جستجوی غیرقطعی:
فرایند تصمیم مارکوف

سید ناصر رضوی n.razavi@tabrizu.ac.ir
۱۳۹۵

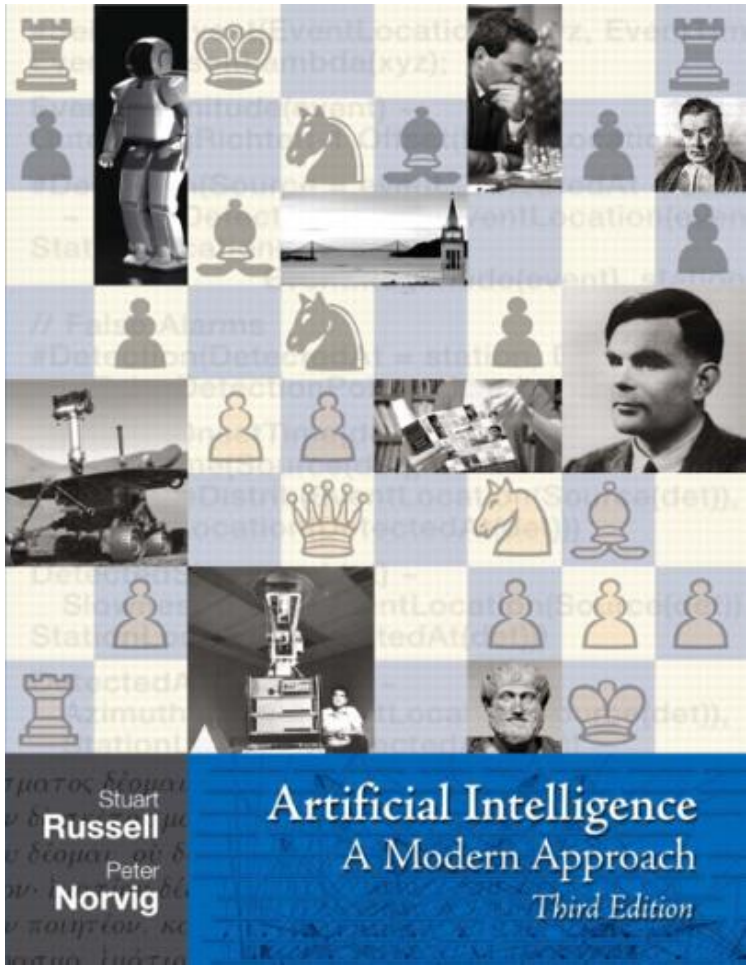
معرفی

n.razavi@tabrizu.ac.ir

- سید ناصر رضوی.
  - عضو گروه مهندسی کامپیوتر؛
  - دانشکده مهندسی برق و کامپیوتر، دانشگاه تبریز؛
  - سرپرست آزمایشگاه تحقیقاتی: «هوش محاسباتی و یادگیری ماشین»

- زمینه‌های پژوهشی.
  - یادگیری ماشین و یادگیری عمیق
  - پردازش زبان طبیعی و مترجم ماشین
  - پردازش تصویر و بینایی ماشین
  - رباتیک و خودروهای هوشمند

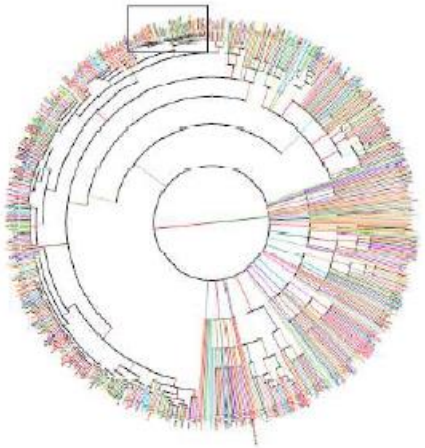
منابع و مراجع

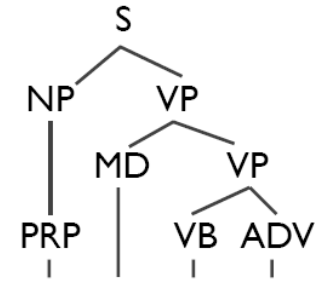


- هوش مصنوعی: یک رویکرد نوین. [ویراست سوم]
  - استوارت راسل و پیتر نورویگ. (۲۰۱۰)

فهرست مطالب

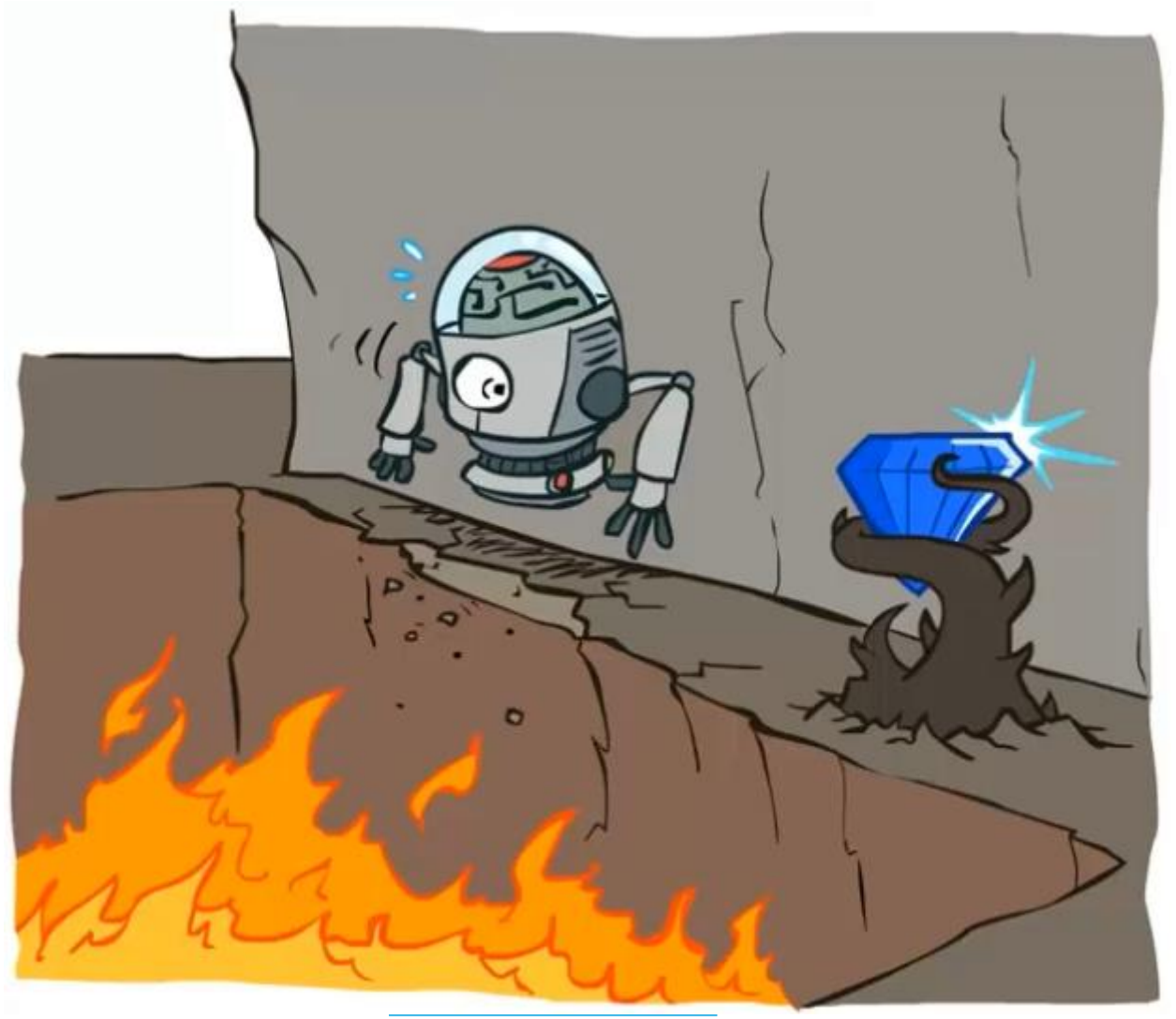
فهرست مطالب: کاربردهای پیشرفته





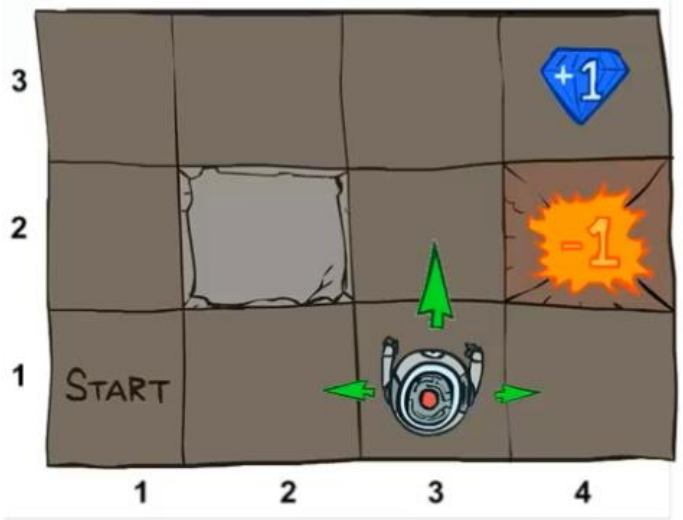
فهرست مطالب: کاربردهای پیشرفته

جستجوی غیرقطعی



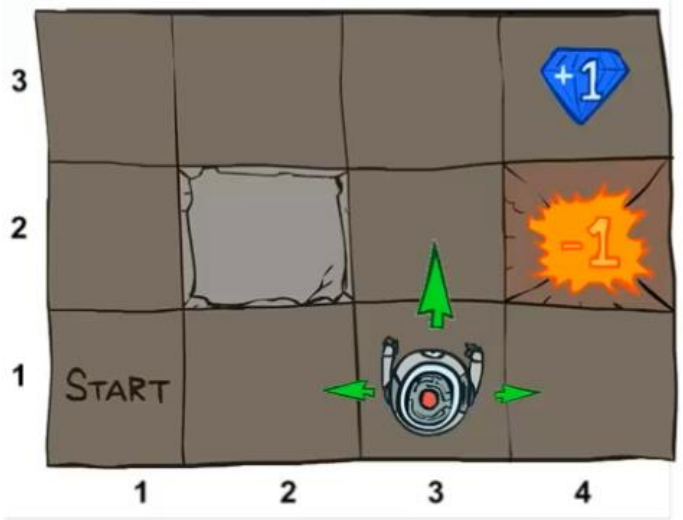
فرایند تصمیم مارکوف

مثال: محیط گرید



- یک مسئله شبیه به مسیریابی.
  - عامل در یک محیط گرید عمل می‌کند.
  - دیوارها مانع از حرکت عامل می‌شوند.

مثال: محیط گرید



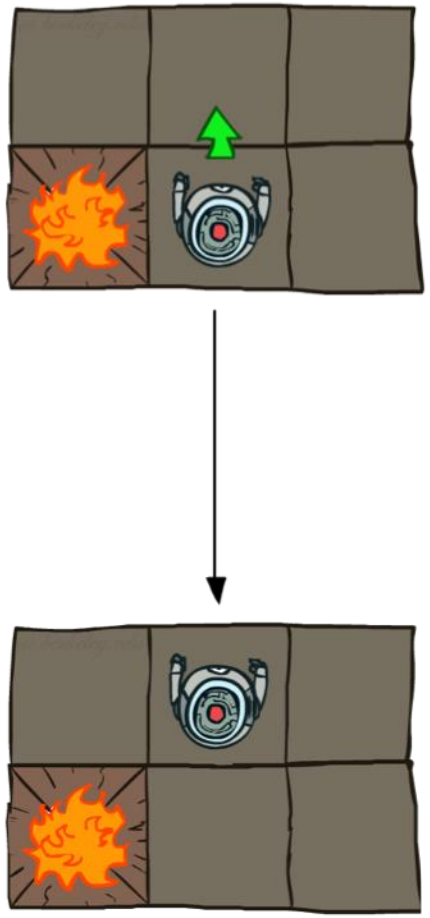
- یک مسئله شبیه به مسیریابی.
  - عامل در یک محیط گرید عمل می‌کند.
  - دیوارها مانع از حرکت عامل می‌شوند.

- حرکت‌های دارای نویز.
  - ۸۰ درصد مواقع انجام عمل North باعث حرکت عامل به خانه بالایی می‌شود.
  - اما ۱۰ درصد مواقع این عمل عامل را به چپ و ۱۰ درصد مواقع عامل را به راست می‌برد.
  - اگر در جهت حرکت عامل دیواری وجود داشته باشد، عامل در مکان فعلی خود باقی می‌ماند.

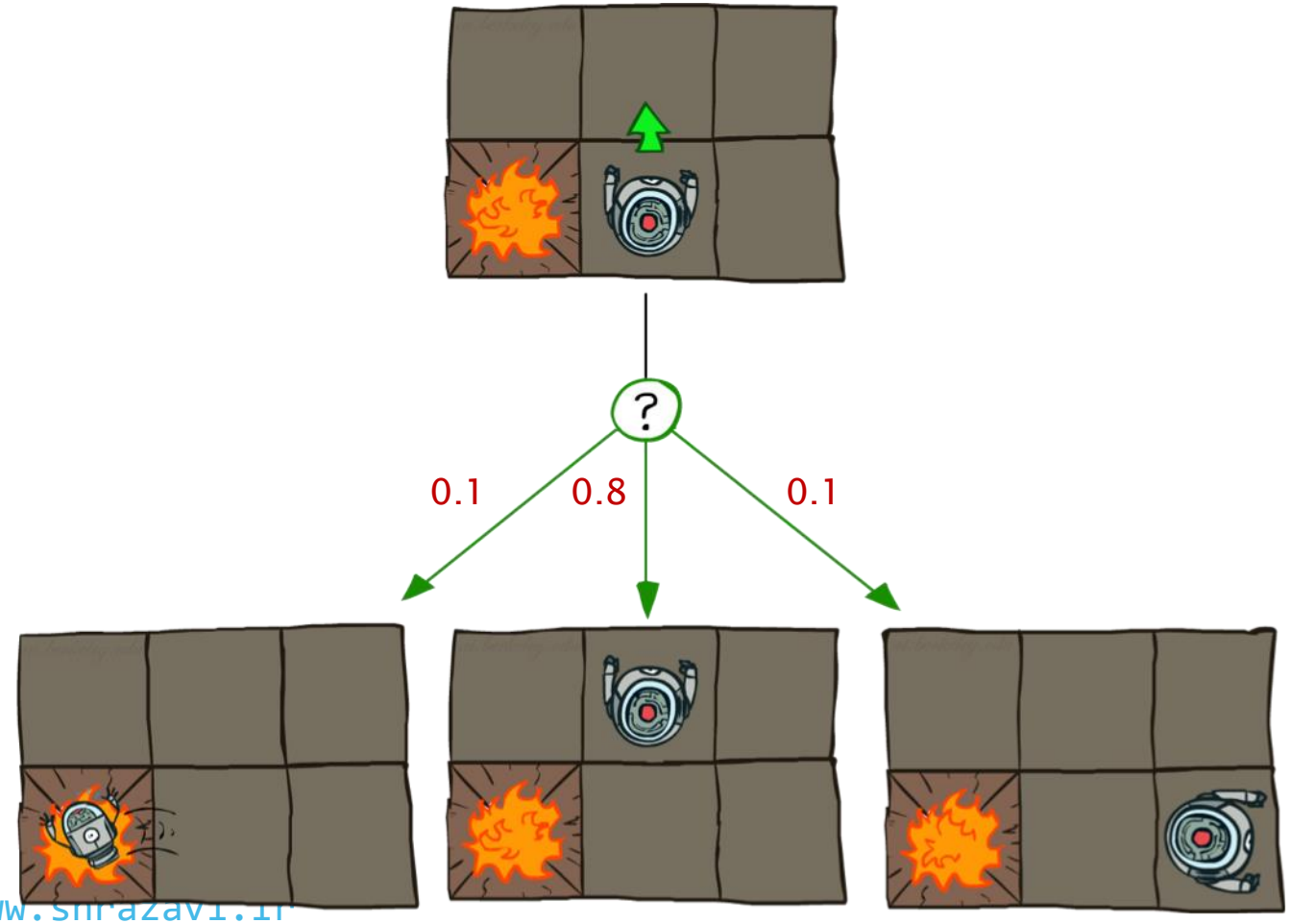
- عامل پس از انجام هر عمل یک پاداش دریافت می‌کند. [پاداش مرحله‌ای و نهایی]

عملیات در محیط گرید

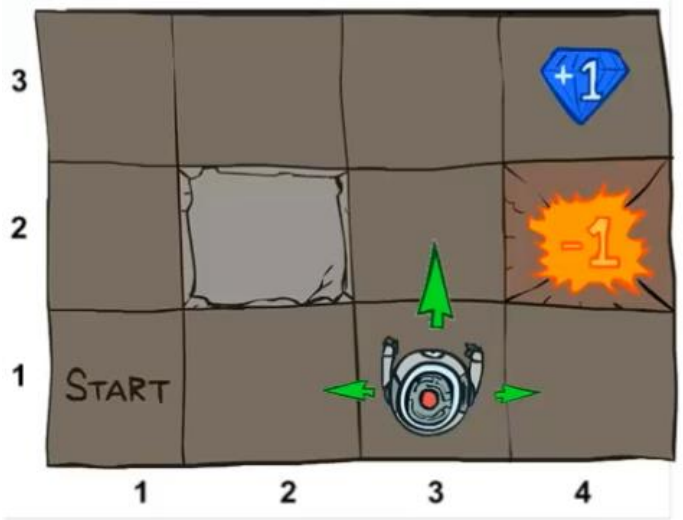
محیط گرید قطعی



محیط گرید اتفاقی



فرایند تصمیم مارکوف



- یک MDP به صورت زیر تعریف می‌شود:
  - یک مجموعه از حالت‌ها $s \in S$
  - یک مجموعه از اعمال $a \in A$
  - یک تابع تغییر حالت $T(s,a,s')$
    - احتمال رفتن از s به s' با انجام عمل a [نام دیگر: مدل یا دینامیک]
  - یک تابع پاداش $R(s,a,s')$
  - یک حالت شروع و شاید یک حالت پایان

- MDP ها بیانگر مسائل جستجوی غیرقطعی هستند.
  - یک روش برای حل این مسائل استفاده از الگوریتم expectimax است.
  - در ادامه با یک روش دیگر آشنا خواهیم شد.

فرایند تصمیم مارکوف

- فرایند «مارکوف» معمولاً به این معنا است که با داشتن حالت فعلی، حالت بعدی و حالت قبلی از یکدیگر مستقل هستند.
- در فرایندهای تصمیم مارکوف، منظور از «مارکوف» این است که نتیجه‌ی هر عمل تنها به حالت فعلی بستگی دارد.

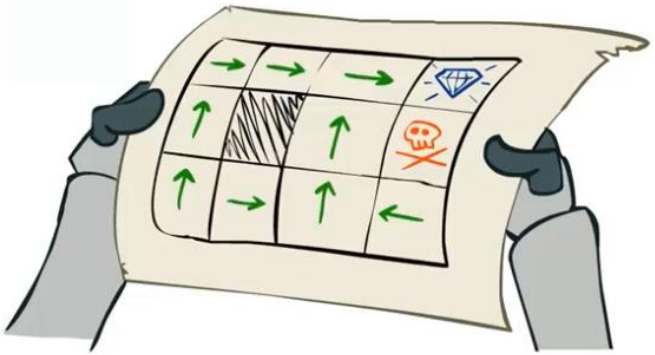
آندره مارکوف
(۱۸۵۶-۱۹۲۲)

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, \dots, S_0 = s_0)$$

$$=$$

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

سیاست‌ها

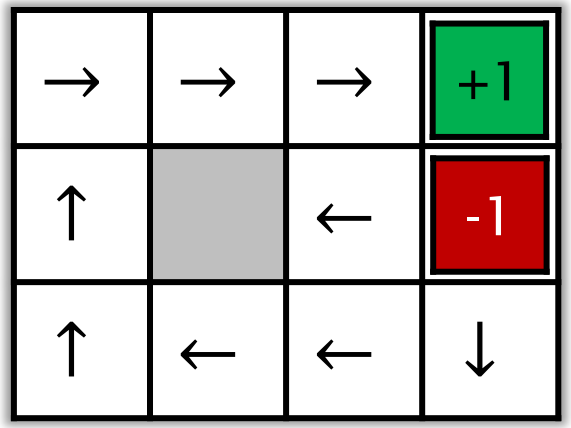


- در مسائل جستجوی تک-عاملی قطعی، ما به دنبال یک برنامه‌ی بهینه بودیم. [یک دنباله از عملیات از حالت شروع به حالت هدف]

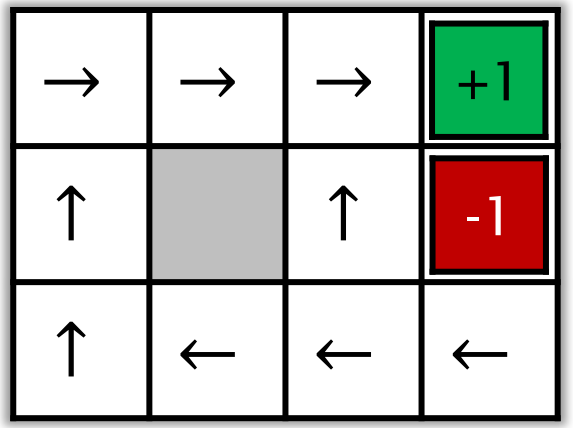
- در مسائل MDP به دنبال یک سیاست بهینه هستیم: $\pi^*: S \rightarrow A$
  - سیاست π تعیین کننده‌ی یک عمل در هر حالت است.
  - سیاستی بهینه است که در صورت دنبال کردن، سودمندی مورد انتظار را بیشینه سازد.
  - یک سیاست صریح، بیانگر یک عامل واکنشی است.

- جستجوی expectimax سیاست‌ها را به طور کامل محاسبه نمی‌کند.
  - تنها برای یک حالت عمل مناسب را محاسبه می‌کند.

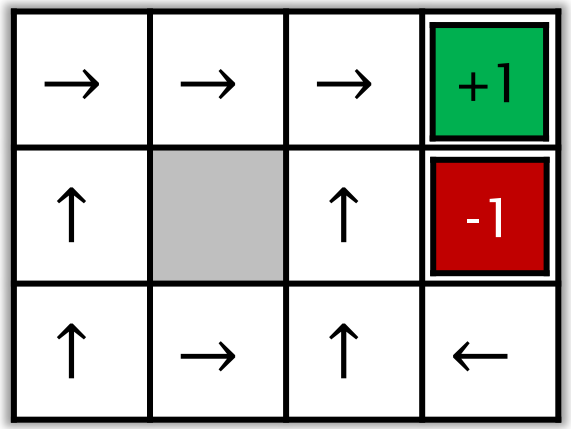
سیاست‌های بهینه



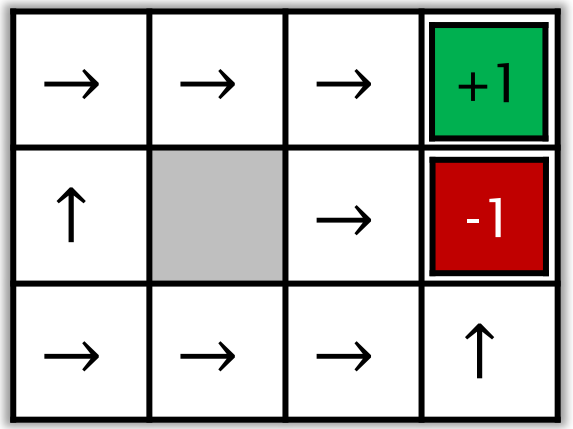
$R(s) = -0.01$



$R(s) = -0.03$

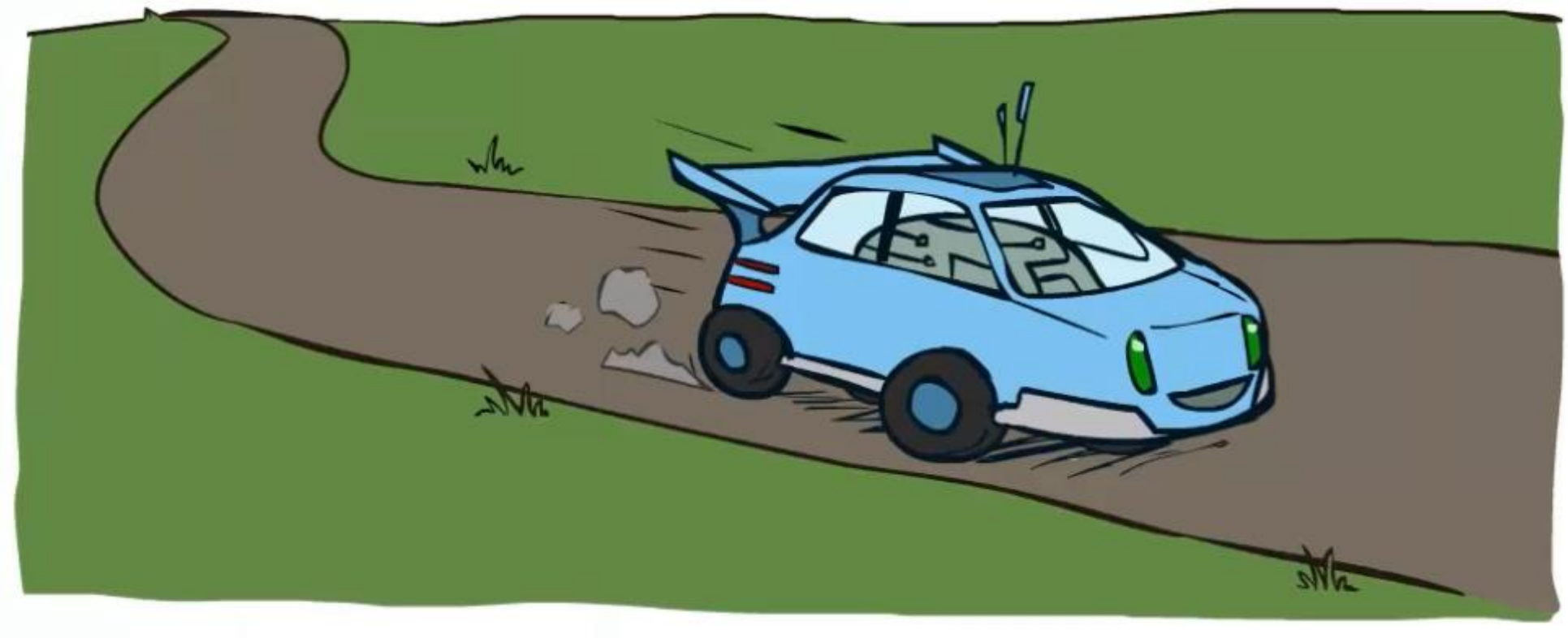


$R(s) = -0.4$



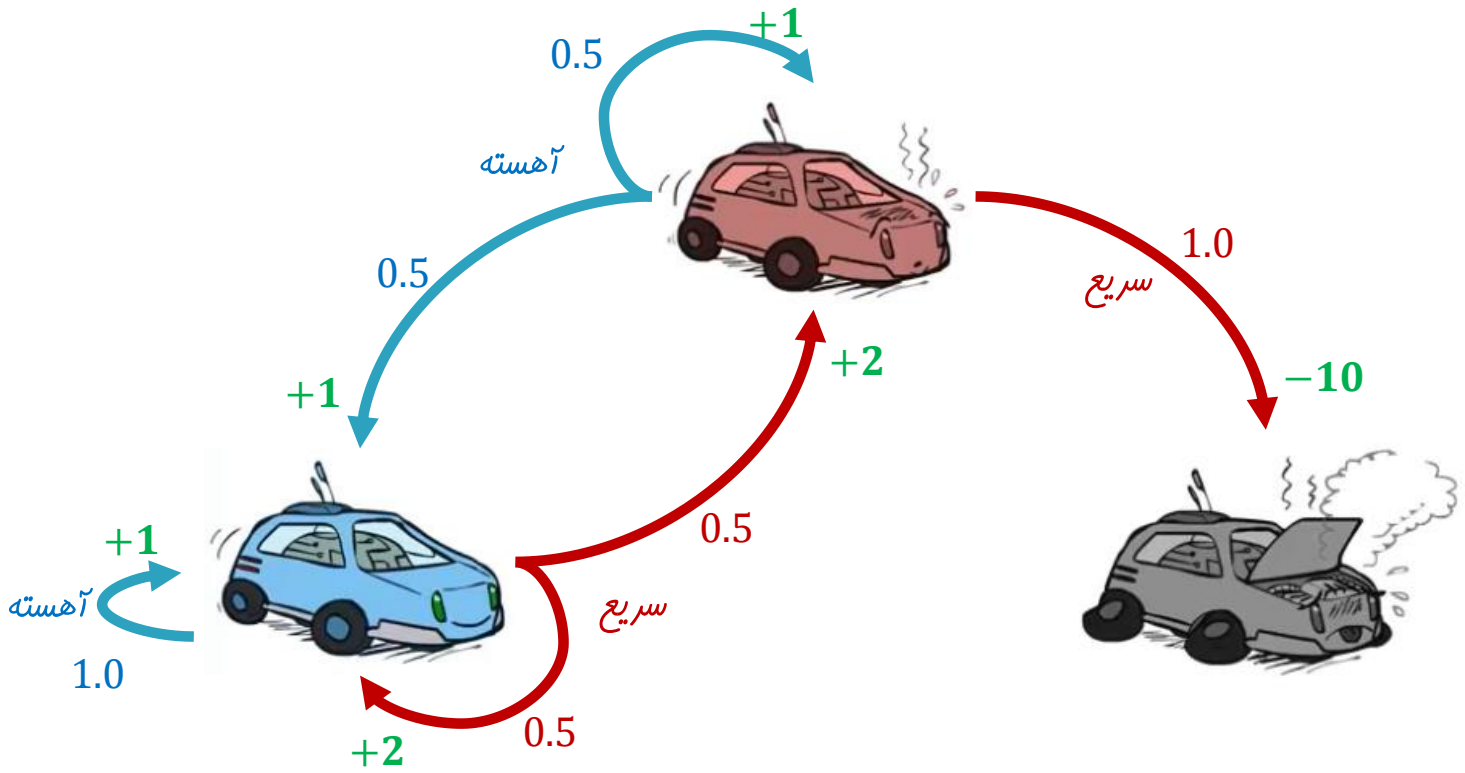
$R(s) = -2.0$

مثال: مسابقه‌ی اتومبیل رانی

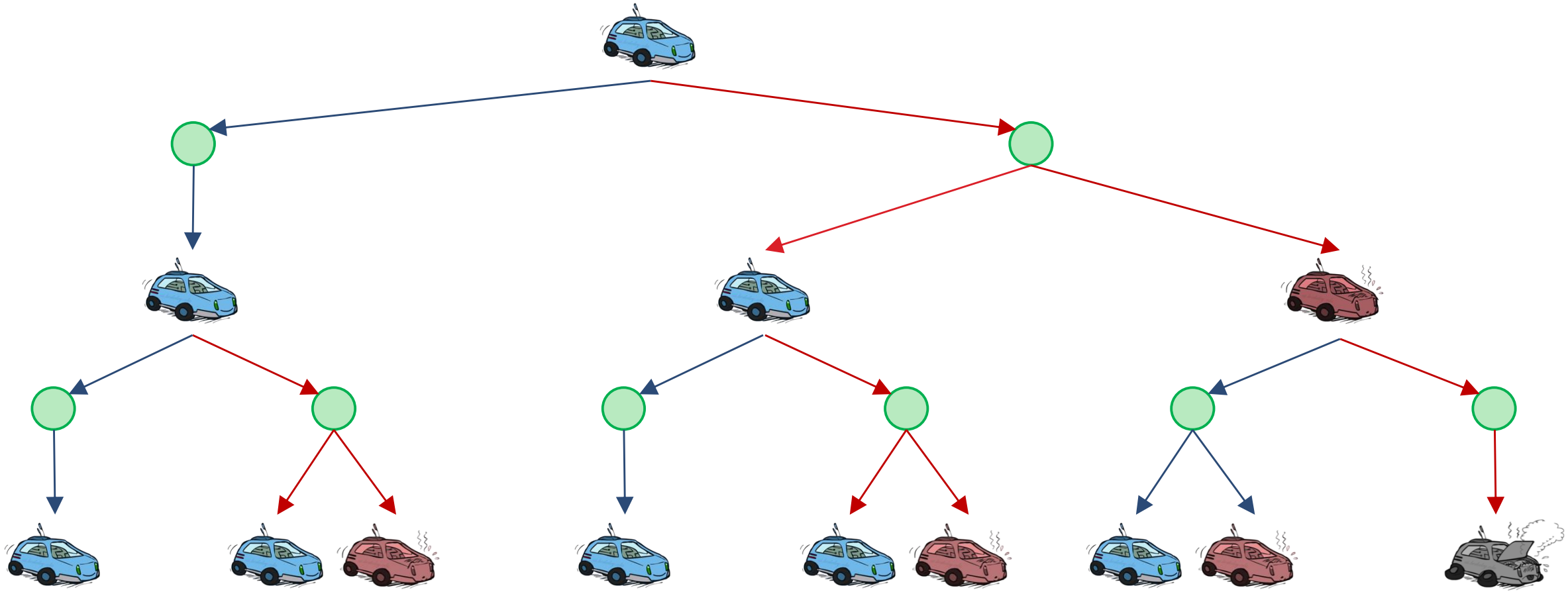


مثال: مسابقه‌ی اتومبیل رانی

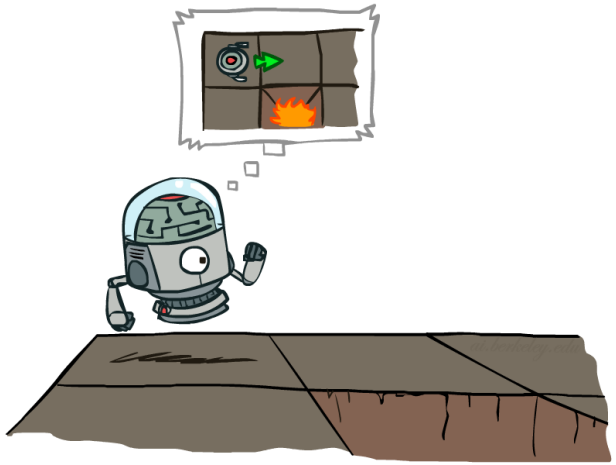
- یک اتومبیل روباتی می‌خواهد یک مسیر طولانی را به سرعت بپیماید.
- حالت‌ها: خنک، گرم، جوش
- اعمال: آهسته، سریع
- سریع رفتن پاداش را دو برابر می‌کند.

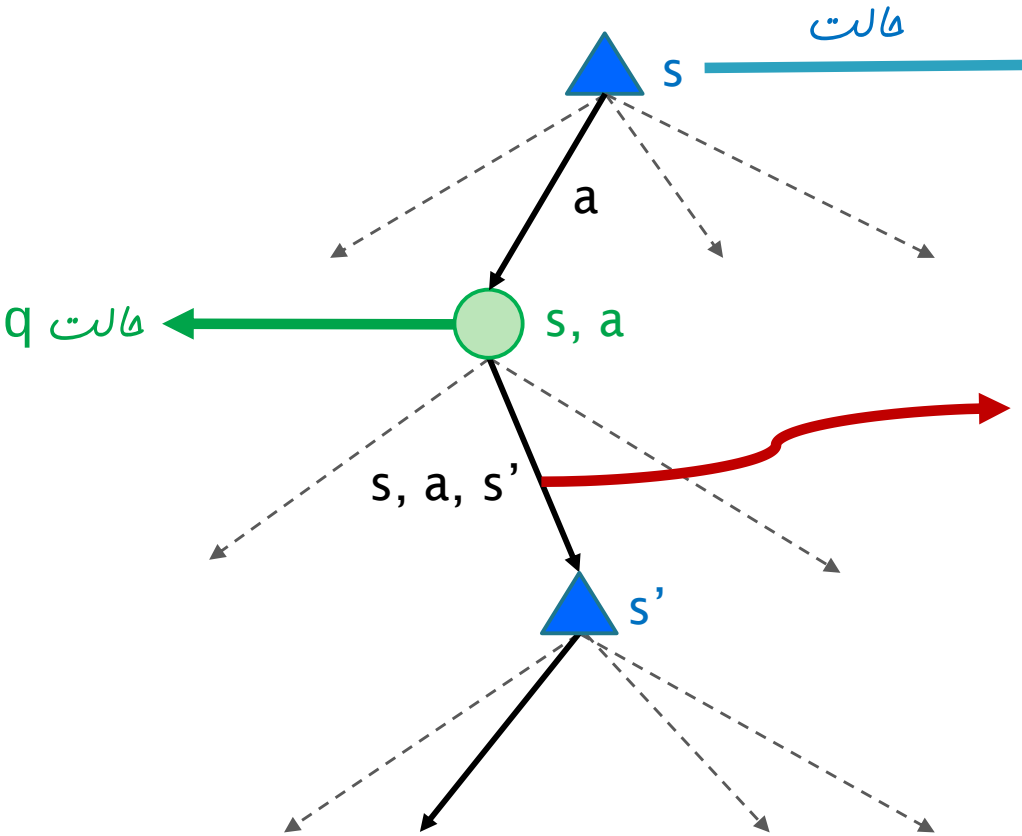


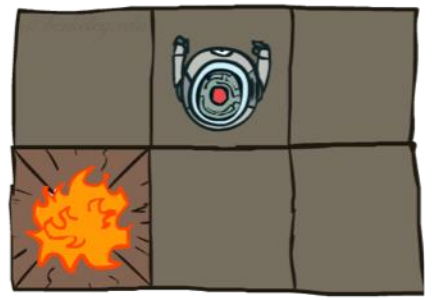
درخت جستجو



درخت جستجو: حالت کلی





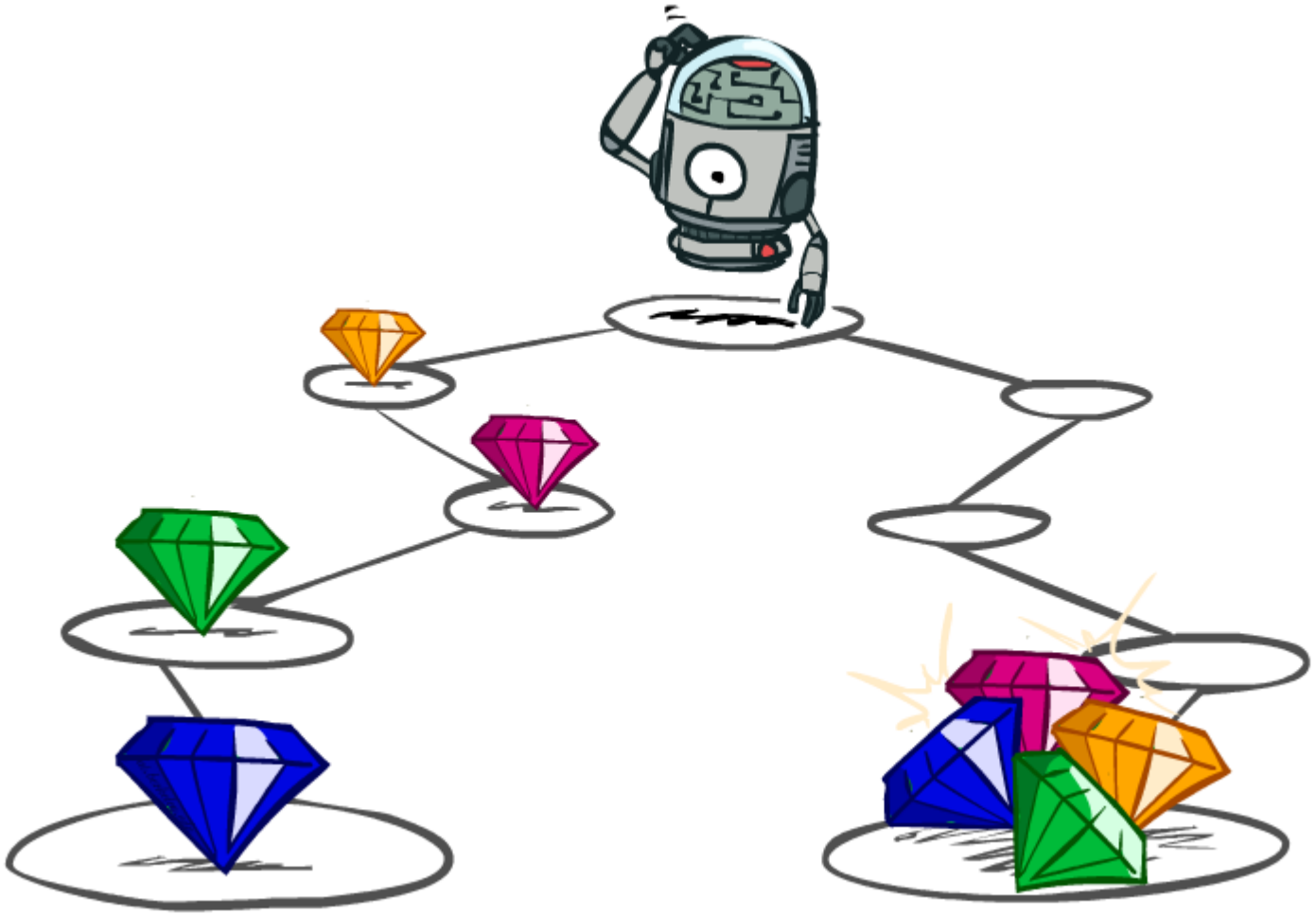


(s, a, s') تغییر حالت

T(s, a, s') = P(s' | s, a)

R(s, a, s')

سودمندی یک دنباله از عملیات

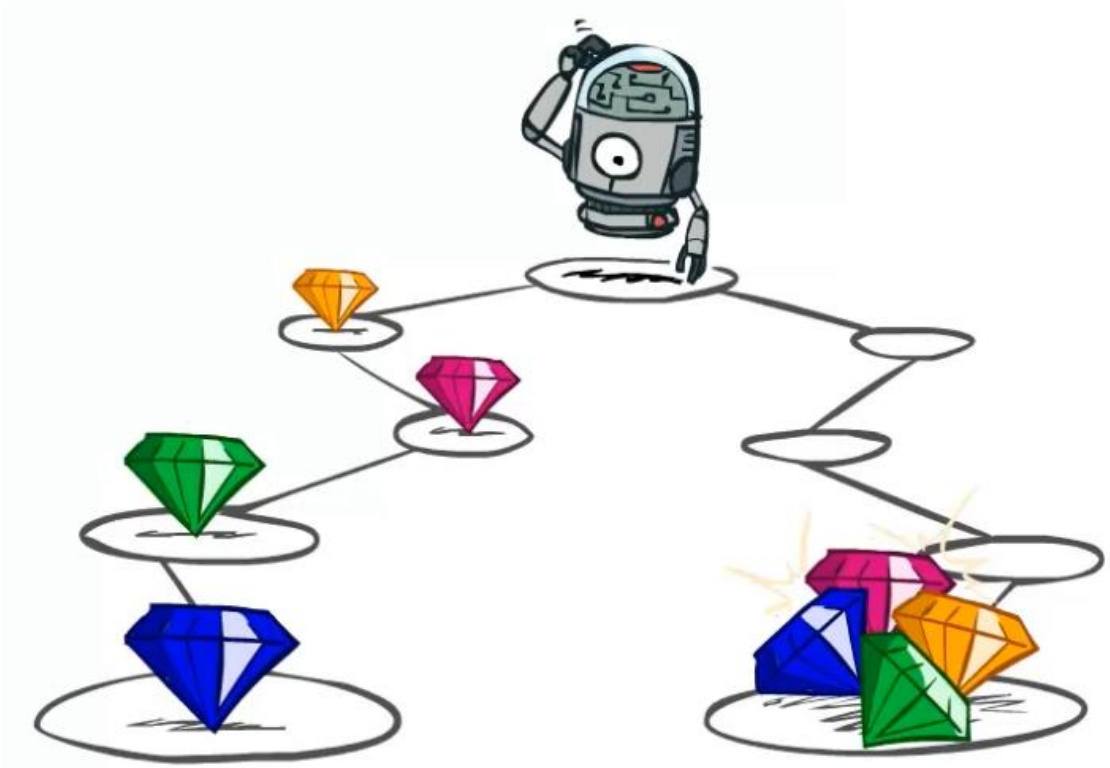


سودمندی یک دنباله از عملیات

- س. عامل باید کدام دنباله از پاداش‌ها را ترجیح دهد؟

- کمتر یا بیشتر؟ [۱, ۲, ۲] یا [۲, ۳, ۴]

- حالا یا بعداً؟ [۱, ۰, ۰] یا [۰, ۰, ۱]



کاهش پاداش‌ها

- س. عامل باید کدام دنباله از پاداش‌ها را ترجیح دهد؟
  - منطقی است که عامل بخواهد مجموع پاداش‌های دریافتی خود را بیشینه سازد.
  - همچنین، منطقی است که عامل پاداش‌های فوری را به پاداش‌های دارای تأخیر ترجیح دهد.

- یک راه‌حل. کاهش ارزش پاداش‌ها با گذشت زمان به صورت نمایی.

1

ارزش کنونی

γ

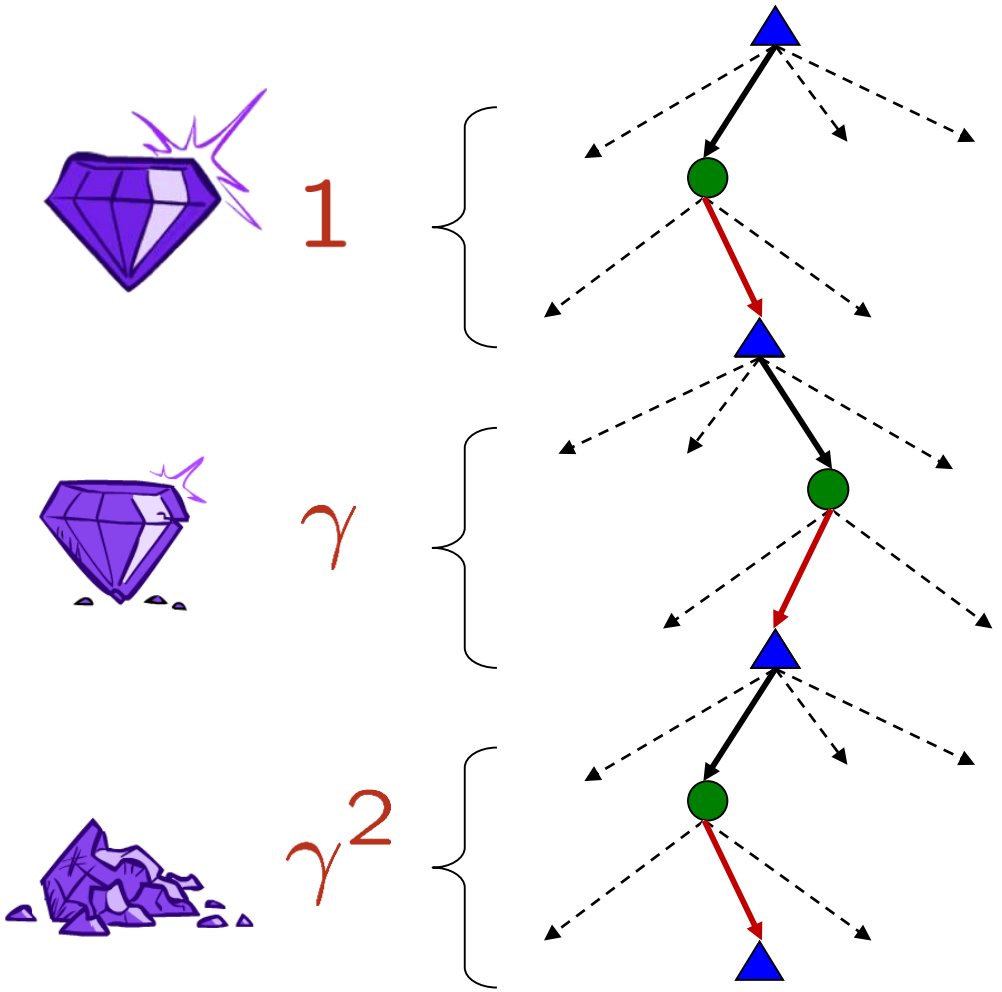
ارزش پس از یک مرحله

γ^2

ارزش پس از دو مرحله

$(\gamma \leq 1)$

کاهش پاداش‌ها

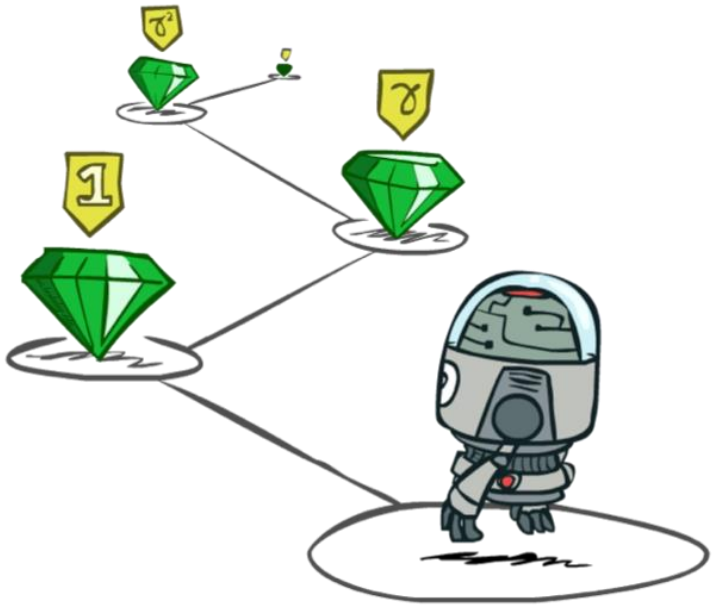


- چگونگی کاهش پاداش‌ها.
  - هر بار که یک سطح پایین می‌رویم، ضریب کاهش γ را یک بار ضرب می‌کنیم.
- چرا باید مقدار پاداش را کاهش دهیم؟
  - پاداش‌های فوری احتمالاً سودمندی بالاتری نسبت به پاداش‌های دارای تأخیر دارند.
  - کمک به همگرایی الگوریتم‌ها.
- مثال: ضریب کاهش ۰/۵

$$U([1,2,3]) = 1 \cdot 1 + 0.5 \cdot 2 + 0.25 \cdot 3$$

$$U([1,2,3]) < U([3,2,1])$$

اولویت‌های ایستا



- قضیه. برای اولویت‌های ایستا:

$$[a_1, a_2, \dots] \succ [b_1, b_2, \dots]$$

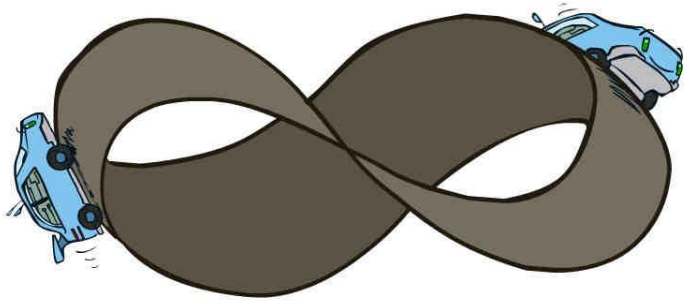
$$\Updownarrow$$

$$[r, a_1, a_2, \dots] \succ [r, b_1, b_2, \dots]$$

- با فرض اولویت‌های ایستا، تنها دو روش برای تعریف سودمندی‌ها ممکن است:
  - سودمندی جمع شونده: $U([r_0, r_1, r_2, \dots]) = r_0 + r_1 + r_2 + \cdots$
  - سودمندی کاهش یابنده: $U([r_0, r_1, r_2, \dots]) = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots$

سودمندی‌های نامتناهی؟!

- مشکل. اگر بازی تا ابد ادامه داشته باشد، آیا پاداش بی‌نهایت دریافت خواهیم کرد؟

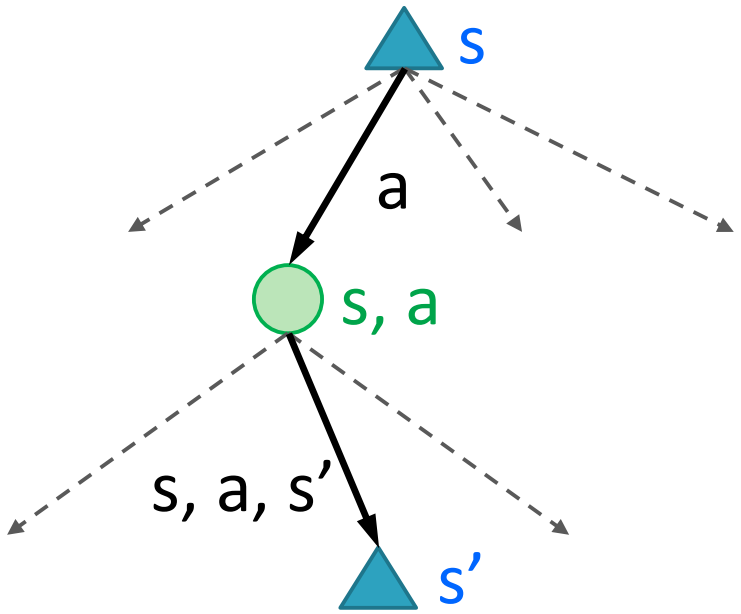


- راه‌حل‌ها.
  - افق متناهی: [همانند جستجوی عمقی محدود]
  - اعمال ضریب کاهش: $0 < \gamma < 1$
    - ضریب کوچک‌تر به معنای افق محدودتر - تمرکز کوتاه مدت

$$U([r_0, r_1, \dots, r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{max}/(1-\gamma)$$

  - حالت‌های جذب کننده: مانند حالت «جوش» در مسابقه‌ی اتومبیل رانی.

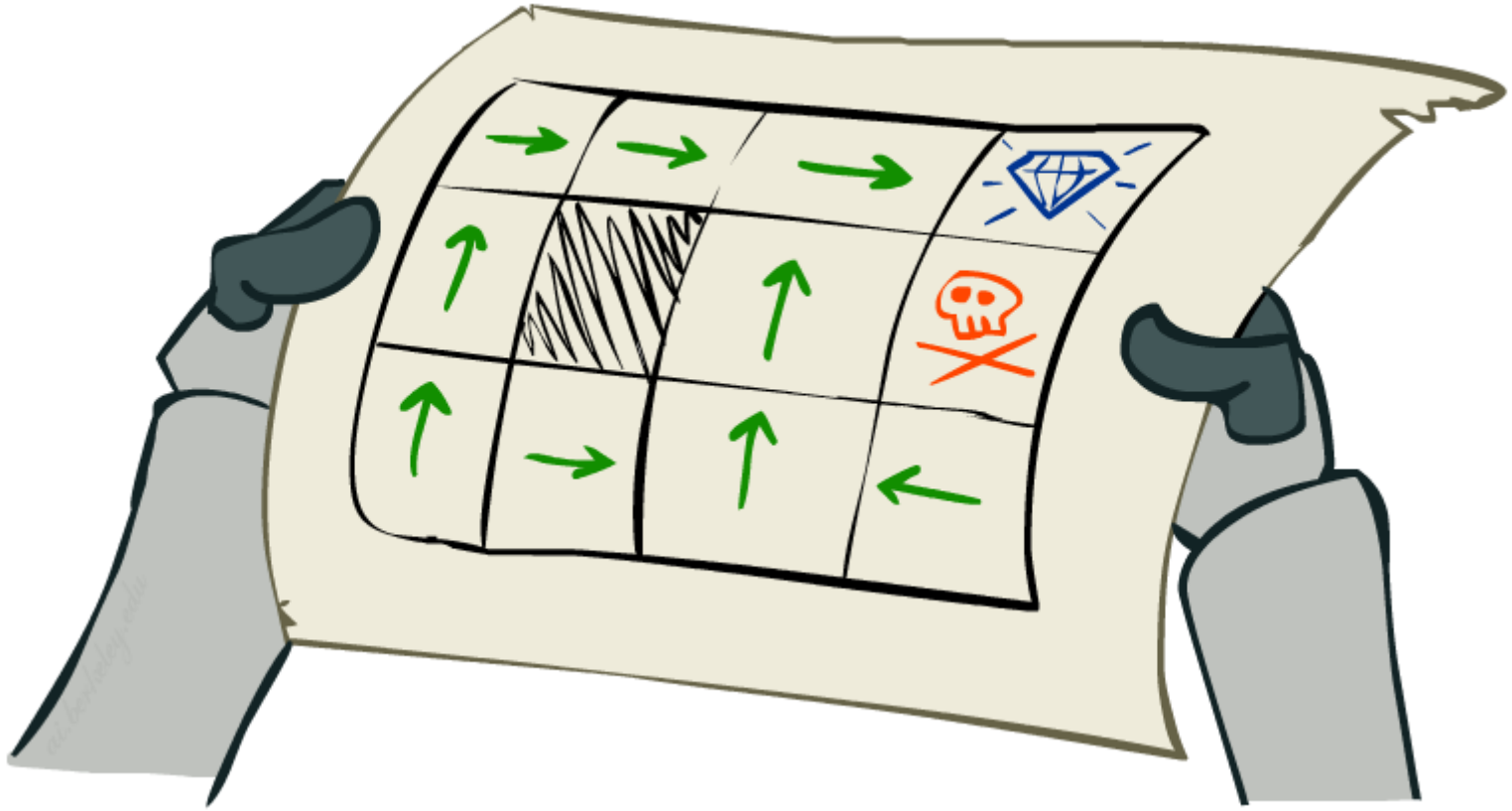
مرور: فرایند تصمیم مارکوف



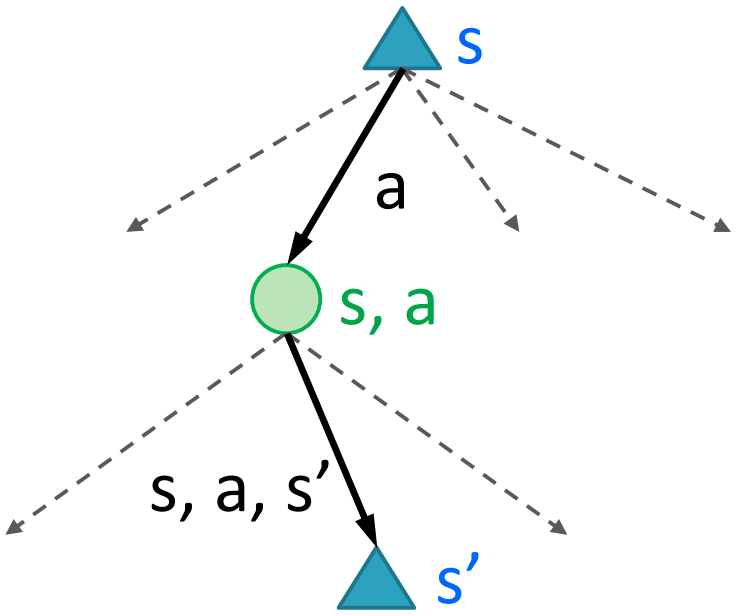
- یک MDP به صورت زیر تعریف می‌شود:
  - یک مجموعه از حالت‌ها S
  - یک مجموعه از اعمال A
  - یک تابع تغییر حالت $T(s,a,s')$
  - یک تابع پاداش $R(s,a,s')$
  - یک حالت شروع s_0

- دو کمیت مهم.
  - سیاست: انتخاب یک عمل به ازای هر حالت.
  - سودمندی: مجموع (کاهش یافته) پاداش‌ها.

حل مسائل MDP



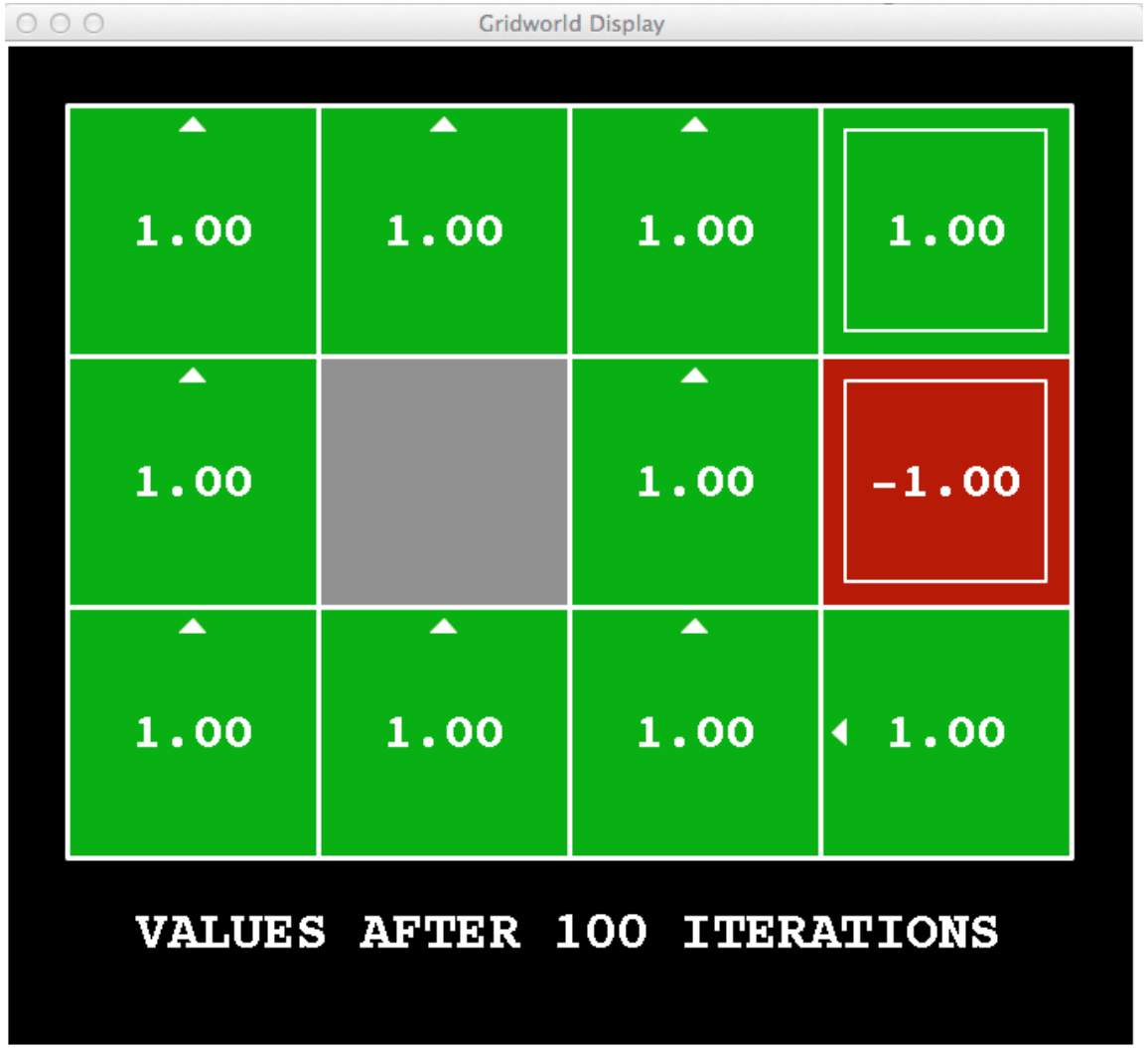
کمیت‌های بهینه



- $V^*(s)$ = ارزش (سودمندی) حالت s:
  - سودمندی مورد انتظار با شروع از s و بهینه عمل کردن.
- $Q^*(s, a)$ = ارزش (سودمندی) حالت q:
  - سودمندی مورد انتظار با شروع از s و انتخاب عمل a و بهینه عمل کردن [از این به بعد]
- سیاست بهینه.
  - $\pi^*(s)$ = عمل بهینه در حالت s.

[اجرای نمایشی – مقادیر گرید]

کمیت‌های بهینه: ارزش حالت‌ها

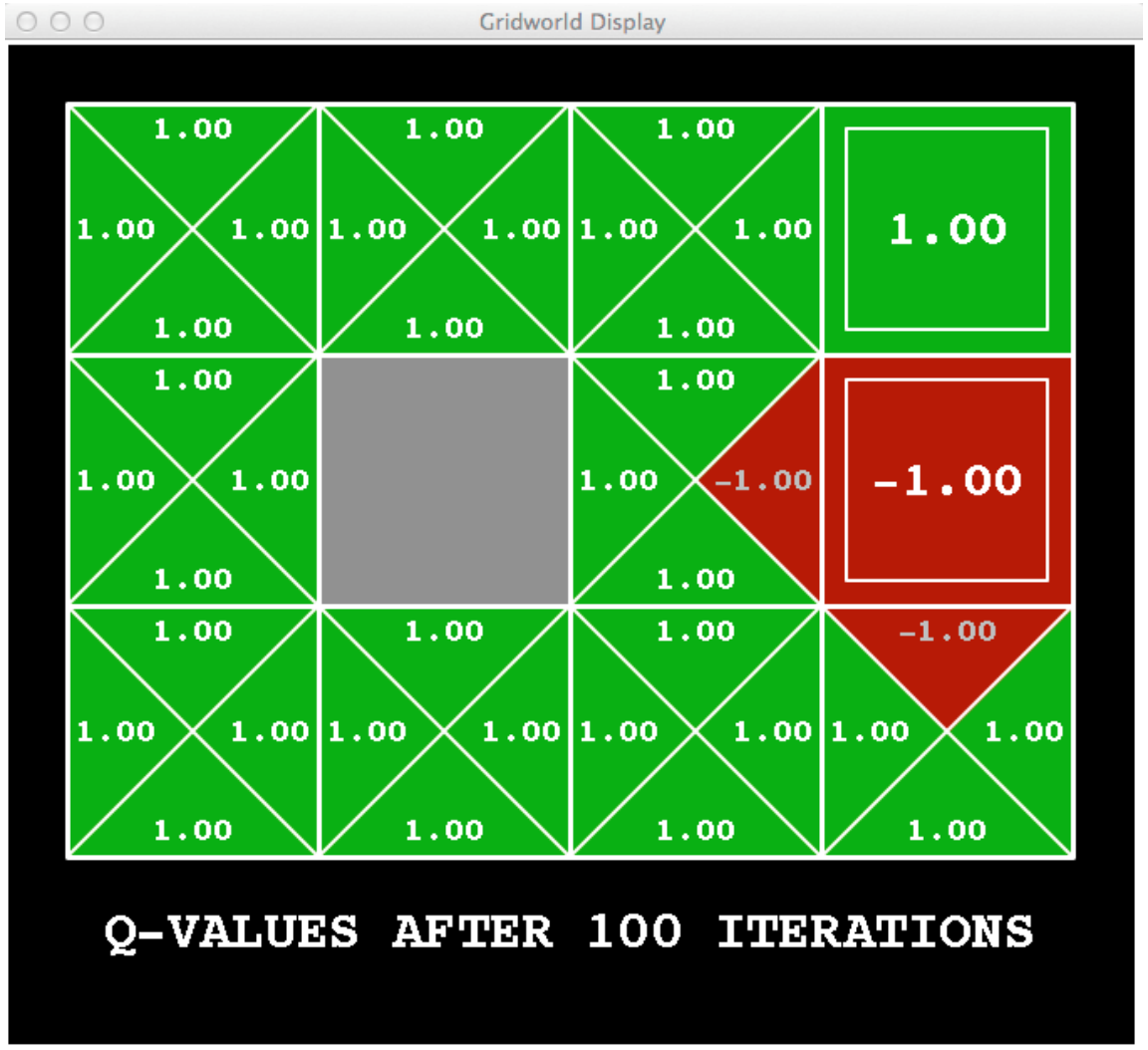


نویز = ۰

ضریب کاهش = ۱

پاداش مرحله‌ای = ۰

کمیت‌های بهینه: ارزش حالت‌های q

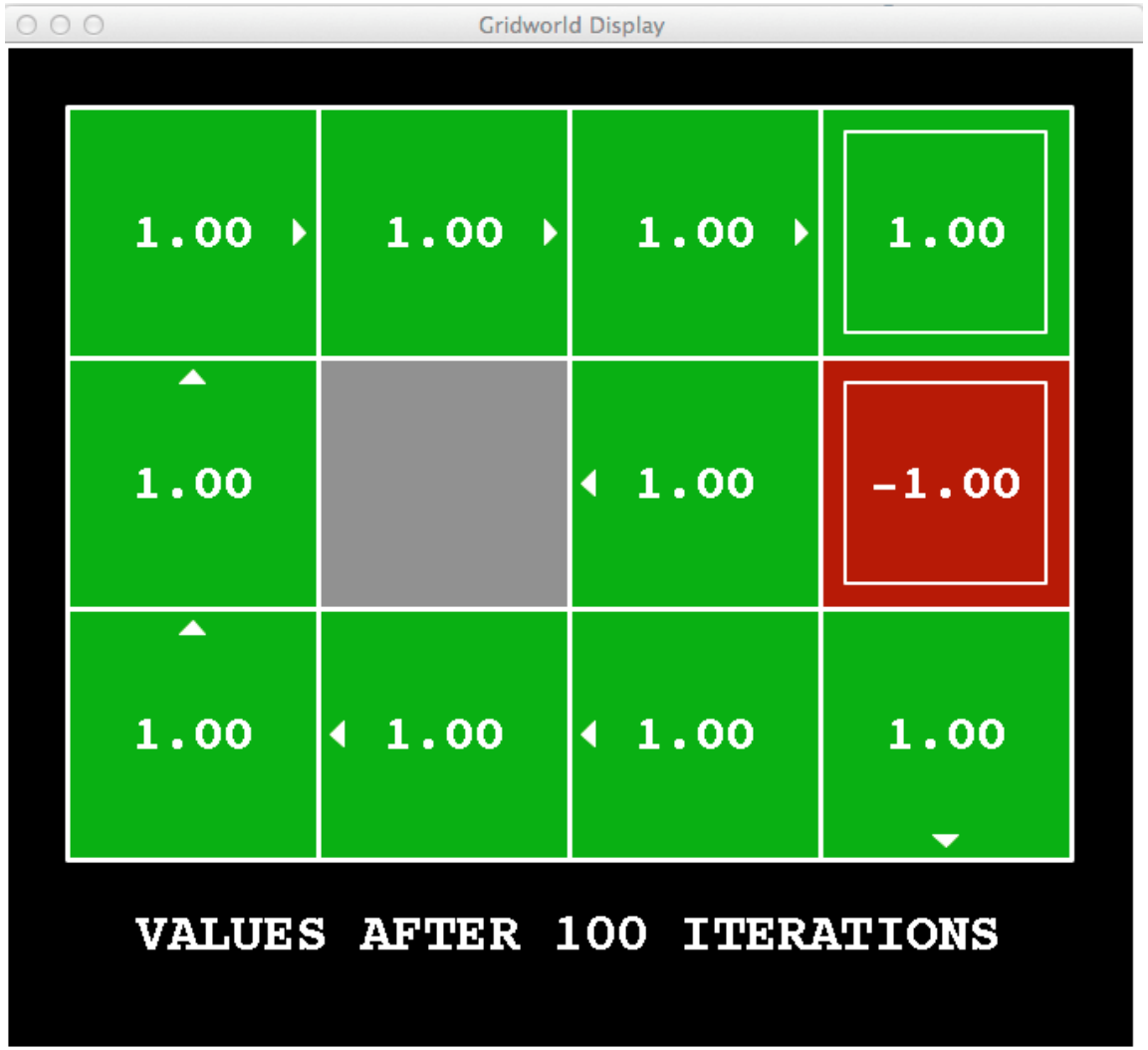


نویز = ۰

ضریب کاهش = ۱

پاداش مرحله‌ای = ۰

کمیت‌های بهینه: ارزش حالت‌ها

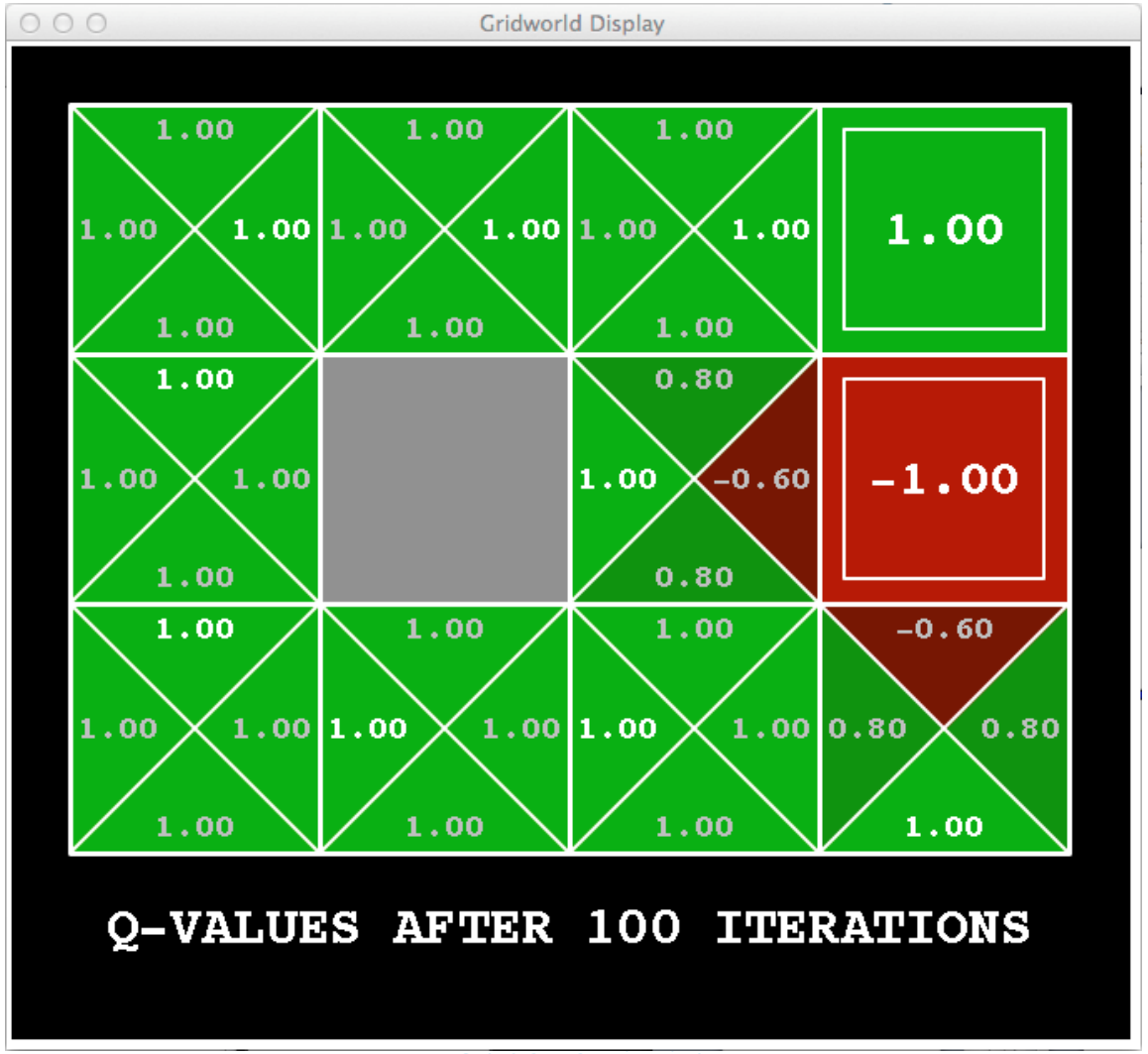


نویز = ۰/۲

ضریب کاهش = ۱

پاداش مرحله‌ای = ۰

کمیت‌های بهینه: ارزش حالت‌های q

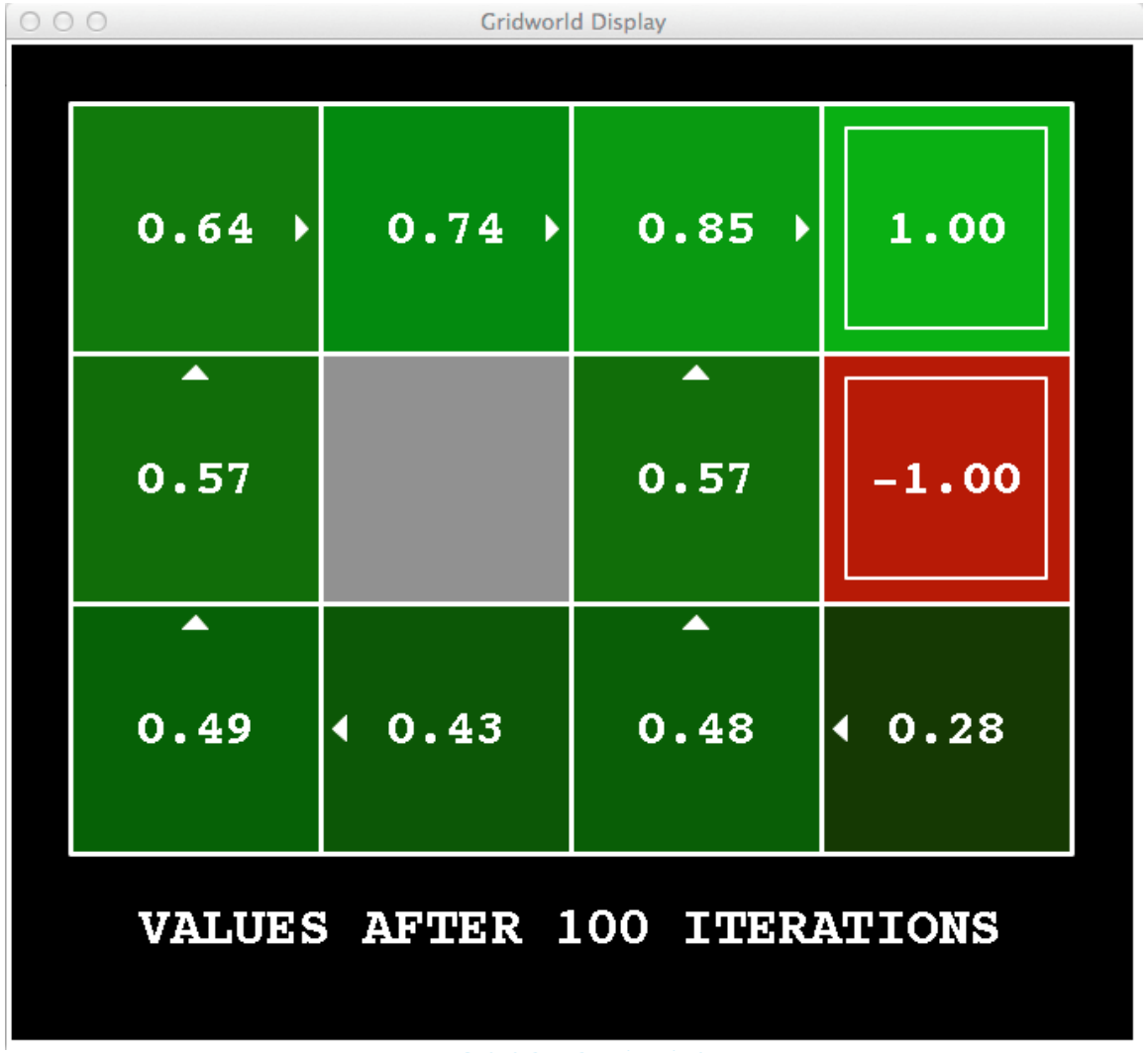


نویز = ۰/۲

ضریب کاهش = ۱

پاداش مرحله‌ای = ۰

کمیت‌های بهینه: ارزش حالت‌ها

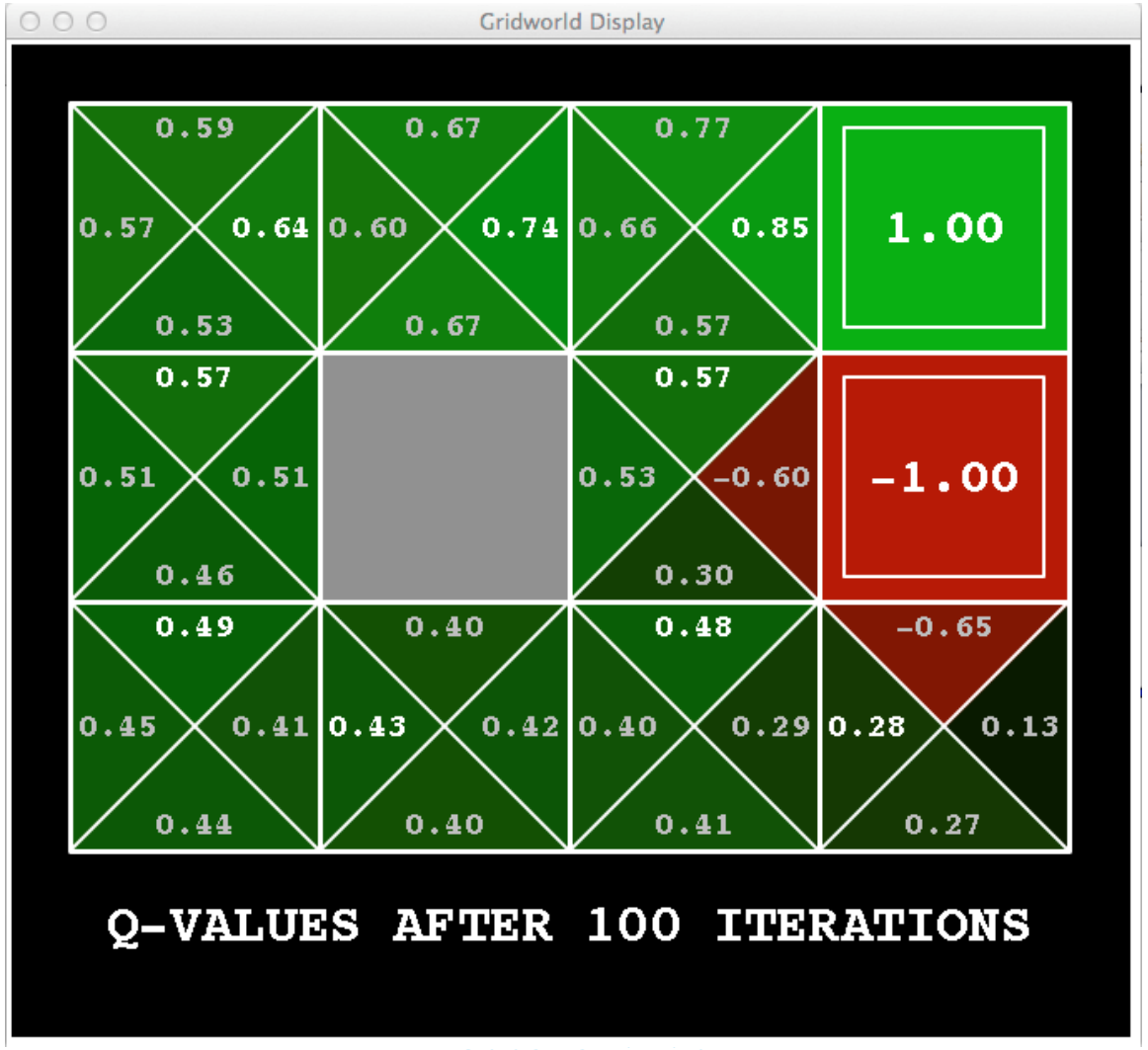


نویز = ۰/۲

ضریب کاهش = ۰/۹

پاداش مرحله‌ای = ۰

کمیت‌های بهینه: ارزش حالت‌های q



نویز = ۲/۰

ضریب کاهش = ۰/۹

پاداش مرحله‌ای = ۰

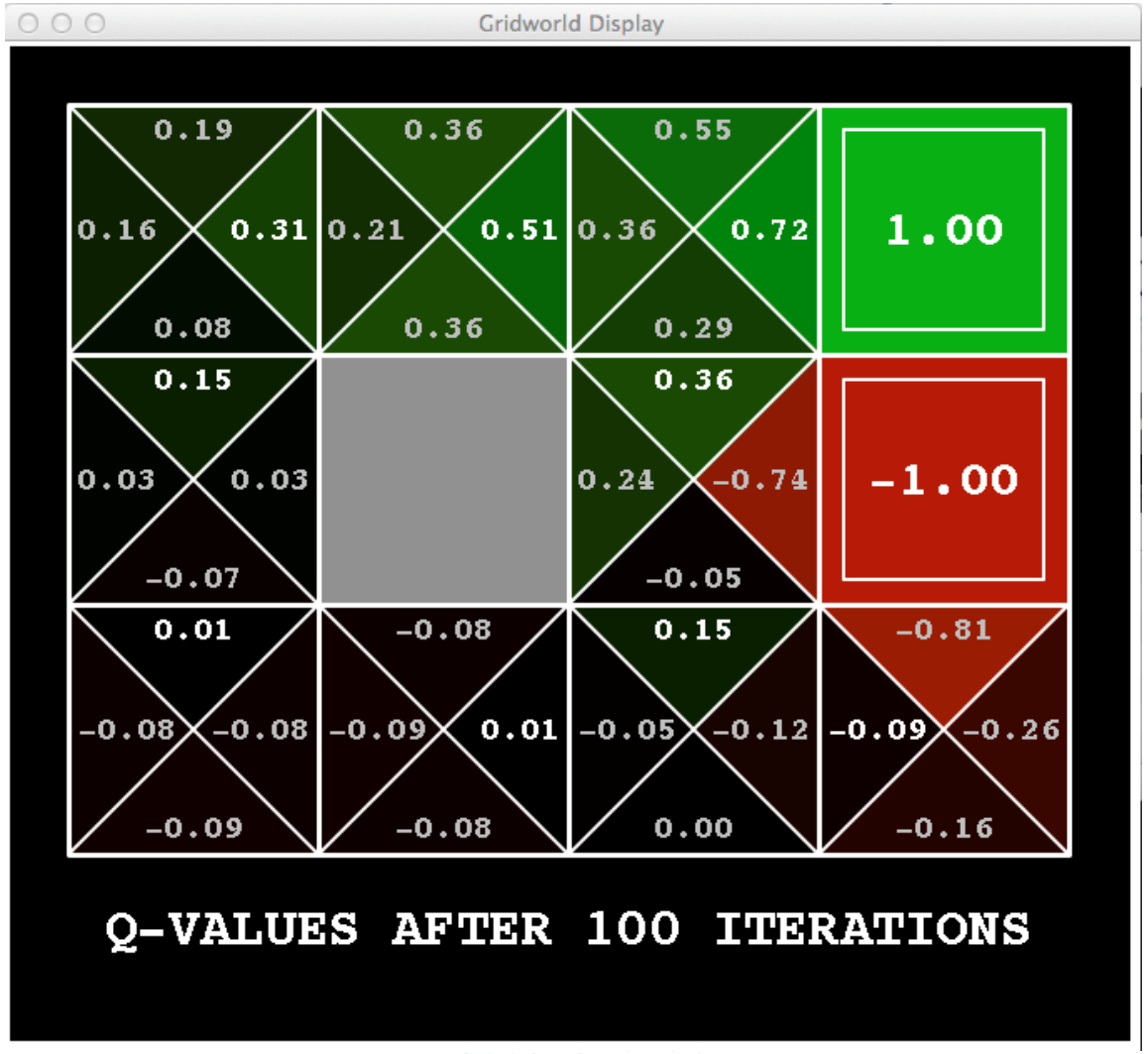
کمیت‌های بهینه: ارزش حالت‌ها

نویز = ۰/۲

ضریب کاهش = ۰/۹

پاداش مرحله‌ای = -۰/۱

کمیت‌های بهینه: ارزش حالت‌های q

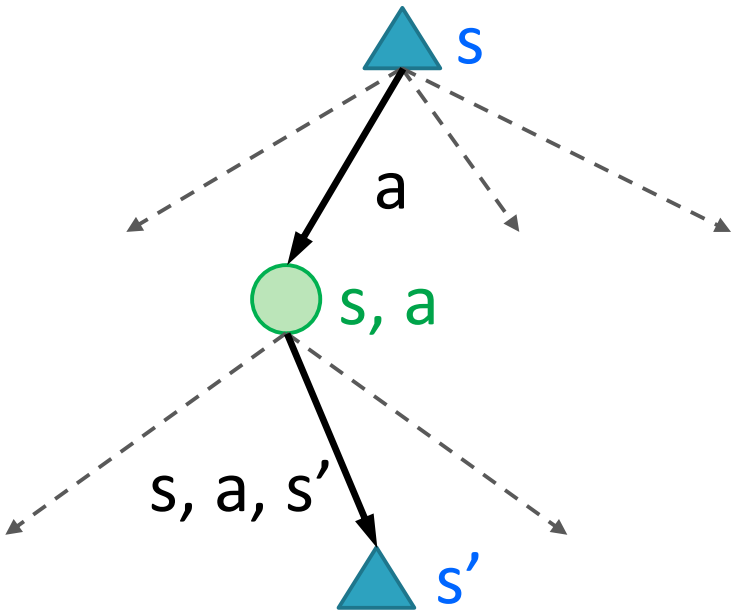


نویز = ۰/۲

ضریب کاهش = ۰/۹

پاداش مرحله‌ای = -۰/۱

محاسبه‌ی ارزش حالت‌ها



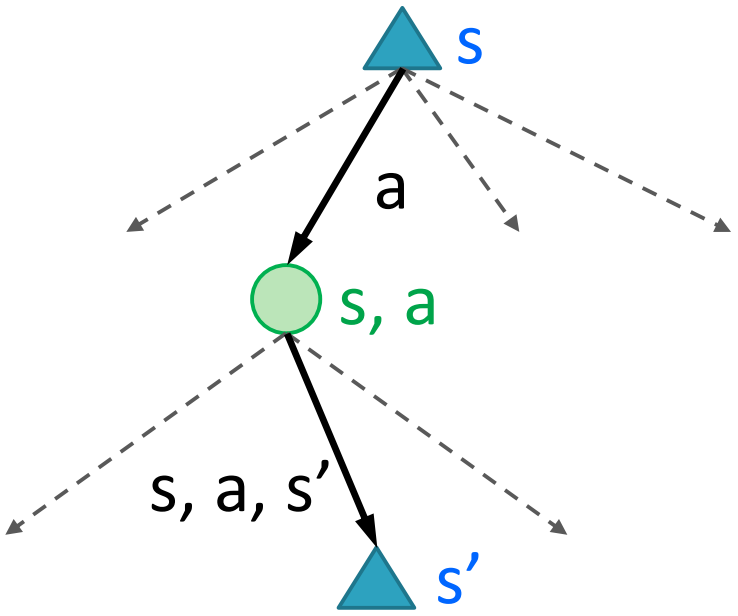
- عمل پایه‌ای: محاسبه‌ی ارزش یک حالت.
  - سودمندی مورد انتظار با انجام عمل بهینه
  - میانگین وزنی پاداش‌ها [با اعمال ضریب کاهش]
  - این همان چیزی است که الگوریتم expectimax محاسبه می‌کند!

- تعریف بازگشتی ارزش حالت‌ها.

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s\prime} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

محاسبه‌ی ارزش حالت‌ها



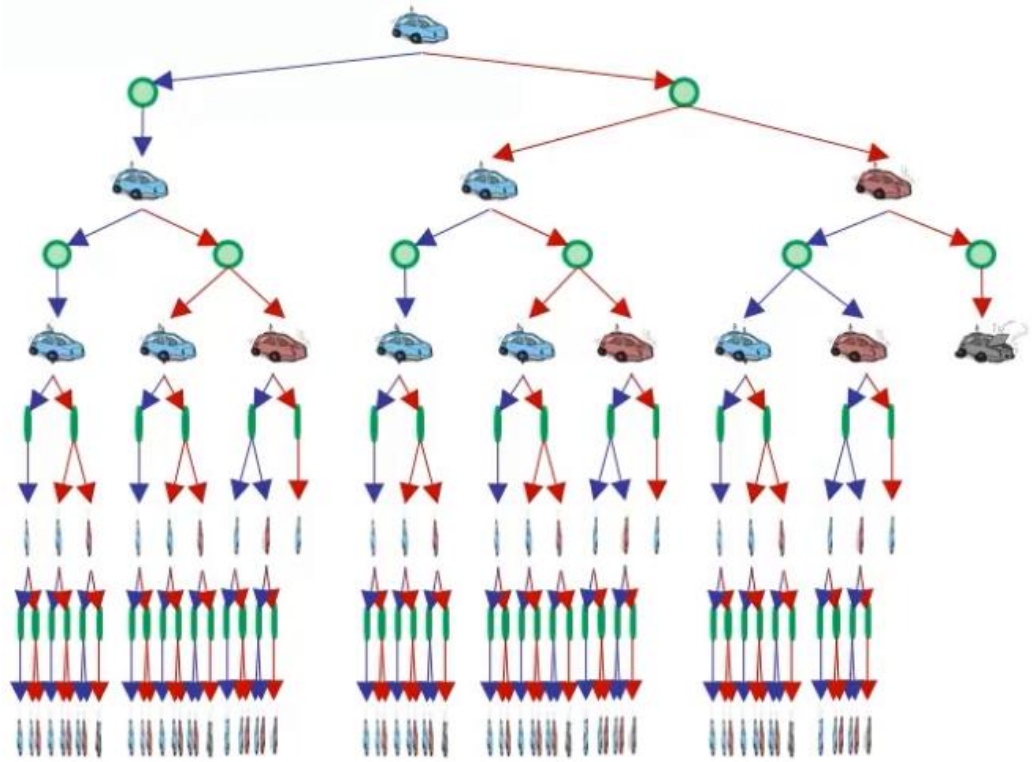
- عمل پایه‌ای: محاسبه‌ی ارزش یک حالت.
  - سودمندی مورد انتظار با انجام عمل بهینه
  - میانگین وزنی پاداش‌ها [با اعمال ضریب کاهش]
  - این همان چیزی است که الگوریتم expectimax محاسبه می‌کند!

- تعریف بازگشتی ارزش حالت‌ها.

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

معادله‌ی بلمن

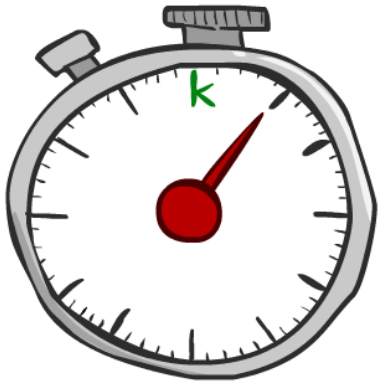
درخت جستجو: مسابقه‌ی اتومبیل رانی



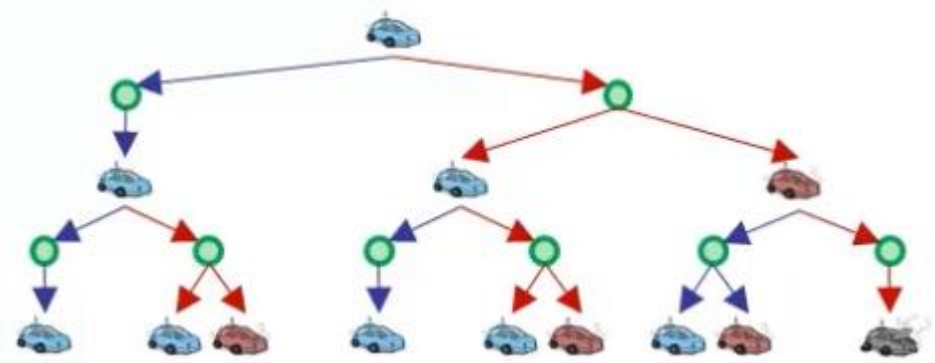
- استفاده از expectimax بسیار ناکارامد است!
- مشکل: حالت‌های تکراری.
  - ایده: هر کمیت مورد نیاز را فقط یک بار محاسبه کن.
- مشکل: درخت نامتناهی.
  - ایده: از محاسبات با عمق محدود استفاده کن و آن قدر عمق را افزایش بده تا تغییرات ناچیز گردند.
  - توجه: قسمت‌های عمیق درخت اهمیت ندارند [ضریب کاهش]

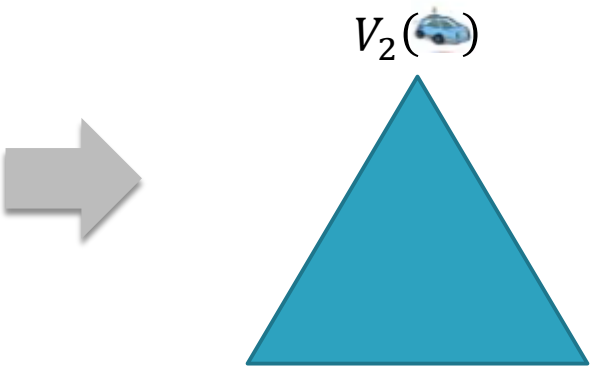
ارزش‌های با زمان محدود

- ایده‌ی کلیدی. ارزش‌های با زمان محدود!



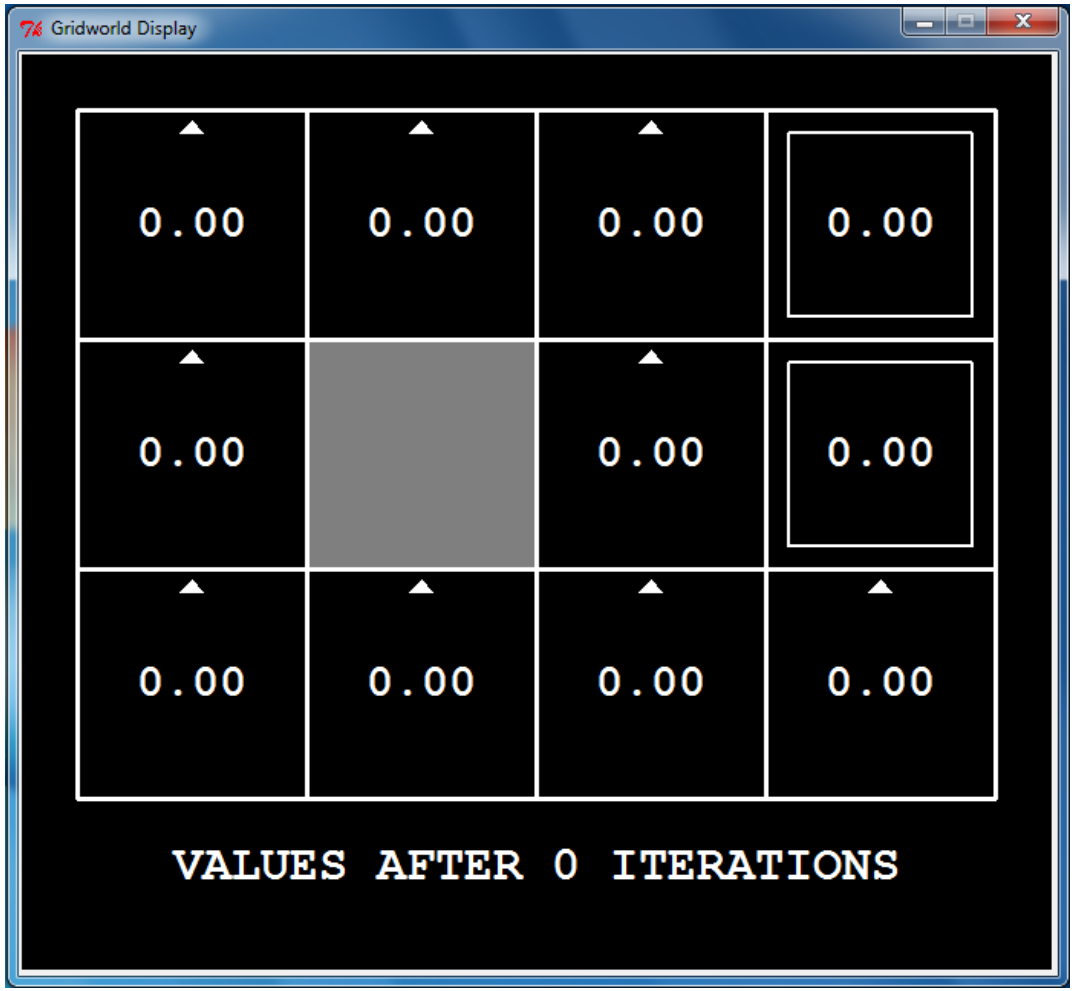
- تعریف $V_k(s)$ به عنوان ارزش بهینه s اگر بازی پس از k مرحله‌ی دیگر خاتمه یابد.
  - معادل اجرای expectimax با عمق محدود k





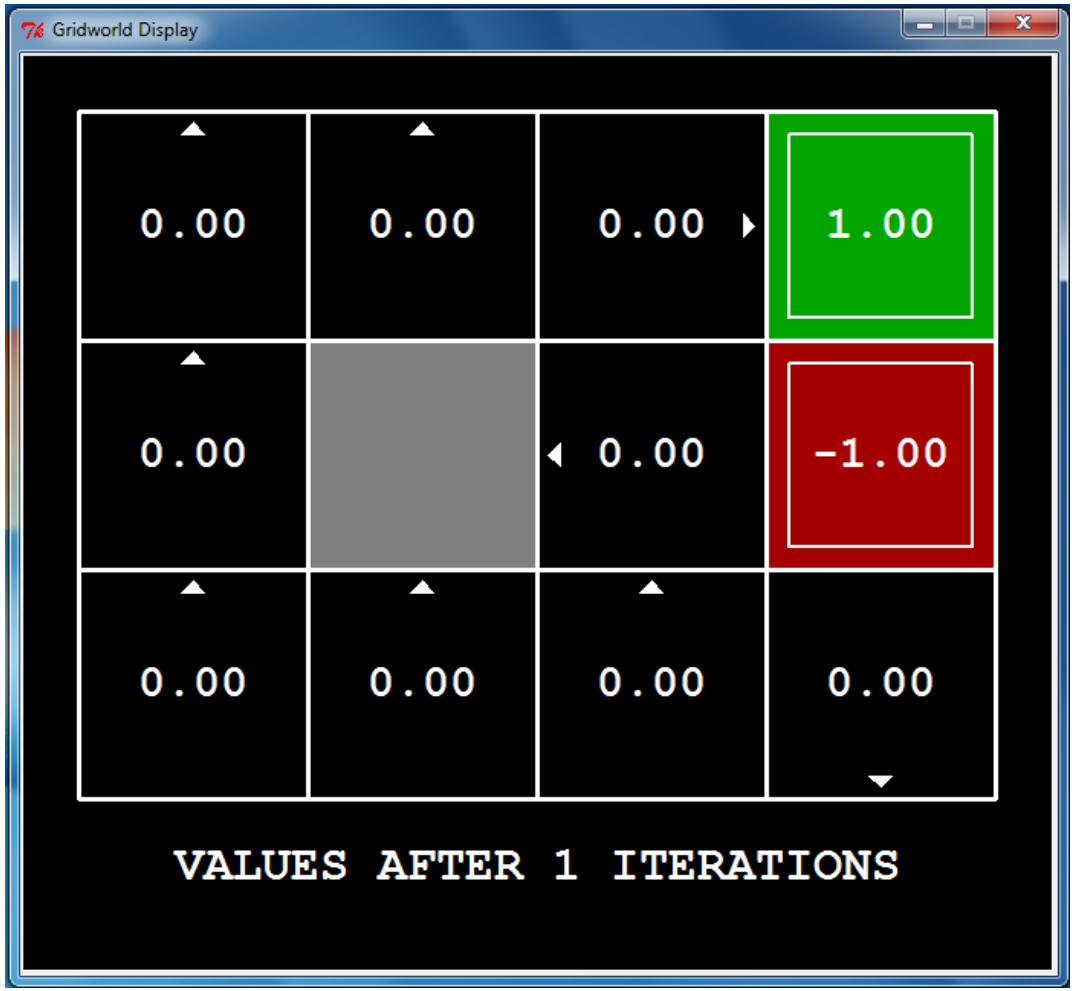
اجرای نمایشی

python gridworld.py -a value -i 100 -v



اجرای نمایشی

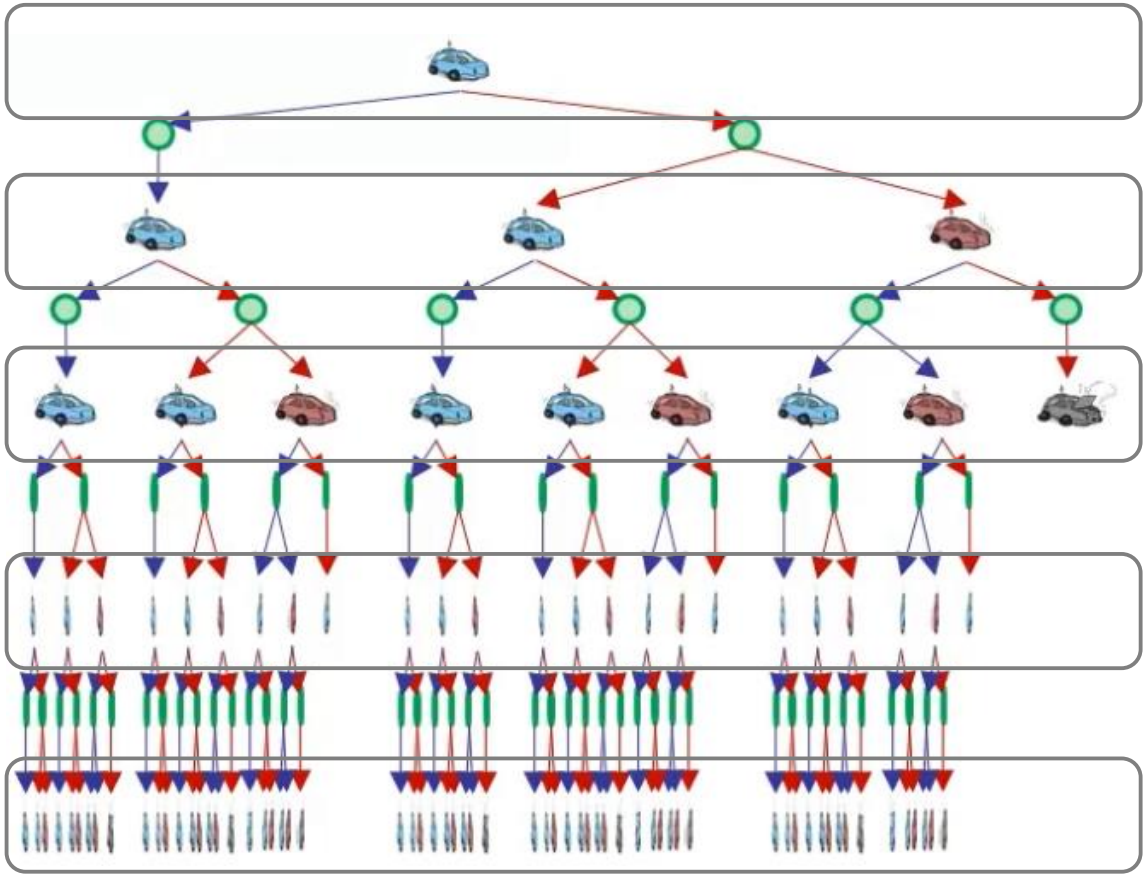

```
python gridworld.py -a value -i 100 -v
```



اجرای نمایشی


```
python gridworld.py -a value -i 100 -v
```

محاسبه‌ی ارزش‌ها



$V_4(\ ) \quad V_4(\ ) \quad V_4(\ )$

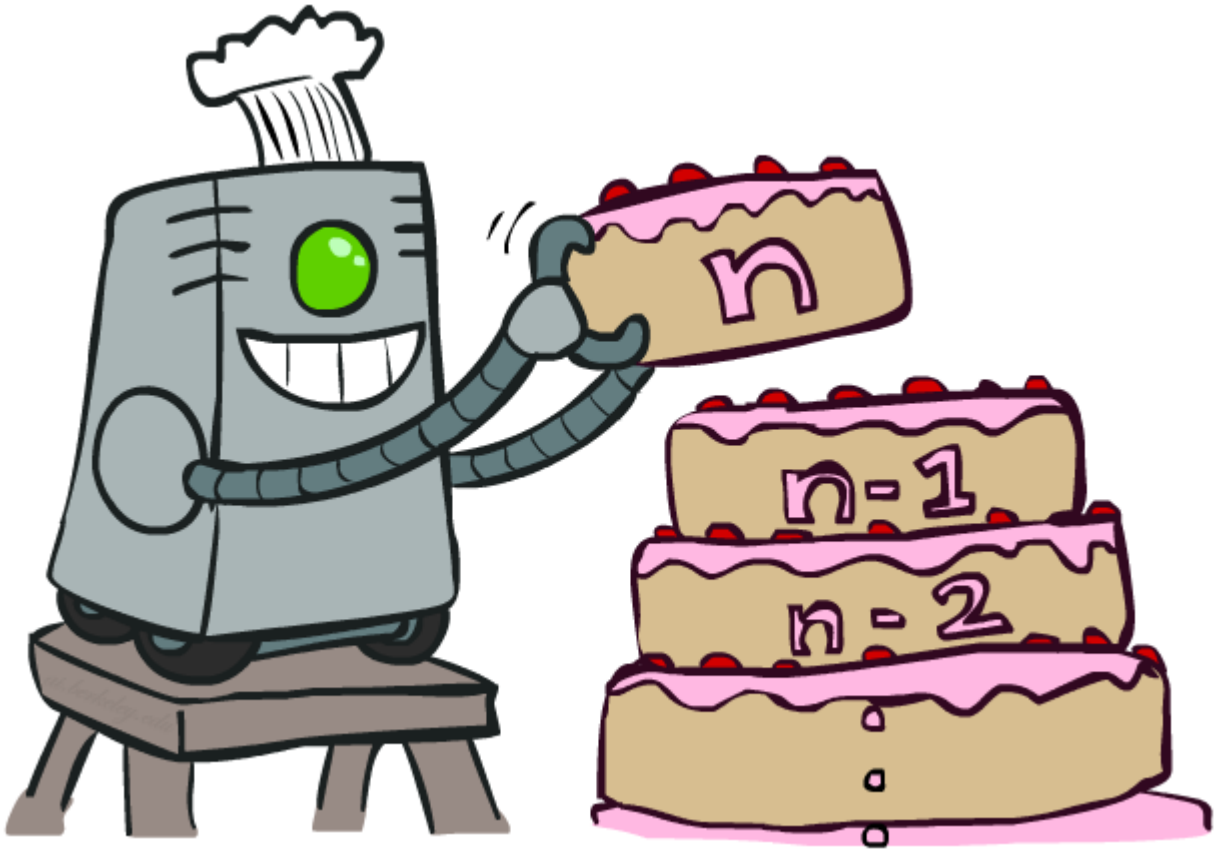
$V_3(\ ) \quad V_3(\ ) \quad V_3(\ )$

$V_2(\ ) \quad V_2(\ ) \quad V_2(\ )$

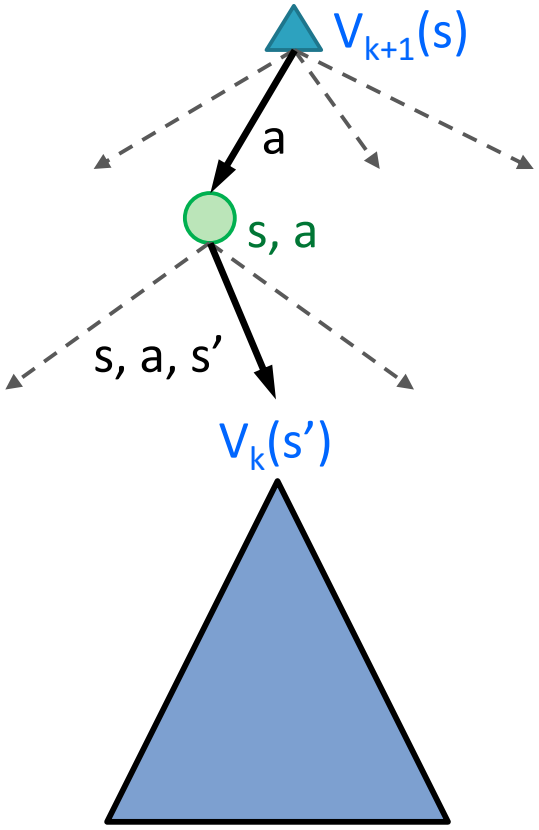
$V_1(\ ) \quad V_1(\ ) \quad V_1(\ )$

$V_0(\ ) \quad V_0(\ ) \quad V_0(\ )$

الگوریتم تکرار مقدار



الگوریتم تکرار مقدار



- الگوریتم تکرار مقدار.
  - با بردار $V_0(s) = 0$ شروع کن.
  - با داشتن بردار $V_k(s)$، برای هر حالت یک لایه از محاسبات expectimax را انجام بده:

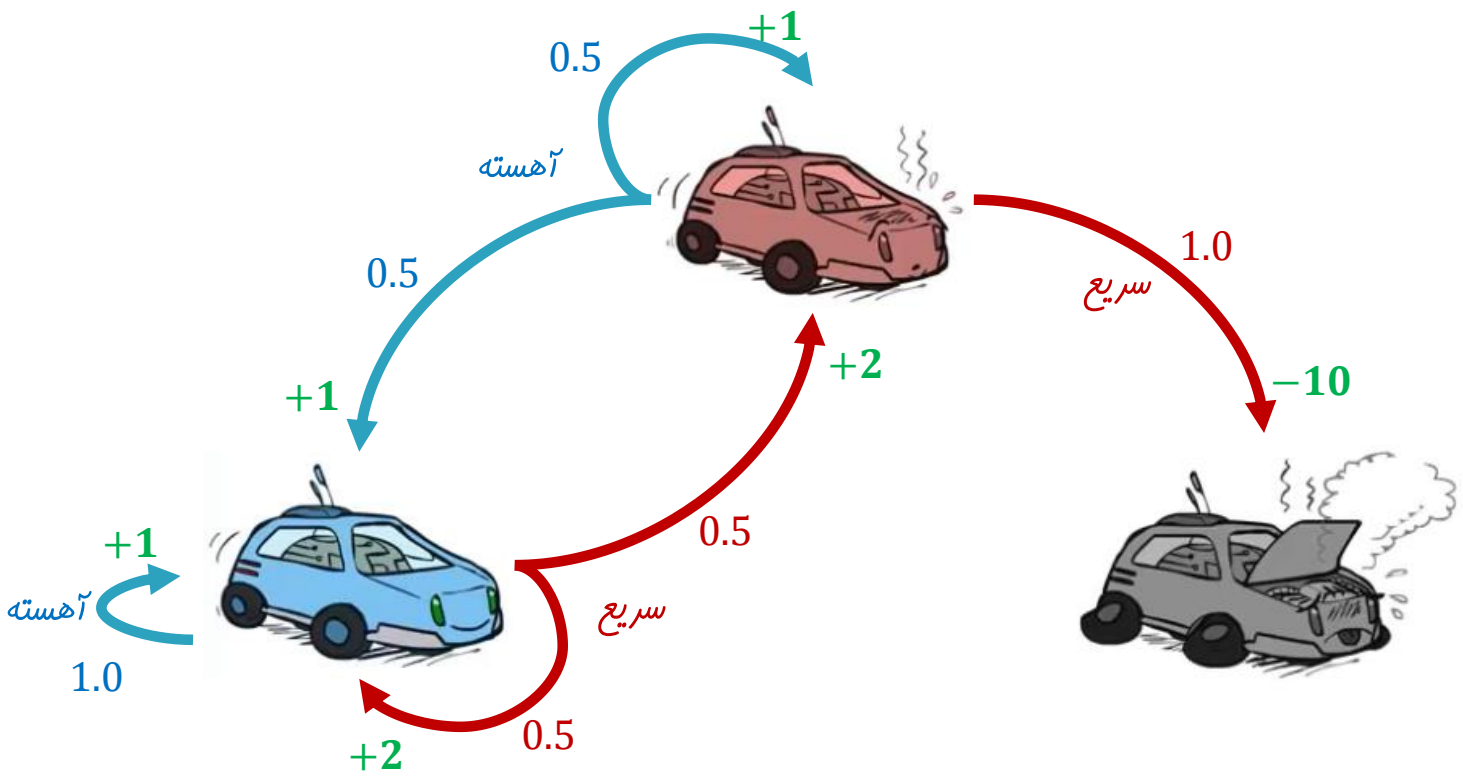
$$V_{k+1}(s) = \max_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V_k(s')]$$

  - محاسبات بالا را تا زمان همگرا شدن مقادیر تکرار کن.

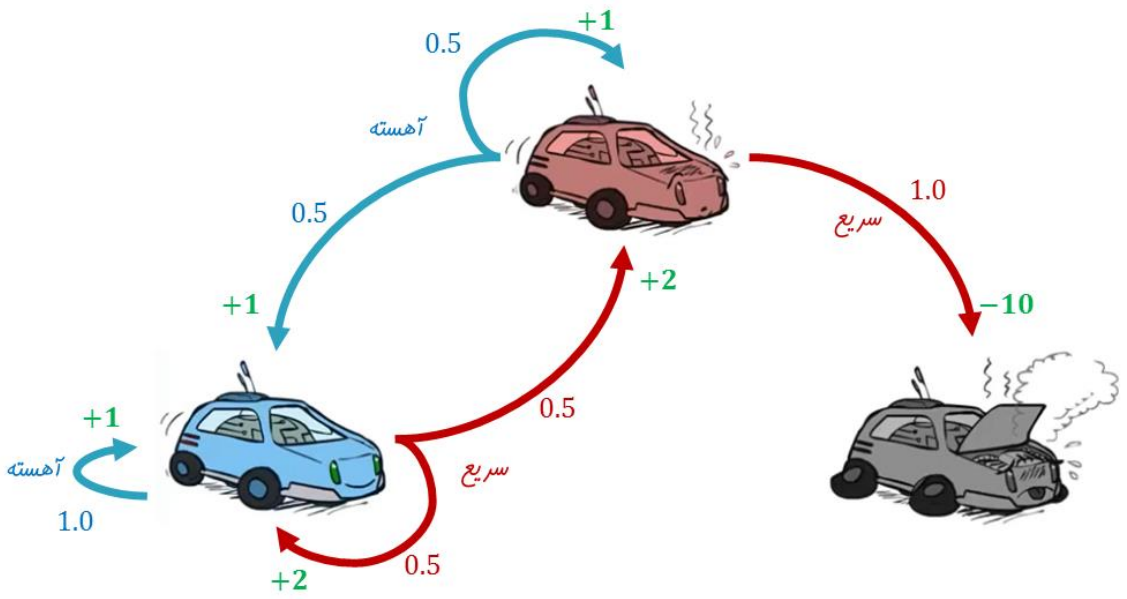
- تحلیل. پیچیدگی زمانی هر تکرار: $O(S^2A)$

- قضیه. این الگوریتم به یک مجموعه‌ی یکتا از مقادیر بهینه همگرا می‌شود:
  - سیاست ممکن است خیلی زودتر از مقادیر همگرا شود.

الگوریتم تکرار مقدار: مثال



الگوریتم تکرار مقدار: مثال



$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

$\gamma = 1$

توجه. همگرا شدن سیاست پس از یک تکرار

V_2	3.5	2.5	0
V_1	2	1	0
V_0	0	0	0

همگرایی

- س. چگونه می‌توان فهمید مقادیر همگرا خواهند شد؟

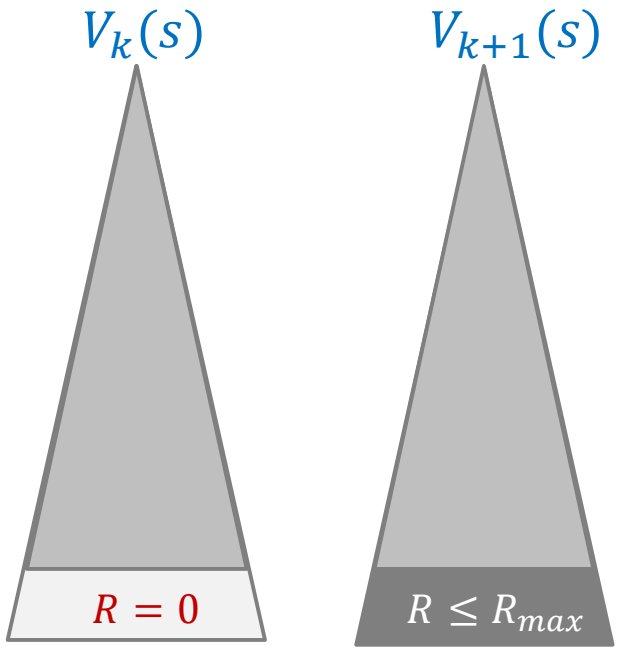
$$K \to \infty \Rightarrow V_k(s) \to V^*(s)$$

- حالت ۱: اگر درخت دارای عمق محدود M باشد،
  - در این صورت بردار V_M شامل مقادیر واقعی است بدون آن که درخت برش خورده باشد.

همگرایی (ادامه)

- س. چگونه می‌توان فهمید مقادیر همگرا خواهند شد؟

- حالت ۲: اگر ضریب کاهش کوچک‌تر از ۱ باشد.
  - طرح کلی: به ازای هر حالت، مقادیر V_k و V_{k+1} را می‌توان به عنوان نتیجه‌ی یک جستجوی expectimax با محدوده‌ی عمقی k+1 در دو درخت جستجوی تقریباً یکسان در نظر گرفت.
  - تفاوت در این است که در آخرین لایه، V_{k+1} شامل پاداش‌های واقعی است اما V_k شامل پاداش‌های مساوی صفر است.
  - پاداش‌ها در این لایه همگی حداکثر R_{max} هستند.
  - و حداقل برابر با R_{min} هستند.
  - اما مقادیر این لایه در مقدار γ^k ضرب می‌شوند.
  - بنابراین اختلاف V_k و V_{k+1} حداکثر برابر با $\gamma^k \max|R|$ است.
  - در نتیجه با افزایش k، مقادیر همگرا می‌شوند.

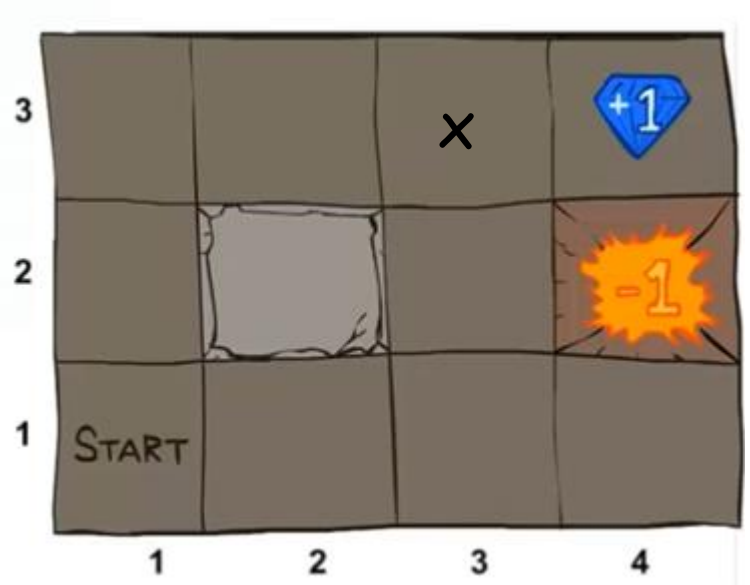


$$|V_{k+1}(s) - V_k(s)| \leq \gamma^k \cdot \max |R|$$

$$\Rightarrow \lim_{k \to \infty} |V_{k+1}(s) - V_k(s)| = 0$$

پرسش کلاسی

$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$



$$\gamma = \frac{1}{2}, \qquad R(s,a,s') = -0.04, \qquad V_0(S) = 0$$

$$V_1(x) = 0.36 \qquad V_2(x) = 0.376$$