**Machine Learning**

**Tugas 3 : Q-learning**

****

Agung Nursatria Banyuwiguna

1301150073

IF-39-03

Program Studi S1 Teknik Informatika – Fakultas Informatika

Universitas Telkom

Jalan Telekomunikasi Terusan Buah Batu, Bandung

Indonesia

# Problem Statement

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.

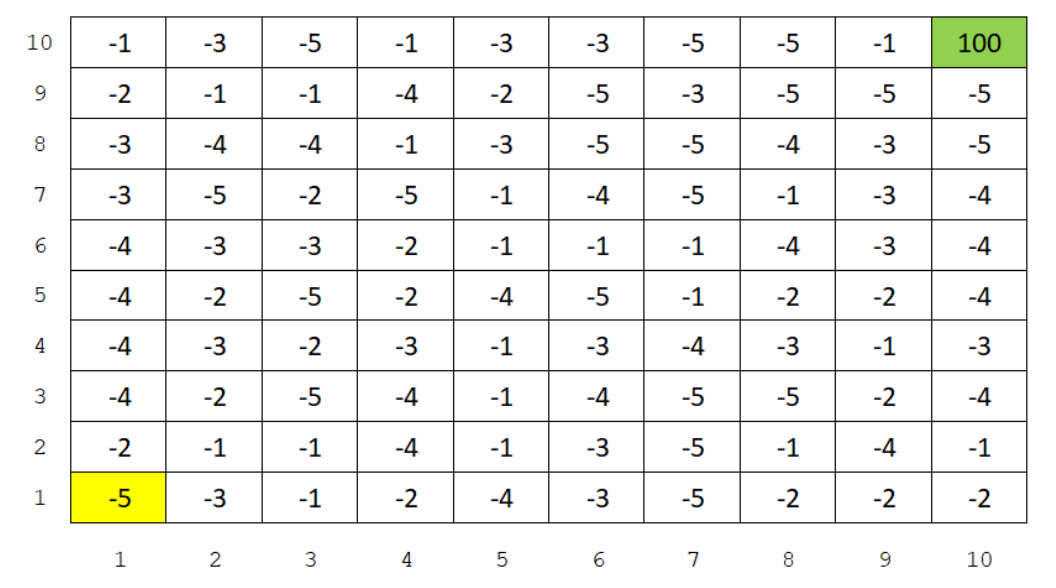


Figure 1: Sebuah grid world ukuran 10 x 10, di mana angka-angka dalam kotak menyatakan reward. Agent berada di posisi Start (1,1) dan Goal di posisi (10,10)

# Langkah Pengerjaan

* Learning

The Q-Learning algorithm goes as follows:

1. Set the gamma parameter, and environment rewards in matrix R.

2. Initialize matrix Q to zero.

3. For each episode:

Select a random initial state.

Do While the goal state hasn't been reached.

* Select one among all possible actions for the current state.
* Using this possible action, consider going to the next state.
* Get maximum Q value for this next state based on all possible actions.
* Compute: Q(state, action) = R(state, action) + Gamma \* Max[Q(next state, all actions)]
* Set the next state as the current state.

End Do

End For

* Running/Testing

Algorithm to utilize the Q matrix:

1. Set current state = initial state.

2. From current state, find the action with the highest Q value.

3. Set current state = next state.

4. Repeat Steps 2 and 3 until current state = goal state.

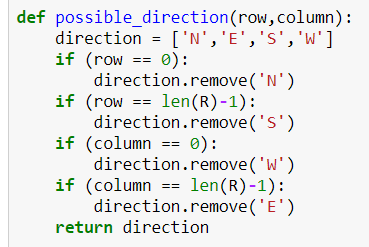
# Screenshot Program

* Library / Dependencies:

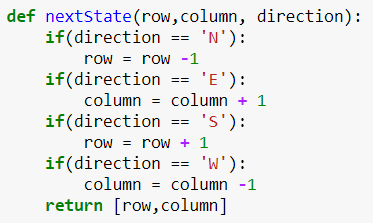


* Function

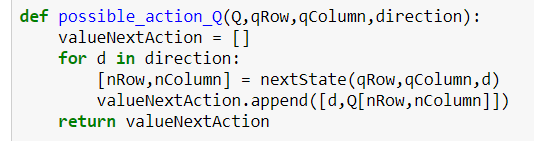
1. Fungsi **possible\_direction** ini berguna untuk menentukan arah (*direction*) mana saja yang bisa diambil untuk berpindah tempat



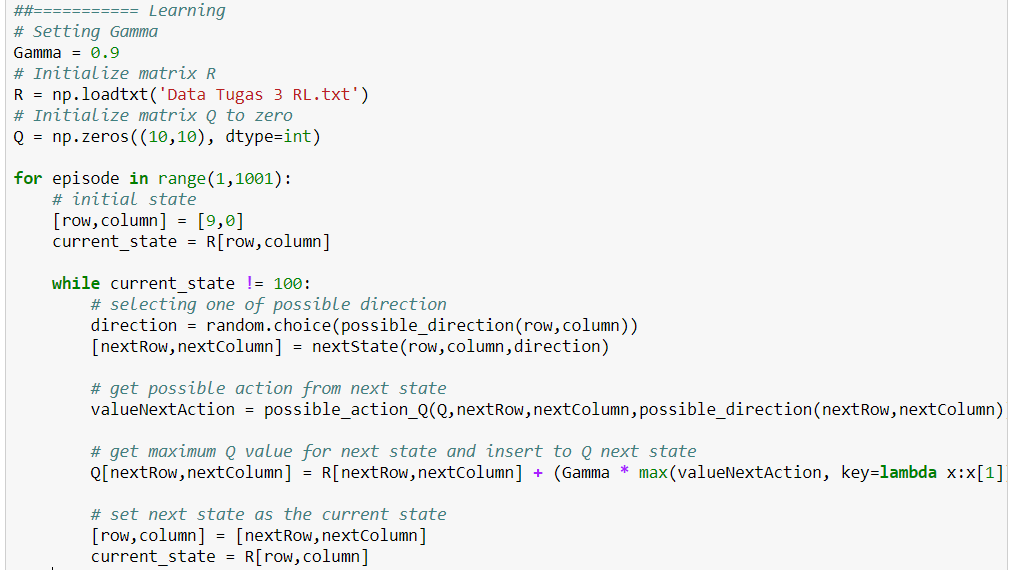
1. Fungsi **nextState** berguna untuk berpindah tempat dari suatu state ke state lain



1. Fungsi **possible\_action\_Q** berguna untuk mencari aksi yang dapat diambil berdasarkan Q



* Learning



Hal yang dilakukan pada tahapan learning yaitu :

1. Setting gamma, dengan rentang nilai 0 <= gamma > 1. Semakin mendekati 0, maka semakin tidak bergantung future reward dan semakin bergantung pada immediate reward, dan sebaliknya. Disini penulis menggunakan gamma 0.9.
2. Me-*load* data matrix reward dari file txt ke dalam variable R.
3. Membuat matrix Q dengan ukuran 10x10.
4. Melakukan learning sebanyak 1000 episode, dengan 1 episode yaitu terjadinya pergerakan dari state awal ke goal state yang mana goal state ditunjuk berupa elemen pada matrix R dengan reward = 100. Yang dapat dilihat pada langkah nomor 5 – 9.
5. Memilih secara acak state selanjutnya dari state saat ini.
6. Mencari daftar aksi yang dapat dilakukan dari state selanjutnya.
7. Memilih aksi terbaik berdasarkan maximum Q dari aksi yang dapat dilakukan dari state selanjutnya.
8. Berpindah ke state selanjutnya.
9. Ulangi kembali dari langkah 5 sampai state saat ini = goal state.

* Running

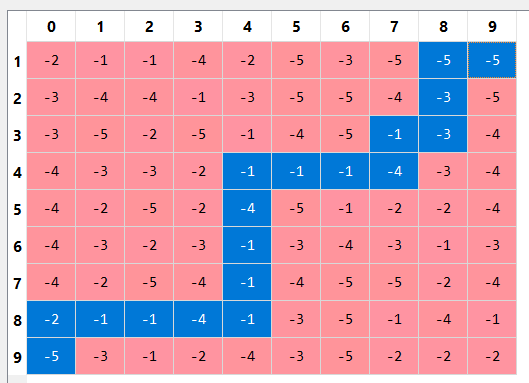


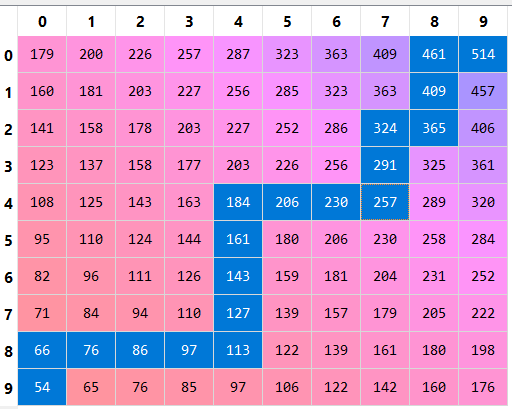
Hal yang dilakukan pada running yaitu :

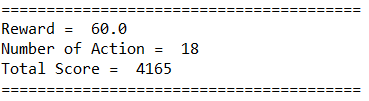
1. Input posisi awal, dan jadikan state saat ini (current state), juga dapatkan score dan reward dari posisi tersebut
2. Lakukan looping hingga state saat ini = goal state yang bernilai 100
3. Dapatkan daftar state kemana saja dari posisi sekarang yang bisa dipilih
4. Pilih state terbaik berdasakan possible action Q terbaik untuk menjadi state selanjutnya
5. State selanjutnya dijadikan state saat ini
6. ulangi dari langkah 3 hingga state saat ini = goal state
7. keluarkan hasil running

# Evaluasi Hasil Running Program

Berikut daftar perpindahan state yang dilakukan oleh program untuk mencapai goal state







Dari hasil running program, didapat reward sebesar **60** dengan jumlah aksi berpindah tempat sebanyak **18x** dan total score Q dari start state ke goal state sebesar **4165**.

# Referensi

<http://mnemstudio.org/path-finding-q-learning-tutorial.htm>