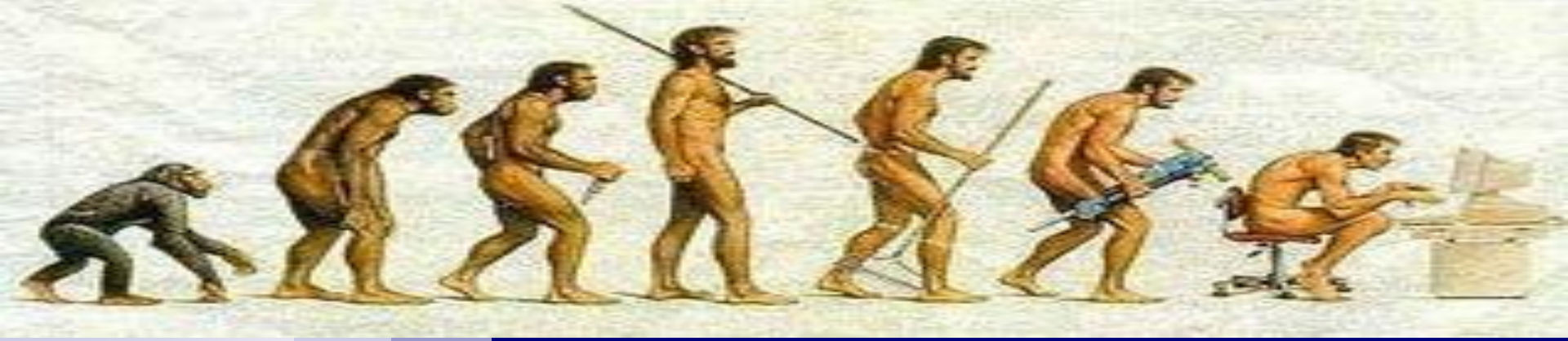


هوش مصنوعی (مقدماتی)

Artificial Intelligence

میلاد سلطانی

فصل چهارم

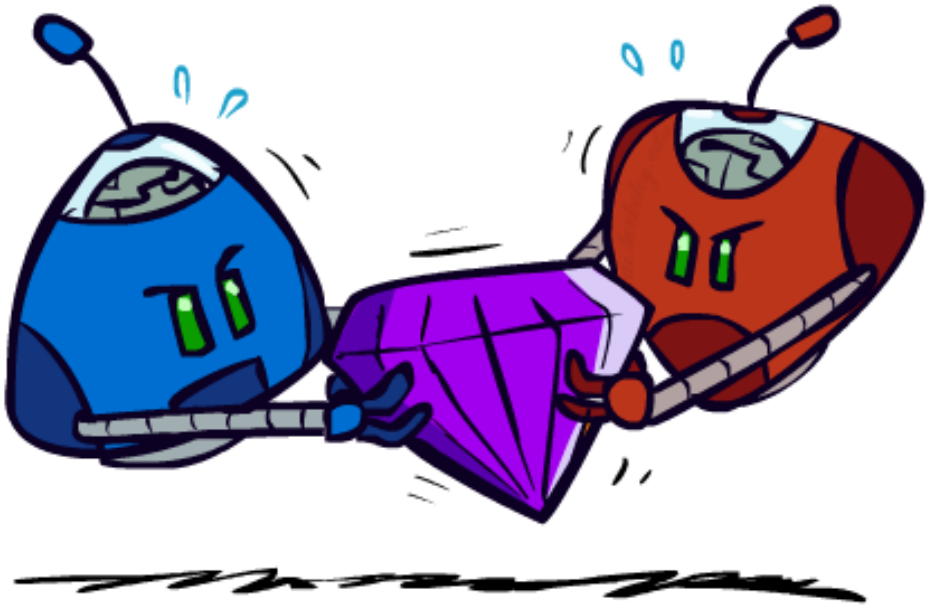
جستجوی تخاصمی

Adversarial Search

- مقدمه
- نظریه بازی ها
- تصمیمات بهینه در بازی ها
- الگوریتم بیشینه کمینه
- هرس آلفا بتا

مقدمه

- عوامل در محیط چند عاملی دو حالت دارند: همکار و رقابتی. در محیط رقابتی که هدف هر عاملی در تعارض با دیگر عوامل است، رفتارهای غیر قابل پیش بینی عوامل دیگر میتواند سبب بروز مقتضیات بسیاری در فرآیند حل مسأله شود که باعث پیدایش مسائل جستجوی تخاصمی شده است.



نظریه بازی ها

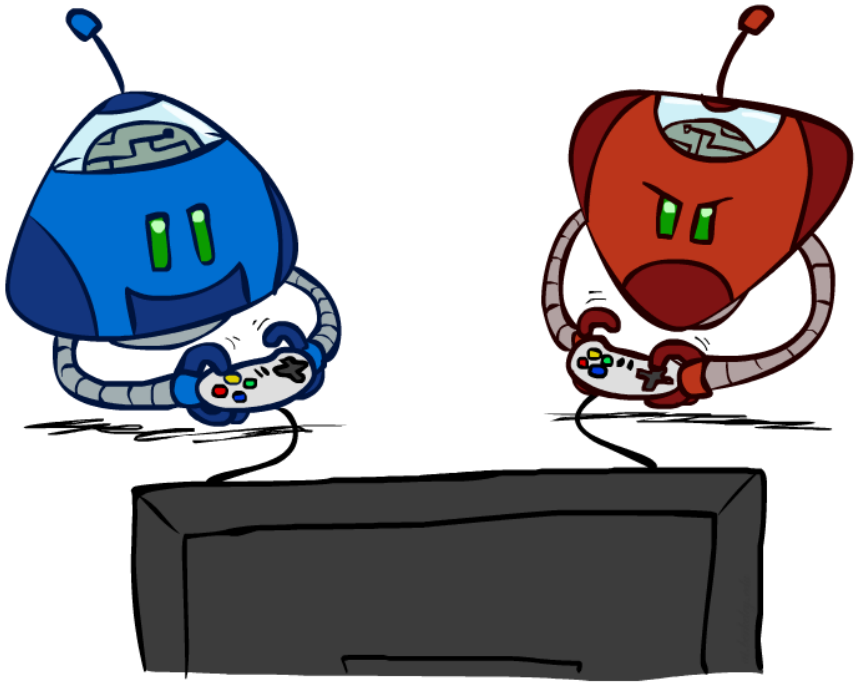
- بحث جستجوی تخاصمی به نام نظریه بازی ها یا تئوری بازی ها نیز مطرح است.

- نظریه بازیهای ریاضی، شاخه ای از علم اقتصاد است که هر محیط چند عاملی را بصورت یک بازی در نظر می گیرد، به شرط اینکه اثر هر عامل بر عامل های دیگر معنی دار باشد، خواه آن عامل رقیب است یا همکار.

- در این مبحث محیط های مسائل، معین و کاملاً رؤیت پذیر است که دو عامل بصورت نوبتی عمل می کنند و در انتهای بازی امتیاز طرفین برابر و مخالف همدیگر خواهد بود.

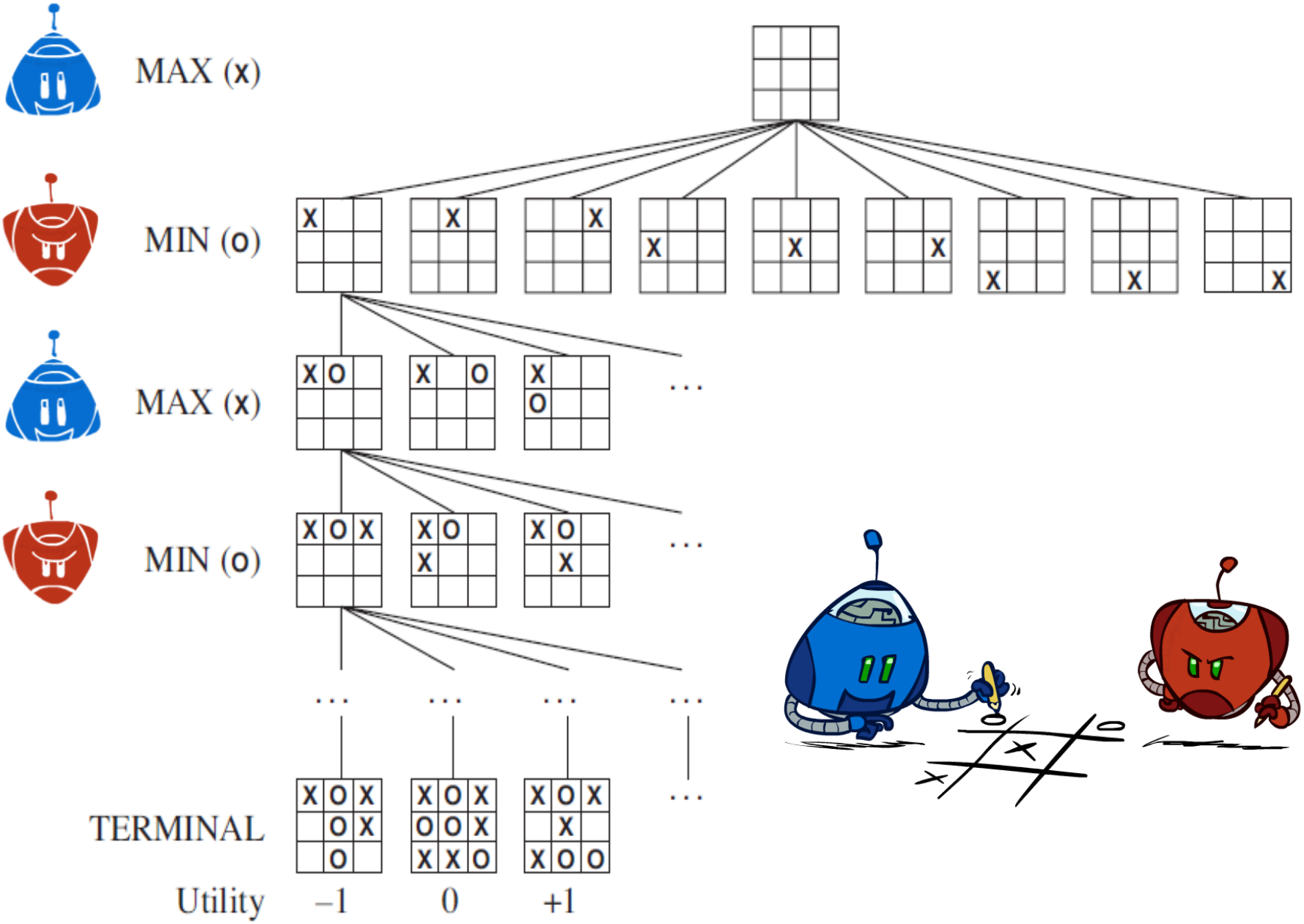
بازی ها در هوش مصنوعی

- در هوش مصنوعی بازیها به مسائلی گفته می شود که متخصصین بازیها، آنها را بازیهای **معین**، **نوبتی**، **دو نفره** و با **اطلاعات دقیق** اطلاق می کنند که در هر لحظه **مجموع امتیازات صفر** باشد.



تصمیمات بهینه در بازی ها

- در بازی های دو نفره، دو طرف بازی را MIN و MAX نامگذاری می کنند که MAX ابتدا بازی را شروع می کند و در انتهای بازی، امتیازاتی به برنده و جریمه هایی برای بازنده در نظر گرفته می شود.
- یک بازی بعنوان یک مسأله جستجو دارای اجزای زیر است:
  - **حالت اولیه:** موقعیت ابتدای بازی و شخص شروع کننده بازی
  - **تابع پسین:** شامل لیست زوج هایی است که حرکت مجاز و نتیجه را نشان می دهد.
  - **آزمون پایانی:** مشخص می کند چه زمانی بازی به پایان می رسد.
  - **تابع سودمندی:** به حالات پایانی یک مقدار عددی اختصاص می دهد.
- حالت اولیه و تصمیمات مجاز برای هر طرف، **درخت بازی** را برای آن بازی تعریف می کند.



قسمتی از درخت جستجوی مربوط به بازی دوز (TIC-TAC-TOE)

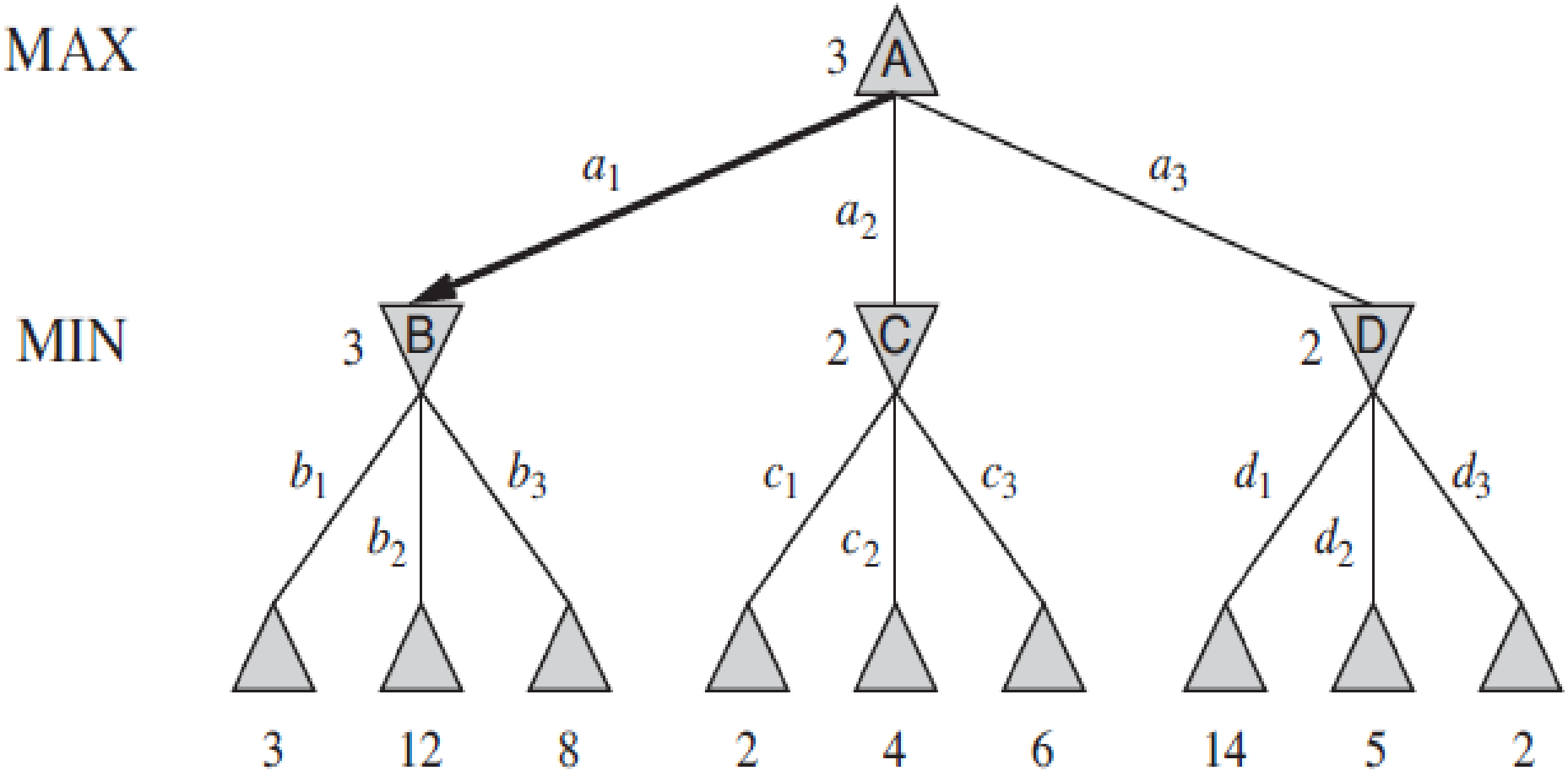
تصمیمات بهینه در بازی ها

- راهبرد بهینه در یک درخت بازی بوسیله محاسبه مقدار بیشینه کمینه مربوط به هر گره (MIN-MAX) مشخص می شود.
- مقدار بیشینه کمینهٔ یک حالت پایانی، برابر مقدار سودمندی آن می باشد.
- MAX تمایل به حرکت به حالت دارای بیشترین مقدار و MIN تمایل به حرکت به حالت کمترین مقدار را دارد.

$$MiniMax-Value(n)=\begin{cases} Utility(n) & if \quad n \quad Final-state \\ \max_{s\in Successor(n)} MiniMax-Value(s) & if \quad n \quad MAX-node \\ \min_{s\in Successor(n)} MiniMax-Value(s) & if \quad n \quad MIN-node \end{cases}$$

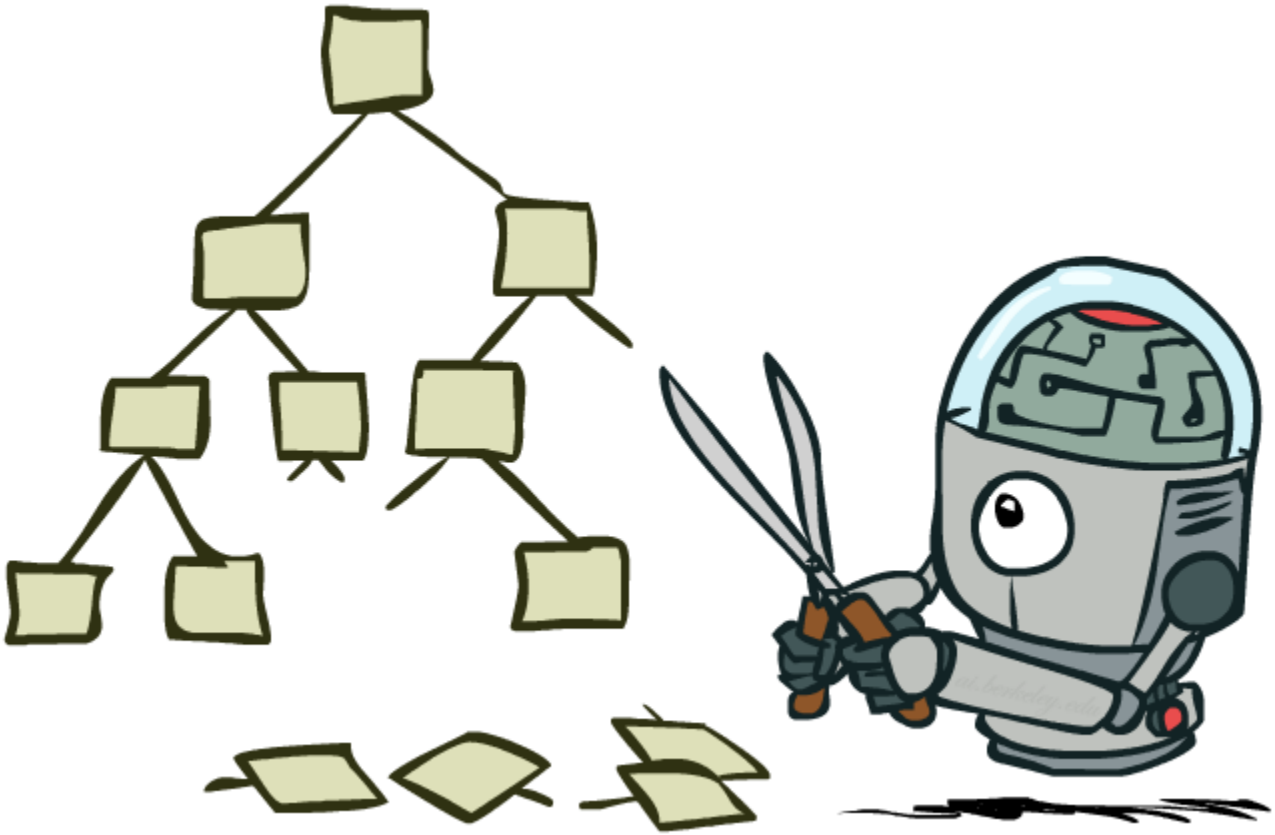
الگوریتم بیشینه کمینه

- در این الگوریتم، تمام مسیرها را رو به پایین و تا رسیدن به برگهای درخت پیشروی می کند و سپس مقادیر بیشینه کمینه مربوط به هر گره، بصورت معکوس از پایین به بالا اختصاص می یابد.
- الگوریتم بیشینه کمینه یک جستجوی کامل اول عمق را در درخت بازی انجام می دهد.
- ویژگی های الگوریتم بیشینه کمینه:
  - اگر عمق درخت m باشد و در هر نقطه بتوان b حرکت انجام داد، پیچیدگی زمانی $O(b^m)$ دارد.
  - با شرایط قبلی، پیچیدگی فضای حافظه $O(bm)$ برای زمانیکه تمام پسین ها یکجا تولید شوند و $O(m)$ برا زمانیکه تنها یک پسین تولید شود، خواهد داشت.



قسمتی از یک درخت بازی دولایه

هرس آلفا بتا



هرس آلفا بتا

- مشکل الگوریتم بیشینه کمینه آن است که تعداد حالات مورد بررسی دارای یک رابطه نمایی بر حسب تعداد حرکات می باشد (تعداد حالات زیاد است).

- می توان با استفاده از ایده هرس کردن، بدون دیدن تمام گره های درخت، بیشینه کمینه صحیح را محاسبه کرد.

- روش هرس آلفا بتا مشابه با روش بیشینه کمینه است با این تفاوت که روش هرس آلفا بتا شاخه هایی را که تأثیری در تصمیم گیری نهایی ندارند، هرس می کند.

هرس آلفا بتا

- روش هرس آلفا بتا در مثال قبل را می توان با ساده کردن الگوریتم بیشینه کمینه، بصورت زیر بیان کرد:

$$
\begin{aligned}
MiniMax-Value(root) &= MAX(MIN(3,12,8), MIN(2,x,y), MIN(14,5,2)) \\
&= MAX(3, MIN(2,x,y), 2) \\
&= MAX(3,z,2) \quad where \quad z \le 2 \\
&= 3
\end{aligned}
$$

- چون نوع جستجو اول عمق می باشد، لذا در هر زمان باید تنها گره هایی که در راستای یک مسیر از درخت باشند، مورد توجه قرار گیرند.

هرس آلفا بتا

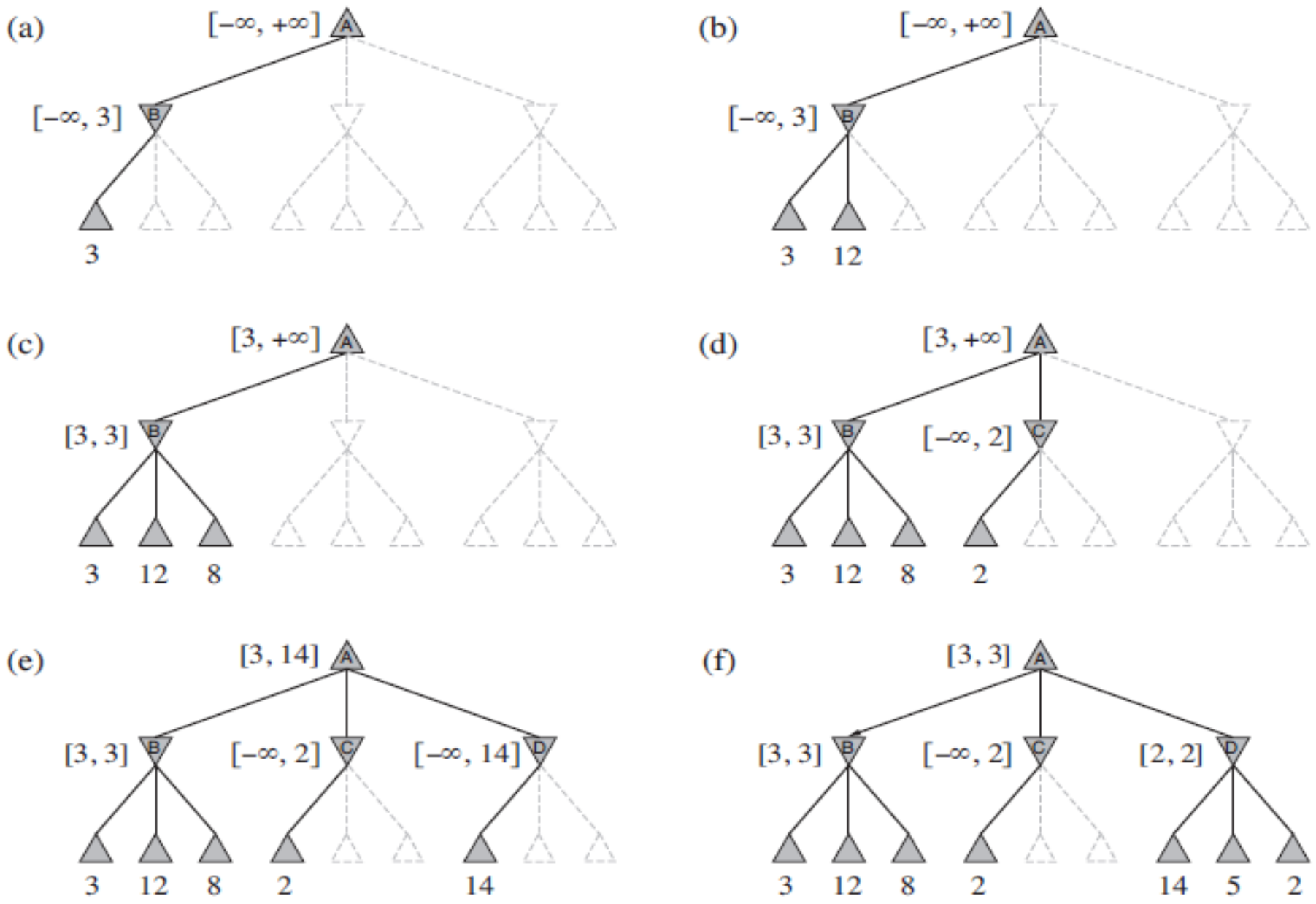
- الگوریتم هرس آلفا بتا دو پارامتر α و β دارد که بصورت زیر تعریف می شوند:

۱) α مقدار بهترین انتخاب ممکن (بالاترین مقدار) در هر نقطه انتخاب در طول مسیر برای MAX که تاکنون بدست آمده است.

۲) β مقدار بهترین انتخاب ممکن (کمترین مقدار) در هر نقطه انتخاب در طول مسیر برای MIN که تاکنون بدست آمده است.

- در الگوریتم هرس آلفا بتا، همزمان با پیشروی در درخت بازی، مقادیر α و β بروزآوری می شوند و به محض رخداد شرط زیر شاخه های باقیمانده در مسیر آن گره هرس می شوند.

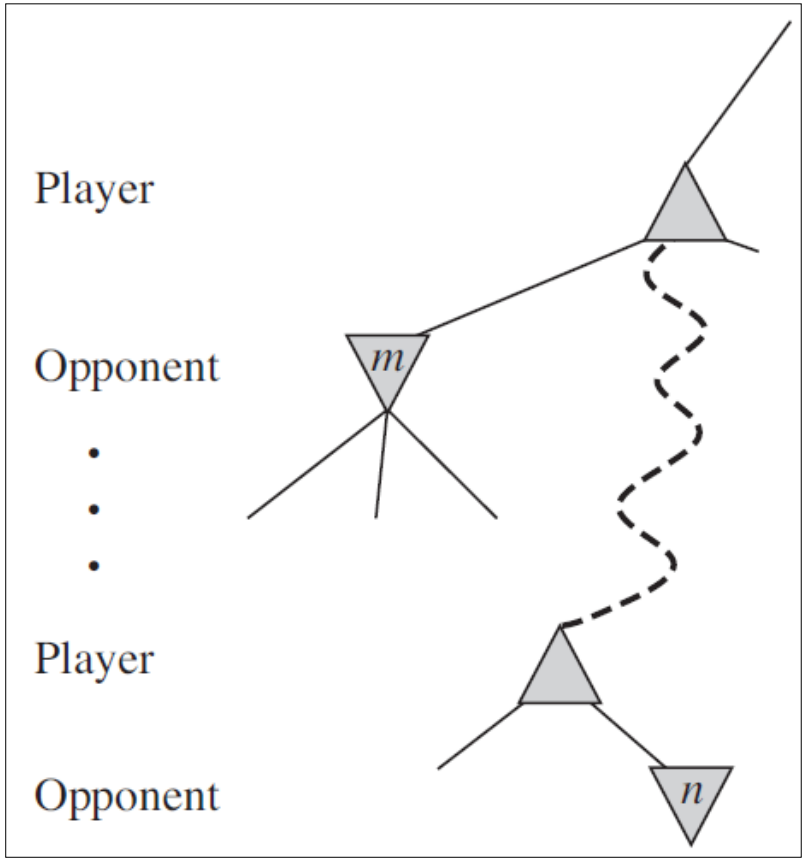
$$\alpha \geq \beta$$



مراحل تصمیم گیری روش هرس آلفا بتا در درخت جستجوی مثال قبل

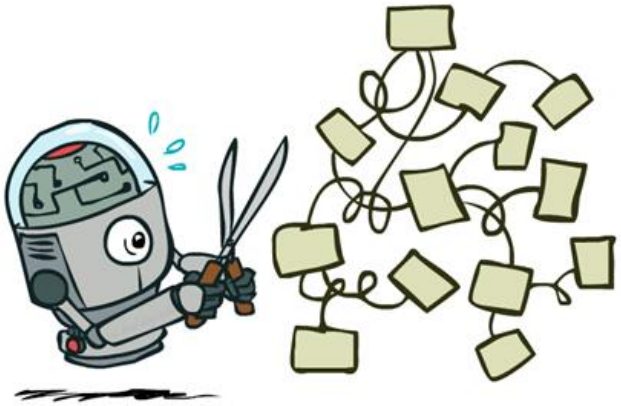
هرس آلفا بتا

- الگوریتم هرس آلفا بتا را می توان به درخت هایی با هر میزان عمق اعمال نمود و در برخی موارد می توان بجای فقط برگها، زیر شاخه های اصلی را نیز هرس کرد.
  - گره ای مثل n را در نظر بگیرید که قابل انتخاب توسط بازیکن باشد. اگر گره m در شاخه والد یا هر شاخه دیگر قبل از n باشد، آنگاه در یک بازی حقیقی هرگز به n نخواهیم رسید. لذا گره n هرس می شود.



هرس آلفا بتا

- ویژگی های الگوریتم هرس آلفا بتا:
  - در شرایط مناسب اجرای اگوریتم هرس آلفا بتا، اگر فاکتور انشعاب b و عمق جستجو برابر d باشد، تعداد $O(b^d)$ بررسی مربوط به الگوریتم بیشینه کمینه به $O(b^{d/2})$ تبدیل می شود.
  - فاکتور انشعاب از b به $\sqrt{b}$ تغییر پیدا می کند و این یعنی اینکه در یک زمان برابر، الگوریتم هرس آلفا بتا تا دو برابر روش بیشینه کمینه میتواند پیش بینی کند.



بررسی الگوریتم های قبلی

- مشکل الگوریتم بیشینه کمینه
  - الگوریتم بیشینه کمینه کل فضای جست و جوی بازی را تولید میکند.
- مشکل الگوریتم هرس آلفا بتا
  - الگوریتم هرس آلفا بتا با وجود هرس درخت، کل مسیر حالتهای پایانی را حداقل برای بخشی از فضای حالت، باید جست و جو کند. ولی این عمق عملی نیست، زیرا حرکات باید در زمانی معقول انجام شود.
- راه حل شانون در سال ۱۹۵۰ برای حل مشکلات بالا:
  - باید جستجو زودتر از زمان معمول قطع شود و بهتر است از گره های غیر پایانه به گره های پایانه پرداخته شود. یعنی به عبارت دیگر **تابع سودمندی** با یک **تابع ارزیاب هیوریستیک** و **آزمون پایانی** با یک **آزمون قطع** جایگزین شوند.

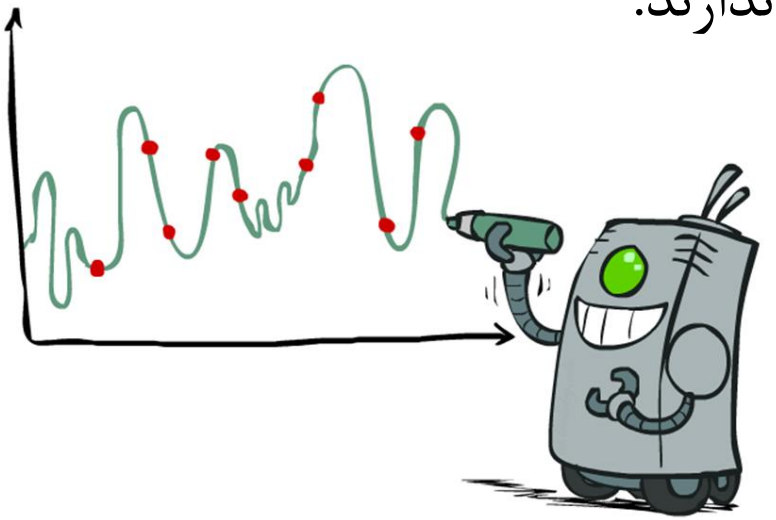
تابع ارزیاب

- چگونه یک تابع ارزیاب مناسب طراحی کنیم؟
  - باید حالات انتهایی را مثل یک تابع سودمندی مرتب کند.
  - محاسبات انجام شده نباید زیاد طولانی باشد.
  - در حالت های غیر پایانی، تابع ارزیاب باید وابستگی زیادی به شانس های حقیقی بُرد داشته باشد.
    - عبارت "شانس های بُرد" یعنی حالات دارای عدم قطعیت، و عدم قطعیت در بازی های قطعی مثل شطرنج ناشی از محدودیت های محاسباتی است نه محدودیت های اطلاعاتی !
- تابع ارزیاب باید فقط در موقعیت های ساکن اعمال شود.
  - **موقعیت ساکن:** موقعیتی است که در آینده نزدیک در آنها احتمال تغییرات شدید در مقادیر ارزیابی، بسیار کم می باشد.

آزمون قطع جستجو

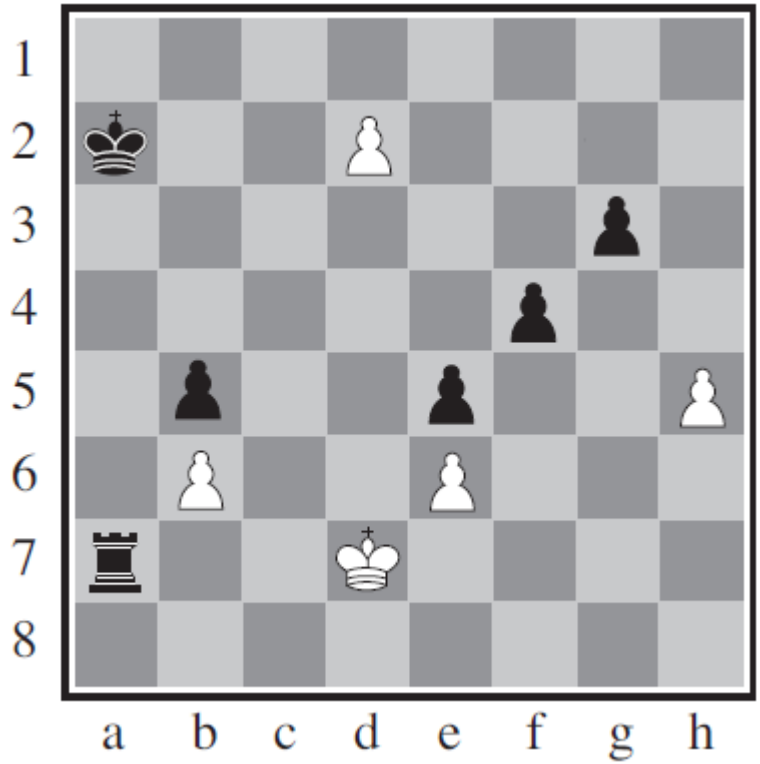
- قطع جستجو می تواند به روش های زیر انجام شود:
  - در نظر گرفتن یک عمق ثابت
  - استفاده از روش عمیق شونده تکراری

- در عمل روش های بالا کارآیی خوبی ندارند.



اثر افق دید

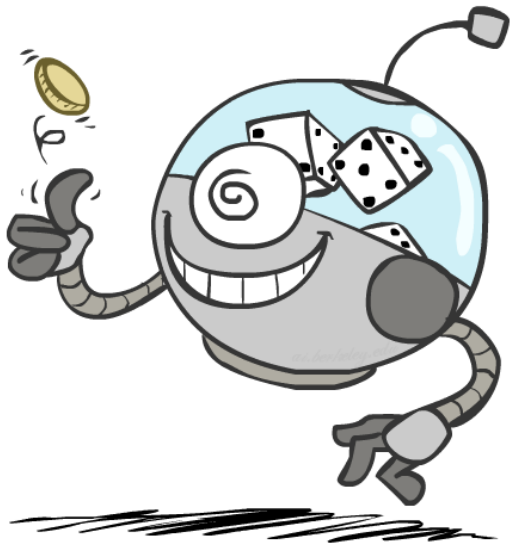
- چنین حالتی زمانی رخ می دهد که برنامه به واسطه یک حرکت حریف با شرایطی روبرو شود که بروز یک لطمه شدید، اجتناب ناپذیر باشد.



بازیهای دارای عامل شانس

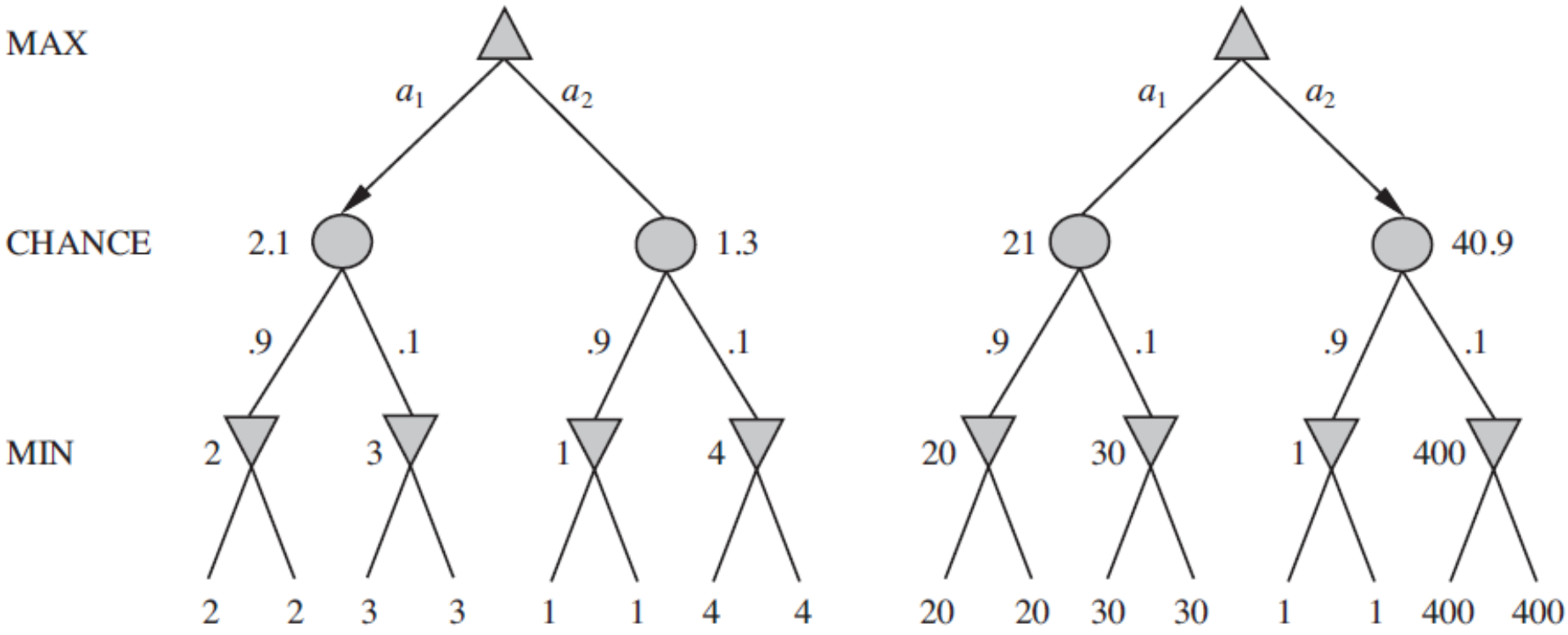
- در زندگی واقعی حوادث احتمالی پیش می آید که باعث ایجاد شرایط پیش بینی نشده می گردد که در بسیاری از بازیها این عنصر احتمالی، مانند پرتاب یک تاس در نظر گرفته می شود.

بازیهای دارای عامل شانس



- با توجه به عنصر احتمالی در بازی، هیچ بازیکنی نمی تواند یک درخت بازی استاندارد بسازد. یعنی باید علاوه بر گره های MIN و MAX دارای **گره های شانس** نیز باشد.

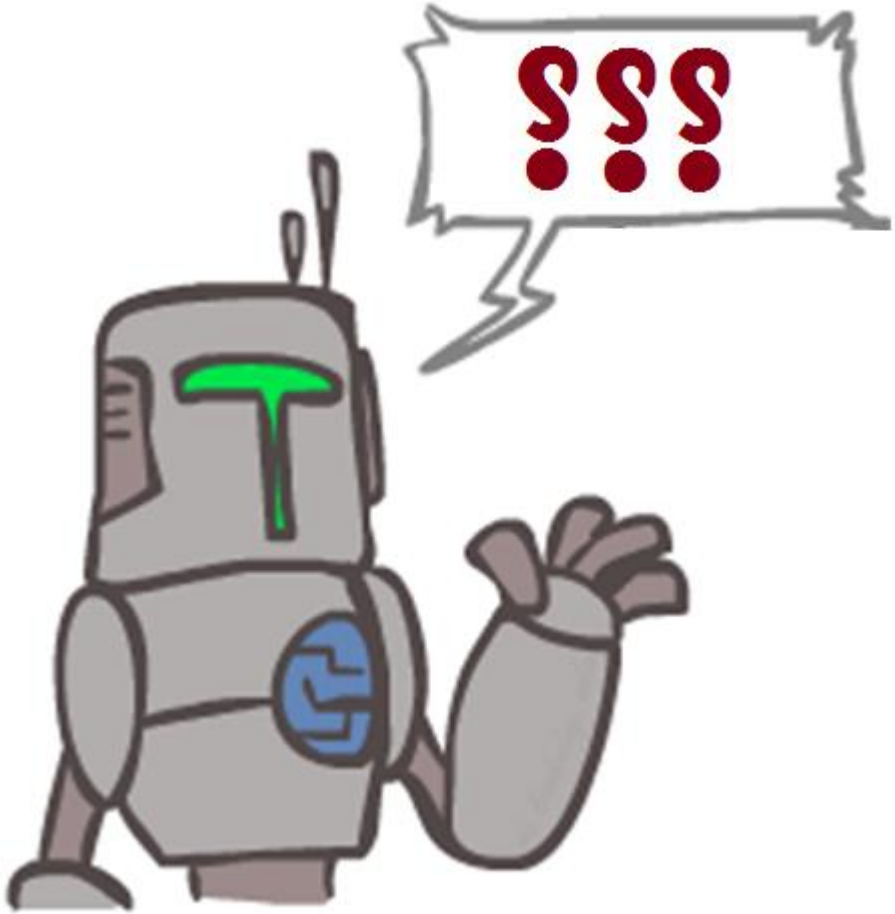
$$ExpectedMiniMax(n) = \begin{cases} Utility(n) & if\ n\ \ Final-state \\ \max_{s \in Successor(n)} ExpectedMiniMax(s) & if\ n\ \ MAX-node \\ \min_{s \in Successor(n)} ExpectedMiniMax(s) & if\ n\ \ MIN-node \\ \sum_{s \in Successor(n)} P(s) \times ExpectedMiniMax(s) & if\ n\ Chance-node \end{cases}$$



مراحل تصمیم گیری در درخت بازی با حضور عنصر شانس

مثالی از بازی های پیشرفته

- **شطرنج** : نرم افزار *Deep Blue* در سال ۱۹۹۷ توانست قهرمان شطرنج جهان " گری کاسپاروف" را در ۶ دور بازی شکست دهد.
  - سازنده :
    - موری کمپبل، فنگ هیسونگ هسو و جوزف هوان – شرکت IBM
  - مشخصات سخت افزاری ماشین برنده :
    - ۳۰ پردازشگر موازی IBM RS/6000 برای اجرای نرم افزار جستجو
    - ۴۸۰ پردازشگر VLSI مخصوص شطرنج برای انجام حرکات و ارزیابی حالت پایانی
  - مشخصات نرم افزاری :
    - جستجوی بطور متوسط ۱۲۶ میلیون گره در هر ثانیه
    - رسیدن به حداکثر ۳۰ میلیارد موقعیت در هر حرکت تا رسیدن به عمق ۱۴ درخت بازی
    - استفاده از الگوریتم استاندارد عمیق شونده تکراری هرس آلفا بتا همراه با جدول جابجایی



سوال ؟