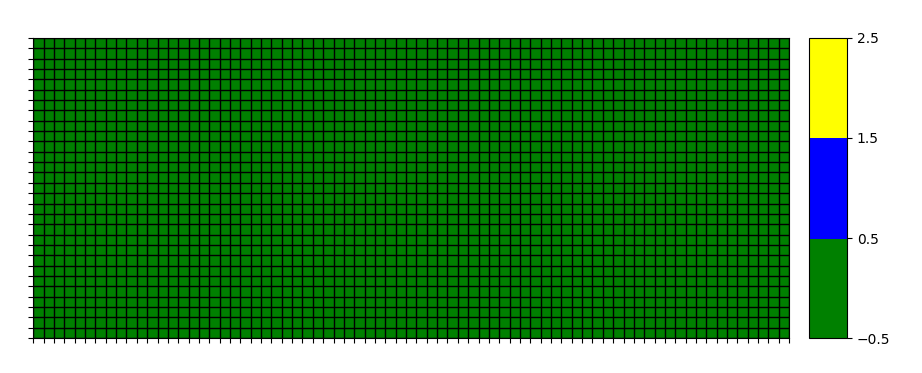
size = (self.num\_states[0], self.num\_states[1],

env.action\_space.n))









**This configuration has been executed twice with exact same result.**

**Since all Q and rewards are 0 and the actions selection is not chosen randomly when tied up, the matrix is never updated and the actions to take is always -1 (left).**

**If this return is randomized in the following way:**

def act(self, state, occurrences):

if np.random.rand() < self.exploration\_rate:

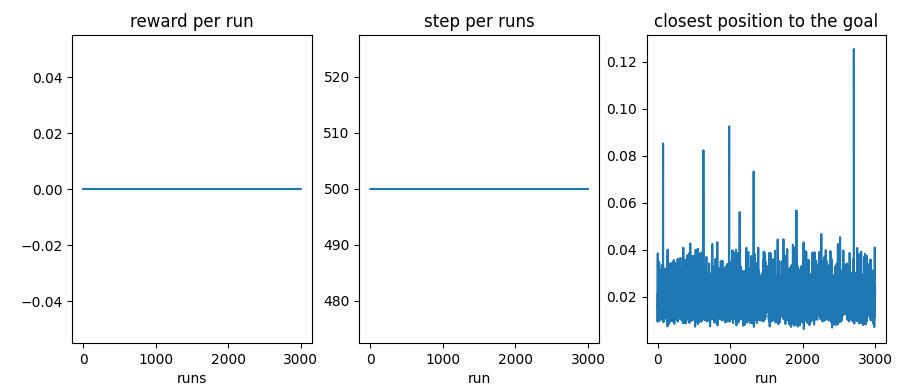
return random.randrange(self.action\_space)

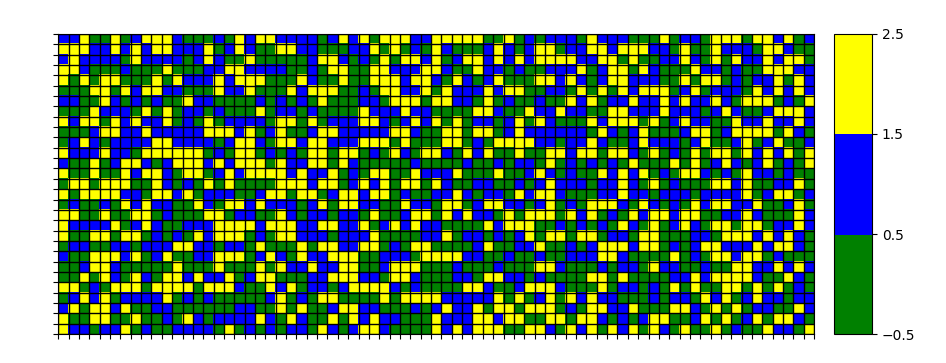
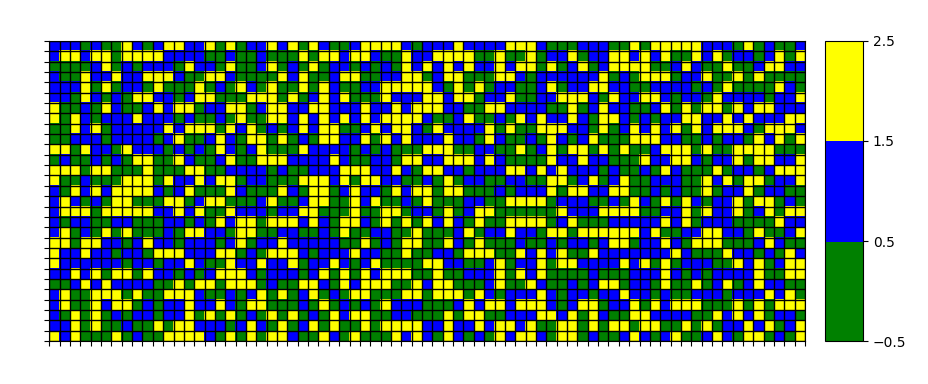
untied\_matrix=np.random.choice(np.flatnonzero(qsolver.q\_values[i][j] == qsolver.q\_values[i][j].max()))

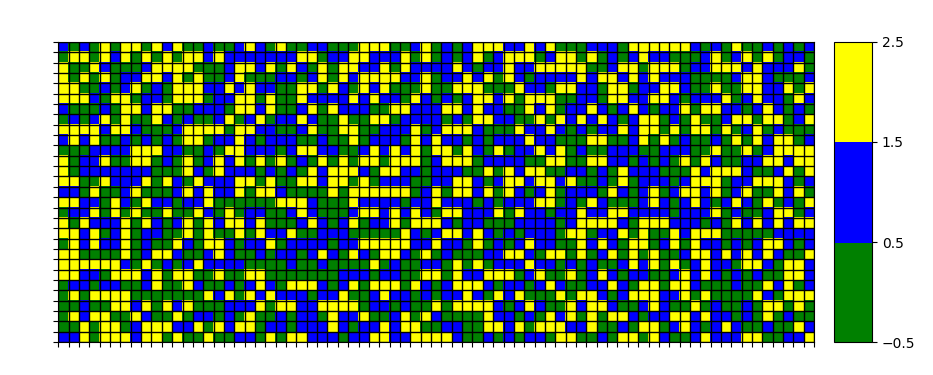
action=untied\_matrix

return action

**the obtained results are still really poor:**

****

****

****

**Note that this is not the case when using exactly the same configuration (including breaking ties up modification) but with a randomly q\_values initialization between -1 and 1:**

