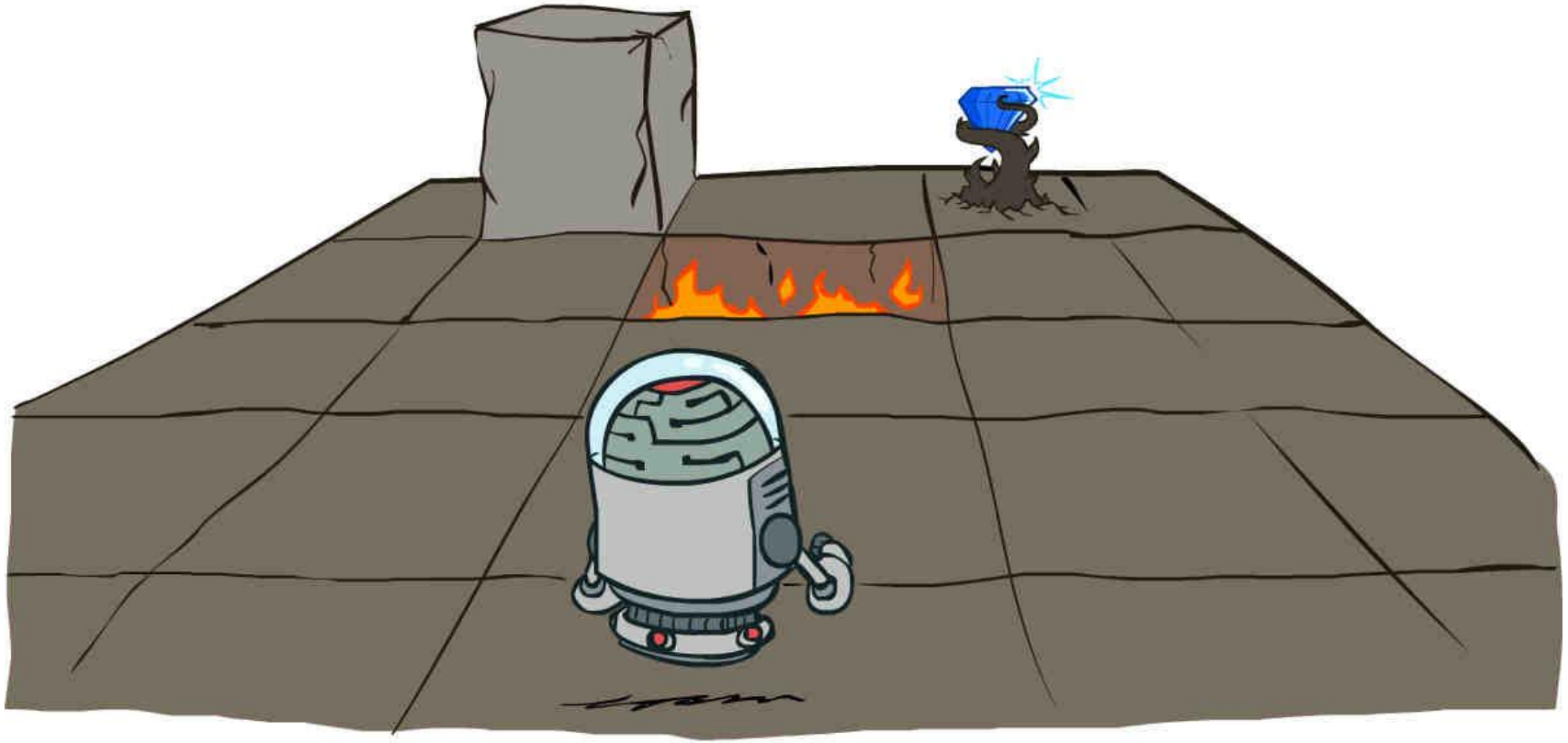
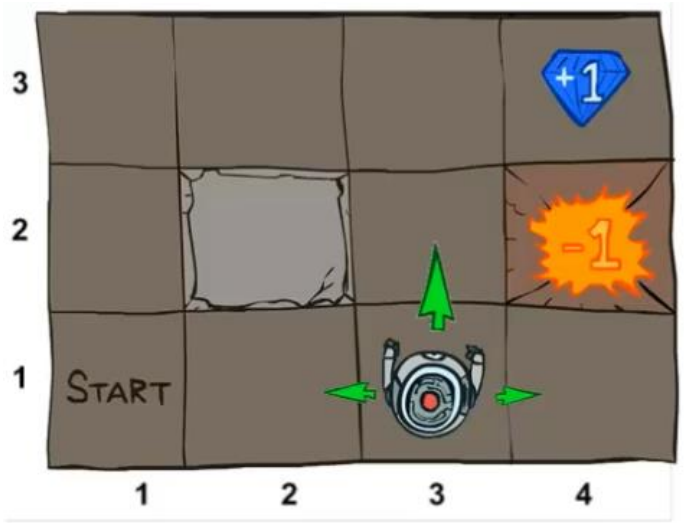
جستجوی غیر قطعی:
فرایند تصمیم مارکوف II

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

فرایند تصمیم مارکوف



مثال: محیط گرید



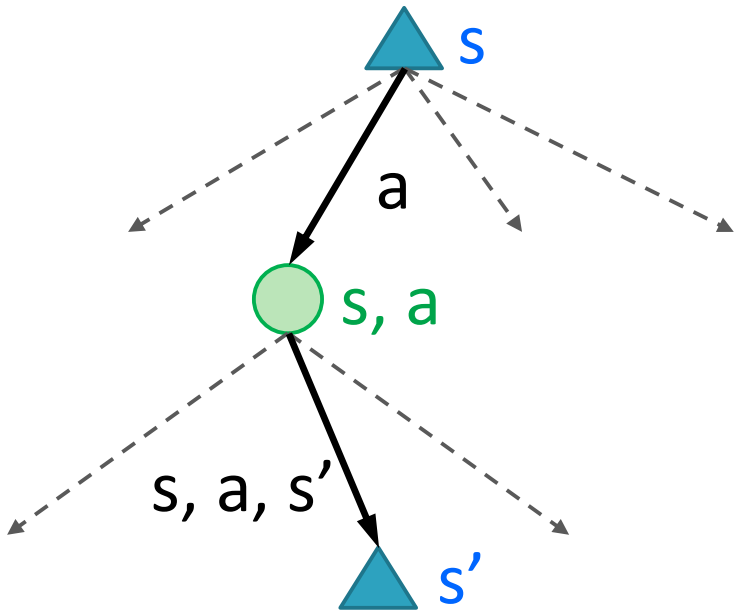
- یک مسئله شبیه مسیریابی.
  - عامل در یک محیط گرید عمل می‌کند.
  - دیوارها مانع از حرکت عامل می‌شوند.

- حرکت‌های دارای نویز.
  - ۸۰ درصد مواقع انجام عمل North باعث حرکت عامل به خانه‌ی 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]

- هدف. بیشینه‌سازی مجموع (کاهش یافته) پاداش‌های دریافتی.

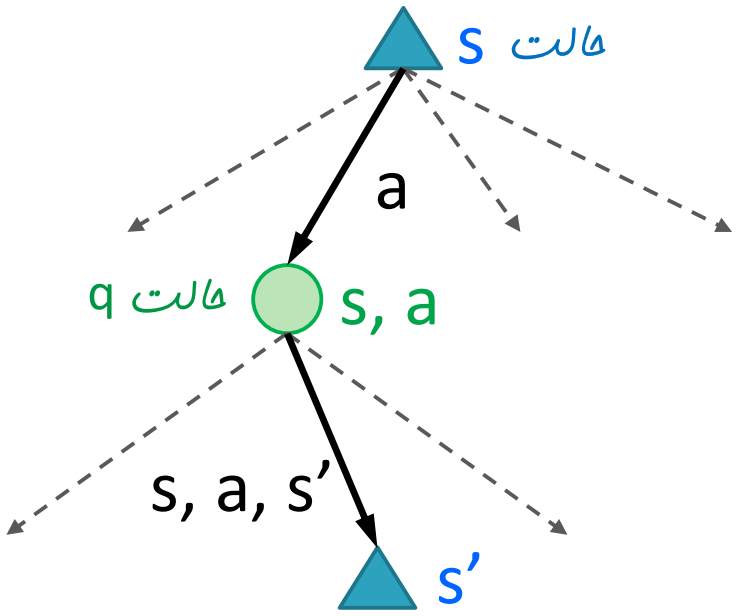
فرایند تصمیم مارکوف (یادآوری)



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

- کمیت‌های مهم.
  - سیاست: انتخاب یک عمل به ازای هر حالت.
  - سودمندی: مجموع کاهش یافته‌ی پاداش‌ها.
  - ارزش حالت‌ها: سودمندی مورد انتظار با شروع از یک حالت [گره ماکزیمم]
  - مقادیر Q: سودمندی مورد انتظار با شروع از یک حالت q [گره شانس]

کمیت‌های بهینه (یادآوری)

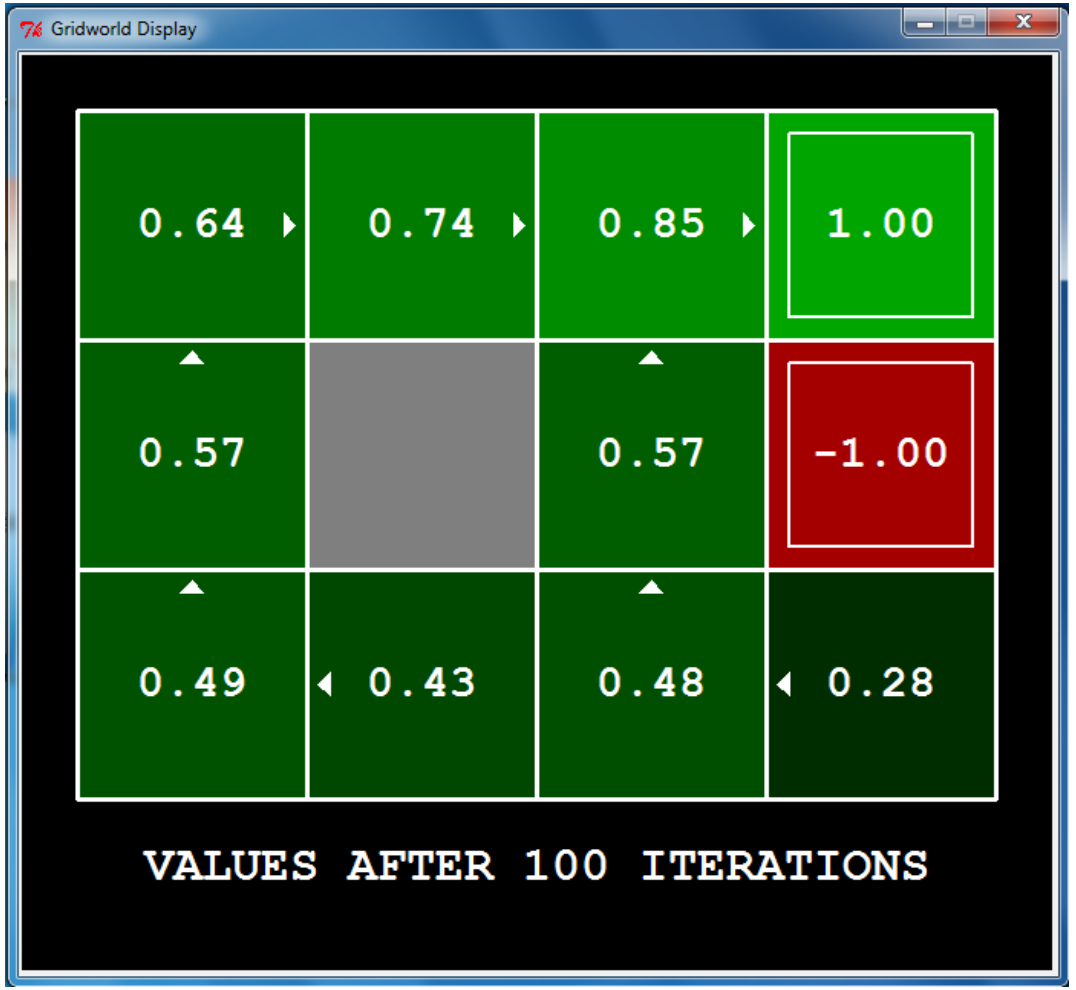


- $V^*(s)$ = ارزش (سودمندی) حالت s:
  - سودمندی مورد انتظار با شروع از s و عمل کردن به صورت بهینه
- $Q^*(s, a)$ = ارزش (سودمندی) حالت q:
  - سودمندی مورد انتظار با شروع از s و انتخاب عمل a و عمل کردن به صورت بهینه [از این به بعد]
- سیاست بهینه.
  - $\pi^*(s)$ = عمل بهینه در حالت s.

[اجرای نمایشی: مقادیر در محیط گرید]

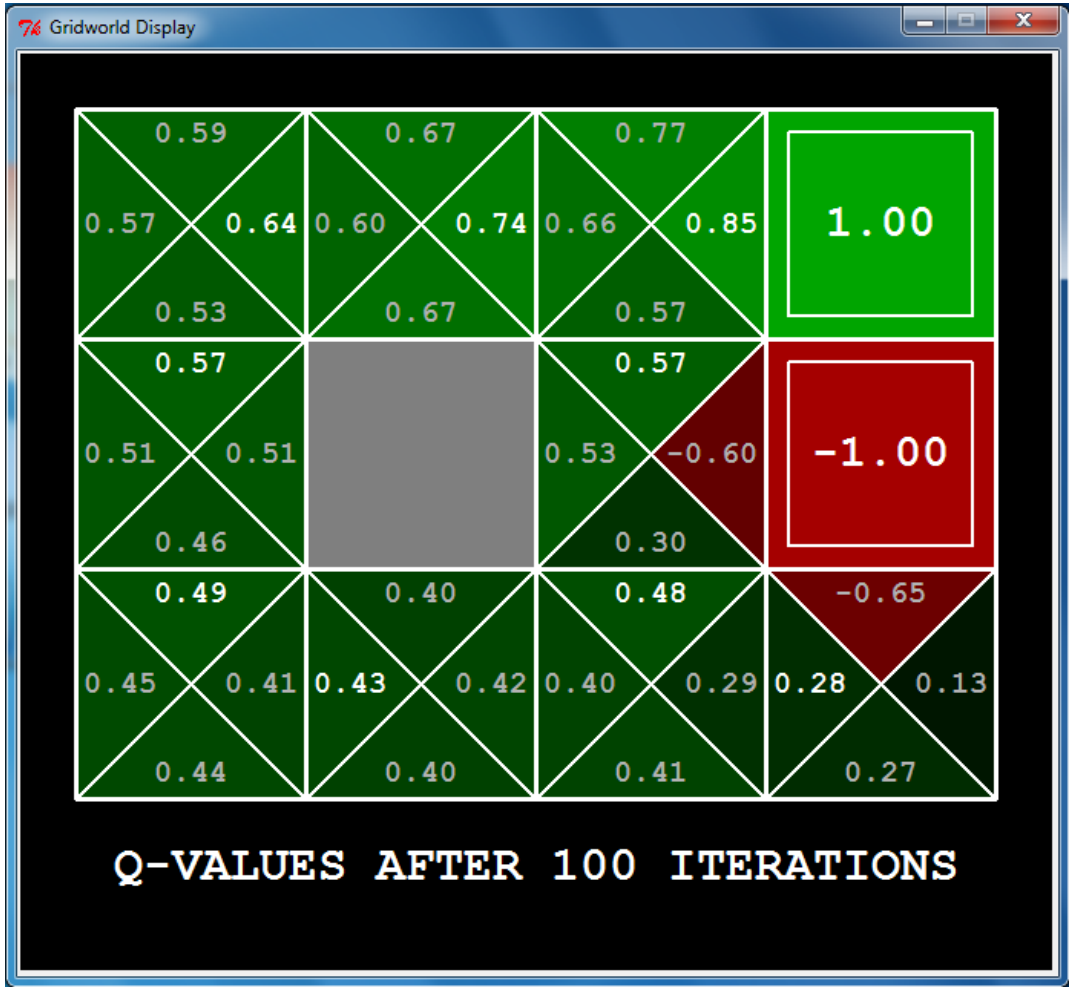
کمیت‌های بهینه: ارزش حالت‌ها


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



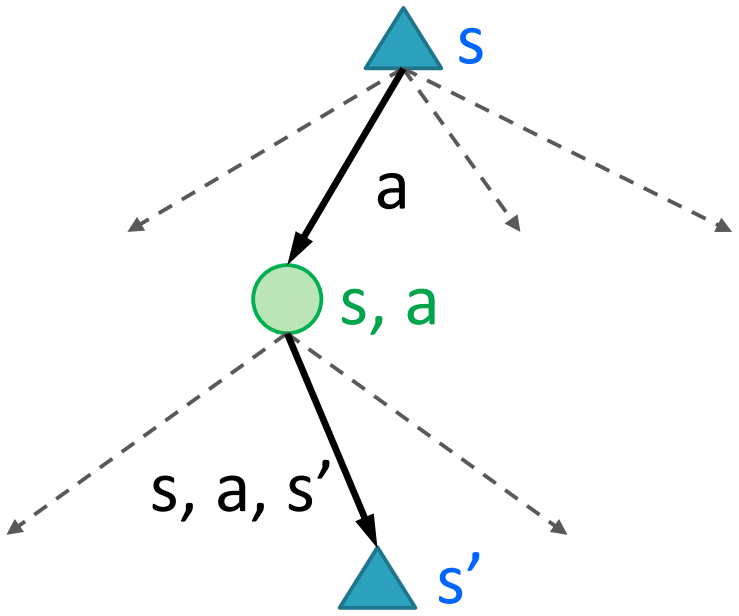
کمیت‌های بهینه: ارزش حالت‌های q


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



معادلات بلمن

معادلات بلمن: محاسبه‌ی ارزش حالت‌ها



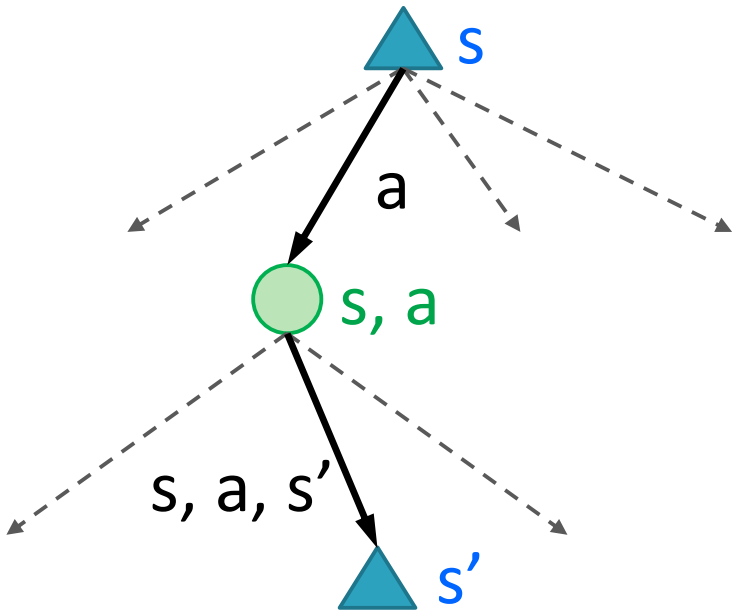
- عمل پایه‌ای: محاسبه‌ی ارزش یک حالت.
  - سودمندی مورد انتظار با انجام عمل بهینه
  - میانگین وزنی پاداش‌ها [با اعمال ضریب کاهش]
  - این همان چیزی است که الگوریتم expectimax محاسبه می‌کند!

- تعریف بازگشتی ارزش حالت‌ها.

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

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

معادلات بلمن: محاسبه‌ی ارزش حالت‌ها



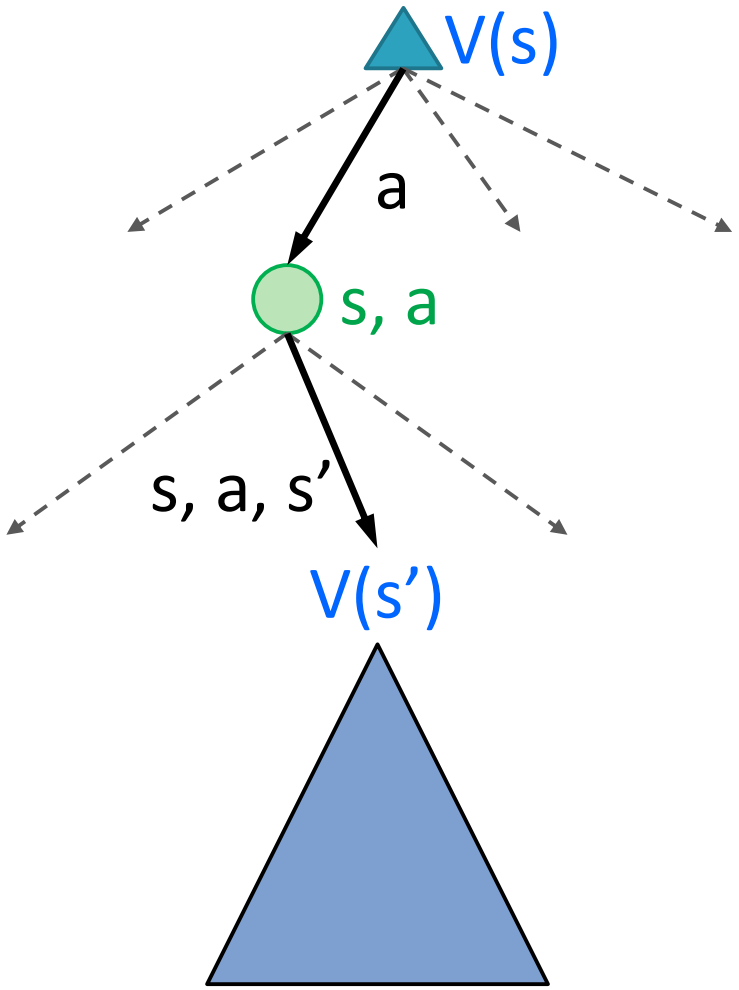
- عمل پایه‌ای: محاسبه‌ی ارزش یک حالت.
  - سودمندی مورد انتظار با انجام عمل بهینه
  - میانگین وزنی پاداش‌ها [با اعمال ضریب کاهش]
  - این همان چیزی است که الگوریتم expectimax محاسبه می‌کند!

- تعریف بازگشتی ارزش حالت‌ها.

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

معادله‌ی بلمن

الگوریتم تکرار مقدار



- معادلات بلمن مقادیر بهینه را تعریف می‌کنند:

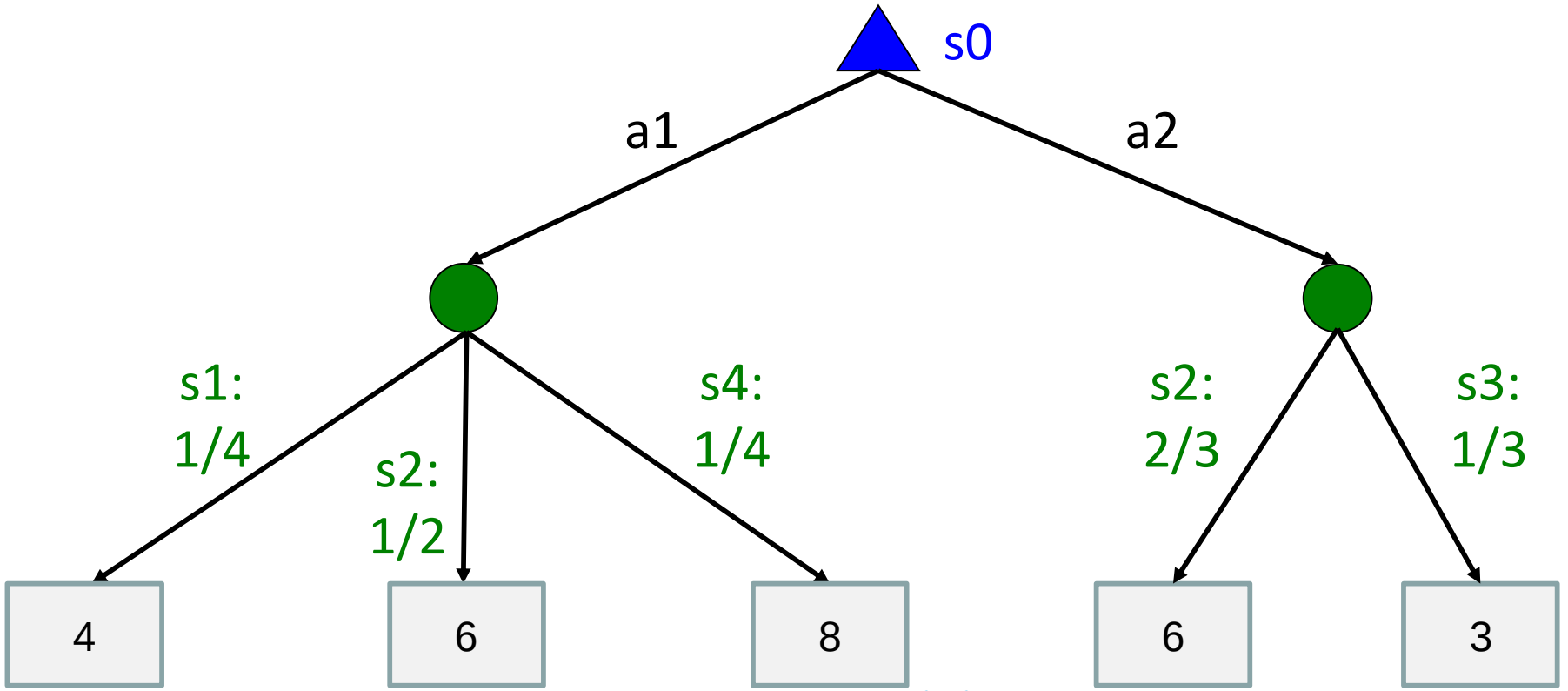
$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- الگوریتم تکرار مقدار، مقادیر بهینه را محاسبه می‌کند:

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

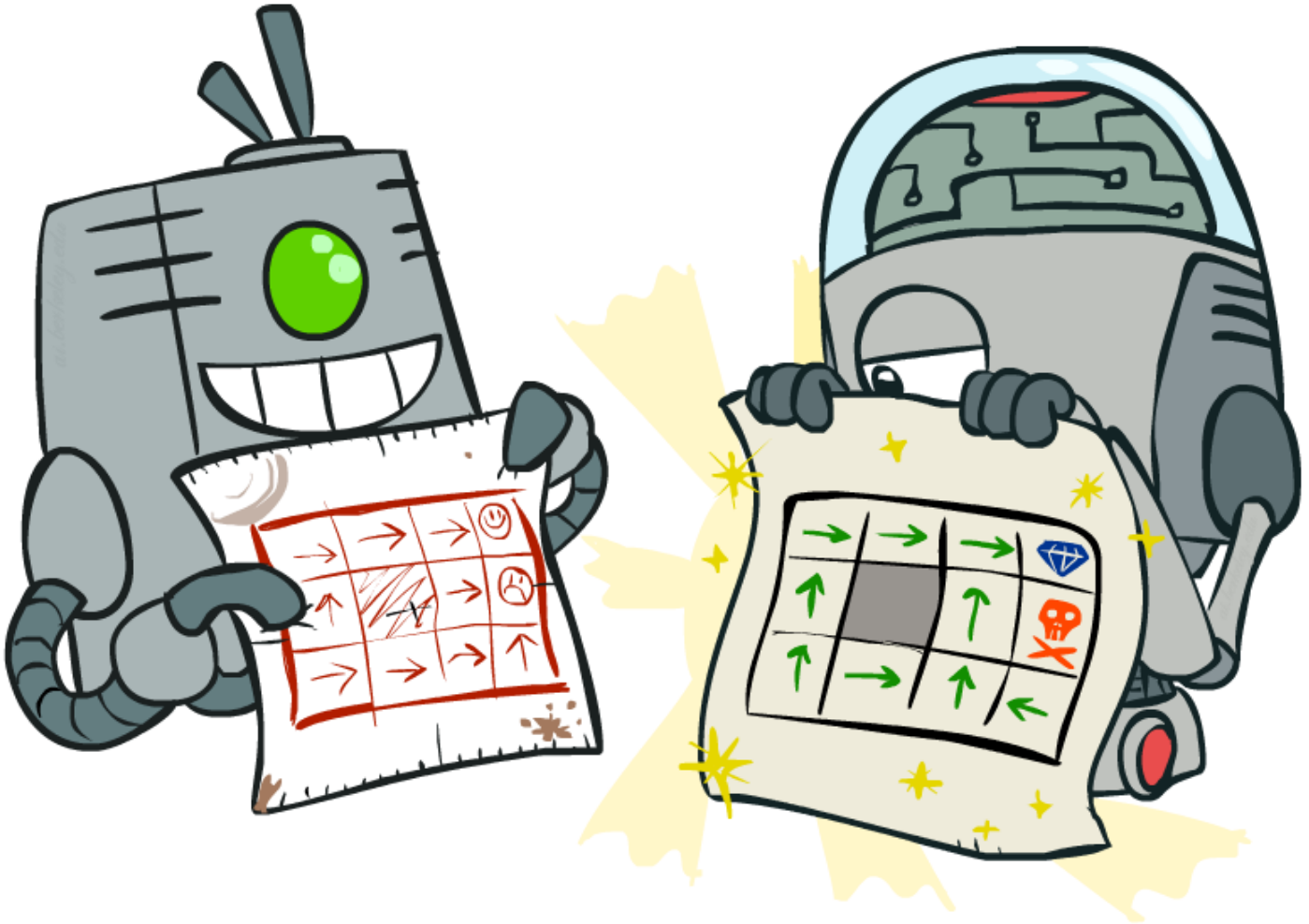
پرسش کلاسی: الگوریتم expectimax

- س. سودمندی مورد انتظار در حالت s0 چیست؟
- س. عمل بهینه در حالت s0 کدام است؟

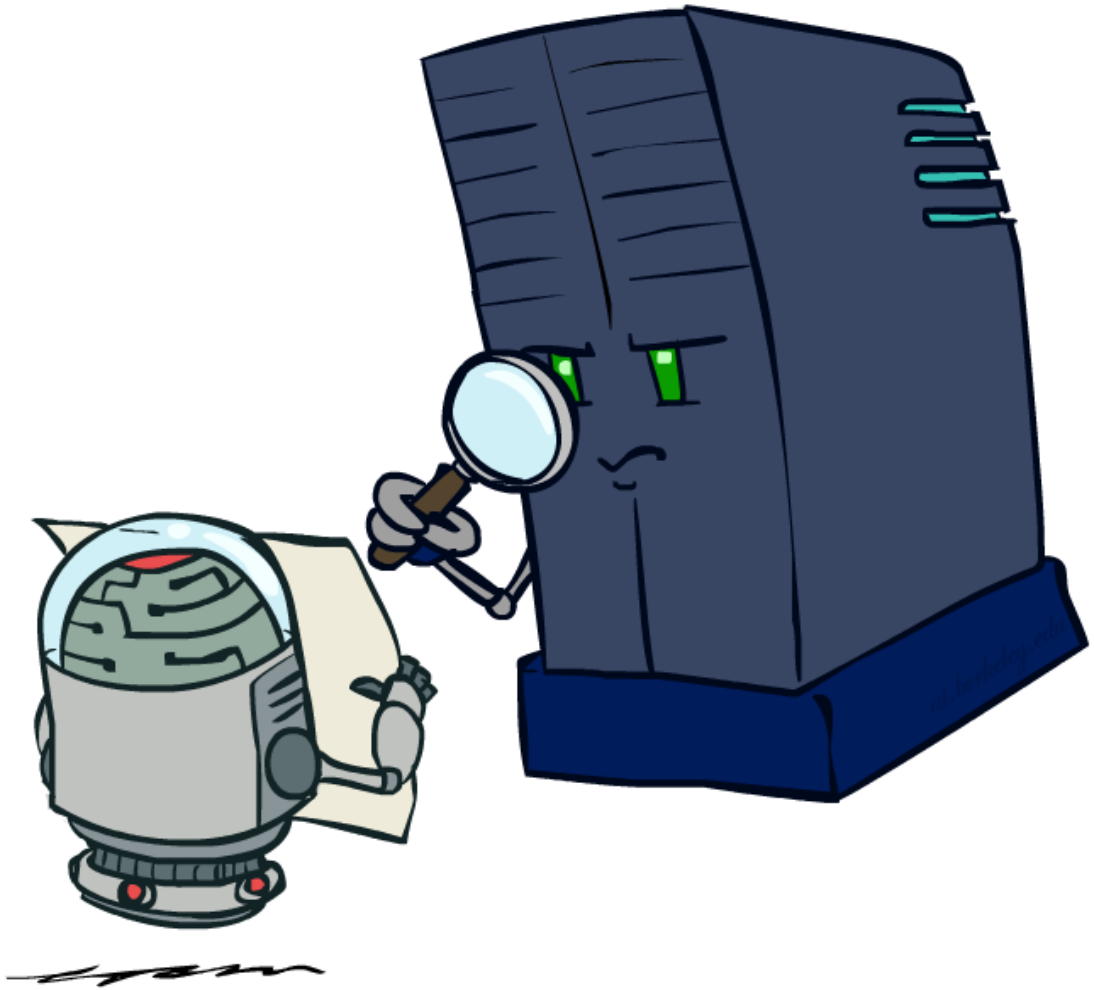


www.snrazavi.ir

روش‌های مبتنی بر سیاست

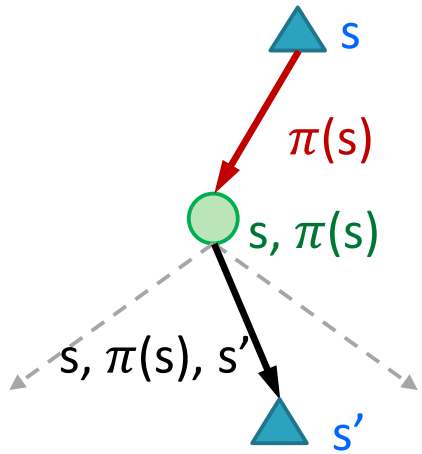


ارزیابی سیاست

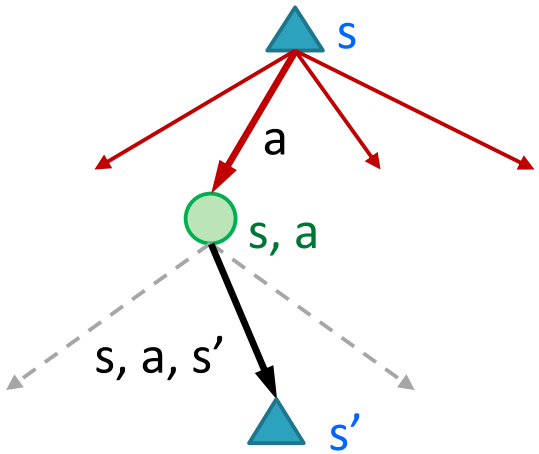


سیاست‌های ثابت

بر طبق سیاست π عمل کن

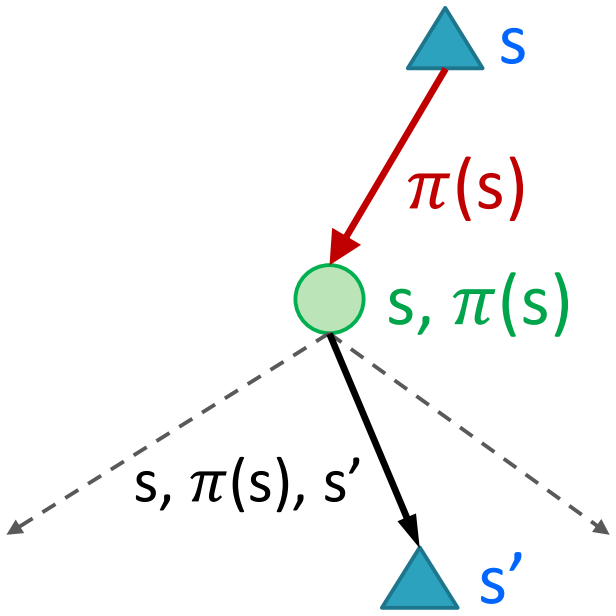


عمل بهینه را انجام بده



- درخت‌های expectimax برای محاسبه‌ی مقادیر بهینه، بر روی تمام عمل‌ها ماکزیمم می‌گیرند.
- اگر از سیاست ثابت $\pi(s)$ استفاده کنیم، درخت ساده‌تر می‌شود. [یک عمل به ازای هر حالت]
  - ... بنابراین ارزش حالت‌ها در درخت بستگی به سیاست انتخاب شده دارد.

محاسبه‌ی ارزش حالت‌ها برای سیاست‌های ثابت



- یک عمل پایه‌ای دیگر. محاسبه‌ی ارزش حالت **s** تحت یک سیاست ثابت [معمولاً غیر بهینه]

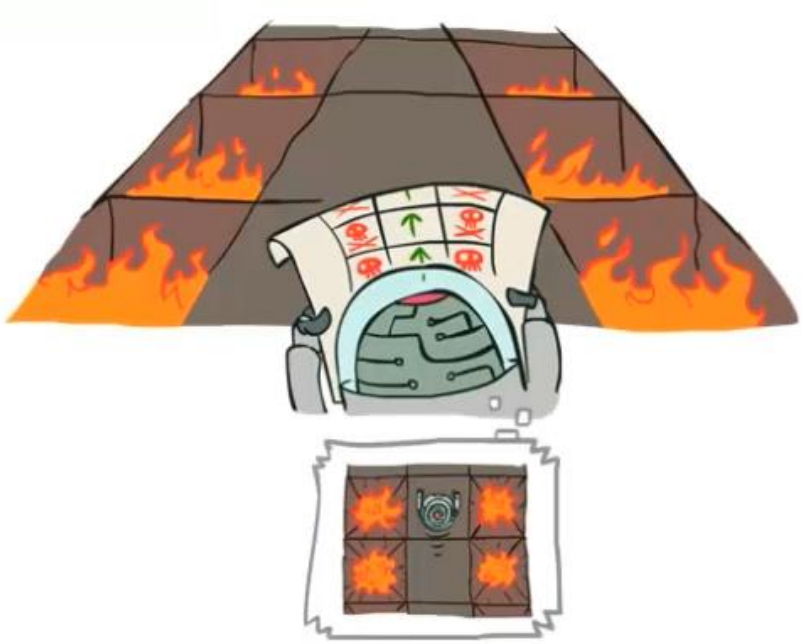
- تعریف ارزش حالت **s** تحت سیاست ثابت π:
  - $V^{\pi}(s)$: سودمندی موردانتظار با شروع از **s** و دنبال کردن π

- تعریف بازگشتی.

$$V^{\pi}(s) = \sum_{S'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

مثال: ارزیابی سیاست

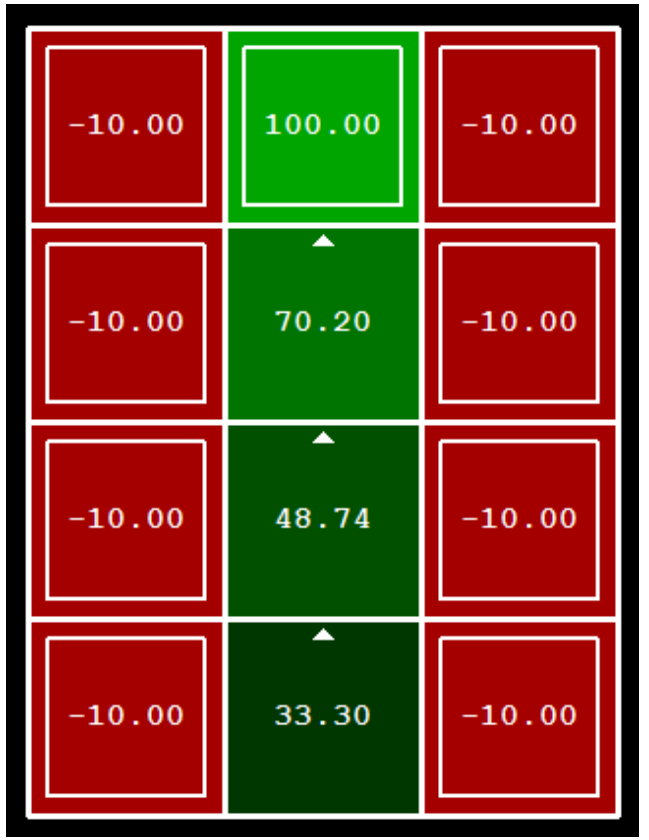
همیشه مستقیم برو



همیشه به راست برو

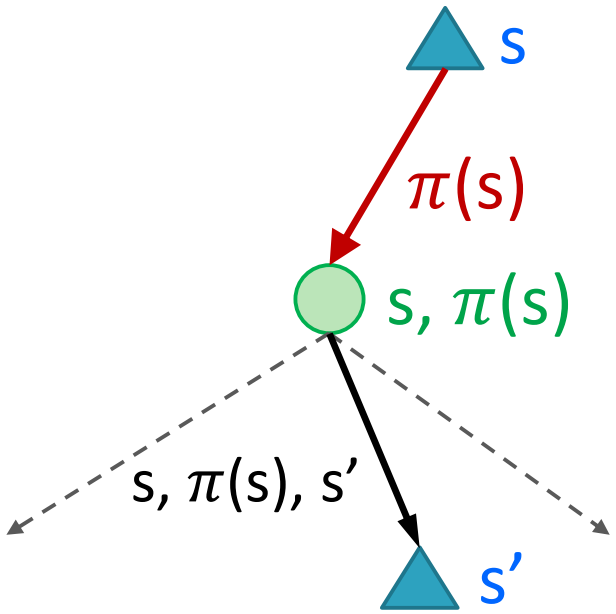
مثال: ارزیابی سیاست

همیشه مستقیم برو



همیشه به راست برو

ارزیابی سیاست



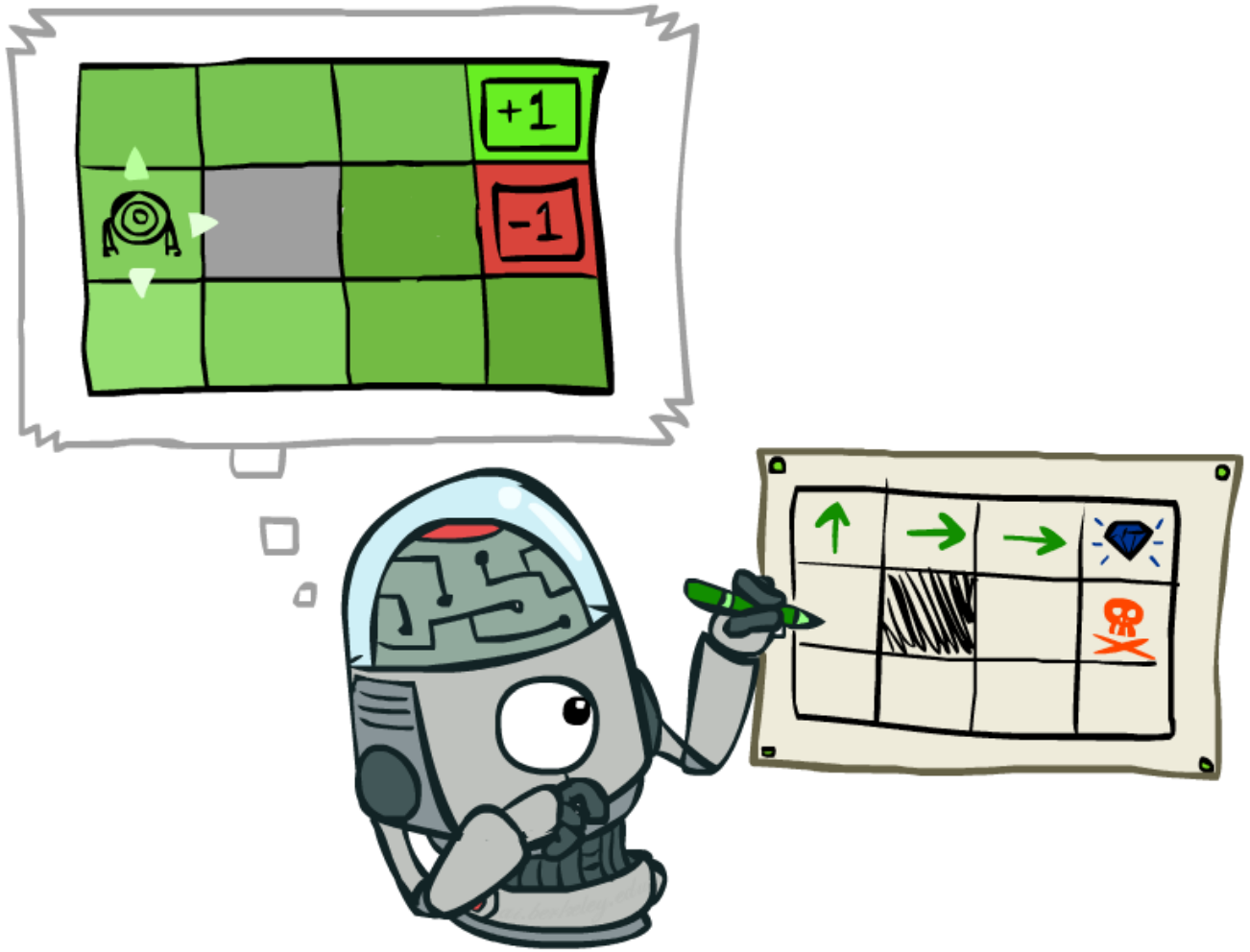
- س. چگونه می‌توان مقادیر V را برای سیاست ثابت π محاسبه نمود؟
- ایده‌ی ۱. تبدیل تساوی در معادلات بلمن به انتساب. [مانند الگوریتم تکرار مقدار]

$$V_0^{\pi}(s) \leftarrow 0$$

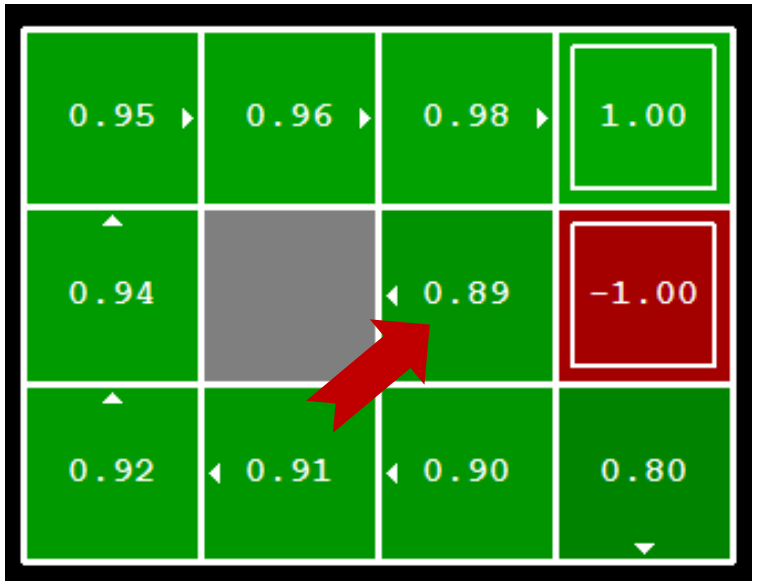
$$V_{k+1}^{\pi}(s) \leftarrow \sum_{S'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$$

- کارایی. پیچیدگی زمانی به ازای هر تکرار $O(S^2)$.
- ایده‌ی ۲. بدون وجود ماکزیمم، معادلات بلمن یک دستگاه معادلات خطی تشکیل می‌دهند.
  - می‌توانید برای حل آن از متلب (یا حل کننده‌ی سیستم‌های خطی مورد علاقه خود) استفاده کنید.

استخراج سیاست



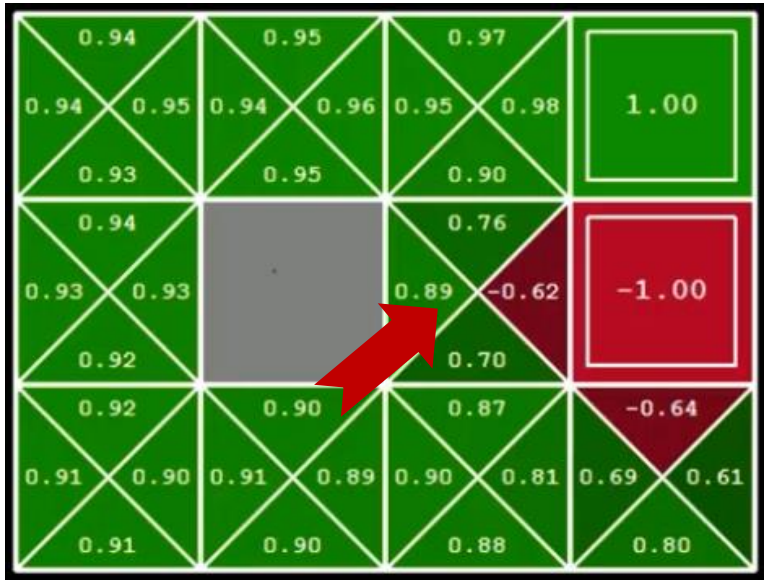
محاسبه‌ی عمل از روی ارزش حالت‌ها



- فرض کنید ارزش بهینه‌ی حالت‌ها را در اختیار داریم.
- س. در هر حالت باید چه عملی را انجام دهیم؟
  - خیلی واضح نیست!
- ج. انجام یک لایه از محاسبات expectimax!

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

محاسبه‌ی عمل از روی ارزش حالت‌های q

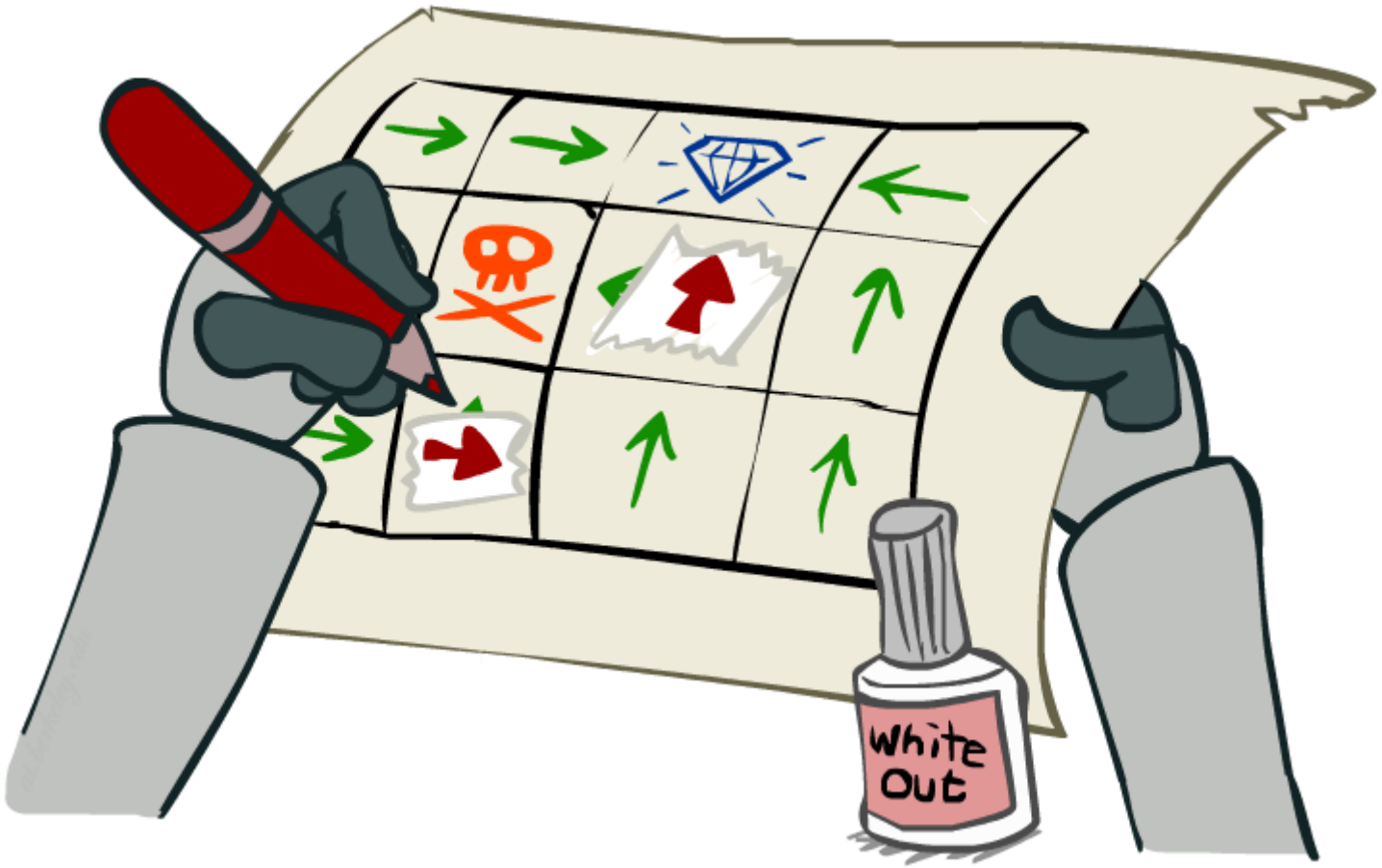


- فرض کنید ارزش بهینه‌ی حالت‌های q را در اختیار داریم.
- س. در هر حالت باید چه عملی را انجام دهیم؟
  - بسیار ساده است!

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

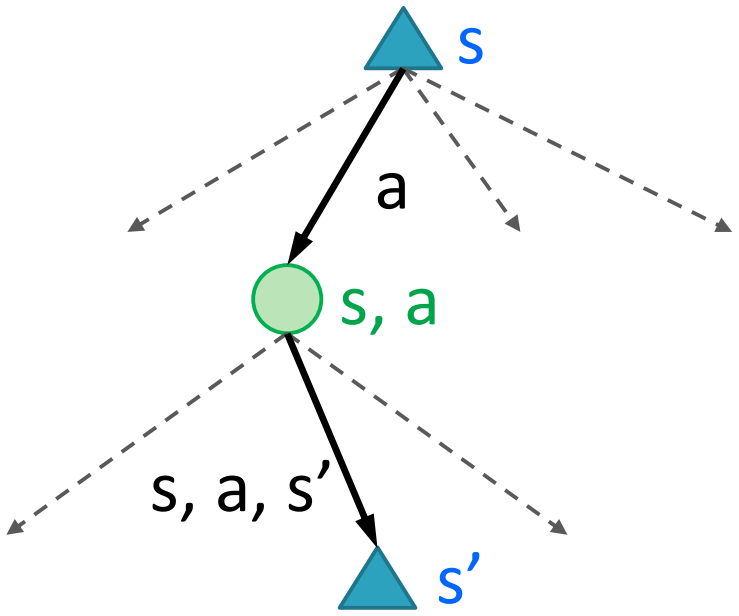
- یک درس مهم. انتخاب عمل از روی مقادیر q ساده‌تر است.

الگوریتم تکرار سیاست



مشکلات الگوریتم تکرار مقدار

- الگوریتم تکرار مقدار، معادله‌ی بلمن را تکرار می‌کند.



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

- مشکل ۱. این الگوریتم کند است - هر تکرار $O(S^2A)$

- مشکل ۲. مقدار «ماکزیمم» در هر حالت به ندرت تغییر می‌کند.

- مشکل ۳. در اغلب موارد، سیاست بسیار سریع‌تر از مقادیر همگرا می‌شود.

[python gridworld.py -a value -i 100 -n 0.2 -r 0.0 -d 0.9 -v]

الگوریتم تکرار سیاست

- الگوریتم تکرار سیاست برای محاسبه‌ی مقادیر بهینه.
  - گام اول: ارزیابی سیاست:

محاسبه‌ی مقادیر حالت‌ها برای یک سیاست ثابت تا زمانی که مقادیر همگرا شوند.

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s') \left[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')\right]$$

  - گام دوم: استخراج سیاست:

به روز رسانی سیاست فعلی با استفاده از مقادیر همگرا شده.

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_i}(s')\right]$$

  - تکرار مراحل بالا تا زمانی که سیاست همگرا شود.

مقایسه

- الگوریتم‌های تکرار مقدار و تکرار سیاست هر دو یک کمیت را محاسبه می‌کنند: مقادیر بهینه!

- در الگوریتم تکرار مقدار:
  - در هر تکرار، مقادیر و سیاست هر دو به روز رسانی می‌شوند.
  - ما به دنبال سیاست نیستیم اما با محاسبه‌ی ماکزیمم بر روی عملیات، به طور ضمنی آن را محاسبه می‌کنیم.

- در الگوریتم تکرار سیاست:
  - چند گذر وجود دارد که در آنها مقادیر سودمندی را تنها برای یک سیاست ثابت به روز رسانی می‌کنیم.
  - پس از ارزیابی سیاست، یک سیاست جدید انتخاب می‌شود. [استخراج سیاست]
  - سیاست جدید نسبت به سیاست قبلی بهتر است. [در غیر این صورت الگوریتم همگرا شده است]

خلاصه: الگوریتم‌های MDP

- به طور خلاصه، اگر می‌خواهید:
  - مقادیر بهینه‌ی حالت‌ها را محاسبه کنید: از الگوریتم تکرار مقدار یا تکرار سیاست استفاده کنید.
  - مقادیر را برای برای یک سیاست خاص محاسبه کنید: از الگوریتم ارزیابی سیاست استفاده کنید.
  - از روی مقادیر، سیاست را به دست آورید: از الگوریتم استخراج سیاست استفاده کنید.

- این الگوریتم‌ها یکسان به نظر می‌رسند.
  - همه‌ی این الگوریتم‌ها گونه‌های مختلف از به روز رسانی معادلات بلمن هستند.
  - همه‌ی آنها از یک لایه از محاسبات expectimax استفاده می‌کنند.
  - تفاوت آنها در این است که آیا از یک سیاست ثابت استفاده می‌کنیم یا در هر حالت بر روی عملیات ممکن ماکزیمم‌گیری می‌کنیم.

مسئله‌ی Double bandits

فرآیند تصمیم مارکوف

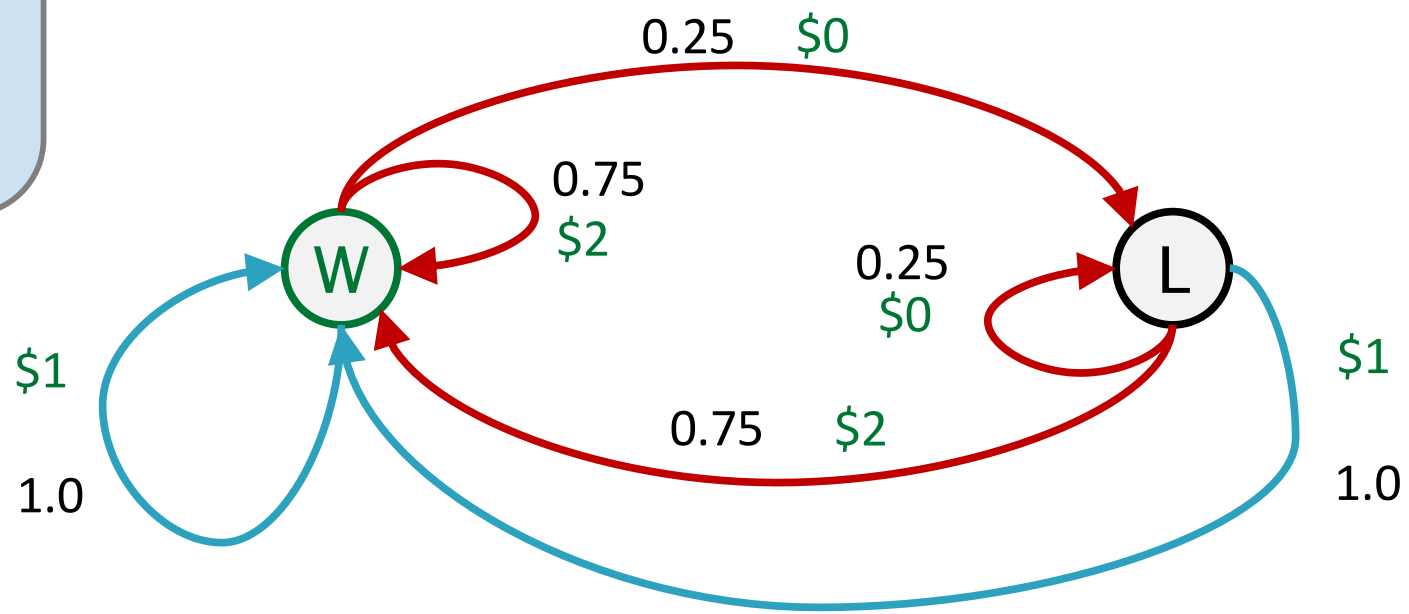
- حالت‌ها: برنده، بازنده
- عمل‌ها: آبی، قرمز

بدون کاهش پاداش

۱۰۰ مرحله زمانی

هر دو حالت دارای

ارزش برابر هستند



برنامه‌ریزی آفلاین

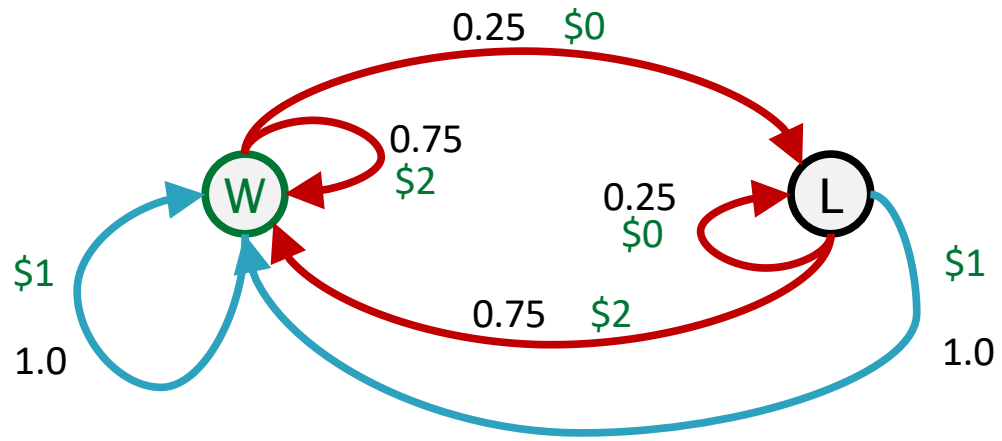
- حل MDP یک برنامه‌ریزی آفلاین است.
  - شما همه‌ی مقادیر را از طریق محاسبه تعیین می‌کنید.
  - باید از جزییات MDP آگاه باشید.
  - بازی را واقعاً بازی نمی‌کنید!

بدون کاهش پاداش

۱۰۰ مرحله زمانی

هر دو حالت دارای

ارزش برابر هستند



	مقدار
انجام آبی	۱۰۰
انجام قرمز	۱۵۰

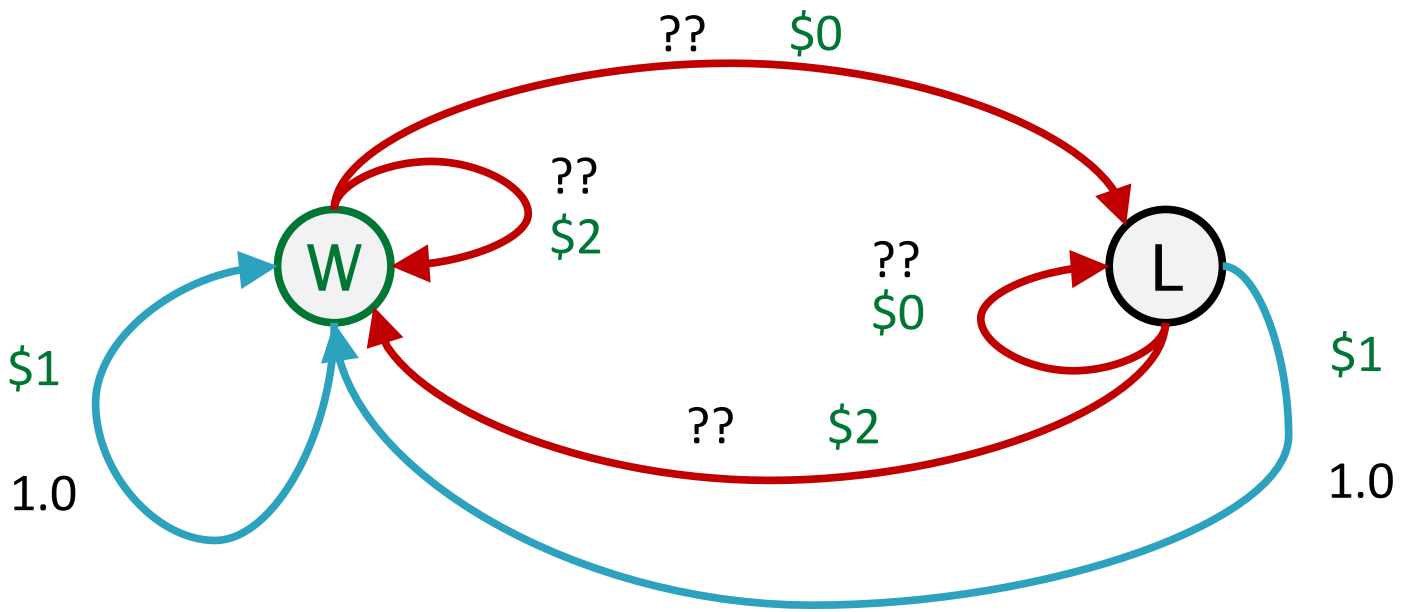
انجام بازی

$2 $2 $0 $2 $2

$2 $2 $0 $0 $2

برنامه‌ریزی آنلاین

- تغییر قوانین بازی! شانس برنده شدن در ماشین قرمز مانند قبل نیست.



انجام بازی

$0 $0 $0 $2 $0

$2 $0 $0 $0 $0

چه اتفاقی افتاد؟

- آنچه که اتفاق افتاد برنامه‌ریزی نبود، بلکه یادگیری بود!
  - به بیان دقیق‌تر، یادگیری تقویتی!
  - در این مثال شما نمی‌توانستید تنها از طریق محاسبه، MDP را حل کنید.
  - برای حل آن مجبور بودید واقعاً عمل کنید.
- ایده‌های مهم در یادگیری تقویتی.
  - کاوش: آزمایش کردن عملیات ناشناخته برای به دست آوردن اطلاعات.
  - بهره‌برداری: سرانجام باید از دانشی که آموخته‌اید استفاده کنید.
  - پشیمانی: حتی اگر هوشمندانه یاد بگیرید، باز هم ممکن است اشتباه کنید.
  - نمونه‌برداری: به دلیل عدم قطعیت، شما باید یک عمل را بارها آزمایش کنید.
  - دشواری: یادگیری می‌تواند بسیار سخت‌تر از حل یک MDP شناخته شده باشد.