REINFORCEMENT LEARNING (Q - LEARNING)

Nama: Chlaudiah Julinar
NIM: 1301150434
Kelas: IF 39 10

A. LAPORAN

1. Analisis Masalah

Masalah:
Bangunlah sebuah sistem Q-learning untuk menemukan optimum policy sehingga Agent yang berada di posisi Start (1,1) mampu menemukan Goal yang berada di posisi (10,10) dengan mendapatkan Total Reward maksimum pada grid world dalam Figure 1 berikut ini. Data pada Figure 1 dapat dilihat di file DataTugasML3.txt. Pada kasus ini, Agent hanya bisa melakukan empat aksi: N, E, S, dan W yang secara berurutan menyatakan North (ke atas), East (ke kanan), South (ke bawah), dan West (ke kiri). Anda boleh menggunakan skema apapun dalam mengimplementasikan sebuah episode.
Keterangan Gambar: Berikut merupakan grid world yang diberikan berukuran 10 x 10, dimana angka-angka didalam kotak menyatakan *reward*,*agen* berada pada *Initial State* = (1,1) dan berakhir pada *Goal State* = (10,10)

Analisis Masalah:
Masalah yang diharus diselesaikan disini yaitu, menemukan *Total Reward* terbaik yang

dihasilkan selama *agent* melakukan learning yaitu berupa cara menuju *Initial State* yaitu (1,1) hingga sampai pada *Goal State* yaitu (10,10). Setiap kali, *agent* selesai melakukan satu kali perjalanan, maka akan dihitung score yang diperoleh dari tiap reward yang didapat selama *agent* melakukan *action* yaitu = up, down, lef, dan right.

Sebelum membuat *agent* melakukan learning, terlebih dahulu kita harus menyediakan atau mendefinisikan *environment* dari jalur yang akan dilalui oleh *agent*. Desain dari *environment* akan dijelaskan pada tahap selanjutnya.

2. Desain Environment

Desain Environment yaitu tahap dimana kita akan mengatur *action* dari *agent* sehingga *agent* dapat memutuskan harus bergerak kearah mana dari *action* yang diberikan. Lalu, kita juga harus mengatur berapa nilai *reward* yang diperoleh jika *agent* melakukan salah satu *action*.

Mengatur Gerak dari Agent

- Saat aksi yang dipilih adalah 0 maka agent akan bergerak kearah atas, artinya nilai dari koordinat y akan bertambah 1 tiap kali agent bergerak ke atas selama belum lebih dari y = 10
- Saat aksi yang dipilih adalah 1 maka agent akan bergerak kearah bawah, artinya nilai dari koordinat y akan berkurang 1 tiap kali agent bergerak kebawah selama belum kurang dari y = 0
- Saat aksi yang dipilih adalah 2 maka agent akan bergerak kearah kanan, artinya nilai dari koordinat x akan bertambah 1 tiap kali agent bergerak selama belum lebih dari x = 10
- Saat aksi yang dipilih adalah 3 maka agent akan bergerak kearah atas, artinya nilai dari koordinat x akan berkurang 1 tiap kali agent bergerak selama belum kurang dari x = 0

Mengisi Nilai Reward dari tiap Aksi yang dilakukan

- Saat aksi yang dipilih adalah 0 maka agent akan bergerak kearah atas, maka reward yang diperoleh adalah saat agent berada pada koordinat y > 0 dan nilai y akan

ditambahkan 1 tiap kali agent bergerak selama belum lebih dari y = 10 dan nilai x tetap

- Saat aksi yang dipilih adalah 1 maka agent akan bergerak kearah bawah,maka reward yang diperoleh adalah saat agent berada pada koordinat y < 10 dan nilai y akan dikurangkan 1 tiap kali agent bergerak selama belum kurang dari y = 0 dan nilai x tetap
- Saat aksi yang dipilih adalah 2 maka agent akan bergerak kearah kanan, maka reward yang diperoleh adalah saat agent berada pada koordinat x < 10 dan nilai x akan ditambahkan 1 tiap kali agent bergerak selama belum lebih dari x = 10 dan nilai y tetap
- Saat aksi yang dipilih adalah 3 maka agent akan bergerak kearah atas, maka reward yang diperoleh adalah saat agent berada pada koordinat x > 0 dan nilai x akan dikurangkan 1 tiap kali agent bergerak selama belum kurang dari x = 0 dan nilai y tetap

B. LANGKAH - LANGKAH PENYELESAIAN Q-LEARNING

1. Import Library

In [1]:
```
import numpy as np
import random
```

2. Membuat Fungsi Utama untuk Update Q-table dengan Bellman Equation

In [2]:
```
def bellman(loc,dec,gamma,q):
    a = (loc[0],loc[1])
    hasil1 = reward_table[a][dec]
    d0 = q[(loc[0],loc[1])][0]
    d1 = q[(loc[0],loc[1])][1]
    d2 = q[(loc[0],loc[1])][2]
    d3 = q[(loc[0],loc[1])][3]
    hasil2 = max(d0, d1, d2, d3)
```

```
    hasil = hasil1 + (gamma * hasil2)
    return hasil
```

3. Mengatur Action dari Agent

In [3]:
```
def move(loc,dec):
    if dec == 0 :
        if loc[1] > 0 :
            loc[1] = loc[1] - 1
    elif dec == 1 :
        if loc[1] < 9 :
            loc[1] = loc[1] + 1
    elif dec == 2 :
        if loc[0] < 9 :
            loc[0] = loc[0] + 1
    elif dec == 3 :
        if loc[0] > 0 :
            loc[0] = loc[0] - 1
```

4. Mengatur Nilai Reward dari Tiap Decision

In [4]:
```
def r_up(loc):
    global reward
    if loc[1] > 0 :
        y = loc[1] - 1
    else :
        return 0
    return reward[loc[0]][y]

def r_down(loc):
    global reward
    if loc[1] < 9 :
        y = loc[1] + 1
    else :
        return 0
```

```
    return reward[loc[0]][y]

def r_right(loc):
    global reward
    if loc[0] < 9 :
        x = loc[0] + 1
    else :
        return 0
    return reward[x][loc[1]]

def r_left(loc):
    global reward
    if loc[0] > 0 :
        x = loc[0] - 1
    else :
        return 0
    return reward[x][loc[1]]
```

5. Mempersiapkan Tiap Variable yang Dibutuhkan dan Loading Data File

In [5]:
```
start = [9,0]
x = 10
y = 10
finish = [0,9]
initial = [0,0,0,0]
q_table = {}
reward = np.loadtxt("DataTugasML3.txt")
states = []
reward_table = {}
maks = -99999
gamma = 0.5

print(reward)
```

```
[[ -1.  -3.  -5.  -1.  -3.  -3.  -5.  -5.  -1. 100.]
 [ -2.  -1.  -1.  -4.  -2.  -5.  -3.  -5.  -5.  -5.]
```

```
[ -3.  -4.  -4.  -1.  -3.  -5.  -5.  -4.  -3.  -5.]
[ -3.  -5.  -2.  -5.  -1.  -4.  -5.  -1.  -3.  -4.]
[ -4.  -3.  -3.  -2.  -1.  -1.  -1.  -4.  -3.  -4.]
[ -4.  -2.  -5.  -2.  -4.  -5.  -1.  -2.  -2.  -4.]
[ -4.  -3.  -2.  -3.  -1.  -3.  -4.  -3.  -1.  -3.]
[ -4.  -2.  -5.  -4.  -1.  -4.  -5.  -5.  -2.  -4.]
[ -2.  -1.  -1.  -4.  -1.  -3.  -5.  -1.  -4.  -1.]
[ -5.  -3.  -1.  -2.  -4.  -3.  -5.  -2.  -2.  -2.]]
```

6. Menginputkan Koordinat X dan Y sebagai State

In [6]:
```
for i in range(x):
    for j in range(y):
        states.append((i,j))
```

7. Membuat Reward Table berukuran 100 x 4

In [7]:
```
for i in range(x*y) :
    reward_table[states[i]] = initial
print(reward_table)
```

```
{(0, 0): [0, 0, 0, 0], (0, 1): [0, 0, 0, 0], (0, 2): [0, 0, 0, 0], (0,
3): [0, 0, 0, 0], (0, 4): [0, 0, 0, 0], (0, 5): [0, 0, 0, 0], (0, 6):
[0, 0, 0, 0], (0, 7): [0, 0, 0, 0], (0, 8): [0, 0, 0, 0], (0, 9): [0,
0, 0, 0], (1, 0): [0, 0, 0, 0], (1, 1): [0, 0, 0, 0], (1, 2): [0, 0, 0,
0], (1, 3): [0, 0, 0, 0], (1, 4): [0, 0, 0, 0], (1, 5): [0, 0, 0, 0],
(1, 6): [0, 0, 0, 0], (1, 7): [0, 0, 0, 0], (1, 8): [0, 0, 0, 0], (1,
9): [0, 0, 0, 0], (2, 0): [0, 0, 0, 0], (2, 1): [0, 0, 0, 0], (2, 2):
[0, 0, 0, 0], (2, 3): [0, 0, 0, 0], (2, 4): [0, 0, 0, 0], (2, 5): [0,
0, 0, 0], (2, 6): [0, 0, 0, 0], (2, 7): [0, 0, 0, 0], (2, 8): [0, 0, 0,
0], (2, 9): [0, 0, 0, 0], (3, 0): [0, 0, 0, 0], (3, 1): [0, 0, 0, 0],
(3, 2): [0, 0, 0, 0], (3, 3): [0, 0, 0, 0], (3, 4): [0, 0, 0, 0], (3,
5): [0, 0, 0, 0], (3, 6): [0, 0, 0, 0], (3, 7): [0, 0, 0, 0], (3, 8):
[0, 0, 0, 0], (3, 9): [0, 0, 0, 0], (4, 0): [0, 0, 0, 0], (4, 1): [0,
0, 0, 0], (4, 2): [0, 0, 0, 0], (4, 3): [0, 0, 0, 0], (4, 4): [0, 0, 0,
0], (4, 5): [0, 0, 0, 0], (4, 6): [0, 0, 0, 0], (4, 7): [0, 0, 0, 0],
```

```
(4, 8): [0, 0, 0, 0], (4, 9): [0, 0, 0, 0], (5, 0): [0, 0, 0, 0], (5,
1): [0, 0, 0, 0], (5, 2): [0, 0, 0, 0], (5, 3): [0, 0, 0, 0], (5, 4):
[0, 0, 0, 0], (5, 5): [0, 0, 0, 0], (5, 6): [0, 0, 0, 0], (5, 7): [0,
0, 0, 0], (5, 8): [0, 0, 0, 0], (5, 9): [0, 0, 0, 0], (6, 0): [0, 0, 0,
0], (6, 1): [0, 0, 0, 0], (6, 2): [0, 0, 0, 0], (6, 3): [0, 0, 0, 0],
(6, 4): [0, 0, 0, 0], (6, 5): [0, 0, 0, 0], (6, 6): [0, 0, 0, 0], (6,
7): [0, 0, 0, 0], (6, 8): [0, 0, 0, 0], (6, 9): [0, 0, 0, 0], (7, 0):
[0, 0, 0, 0], (7, 1): [0, 0, 0, 0], (7, 2): [0, 0, 0, 0], (7, 3): [0,
0, 0, 0], (7, 4): [0, 0, 0, 0], (7, 5): [0, 0, 0, 0], (7, 6): [0, 0, 0,
0], (7, 7): [0, 0, 0, 0], (7, 8): [0, 0, 0, 0], (7, 9): [0, 0, 0, 0],
(8, 0): [0, 0, 0, 0], (8, 1): [0, 0, 0, 0], (8, 2): [0, 0, 0, 0], (8,
3): [0, 0, 0, 0], (8, 4): [0, 0, 0, 0], (8, 5): [0, 0, 0, 0], (8, 6):
[0, 0, 0, 0], (8, 7): [0, 0, 0, 0], (8, 8): [0, 0, 0, 0], (8, 9): [0,
0, 0, 0], (9, 0): [0, 0, 0, 0], (9, 1): [0, 0, 0, 0], (9, 2): [0, 0, 0,
0], (9, 3): [0, 0, 0, 0], (9, 4): [0, 0, 0, 0], (9, 5): [0, 0, 0, 0],
(9, 6): [0, 0, 0, 0], (9, 7): [0, 0, 0, 0], (9, 8): [0, 0, 0, 0], (9,
9): [0, 0, 0, 0]}
```

8. Membangun Environtment pada Reward Table

In [8]:
```
for dec in range(0,4):
    for i in range(len(reward)):
        for j in range(len(reward[0])):
            loc = [i,j]
            temp = []
            temp.clear
            temp.append(r_up(loc))
            temp.append(r_down(loc))
            temp.append(r_right(loc))
            temp.append(r_left(loc))
            reward_table[(i,j)] = temp
```

9. Membuat Q Table dengan Value Seluruhnya adalah 0 berukuran 100 x 4

In [9]:
```
for i in range(x*y) :
    q_table[states[i]] = initial
print(q_table)
```

```
{(0, 0): [0, 0, 0, 0], (0, 1): [0, 0, 0, 0], (0, 2): [0, 0, 0, 0], (0,
3): [0, 0, 0, 0], (0, 4): [0, 0, 0, 0], (0, 5): [0, 0, 0, 0], (0, 6):
[0, 0, 0, 0], (0, 7): [0, 0, 0, 0], (0, 8): [0, 0, 0, 0], (0, 9): [0,
0, 0, 0], (1, 0): [0, 0, 0, 0], (1, 1): [0, 0, 0, 0], (1, 2): [0, 0, 0,
0], (1, 3): [0, 0, 0, 0], (1, 4): [0, 0, 0, 0], (1, 5): [0, 0, 0, 0],
(1, 6): [0, 0, 0, 0], (1, 7): [0, 0, 0, 0], (1, 8): [0, 0, 0, 0], (1,
9): [0, 0, 0, 0], (2, 0): [0, 0, 0, 0], (2, 1): [0, 0, 0, 0], (2, 2):
[0, 0, 0, 0], (2, 3): [0, 0, 0, 0], (2, 4): [0, 0, 0, 0], (2, 5): [0,
0, 0, 0], (2, 6): [0, 0, 0, 0], (2, 7): [0, 0, 0, 0], (2, 8): [0, 0, 0,
0], (2, 9): [0, 0, 0, 0], (3, 0): [0, 0, 0, 0], (3, 1): [0, 0, 0, 0],
(3, 2): [0, 0, 0, 0], (3, 3): [0, 0, 0, 0], (3, 4): [0, 0, 0, 0], (3,
5): [0, 0, 0, 0], (3, 6): [0, 0, 0, 0], (3, 7): [0, 0, 0, 0], (3, 8):
[0, 0, 0, 0], (3, 9): [0, 0, 0, 0], (4, 0): [0, 0, 0, 0], (4, 1): [0,
0, 0, 0], (4, 2): [0, 0, 0, 0], (4, 3): [0, 0, 0, 0], (4, 4): [0, 0, 0,
0], (4, 5): [0, 0, 0, 0], (4, 6): [0, 0, 0, 0], (4, 7): [0, 0, 0, 0],
(4, 8): [0, 0, 0, 0], (4, 9): [0, 0, 0, 0], (5, 0): [0, 0, 0, 0], (5,
1): [0, 0, 0, 0], (5, 2): [0, 0, 0, 0], (5, 3): [0, 0, 0, 0], (5, 4):
[0, 0, 0, 0], (5, 5): [0, 0, 0, 0], (5, 6): [0, 0, 0, 0], (5, 7): [0,
0, 0, 0], (5, 8): [0, 0, 0, 0], (5, 9): [0, 0, 0, 0], (6, 0): [0, 0, 0,
0], (6, 1): [0, 0, 0, 0], (6, 2): [0, 0, 0, 0], (6, 3): [0, 0, 0, 0],
(6, 4): [0, 0, 0, 0], (6, 5): [0, 0, 0, 0], (6, 6): [0, 0, 0, 0], (6,
7): [0, 0, 0, 0], (6, 8): [0, 0, 0, 0], (6, 9): [0, 0, 0, 0], (7, 0):
[0, 0, 0, 0], (7, 1): [0, 0, 0, 0], (7, 2): [0, 0, 0, 0], (7, 3): [0,
0, 0, 0], (7, 4): [0, 0, 0, 0], (7, 5): [0, 0, 0, 0], (7, 6): [0, 0, 0,
0], (7, 7): [0, 0, 0, 0], (7, 8): [0, 0, 0, 0], (7, 9): [0, 0, 0, 0],
(8, 0): [0, 0, 0, 0], (8, 1): [0, 0, 0, 0], (8, 2): [0, 0, 0, 0], (8,
3): [0, 0, 0, 0], (8, 4): [0, 0, 0, 0], (8, 5): [0, 0, 0, 0], (8, 6):
[0, 0, 0, 0], (8, 7): [0, 0, 0, 0], (8, 8): [0, 0, 0, 0], (8, 9): [0,
0, 0, 0], (9, 0): [0, 0, 0, 0], (9, 1): [0, 0, 0, 0], (9, 2): [0, 0, 0,
0], (9, 3): [0, 0, 0, 0], (9, 4): [0, 0, 0, 0], (9, 5): [0, 0, 0, 0],
(9, 6): [0, 0, 0, 0], (9, 7): [0, 0, 0, 0], (9, 8): [0, 0, 0, 0], (9,
9): [0, 0, 0, 0]}
```

10. Q Learning

In [11]:
```
for i in range(0,100):
    score = 0
    q = q_table
    end = False
    loc = [9,0]
    while end == False:
        # up = 0 , down = 1 , right = 2 ,left = 3
        dec = random.randint(0,3)
        bell = bellman(loc,dec,gamma,q)
        q[(loc[0],loc[1])][dec] = bell
        if dec == 0 :
            move(loc,dec)
            score = score + r_up(loc)
        elif dec == 1:
            move(loc,dec)
            score = score + r_down(loc)
        elif dec == 2:
            move(loc,dec)
            score = score + r_right(loc)
        elif dec == 3 :
            move(loc,dec)
            score = score + r_left(loc)

        if loc[0] == finish[0] and loc[1] == finish[1] :
            end = True
    print("score = ",score)
    if maks < score :
        maks = score
    q_table = q

print()
print("Maximum: ",maks)
```

```
score =  -440.0
score =  -583.0
score =  -723.0
score =  -859.0
score =  -673.0
score =  -630.0
```

```
score =  -623.0
score =  -862.0
score =  -450.0
score =  -2568.0
score =  -2257.0
score =  -898.0
score =  -625.0
score =  -2530.0
score =  -1590.0
score =  -394.0
score =  -366.0
score =  -1153.0
score =  -141.0
score =  -98.0
score =  -1238.0
score =  -1036.0
score =  -1873.0
score =  -1451.0
score =  -2399.0
score =  -778.0
score =  -3658.0
score =  -4017.0
score =  -1319.0
score =  -480.0
score =  -581.0
score =  -770.0
score =  -1369.0
score =  -1961.0
score =  -502.0
score =  -695.0
score =  -190.0
score =  -1322.0
score =  -2697.0
score =  -3120.0
score =  -1469.0
score =  -2790.0
score =  -3038.0
score =  -187.0
score =  -775.0
```

```
score =  -2487.0
score =  -1050.0
score =  -1238.0
score =  -3356.0
score =  -13.0
score =  -2749.0
score =  -1229.0
score =  -639.0
score =  -331.0
score =  -5560.0
score =  -1502.0
score =  -2262.0
score =  -2988.0
score =  -1662.0
score =  -635.0
score =  -2873.0
score =  -806.0
score =  -647.0
score =  -484.0
score =  -14.0
score =  -918.0
score =  -845.0
score =  -1835.0
score =  -6859.0
score =  -570.0
score =  -811.0
score =  -90.0
score =  -473.0
score =  48.0
score =  -2257.0
score =  -96.0
score =  -974.0
score =  -394.0
score =  -803.0
score =  -3138.0
score =  -1441.0
score =  -1569.0
score =  -3419.0
score =  -562.0
```

```
score =  -5137.0
score =  -1973.0
score =  -841.0
score =  -216.0
score =  -1829.0
score =  -331.0
score =  -362.0
score =  -148.0
score =  -1374.0
score =  -228.0
score =  -1977.0
score =  -491.0
score =  -873.0
score =  -1484.0
score =  -1805.0
score =  -275.0

Maximum:  48.0
```

C. HASIL EKSPERIMEN

Hasil eksperimen yang diperoleh adalah nilai total reward paling bagus selama melakukan eksperimen Q-Learning pada *agent* adalah 48.0 dengan percobaan melakukan episode sebanyak 100 kali episode. Hal ini menyatakan bahwa, *agent* dengan 100 kali episode belum cukup baik memahami environment yang dibuat.

Hal ini terjadi dikarenakan beberapa kemungkinan yang saya lakukan pada pemrograman. Beberapa hal tersebut saya perkirakan adalah:

1. Aksi yang akan dilakukan diputuskan secara random (dapat dilihat pada bagian 10. Q-Learning)
2. Pada saat perhitungan score dengan reward yang diperoleh (dapat dilihat pada bagian 10. Q-Learning)

3. Pada penentuan reward pada reward table sebagai environment yang harus dipelajari oleh *agent* (dapat dilihat pada bagian 8. Membangun environment pada reward table)