

# Markov Decision Process and Dynamic Programming

Tags

## Learnig Objectives

1. Dapat mendeskripsikan Markov Decision Proses (MDP)
2. Dapat menyusun MDP dalam merumuskan RL secara formal
3. Mampu menentukan value sebuah policy.
4. Mampu membuat dynamic programming tipe value iteration.

Definisi Machine Learning (Supervised Learning, Un-Supervised Learning dan Reinforcement Learning), dan contoh aplikasinya dalam berbagai bidang

## Apa itu Markov Decision Process?

MDP merupakan formulasi matematis dari permasalahan RL.

MDP adalah proses pengambilan keputusan yang mengacu pada teori

Markov yang melahirkan Markov Properti

## Markov Property

Markov Decision Process merupakan sebuah tuple  $(S, A, P, R, \gamma)$ .

Dimana:

- $S$  merupakan state
- $A$  merupakan action
- $P$  merupakan state transition probability function (transition probability)
- $R$  merupakan reward function

- $\gamma$  merupakan discount factor ( $\gamma \in [0, 1]$ )

| The future is independent of the past given the present

Sebuah keadaan  $S_t$  bisa disebut *Markov* jika dan hanya jika:

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, \dots, S_t)$$

- Sebuah state memiliki informasi dari sejarah
- Ketika state diketahui, sejarah dapat diabaikan

## Markov Decision Process

- Saat step  $t = 0$ , keadaan awal (initial state) environment adalah  $s_0 \sim p(s_0)$
- Kemudian, untuk  $t = 0$  sampai selesai, agent harus memilih beberapa action:
  - Agent memilih action  $a_t$
  - Environment memberi reward  $r_t = R(\cdot | s_t, a_t)$
  - Environment memberi next state  $s_{t+1} = P(\cdot | s_t, a_t)$
  - Agent menerima reward  $r_t$  dan next state  $s_{t+1}$
- Policy adalah sebuah fungsi dari  $S$  to  $A$  yang menentukan aksi apa yang harus diambil di setiap state.
- Objektif: menemukan policy yang memaksimalkan diskon reward kumulatif

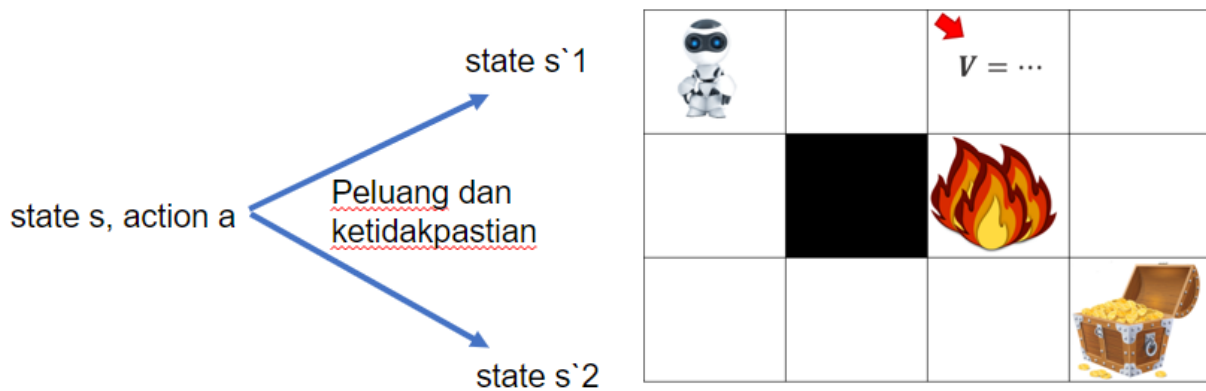
**Nah, apabila sudah mengenal MDP, kita sudah dapat membahas RL**



- Markov Decision Processes mendeskripsikan secara formal lingkungan untuk RL
- Secara spesifik biasanya dibuat saat environment fully observable
- Hampir semua RL problems dapat diformalisasi menggunakan MDP
- Optimal Control
- Partially Observable problems
- Bandits problem

**Bayangkan anda pergi kuliah**





- Moda transportasi apa yang akan anda gunakan?
  - Jalan kaki?
  - Sepeda?
  - Angkutan Umum?
  - Taksi online?
  - Taksi






### Aplikasi RL dalam Dunia Nyata

- Robotik: Memutuskan bergerak kemana.
- Alokasi Sumber Daya: Memutuskan apa yang diproduksi
- Pertanian: Menentukan apa yang ditanam, tapi tidak tahu cuaca dan hasil tanam

### Contoh MDP: Robotik

		 $V = \dots$	
		 $-100$	
			 $+100$

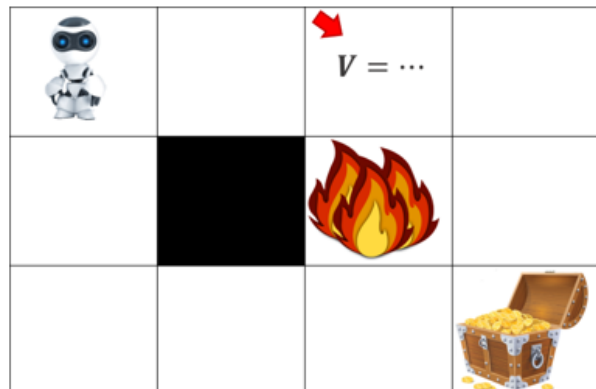
$$V(s) = \max_a (R(s, a) + \gamma V(s'))$$

		 $V = \dots$	
			

## Solusi untuk MDP?

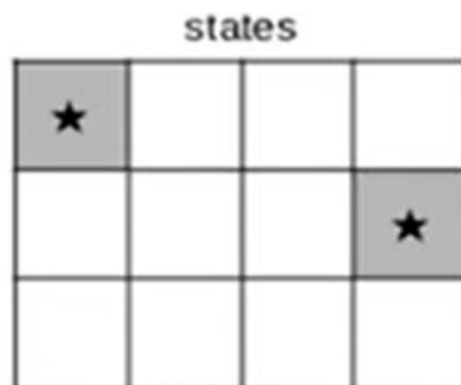
- Policy adalah fungsi  $\pi$  yang memetakan untuk setiap state  $s \in States$  ke action  $a \in Actions(s)$
- Contoh robot pencarian harta karun:

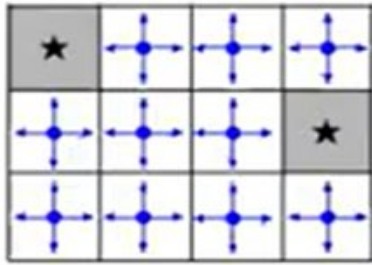
(1,1)      Bawah  
 (2,1)      Bawah  
 (3,1)      Kanan  
 ....      ...



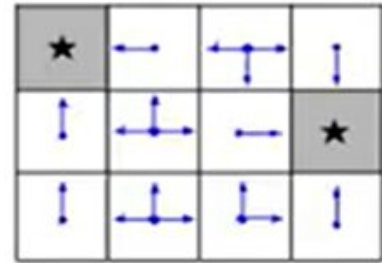
- Contoh MDP: Grid World

actions = {  
 1. right →  
 2. left ←  
 3. up ↑  
 4. down ↓  
 }





Random Policy



Optimal Policy

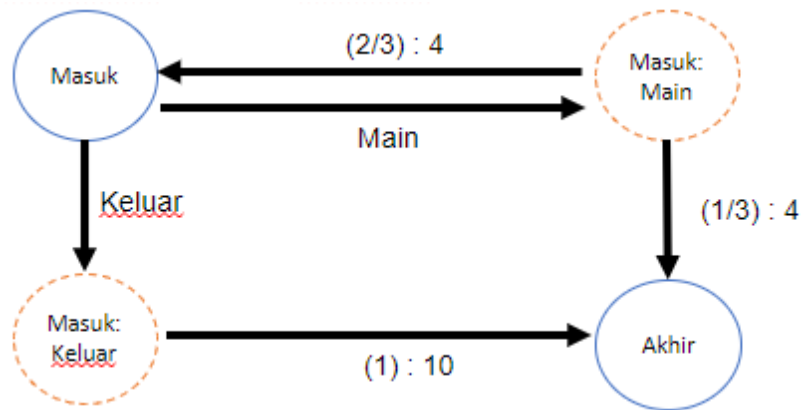
## Policy Optimal

- Masalahnya..., bagaimana handle randomness terkait initial state (keadaan awal) dan transition probability.
- Caranya dengan menghitung jumlah expected reward (reward yang diharapkan) di masa depan.

$$\pi^* = \operatorname{argmax}_{\pi} E \left[ \sum_{t \geq 0} \gamma^t r_t | \pi \right] \text{ with } s_0 \sim p(s_0), a_t \sim \pi(\cdot | s_t), s_{t+1} \sim p(\cdot | s_t, a_t)$$

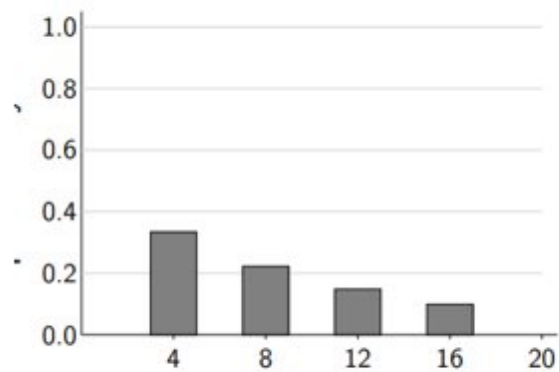
## Contoh MDP: Permainan Dadu

- Untuk setiap ronde:  $r = 1, 2, \dots$ 
  - Bisa memilih untuk main atau keluar
  - Jika keluar, mendapatkan 10 koin dan permainan berakhir
  - Jika main, mendapatkan koin 4 dan dadu dilempar
    - Jika dadu keluar angka 1 atau 2, permainan berakhir
    - Selain itu, lanjutkan ronde berikut



◦ Rewards

- Jika mengikuti policy “main”

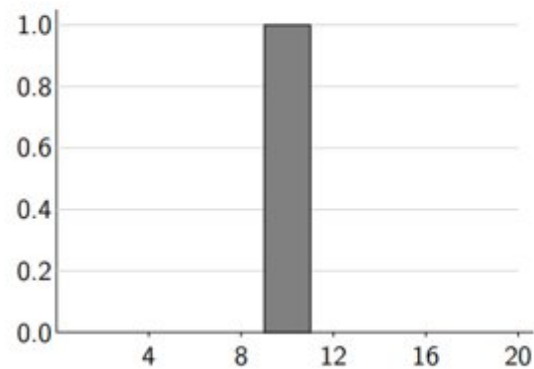


- Expected return:

$$\frac{1}{3}(4) + \frac{2}{3} \cdot \frac{1}{3}(8) + \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}(12) + \dots = 12$$



- Jika mengikuti policy “keluar”

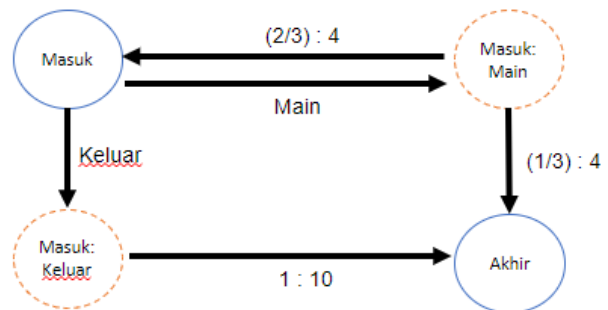


- Expected return:

$$1(10) = 10$$

### MDP untuk permainan Dadu

- State (s) node
- Action (a)
- State : Action q(s, a) node
- Probability  $P(s, a, s')$
- Reward  $r(s, a, s')$



- States: himpunan dari state
- $S_{start} \in States$  : state awal
- $Action(s)$  : actions yang tersedia dari state s
- $P(s, a, s')$  : probabilitas ke s' jika mengambil action a dari state s
- Reward (s, a, s') : idem dengan diatas
- $isEnd(s)$  : apakah state berakhir
- $0 \leq \gamma \leq 1$  faktor diskon

## Probabilitas (atau Transition)

- Transiti Probabilitas  $P(s, a, s')$  menentukan kemungkinan dari state yang didatangi dalam  $s'$  jika melakukan action  $a$  dari state  $s$
- Contoh:

<b>s</b>	<b>a</b>	<b>s'</b>	<b>P(s, a, s')</b>
Masuk	<u>Keluar</u>	Akhir	1
Masuk	Main	Masuk	2/3
Masuk	Main	Akhir	1/3

- Untuk setiap state  $s$  dan action  $a$ :

$$\sum_{s' \in States} P(s, a, s') = 1$$

<b>s</b>	<b>a</b>	<b>s'</b>	<b>P(s, a, s')</b>
Masuk	<u>Keluar</u>	Akhir	1
Masuk	Main	Masuk	2/3
Masuk	Main	Akhir	1/3

- Matrik probabilitas transisi

<b>s</b>	<b>a</b>	<b>s'</b>	<b>P(s, a, s')</b>
Masuk	Keluar	Akhir	1
Masuk	Main	Masuk	2/3
Masuk	Main	Akhir	1/3

<b>action = Main</b>	Masuk	Akhir	<b>action = Keluar</b>	Masuk	Akhir
Masuk	2/3	1/3	Masuk	0	0
Akhir	0	1	Akhir	0	1

### Contoh Transportasi

- Sebuah kota yang mempunyai nomor blok 1 sampai n
- Berjalan dari s ke s+1 membutuhkan waktu 1 menit
- Naik tram ajaib dari s ke 2s membutuhkan waktu 2 menit
- Bagaimana berpindah dari 1 ke n dalam waktu paling sedikit?
- Tram punya kemungkinan gagal 0.5
- Jika gagal, waktu habis 2 menit tapi tidak pindah blok

### Evaluasi sebuah policy

- Mengikuti policy akan bisa menghasilkan jalur yang acak (mengapa?)
- Return (utility) dari sebuah policy adalah Jumlah dari reward selama mengikuti jalur (nilai yang acak)
- Contoh policy = “main” pada permainan dadu:

Jalur	Return
[masuk;main,4,akhir]	4
[masuk;main,4, masuk;main,4, masuk;main,4,akhir]	12
[masuk;main,4, masuk;main,4, <u>akhir</u> ]	8
[masuk;main,4, masuk;main,4, masuk;main,4, masuk;main,4, <u>akhir</u> ]	16

## Return (Utility)

Return adalah  $G_t$  total reward yang di-diskon dari setiap waktu  $t$

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Jalur	Return
[masuk;main,4,akhir]	4
[masuk;main,4, masuk;main,4, masuk;main,4,akhir]	12
[masuk;main,4, masuk;main,4, <u>akhir</u> ]	8
[masuk;main,4, masuk;main,4, masuk;main,4, masuk;main,4, <u>akhir</u> ]	16

## Diskon $\gamma$

- Definisi: return
- Jalur:  $S_0, a_1, \gamma_1, s_1, a_2, \gamma_2, s_2, \dots$  (*action, reward, statebaru*)
- Return dengan diskon adalah:
 
$$u_1 = \gamma_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \dots$$
- Diskon  $\gamma = 1$  (sangat memperhatikan masa depan)
 

[masuk, masuk, masuk, masuk]:  $4+4+4+4 = 16$
- Diskon  $\gamma = 0$  (hidup cuma sekali)
 

[masuk, masuk, masuk, masuk] :  $4 + 0.(4 + 0.(4 + \dots)) = 4$
- Diskon  $\gamma = 0.5$  (hidup yang seimbang)
 

[masuk, masuk, masuk, masuk] :  $4 + 0.5(4 + 0.5(4 + \dots)) = 7.5$

## Apa guna diskon? $\gamma$

Biasanya Markov reward dan MDP mengandung diskon, mengapa?

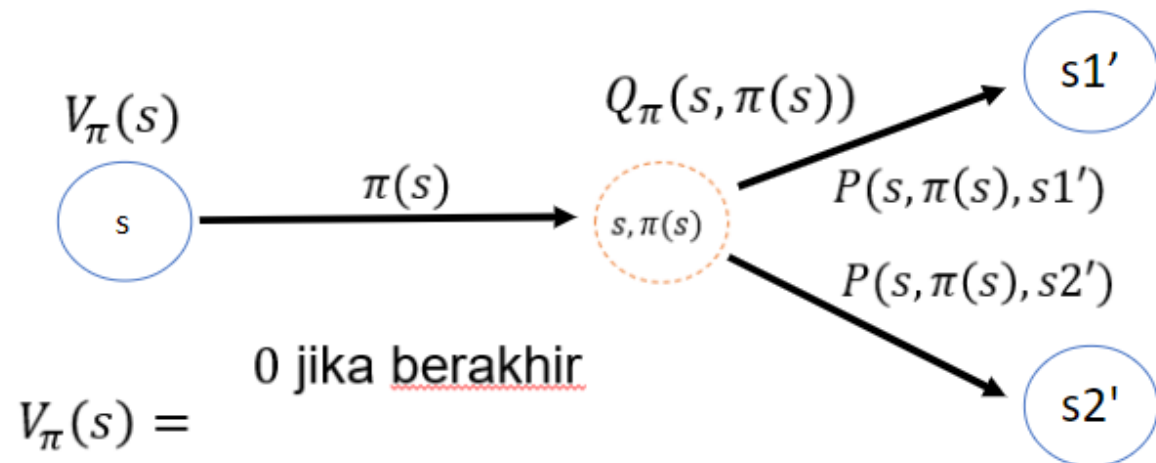
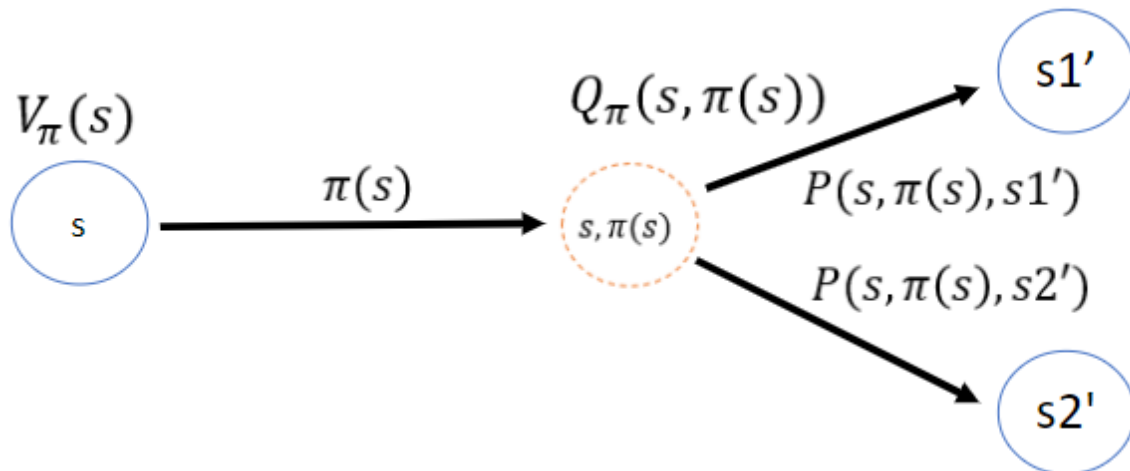
- Secara matematik memberikan konstanta untuk diskon itu mudah
- Menghindari return tak terhingga dalam proses Markov yang berbentuk siklus
- Masa depan yang tidak pasti mungkin tidak diwakili model
- Dalam hal finansial, reward langsung lebih menarik dibanding reward nanti
- Perilaku binatang / manusia biasanya lebih memilih reward langsung
- tetap memungkinkan untuk menggunakan markov reward yang tidak didiskon ( $\gamma = 1$ ) (jika semua alur tidak berulang)

## Recap and Question?

- Policy  $\pi(s)$
- Transition Probability M
- Jalur (path)
- Return (Utility)  $G_t$
- Value
- Excepted
- State (s) node
- Action (a)
- State: Action q(s,a) node
- Probability P(s, a, s')
- Reward r(s, a, s')

## Evaluasi sebuah Policy

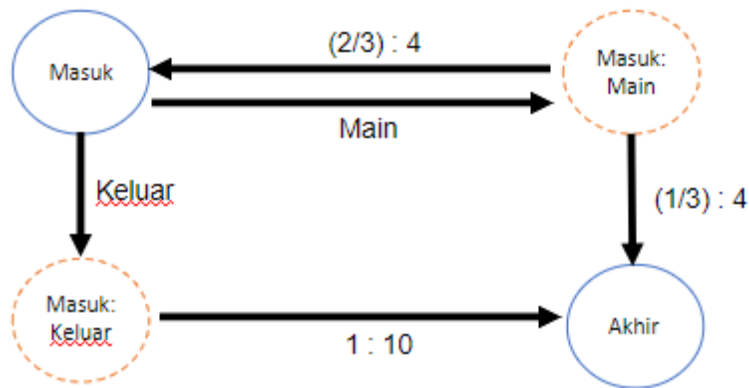
- Definisi: Value dari policy



$$V_\pi(s) = \begin{cases} 0 & \text{jika berakhir} \\ Q_\pi(s, \pi(s)) & \text{selain berakhir} \end{cases}$$

$$Q_\pi(s, a) = \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V_\pi(s')]$$

**Menghitung value Permainan Dadu**



- Misalnya kita pakai policy  $\pi$  “main” maka  $\pi(masuk) = main$

$$V_{\pi}(masuk) = \frac{1}{3}(4 + V_{\pi}(akhir)) + \frac{2}{3}(4 + V_{\pi}(masuk))$$

$$V_{\pi}(akhir) = 0$$

- Bisa kita selesaikan dengan policy  $\pi$  “main” Value adalah:  
(hitung dengan analitik)

$$V_{\pi}(masuk) = 12$$

## Value

- Value adalah harapan dari return (Expected return)

$$Q_{\pi}(s, a) = \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V_{\pi}(s')]$$

- Value dari policy adalah harapan dari return (Expected return) ketika mengikuti suatu policy

$$V_{\pi}(S) = E[G_t | S_t = s]$$

## Value Function

Policy akan menghasilkan sample trajectories (paths) i.e:

$$S_0, a_0, r_0, s_1, a_1, r_1$$

Seberapa bagus suatu state itu?

- Value function akan mengukur seberapa bagus situasi dari lingkungan (state)
- Value function pada state  $s$ , adalah komulatif reward yang diharapkan dari policy saat state  $s$ ;

$$V^\pi(s) = E \left[ \sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi \right]$$

## Q-Value Function

Seberapa bagus pasangan state-action?

- Q-Value function untuk mengukur seberapa bagus pasangan state-action (state-action pair).
- Q-Value function pada state  $s$  dan action  $a$ , adalah komulatif reward yang diharapkan (expected cumulative reward) dari mengambil action  $a$  pada state  $s$  dan mengikuti policy:

$$Q^\pi(s, a) = E \left[ \sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

- Formula ini merupakan Bellman Equation

## Bellman Equation untuk Optimal Policy

- Q-Value function yang paling optimal ( $Q^*$ ) adalah nilai maksimum komulatif expected reward yang dapat dicapai dari suatu pasangan state-action yang diberikan.

$$Q^\pi(s, a) = E \left[ \sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

- $Q^*$  memenuhi persamaan Bellman berikut:

$$Q^*(s, a) = E_{s'} \sim \xi \left[ r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$



## Optimal Policy

- Agent dapat diarahkan dengan Q-value yang telah bisa diselesaikan secara iteratif.
- Value iteration algorithm menggunakan Bellman equation as an iterative update.

$$Q_i + 1(s, a) = E \left[ r + \gamma \max_{a'} Q_i(s', a') | s, a \right]$$

- Setiap state-action pair akan dihitung semuanya, pada state-action yang terbatas ini memungkinkan. Permasalahn timbul bila state-action tidak terbatas (spt pada game Go), kemungkinannya banyak. Tidak scalable.
- Solusinya: dengan fungsi aproximator untuk mengestimasi Q(s,a) i.e. neural network.

## Menghitung value secara iterasi

- Algoritme menggunakan iterasi:
  - Inisialisasi  $V_{\pi}^{(0)}(s) \leftarrow 0$  untuk semua state
  - Untuk iterasi  $t = 1, \dots, t_{end}$ :
    - Untuk setiap state s:

$$V_{\pi}^{(t)}(s) \leftarrow \sum_{s'} P(s, \pi(s), s') \left[ R(s, \pi(s), s') + \gamma V_{\pi}^{(t-1)}(s') \right]$$

## Evaluasi Policy (contoh main dadu)

- $V_{\pi}^{(t)}(s) \leftarrow \sum_{s'} P(s, \pi(s), s') \left[ R(s, \pi(s), s') + \gamma V_{\pi}^{(t-1)}(s') \right]$
- $V_{\pi}(\text{masuk}) = \frac{1}{3}(4 + V_{\pi}(\text{akhir})) + \frac{2}{3}(4 + V_{\pi}(\text{masuk}))$
- Untuk  $\pi(\text{masuk}) = \text{main}$

0	0	0	0	0	0	0
0	4	6.67	8.44	9.6296	10.419	10.947

## Kapan iterasi selesai?

- Saat value iterasi t dikurangi value iterasi t-1 untuk semua state kurang dari konstanta  $\epsilon$ . Secara matematik dapat dituliskan demikian:

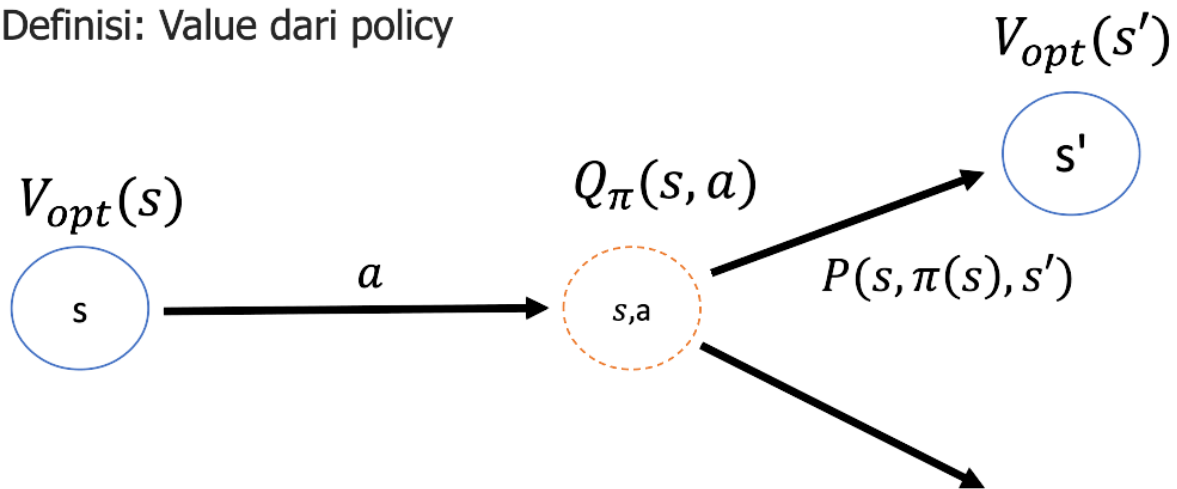
$$\max_{s \in States} |V_{\pi}^{(t)}(s) - V_{\pi}^{(t-1)}(s)| \leq \epsilon$$

- Tidak perlu menyimpan semua nilai  $V_{\pi}^{(t)}$  cukup simpan nilai t dan t-1:

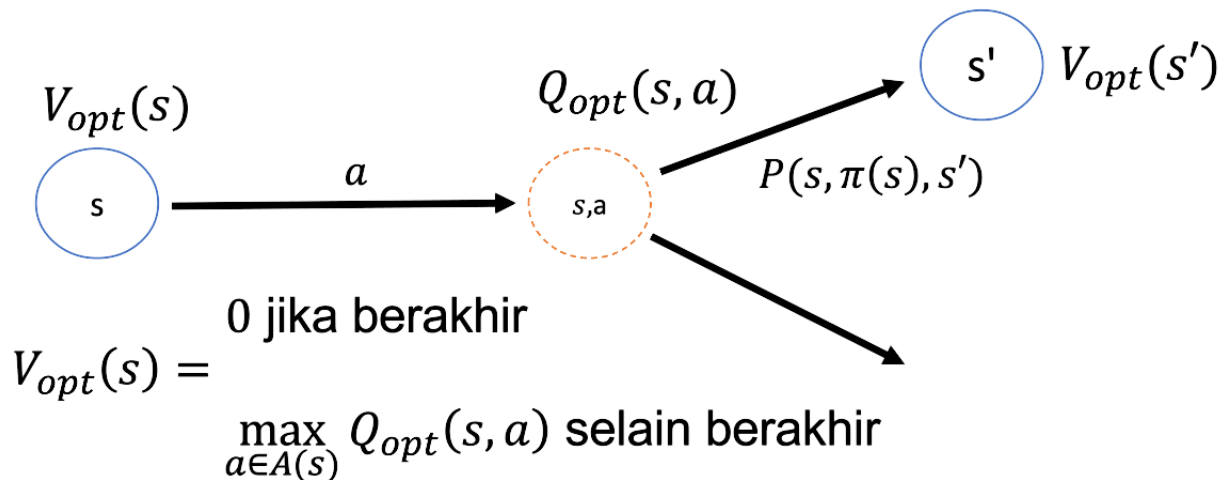
$$V_{\pi}^{(t)} \text{ dan } V_{\pi}^{(t-1)}$$

## Value dan policy yang Optimal

- Definisi: Value dari policy



$$Q_{opt}(s, a) = \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V_{opt}(s')]$$



$$Q_{opt}(s, a) = \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V_{opt}(s')]$$

- $V_{opt}(s) = \max_{a \in A(s)} Q_{opt}(s, a)$

## Dynamic Programming

- Dynamic programming jenis **Value Iteration**
- Algoritme menggunakan iterasi:
- Inisialisasi  $V_{opt}^{(0)}(s) \leftarrow 0$  untuk semua state
- Untuk iterasi  $t = 1, \dots, t_{end}$ :
  - Untuk setiap state  $s$  :

$$V_{opt}^{(t)}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s, \pi(s), s') \left[ R(s, \pi(s), s') + \gamma V_{\pi}^{(t-1)}(s') \right]$$

- Kompleksitas adalah  $O(t_{end} S A S')$

## Konvergensi

- Bagaimana agar value iteration mencapai konvergensi?
  - Diskon  $\gamma < 1$  atau
  - Grafik MDP yang tidak berbentuk siklus
- Maka value iteration akan mencapai konvergensi

## Kesimpulan algoritma

- Evaluasi Policy itu adalah  $(MDP, \pi) \longrightarrow V_\pi$
- Value Iteration itu adalah  $MDP \longrightarrow V_{opt}, \pi_{opt}$

## Markov Transition Graph

- Belajar meng-koding Markov dengan Python
- Studi kasus “Travelling to Capital Cities” atau Gambler Ruin

Seorang bisnisman akan melakukan perjalanan ke ibukota beberapa negara

