بسم الله الرحمن الرحیم

دانشگاه فنی وحرفه ای

هوش مصنوعی(فصل هفتم)

میرمحمود ابوالحسنی

بازی‌ها

فهرست مطالب

- تعریف و انواع بازی
- الگوریتم MiniMax
- هرس آلفا و بتا
- بازی در محیط بخشی قابل مشاهده

بازی‌ها

بازی چیست؟

- بازی یک محیط چند عامله است.
- محیط‌های رقابتی / همکاری چند عامله (شطرنج رقابتی، فوتبال رقابتی همکاری)
- محیط چند عامله رقابتی را جستجوی خصمانه گویند.
- چرا بازی‌ها مطالعه می‌شوند؟
  - از نظر تاریخی سرگرم کننده است.
  - به دلیل پیچیدگی حوزه جذاب برای پژوهش است.
  - به سادگی قابل بیان است و تعداد اعمال محدود است.

بازی‌ها

بازی(جستجوی رقابتی)

- محیط بازیها اغلب چندعامله ،و رقابتی است.
- بازی‌ها،اغلب به صورت دو نفره و نوبتی یک در میان هستند،و مقادیر سودمندی مجموع صفر یا مجموع ثابت هستند.
- تضاد بین،توابع سودمندی در فضای مساله باعث یک جو رقابتی می شود.

Games as search:

- Initial state: e.g. board configuration of chess
- Successor function: list of (move, state) pairs specifying legal moves.
- Terminal test: Is the game finished?
- Utility function: Gives numerical value of terminal states. E.g. win (+1), loose (-1) and draw (0) in tic-tac-toe (next)

بازی‌ها

تعریف بازی به صورت یک مساله جستجو

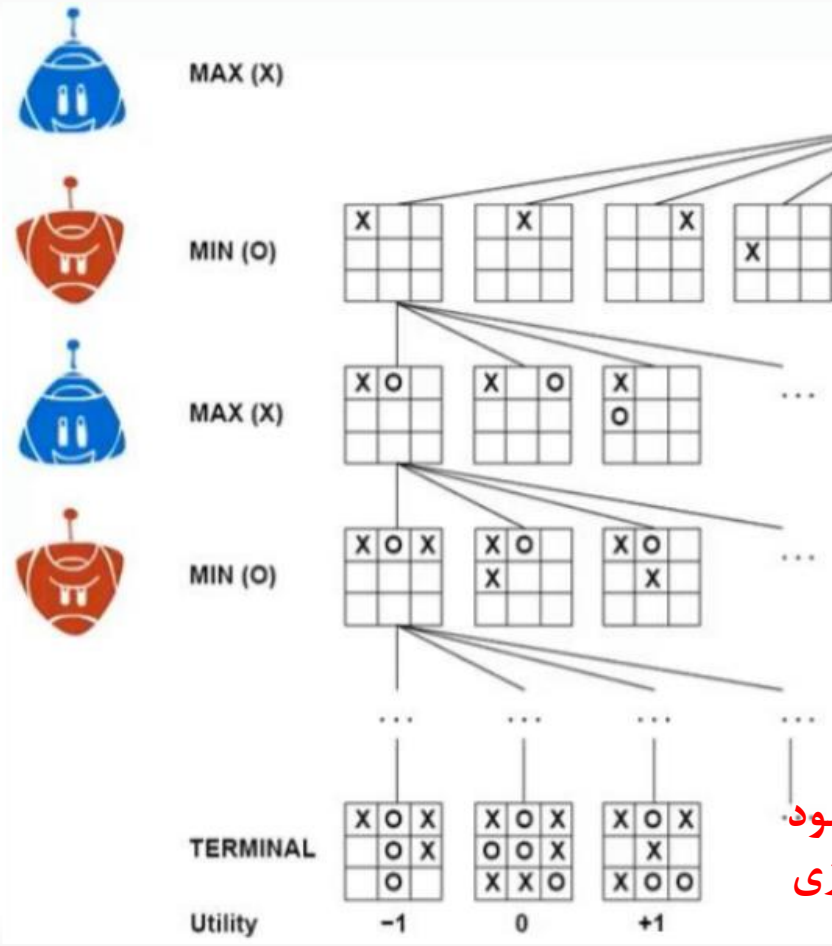
- حالت اولیه:وضعیت اولیه صفحه بازی و مشخص نمودن نوبت بازیکنان
- تابع پسین:لیستی از جفت‌های(حرکت،حالت)،هرکدام نشان دهنده یک حرکت مجاز و حالت حاصل از آن است.
- تست پایان بازی:تعیین کننده، پایان بازی است.
- تابع سودمندی(تابع هدف):عددی را برای هر حالت پایانی برمی‌گرداند.

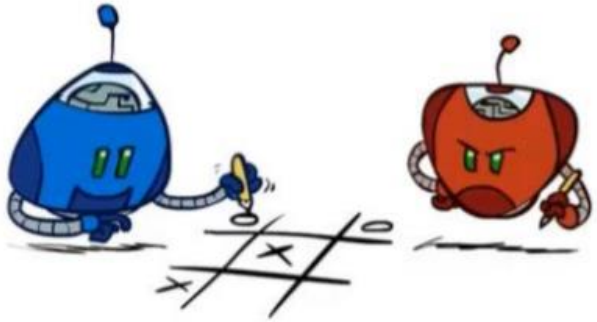
مشخصات بازی

- دو بازیکن: MAX و MIN
- MAX بازی را شروع می‌کند و بازی تا پایان ادامه می‌یابد. برنده جایزه می‌گیرد و بازنده جریمه می‌شود. MAX برای تعیین حرکت بعدی از درخت جستجو استفاده می‌کند.

بازی‌ها

بازی دوز Tic-Toc-To





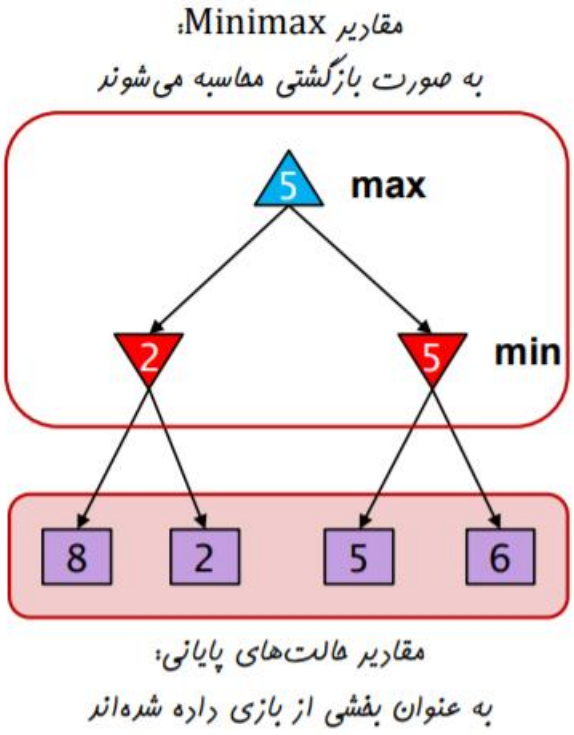
در نهایت مقدار تابع سودمندی، اگر MAX برنده شود مساوی ۱ و اگر MIN برنده شود مساوی ۱- و اگر بازی مساوی شود برابر صفر خواهد شد.

بازی‌ها

الگوریتم Minimax

- بازی‌های قطعی، مجموع صفر.
  - دوز، شطرنج، چکرز
  - یک بازیکن سودمندی خود را بیشینه می‌سازد.
  - بازیکن رقیب سودمندی او را کمینه می‌سازد.

- جستجوی Minimax:
  - یک درخت بازی
  - بازیکن‌ها به نوبت بازی می‌کنند.
  - محاسبه مقدار Minimax گره‌ها: بهترین سودمندی قابل حصول در برابر یک رقیب منطقی (بهینه).



بازی‌ها

MINIMAX

- هدف پیدا کردن یک استراژی برای MAXاست.
- فرض این است که هر دو بازیکن **بهینه** بازی می‌کنند.

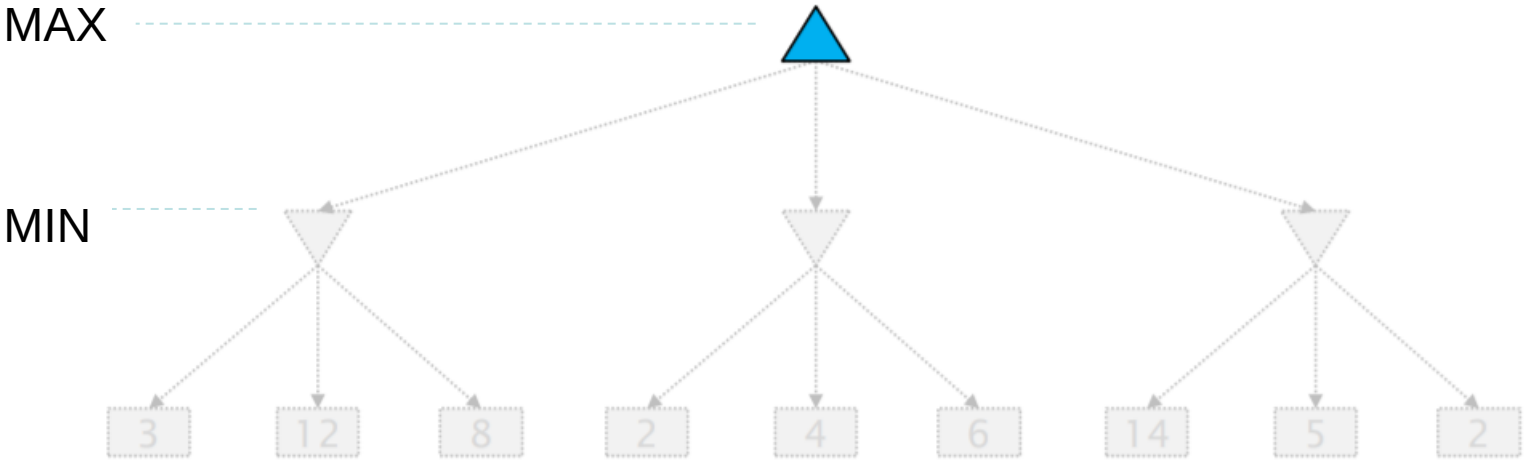
```
def value(state):
    if state is a TERMINAL: return the state's utility
    if the next agent is MAX:        return max-value(state)
    if the next agent is MIN:        return min-value(state)
```

```
def max-value(state):
    initialize v = −∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

بازی‌ها

الگوریتم Minimax بازی فرضی دو نفره

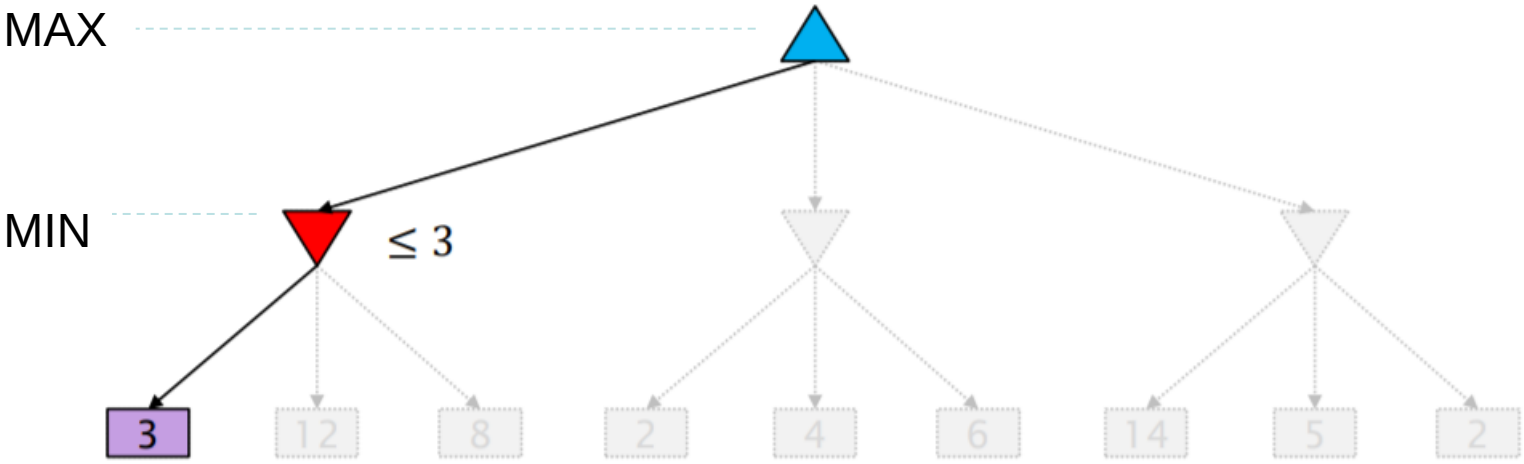


فرض کنید بازی را **MAX** شروع می‌کند و در این بازی **MAX** و **MIN** هرکدام فقط یک حرکت انجام می‌دهند و بازی تمام می‌شود و در هر مرحله **MAX** و **MIN** سه حالت را می‌توانند انتخاب کنند.

ابتدا مقدار تابع سودمندی را برای گره‌های پایانی بدست می‌آوریم.

بازی‌ها

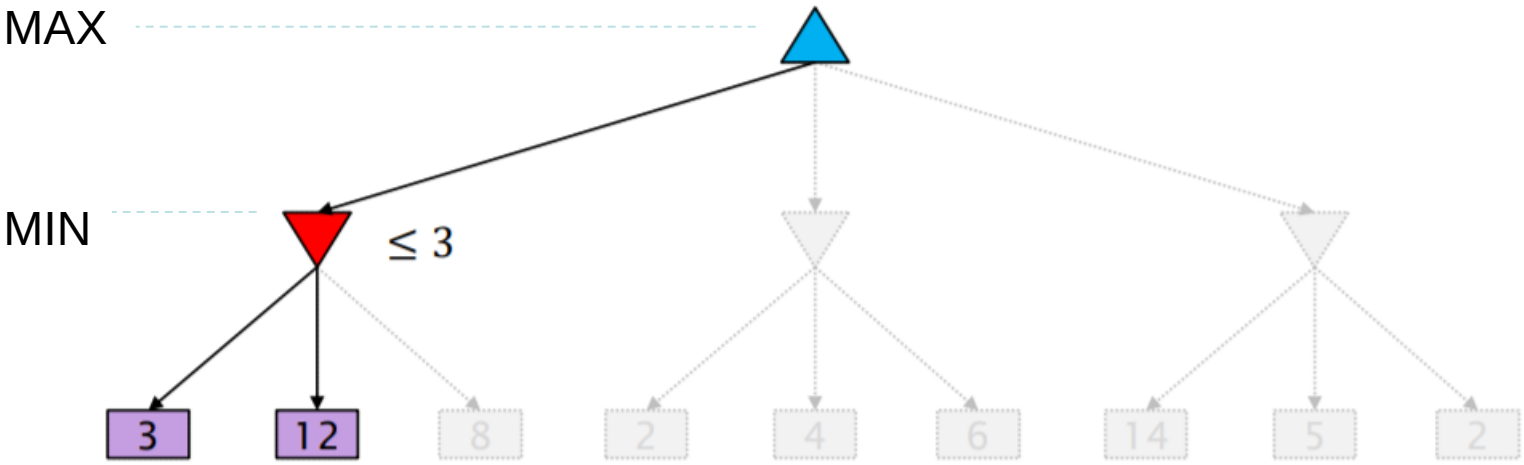
الگوریتم Minimax بازی فرضی دو نفره



در الگوریتم **MINMAX** به گره‌های پایانی یک عدد را نسبت می‌دهد که عدد متناسب با میزان امتیازی است که ما می‌گیریم.حال فرض کنید اعداد زیر امتیازاتی هستند که ما می‌توانیم بگیریم. مثلا اگر ما اولین حرکت از سمت چپ را انجام دهیم و حریف هم اولین حرکت از سمت چپ را انجام دهد، ما ۳ امتیاز خواهیم گرفت.

بازی‌ها

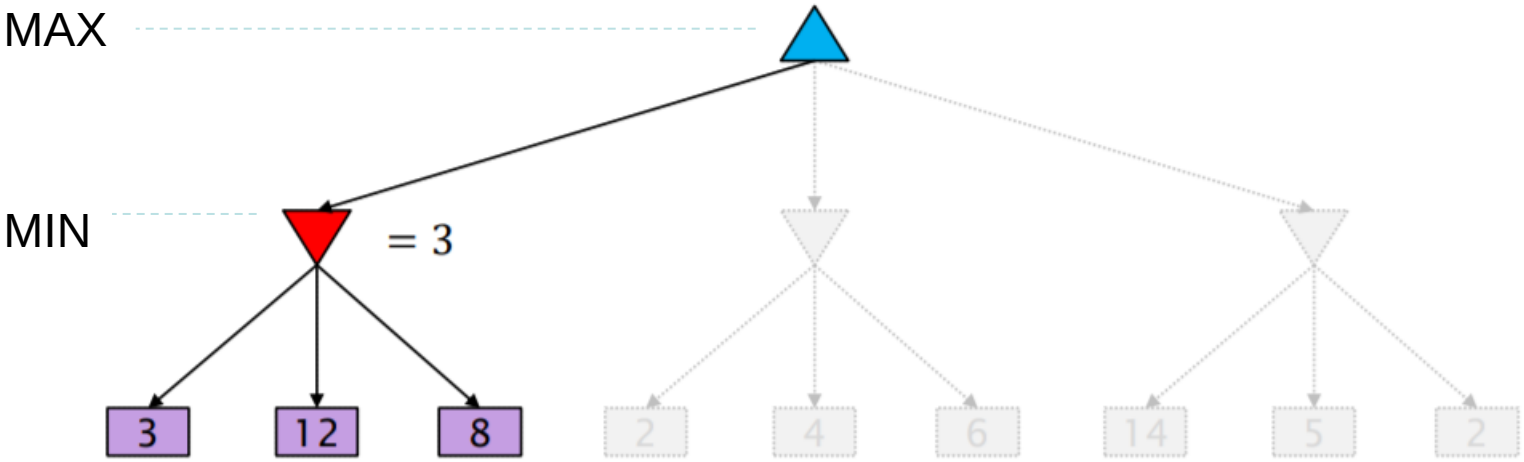
الگوریتم Minimax بازی فرضی دو نفره



بعد از اینکه تابع **UTILITY** ارزش گره‌های پایانی را حساب کرد. درسطح گره **MIN** ، مینیمم مقادیر ممکن را در نظر می گیریم. پس هرکدام از گره‌های **MIN** ، برابر با مینیمم فرزندانش خواهد شد.

بازی‌ها

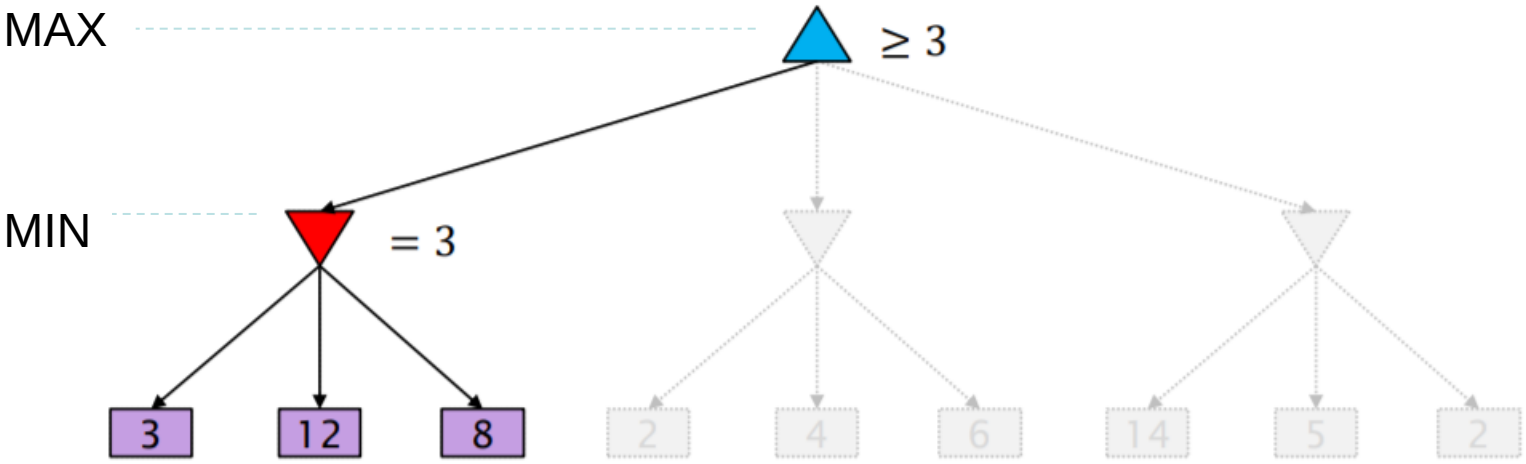
الگوریتم Minimax بازی فرضی دو نفره



بعد از اینکه تابع **UTILITY** ارزش گره‌های پایانی را حساب کرد. درسطح گره **MIN** ، مینیمم مقادیر ممکن را در نظر می گیریم. پس هرکدام از گره‌های **MIN** ، برابر با مینیمم فرزندانش خواهد شد.

بازی‌ها

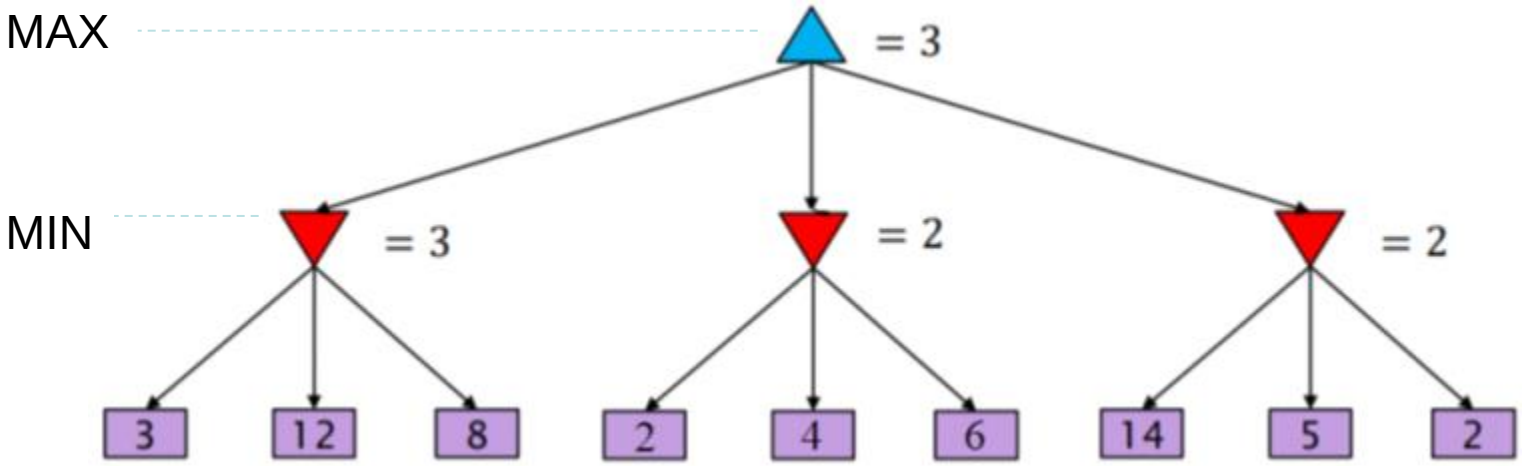
الگوریتم Minimax بازی فرضی دو نفره



در سطح گره MAX ، ماکیسمم مقادیر فرزندانش را می‌گیرد.

بازی‌ها

الگوریتم Minimax بازی فرضی دو نفره

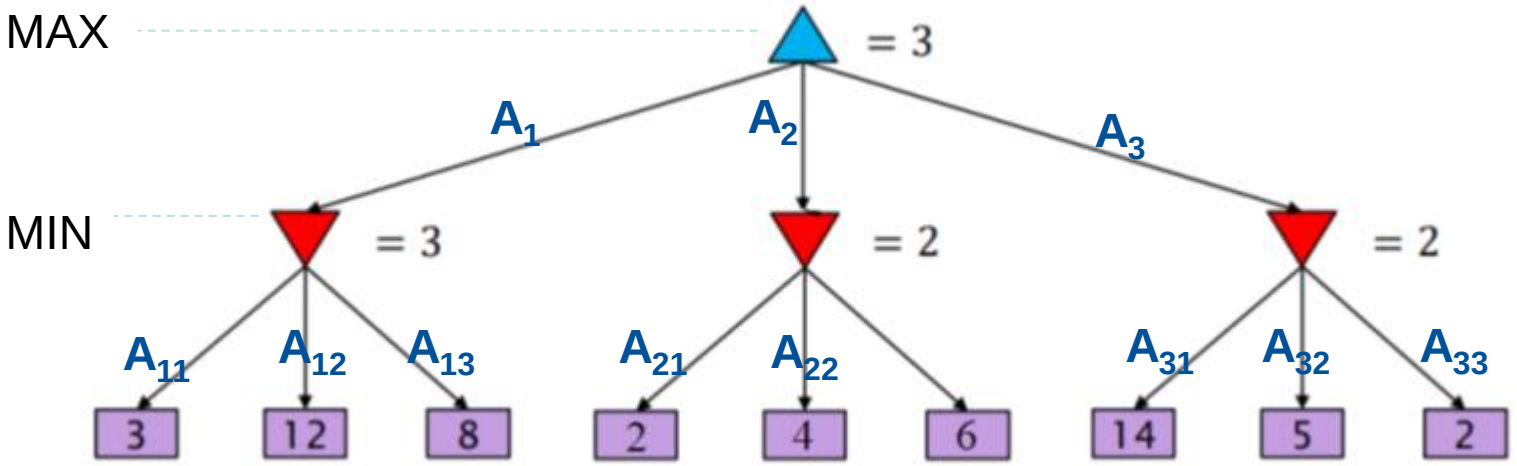


در سطح گره **MAX** ، ماکیسمم مقادیر فرزندانش را می‌گیرد.

هرکدام از گره‌های **MIN** ، برابر با مینیمم فرزندانش خواهد شد.

بازی‌ها

الگوریتم Minimax بازی فرضی دو نفره

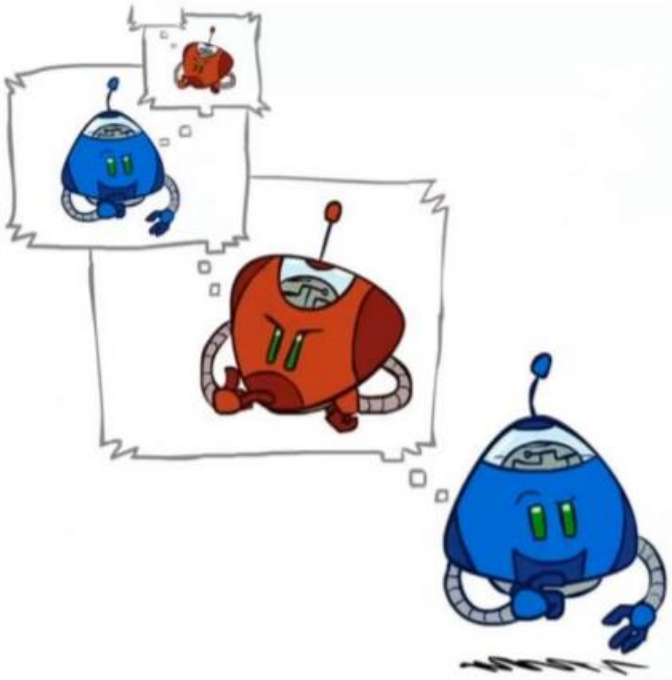


در سطح گره MAX ، ماکیسمم مقادیر فرزندانش را می‌گیرد.
یعنی MAX از بین حرکات A_1, A_2, A_3 حرکت A_1 را باید انجام دهد و حریف (MIN) با فرض بهینه بازی کردن باید حرکت A_{11} را انجام خواهد داد ولذا بیشترین امتیازی که MAX می‌گیرد ۳ است.

در بدترین حالت امتیازی که MAX می گیرد، بهترین است.

بازی‌ها

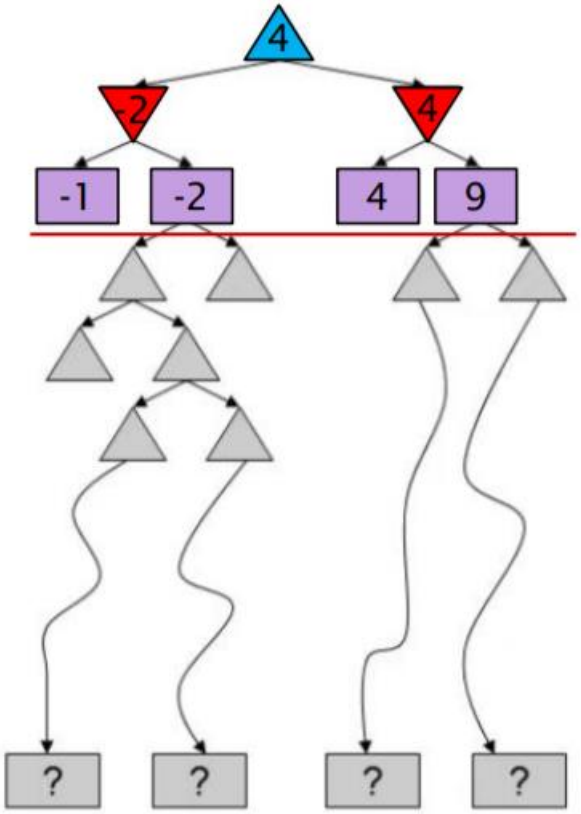
ارزیابی Minimax



- کامل. بله [به شرطی که درخت بازی محدود باشد]
- بهینه. بله [به شرطی که رقیب هم بهینه بازی کند]
- کارایی. درست همانند جستجوی عمقی
  - زمان: $O(b^m)$
  - حافظه: $O(bm)$
- مثال. شطرنج $[b \approx 35, m \approx 100]$
  - محاسبه راه‌حل دقیق عملی نیست.
- س. آیا بررسی تمام درخت بازی لازم است؟

بازی‌ها

- مشکل. در بازی‌های واقعی، نمی‌توان درخت را تا زمان رسیدن به برگ‌ها تولید نمود.

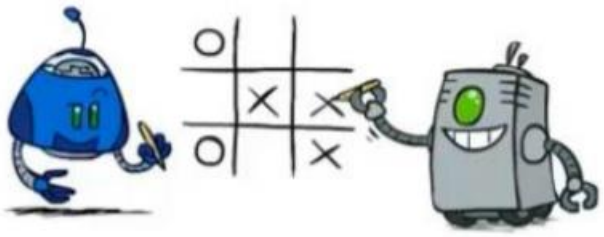


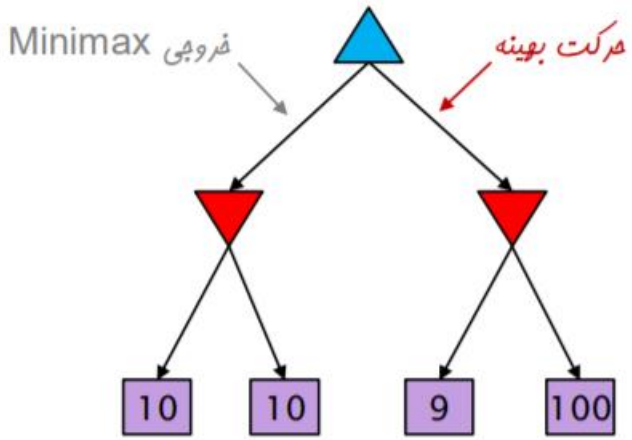
- راه‌حل. جستجوی عمقی با عمق محدود
  - جستجو تا یک عمق محدود از درخت
  - استفاده از یک تابع ارزیابی تخمینی برای گره‌های غیرپایانی.

- مثال. شطرنج $[b \approx 35, m \approx 100]$
  - برای هر حرکت ۱۰۰ ثانیه وقت داریم و
  - می‌توانیم ۱۰۰۰۰ گره در ثانیه تولید کنیم.
  - بنابراین می‌توانیم ۴ حرکت را پیش‌بینی کنیم.

بازی‌ها

- س. اگر رقیب بهینه بازی نکند، چه می‌شود؟





ج: نتیجه‌ی که MAX بدست می‌آورد بهتر نیز خواهد شد.

بازی‌ها

- هیچ تضمینی برای بهینگی پاسخ وجود ندارد.
- جستجوی تنها چند لایه دیگر می‌تواند تفاوت زیادی در نتیجه ایجاد کند.
- اگر بخواهیم به ازای هر مدت زمان دلخواه حتماً یک راه‌حل پیدا کنیم، می‌توانیم از جستجوی عمیق کننده تکراری استفاده کنیم. [یک الگوریتم هر زمانی]

پیچیدگی زمانی $O(b^m)$ زیرا باید همان اول تابع سودمندی را برای گره‌های پایانی محاسبه کنیم، پس لازم است کل درخت را بسط دهیم واین بدترین حالت است.

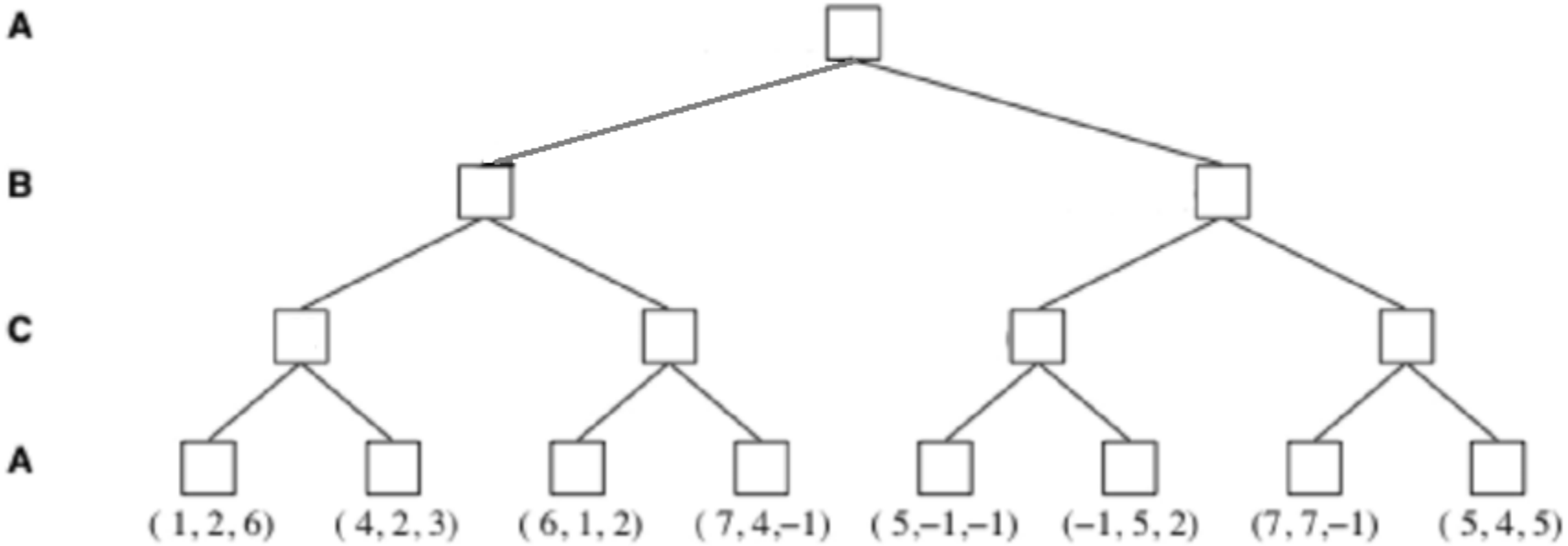
پیچیدگی فضا $O(bm)$ خطی است زیرا می‌توانیم بصورت عمقی درخت را بسط دهیم .

با آنکه فضای خطی مصرف می‌کند ولی به دلیل مرتبه بالای زمانی قابل استفاده در بسیاری از موارد نیست.
برای مثال: در بازی شطرنج نمی‌توان استفاده کرد.

بازی‌ها

بازی چند نفره

(فرض کنید سه نفر باهم بازی می‌کنند و هرکدام فقط یک حرکت می‌تواند انجام دهد و ۲ تا انتخاب دارند.)

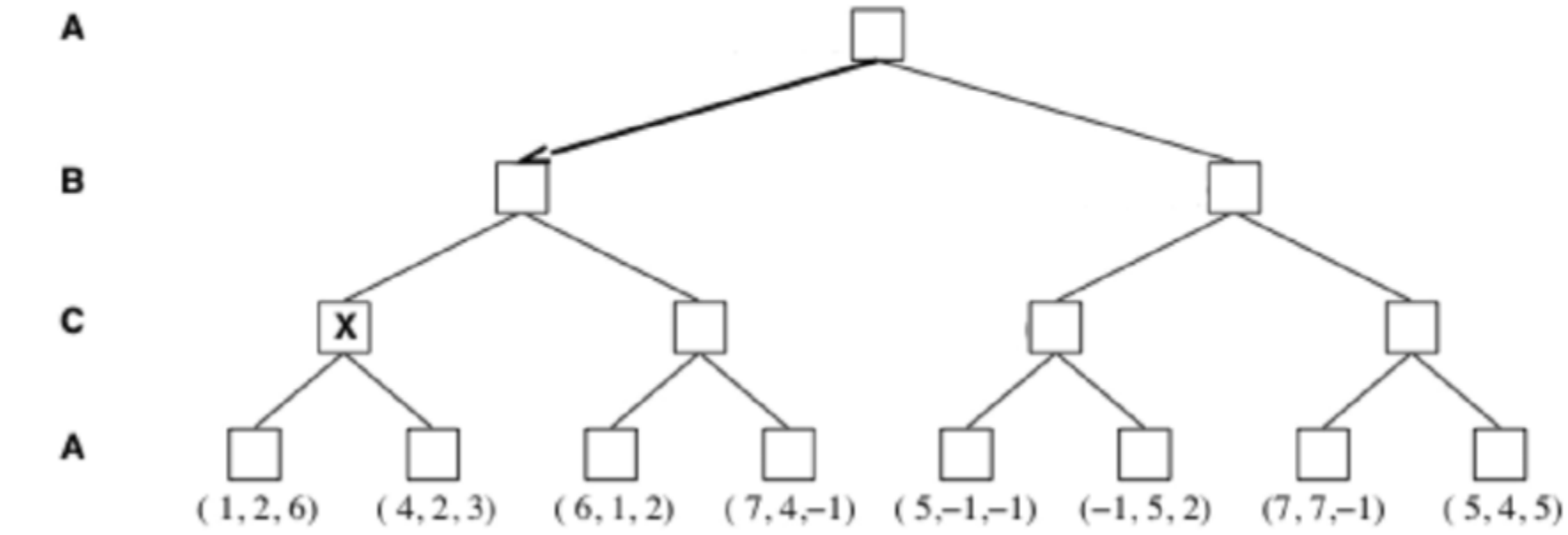


ابتدا مقدار تابع سودمندی را برای گره‌های پایانی بدست می‌آوریم.

بازی‌ها

بازی چند نفره

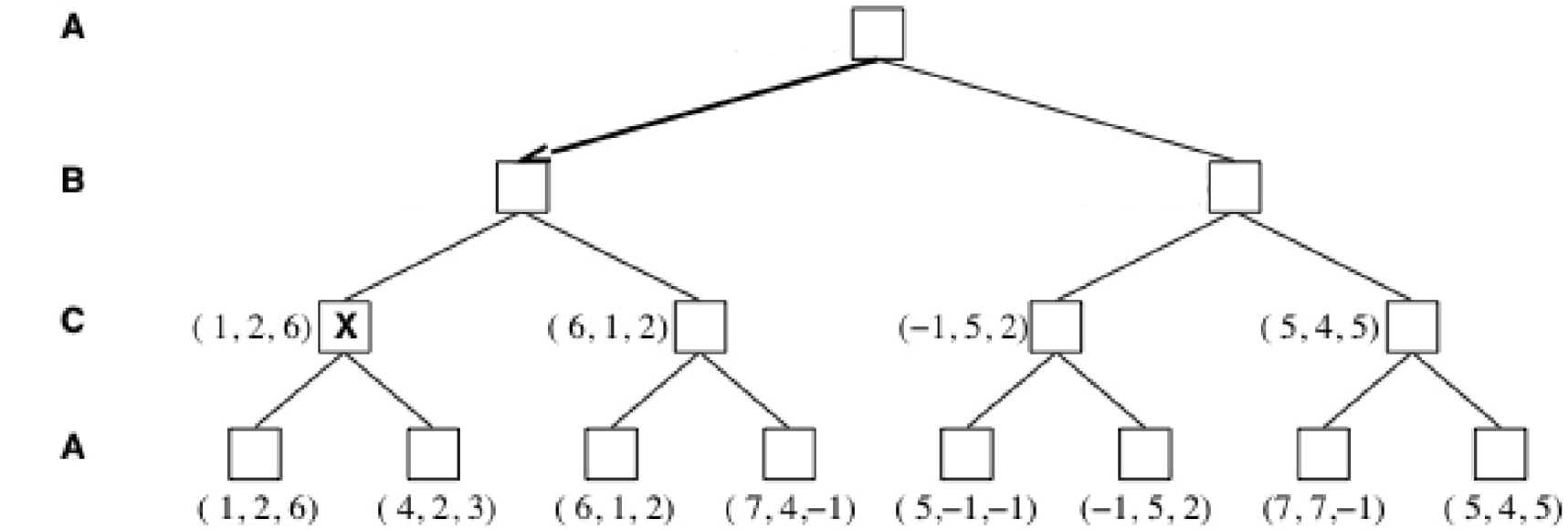
(ادامه ره حل بازی سه نفر)



بازی‌ها

بازی چند نفره

(ادامه ره حل بازی سه نفر)

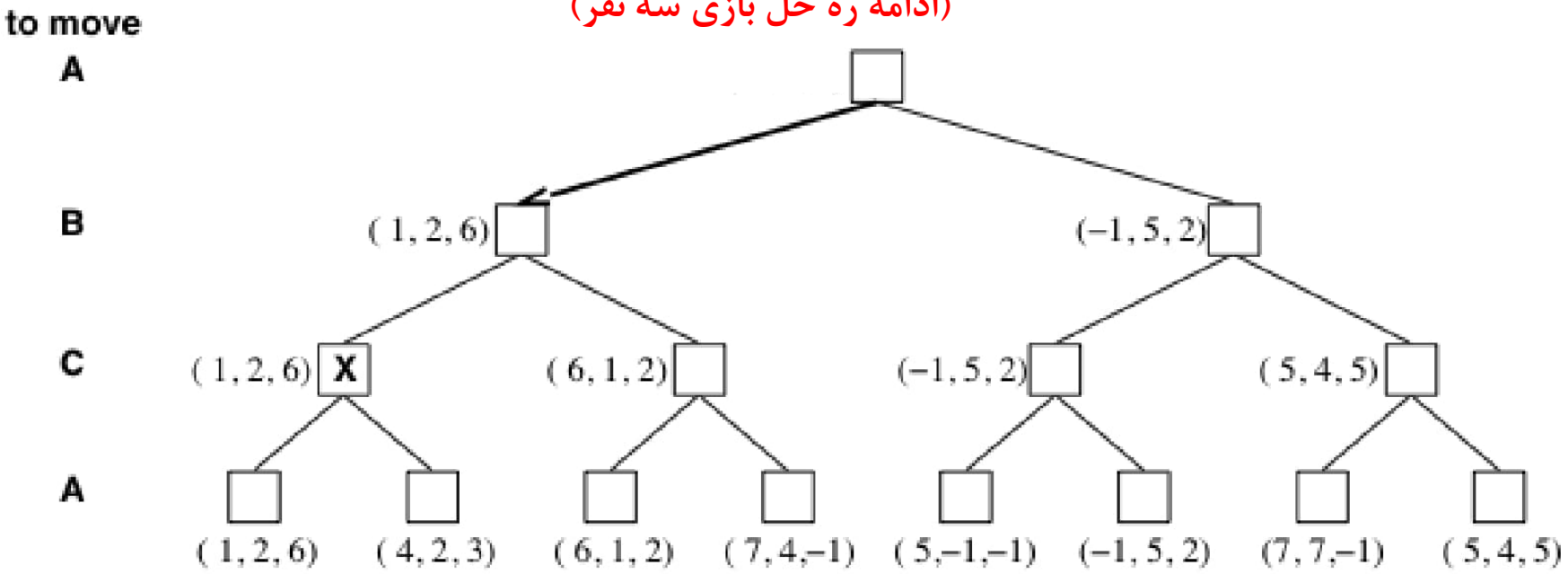


برای گره‌های سطح C ماکسیمم مقدار تابع سودمندی را برای بازیکن C را بدست می‌آوریم.

بازی‌ها

بازی چند نفره

(ادامه ره حل بازی سه نفر)

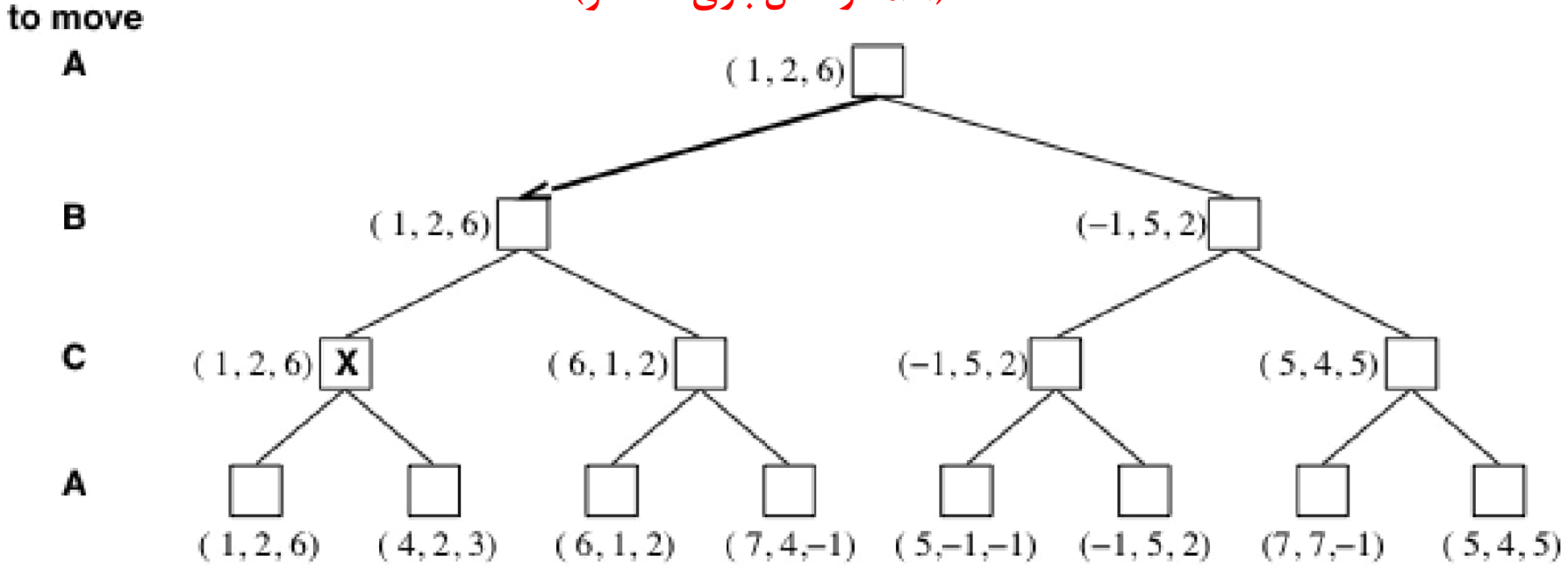


برای گره‌های سطح B ماکسیمم مقدار تابع سودمندی را برای بازیکن B را بدست می‌آوریم.

بازی‌ها

بازی چند نفره

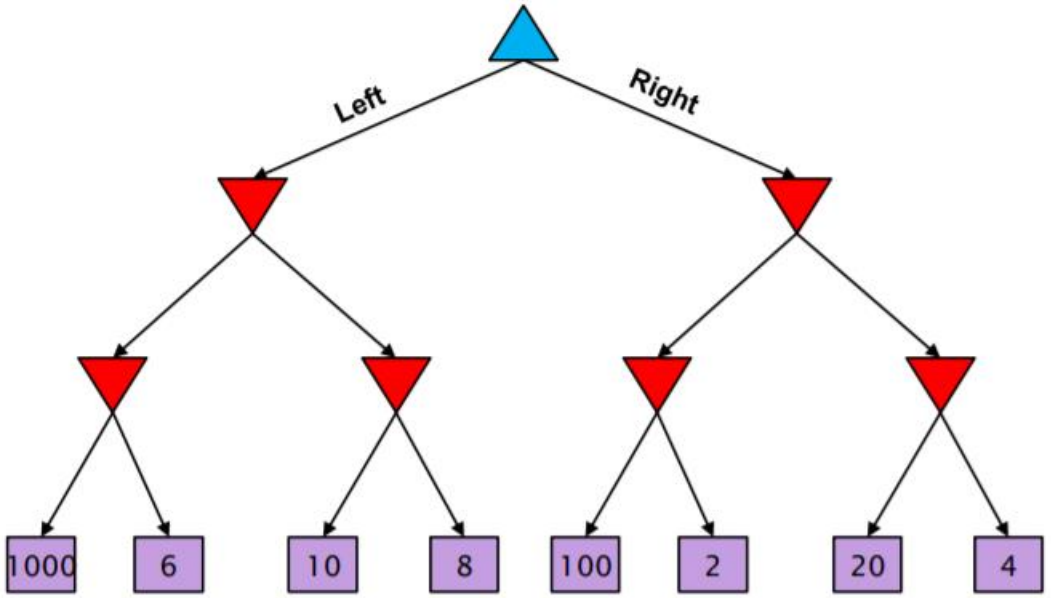
(ادامه ره حل بازی سه نفر)



برای گره‌های سطح A ماکسیمم مقدار تابع سودمندی را برای بازیکن A را بدست می‌آوریم.

بازی‌ها

- درخت بازی زیر را در نظر بگیرید:



- مقدار **minimax** برای این درخت چیست؟
- بر اساس استراتژی **minimax** بازیکن بیشینه‌ساز کدام عمل را انتخاب می‌کند؟

بازی‌ها

مقادیر سودمندی

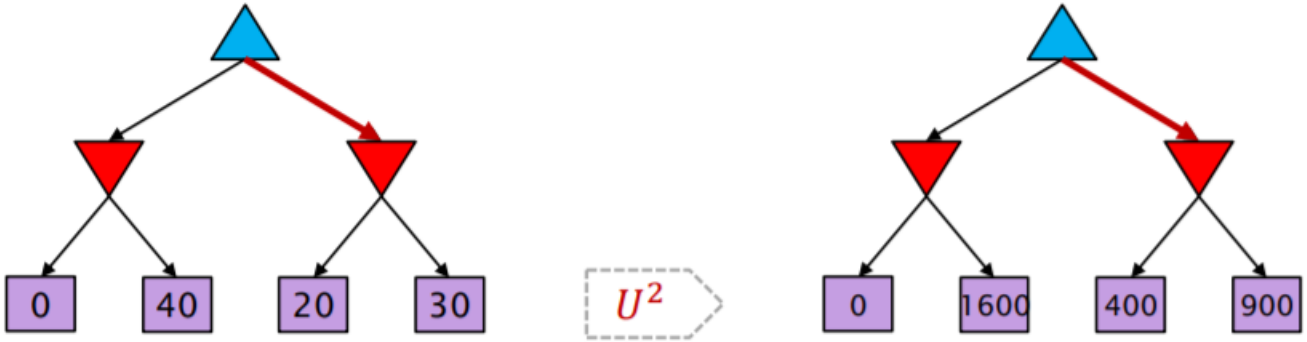
- مقادیر سودمندی. تابعی از حالات محیط به اعداد حقیقی به گونه‌ای که این اعداد بیانگر میزان مطلوب بودن این حالات برای عامل هستند.

- مقادیر سودمندی از کجا می‌آیند؟
  - برای یک بازی، تعریف مقادیر سودمندی می‌تواند ساده باشد (برد: ۱+ و باخت: ۱-)
  - مقادیر سودمندی خلاصه کننده اهداف عامل هستند.
  - قضیه: هر گونه اولویت‌های «منطقی» را می‌توان با استفاده از توابع سودمندی خلاصه نمود.

- ما فقط سودمندی‌ها را تعریف می‌کنیم و اجازه می‌دهیم رفتار مناسب پدیدار گردد.

بازی‌ها

مقادیر سودمندی



- در جستجوی Minimax مقادیر دقیق سودمندی‌ها مهم نیستند و تنها ترتیب نسبی آنها اهمیت دارد.
  - فقط نیاز داریم حالت‌های بهتر سودمندی بالاتری داشته باشند.
  - بنابراین، هر تبدیلی که ترتیب نسبی مقادیر سودمندی را حفظ کند، مجاز است.

بازی‌ها

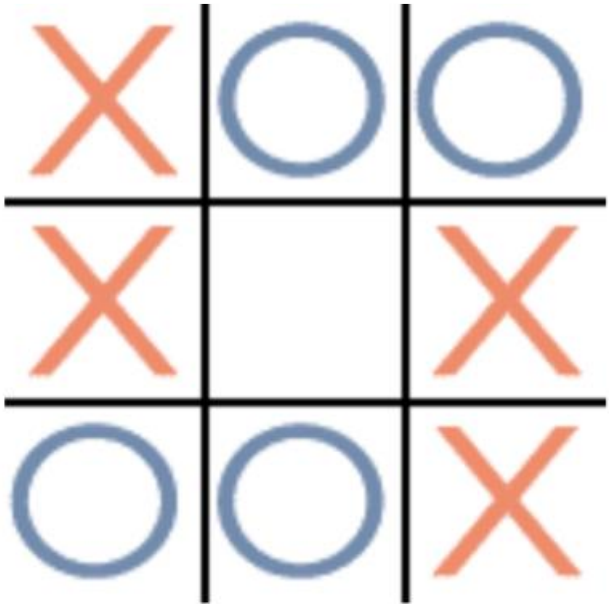
ویژگی‌های تابع ارزیابی

- تابع ارزیابی(سودمندی)،تخمینی از سودمندی یک حالت مشخص در بازی به ما می‌دهد.
- تابع ارزیابی باید مانند تابع سودمندی عمل کند،و متناسب با تابع سودمندی برای حالتهای پایانی عمل کند
- تابع ارزیابی نباید زمان زیادی داشته باشد.
- در حالت‌های غیرپایانی تابع ارزیابی باید شانس برد را به درستی تخمین بزند.

زمانیکه Cutoffرخ می‌دهد آنگاه Evaluation انجام می‌شود.

بازی‌ها

تابع ارزیابی بازی دوز Tic-Toc-Toe



X بازیکن اول و O بازیکن دوم است و بازی را X شروع می کند.

در اغلب بازی‌ها جستجوی کل درخت میسر نیست ولذا از تابع ارزیابی به عنوان هیوریستیک استفاده می‌کنیم.

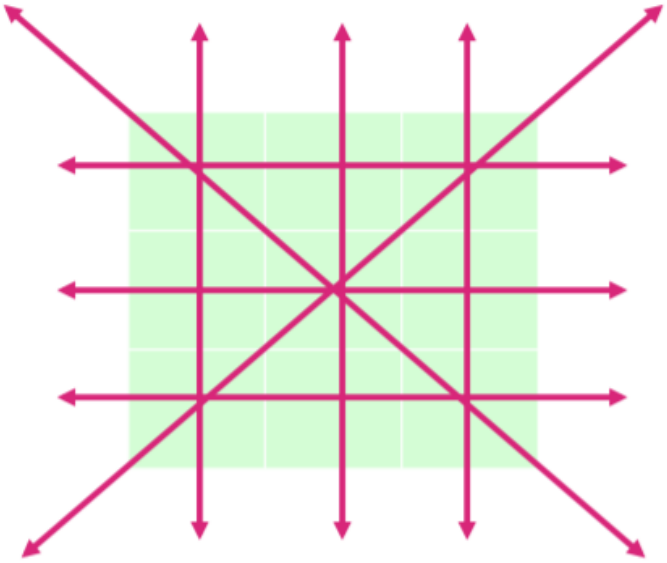
تابع ارزیابی ۱:

تعداد خطوطی که بازیکن X می‌تواند برنده شود، منهای تعداد خطوطی که بازیکن O می‌تواند برنده شود.

بازی‌ها

تابع ارزیابی بازی دوز Tic-Toc-Toe

```
evalX = (number of lines where X can win)  -
        (number of lines where O can win)
```

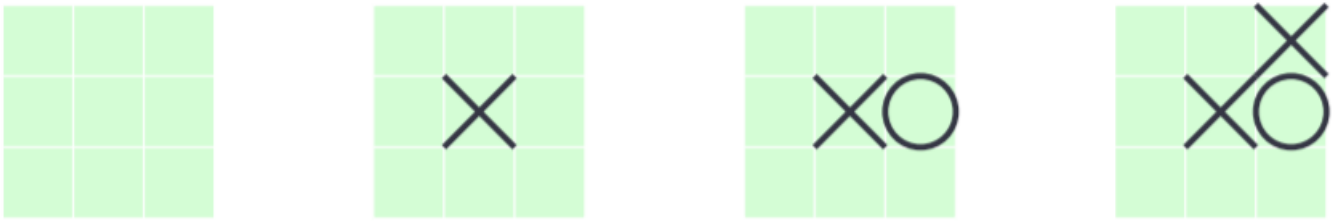


8-8 = 0

بازی‌ها

تابع ارزیابی بازی دوز Tic-Toc-Toe

```
evalX = (number of lines where X can win)  -
        (number of lines where O can win)
```



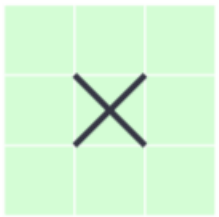
8-8 = 0 | 8-4 = 4 | 6-4 = 2 | 6-2 = 4

بازی‌ها

تابع ارزیابی بازی دوز Tic-Toc-Toe

```
eval(O) = (number of lines where O can win)  -
          (number of lines where X can win)
```


8-8 = 0



4-8 = -4

4-6 = -2

2-6 = -4

بازی‌ها

تابع ارزیابی بازی دوز Tic-Toc-Toe

تابع ارزیابی ۲:

```
Eval = 3 * X2 + X1 - (3 * O2 + O1)
```

Where

X2 is the number of lines with 2 X's and a blank
X1 is the number of lines with 1 X and 2 blanks
O2 is the number of lines with 2 O's and a blank
O1 is the number of lines with 1 O and 2 blanks

Return MaxInt (or Infinity) if X wins, -MaxInt (or –Infinity) if O wins

بازی‌ها

تابع ارزیابی بازی دوز Tic-Toc-Toe

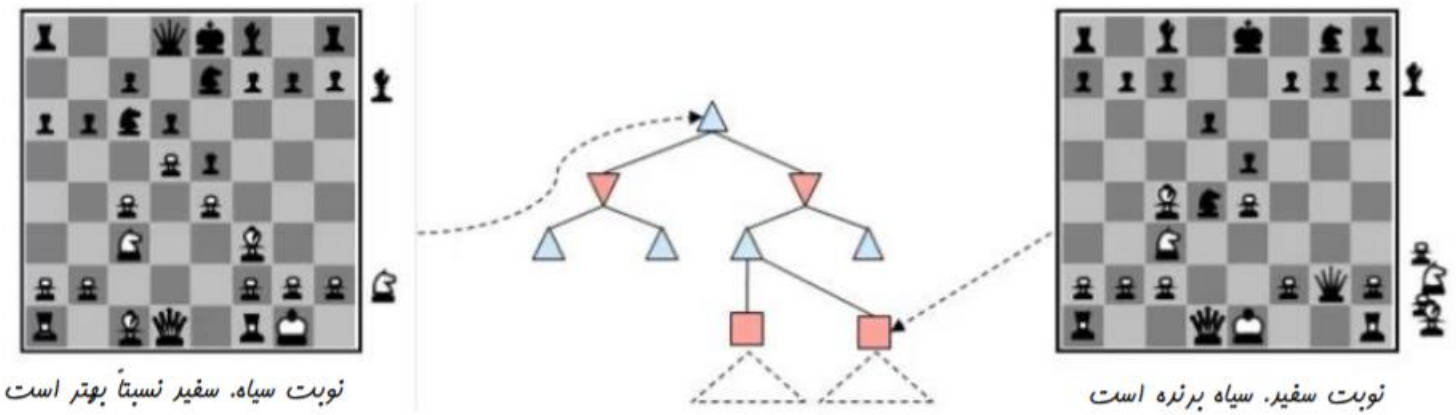
این یک تابع استاتیک است که فقط یک مقدار برمی‌گرداند. مثبت درصورت پیش افتادن بازیکن X و منفی درصورت پیش افتادن بازیکن O و صفر برای حالت مساوی.

یک کد ساده برای بازی دوز:

http://www.cs.olemiss.edu/~dwilkins/CSCI531/tic.c

بازی‌ها

- توابع ارزیابی. ارزیابی گره‌های غیر پایانی در جستجوی عمقی محدود.



نوبت سفید. سیاه برنده است

نوبت سیاه. سفید نسبتاً بهتر است

- تابع ایده‌آل. مقدار واقعی minimax موقعیت داده شده را برمی‌گرداند.
- در عمل. معمولاً از یک مجموع خطی وزن‌دار از ویژگی‌ها استفاده می‌شود:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s)$$

بازی‌ها

برای مثال، اگر **MAX** بازیکن مهره سفید باشد، تفاضل (سفید – سیاه) برای هر نوع مهره در صفحه شطرنج را با f_i نشان دهیم و برای هر مهره وزنی درنظر گرفته و آن را با w_i نشان می‌دهیم.

البته باید توجه داشته باشیم که هر قدر تعداد مهره‌ها در صفحه کمتر می‌شود ارزش بعضی از مهره‌ها مانند
مهره فیل یا رخ یا وزیر بیشتر می‌شود چون قدرت مانور بیشتری پیدا می‌کنند و تابع ارزیابی کم رنگتر می‌شود .

به عبارت دیگر هرچقدر در عمق بیشتر پیش برویم کیفیت تابع ارزیابی اهمیت کمتری پیدا می‌کند.

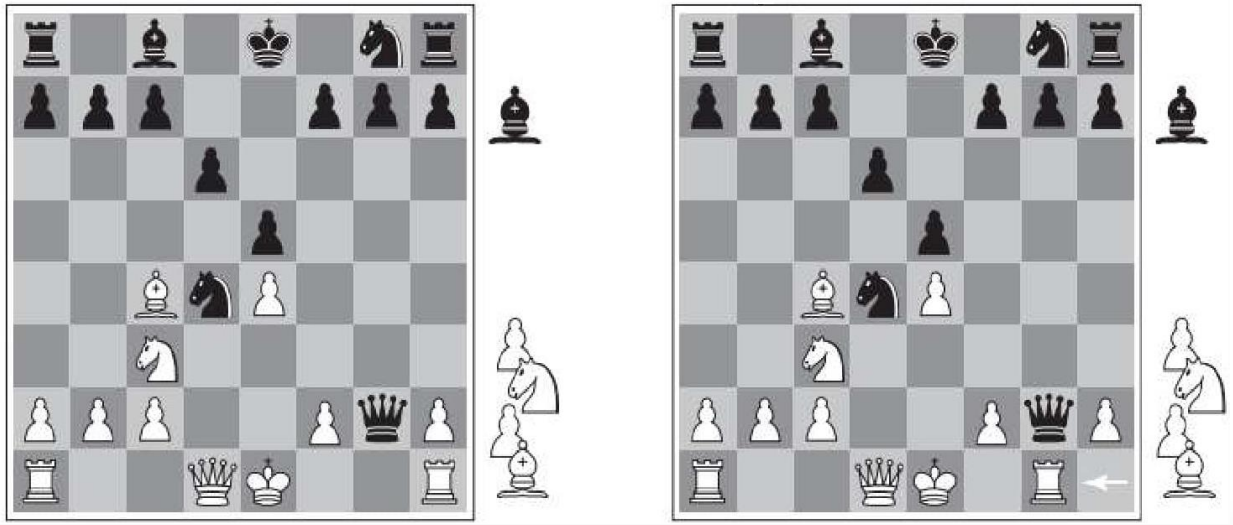
توابع ارزیابی هرگز عالی وبی نقص نیستند.

بازی‌ها

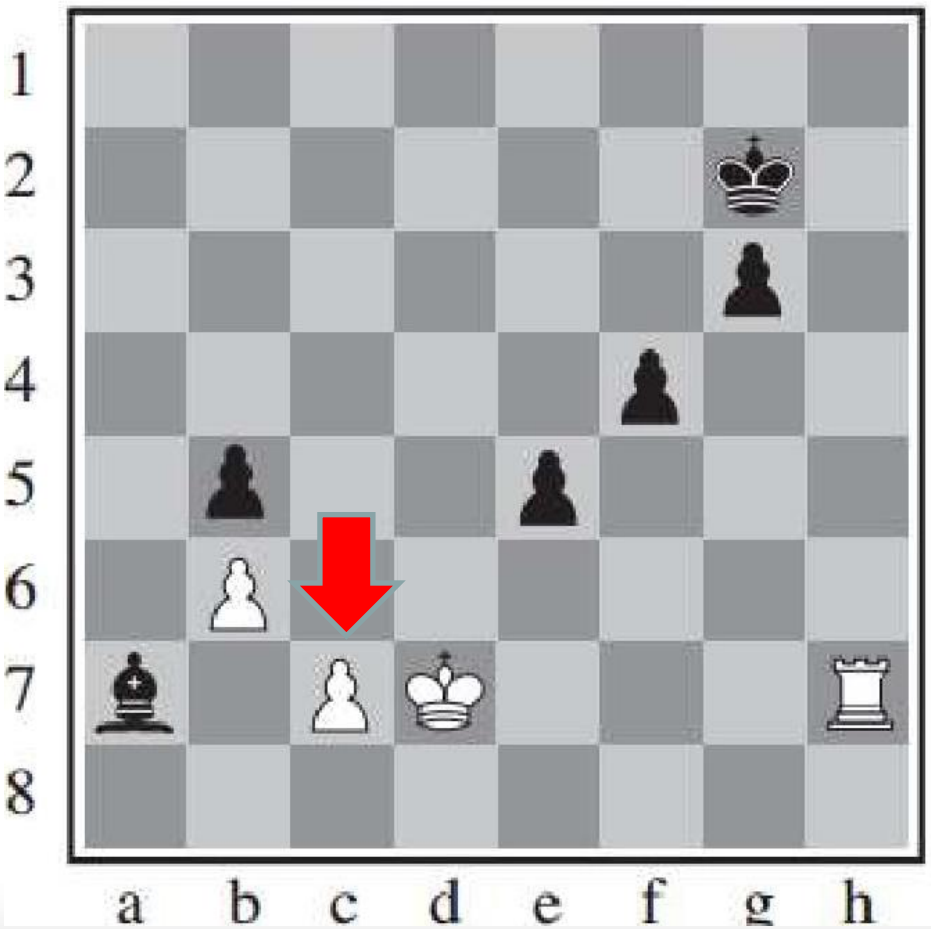
مشکل قطع جستجو

به عنوان مثال، در بازی شطرنج ممکن است حالتی پیش بیاید که پس از قطع جستجو حریف بتواند حرکتی را انجام دهد که تمام پیش بینی‌های ما را به خطر بیندازد و این به خاطر آن است که ما جستجو را باید ادامه می دادیم و درآن موقعیت قطع نمی‌کردیم.(یعنی قطع جستجو باید وقتی انجام شود که مطمئن شویم یک یا دو حرکت بعد اتفاق خاصی نمی‌افتد و تابع ارزیابی تغییر محسوسی نمی‌کند. به این حالت وضعیت ساکن گوییم.)

در شکل زیر با آنکه مهره سیاه بیشتری داخل صفحه است ولی مهره سفید می‌تواند با حرکت دادن رخ یک خانه به طرف راست وزیر سیاه را در حرکت بعدی حذف نماید.



بازی‌ها



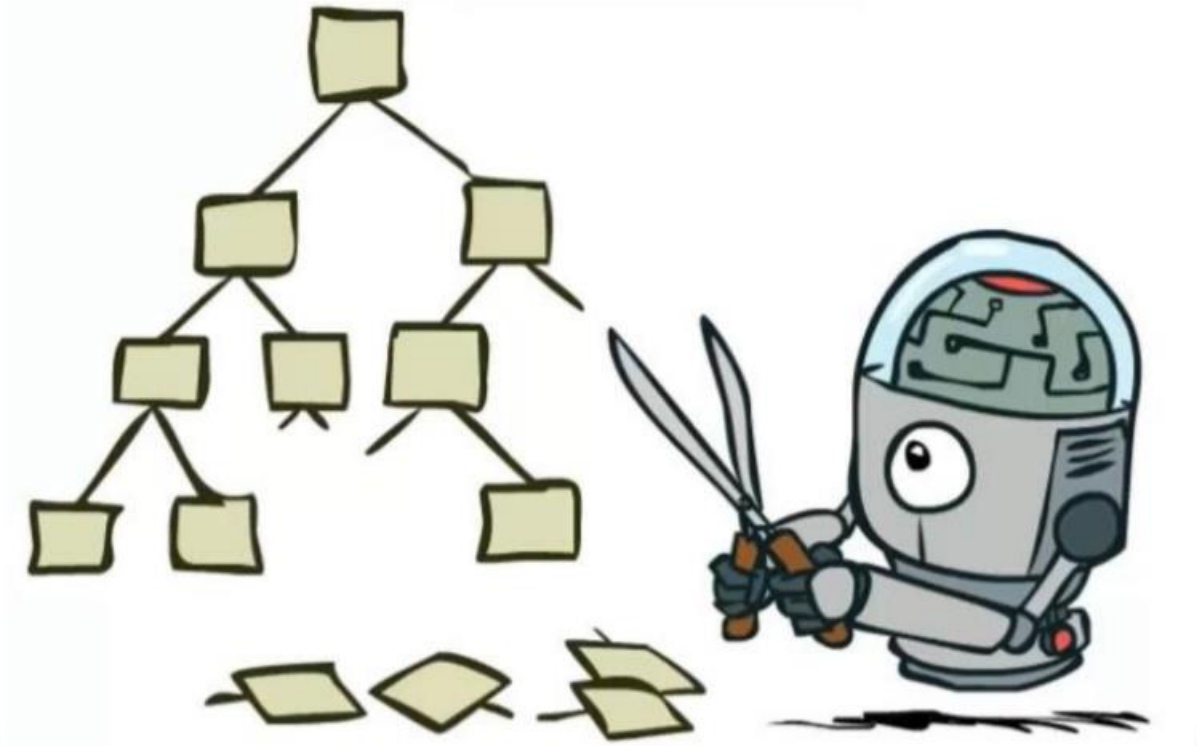
مشکل تابع ارزیابی

فرض کنید تابع ارزیابی ما براساس تعداد مهره‌ها باشد. در شکل مقابل می‌بینیم که تعداد مهره‌های سیاه بیشتر از مهره‌های سفید است ولذا تابع ارزیابی شانس بهتری را برای مهره سیاه تخمین می‌زند ولی اگر بازیکن سفید مهره سرباز خود را به خانه آخر برساند می‌تواند آن را با مهره وزیر جایگزین نموده و در موقعیت بهتری نسبت به بازیکن سیاه قرارگیرد.

در نتیجه تابع ارزیابی باید به گونه‌ای تعریف شود که در هر مرحله ارزیابی صحیحی از موقعیت حریف داشته باشد و حرکت مناسبی را برای ما پیشنهاد دهد.

بازی‌ها

هرس کردن درخت



بازی‌ها

MINIMAX

- مشکل: تعداد حالت‌های بازی نسبت به تعداد حرکت‌ها نمایی است.
- راه حل: همه گره‌ها بررسی نشوند.

هرس آلفا – بتا

- آلفا: بیشترین مقدار در گره MAX و بتا: کمترین مقدار در گره MIN

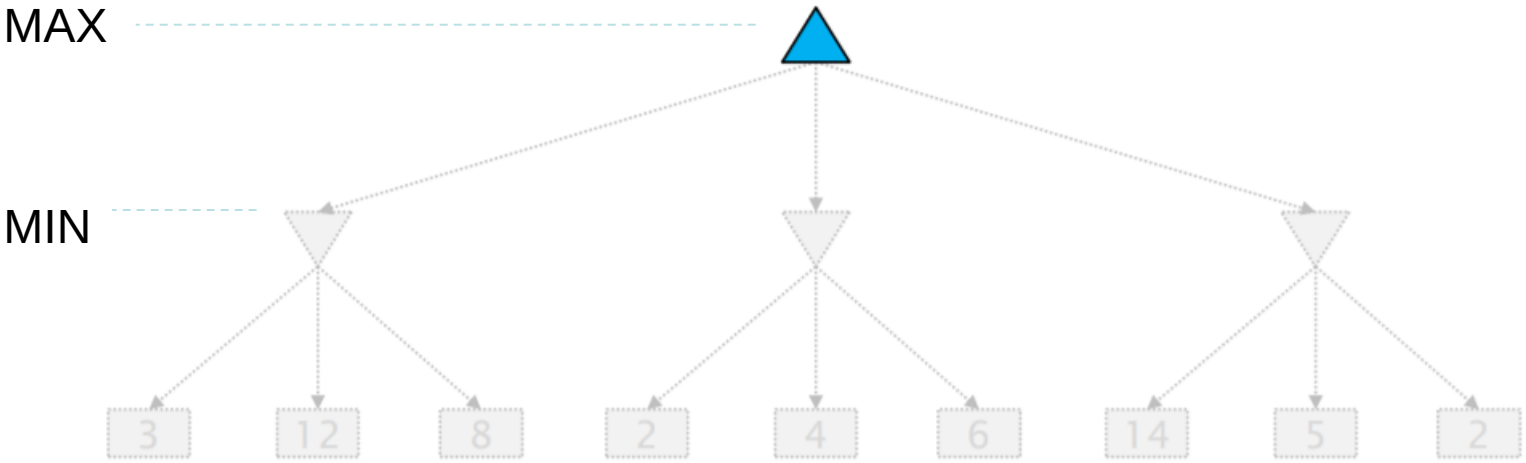
✓ قانون اول: اگر در یک گره MIN بتا از آلفای پدرش کمتر باشد. (هرس باید انجام شود)

قانون دوم: اگر در یک گره MAX آلفا از بتای پدرش بیشتر باشد. (هرس باید انجام شود)

بازی‌ها

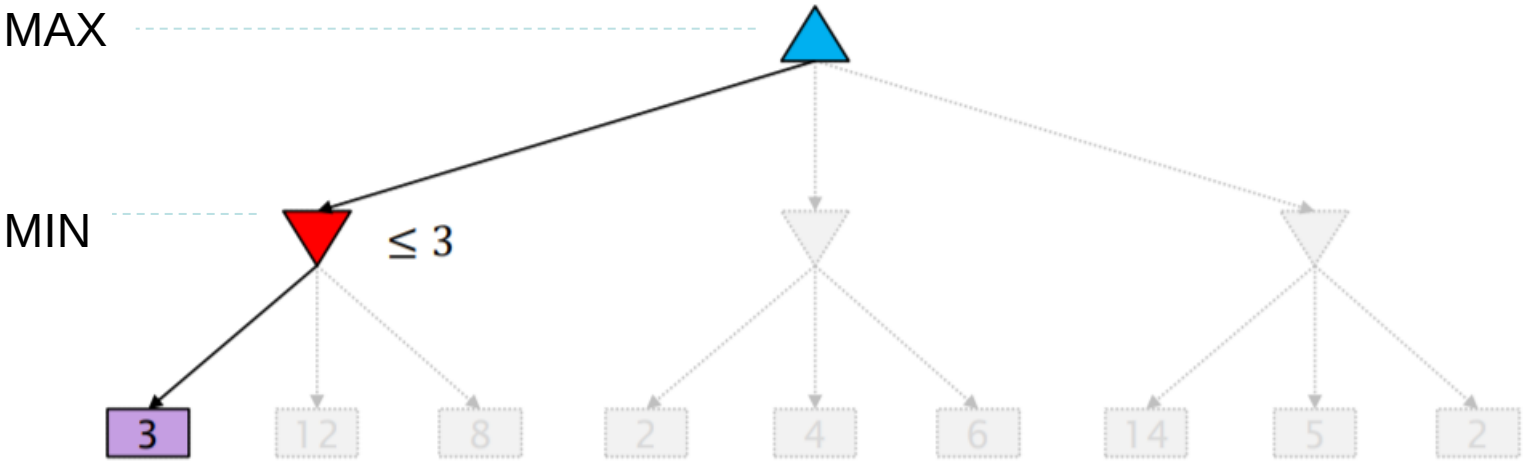
الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره

مثال: فرض کنید بازی را **MAX** شروع می‌کند و در این بازی **MAX** و **MIN** هرکدام فقط یک حرکت انجام می‌دهند و بازی تمام می‌شود و در هر مرحله **MAX** و **MIN** سه حالت را می‌توانند انتخاب کنند.



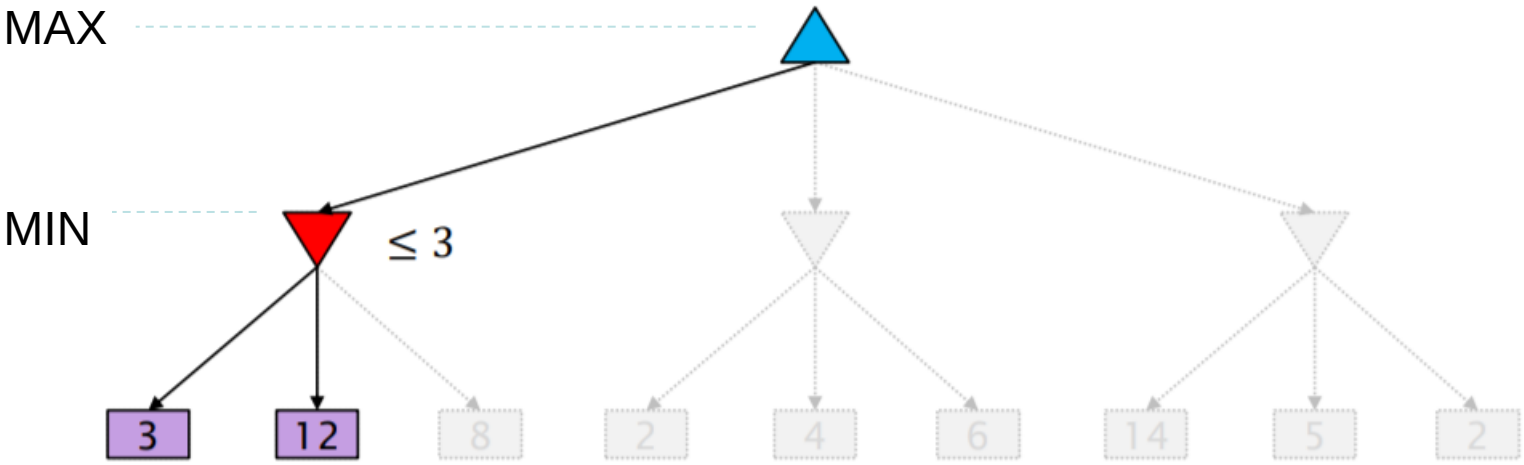
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



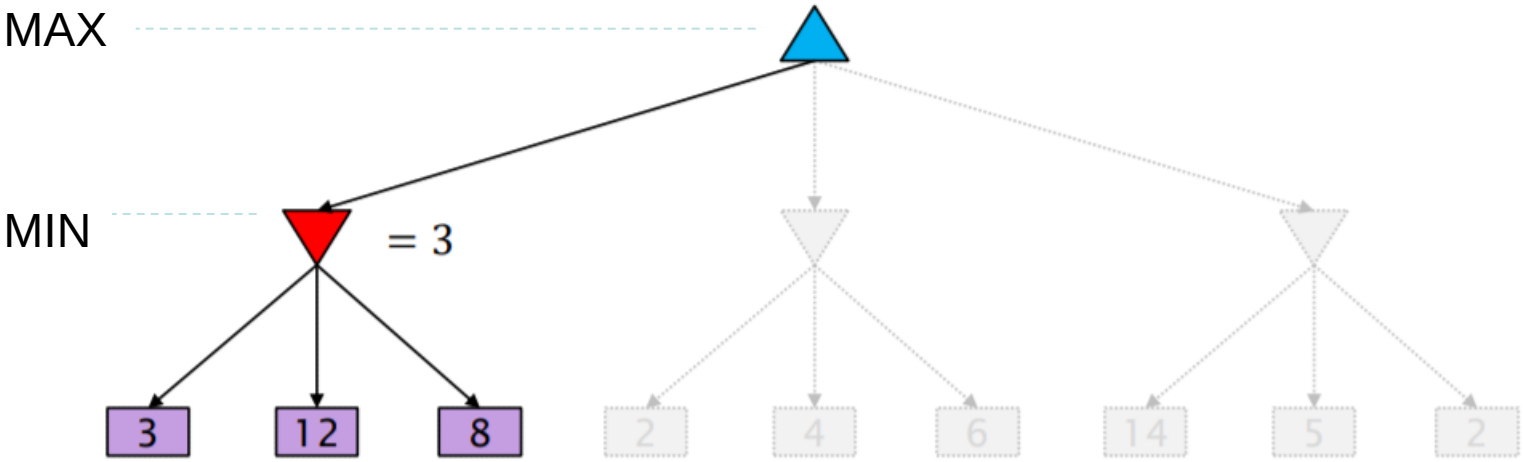
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



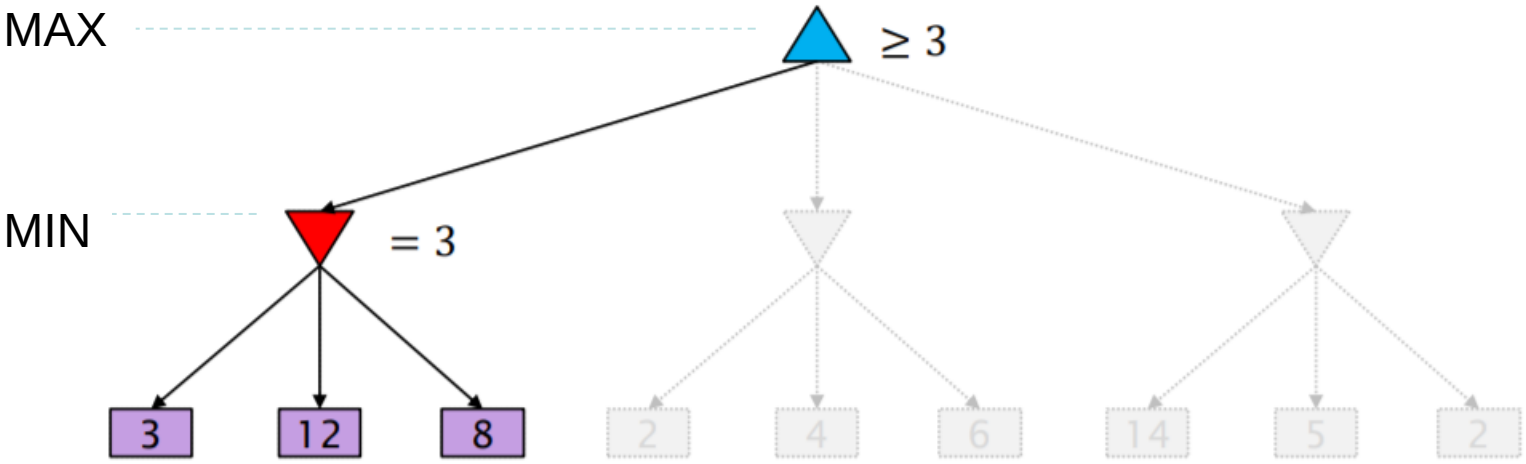
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



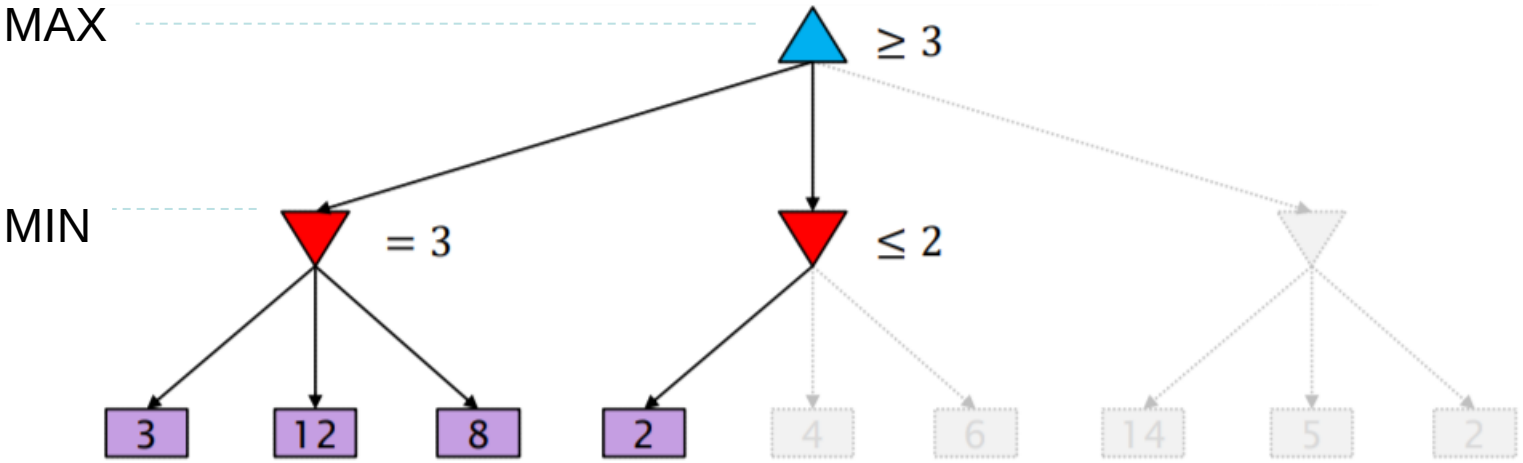
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره

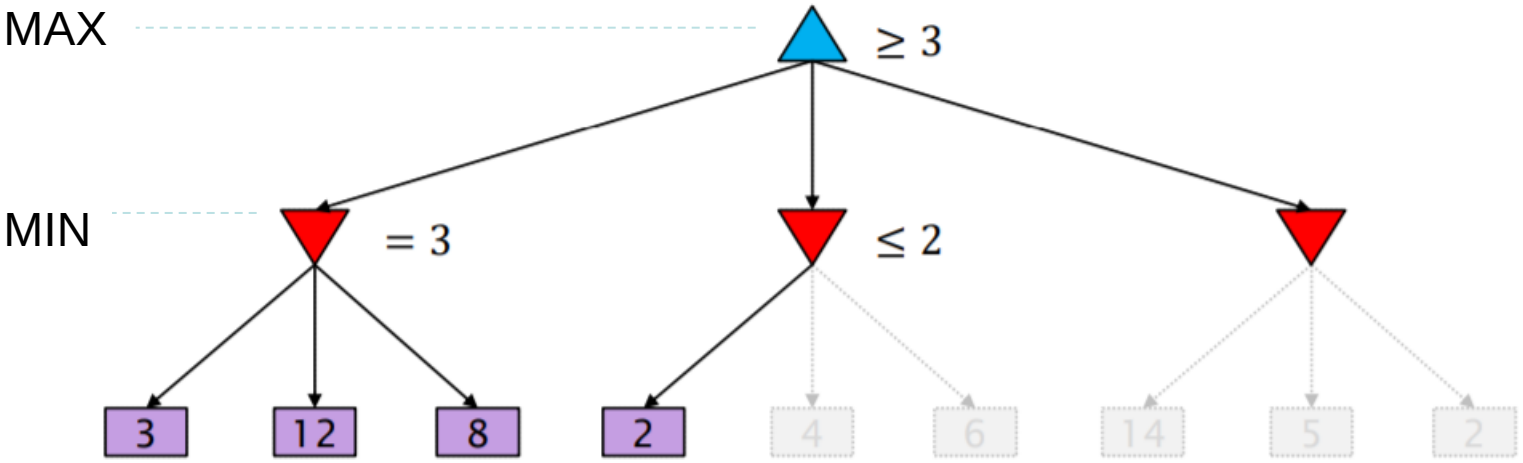


«وقتی فهمیدی چیزی بد است، دیگر برای آن که بفهمی آن چیز واقعاً چقدر بد است وقت تلف نکن».

-- پاتریک وینستون

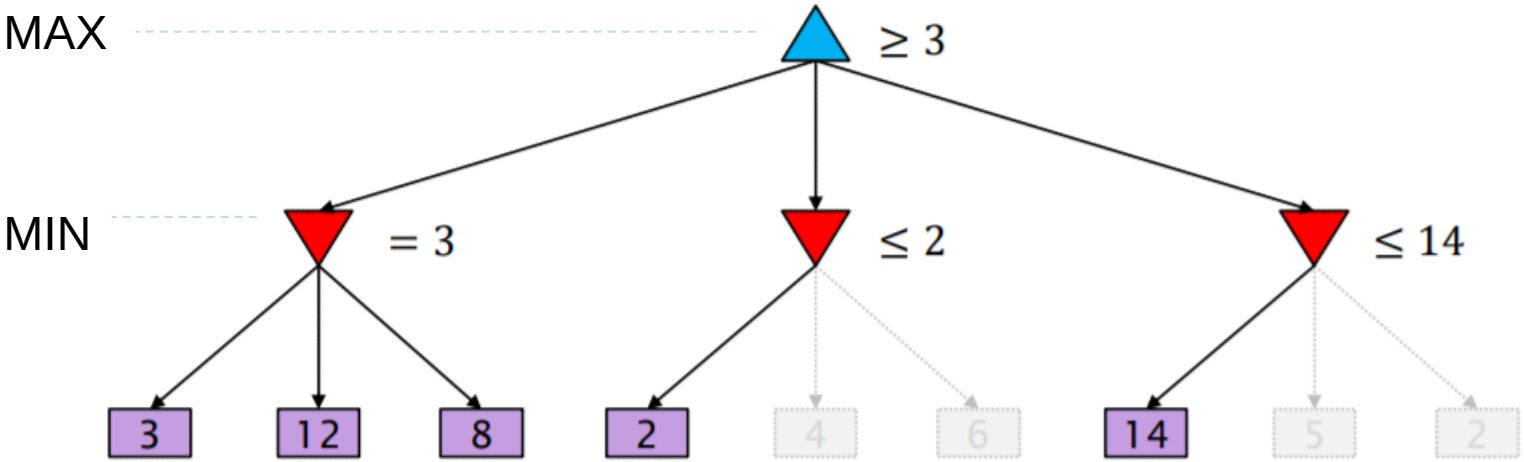
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



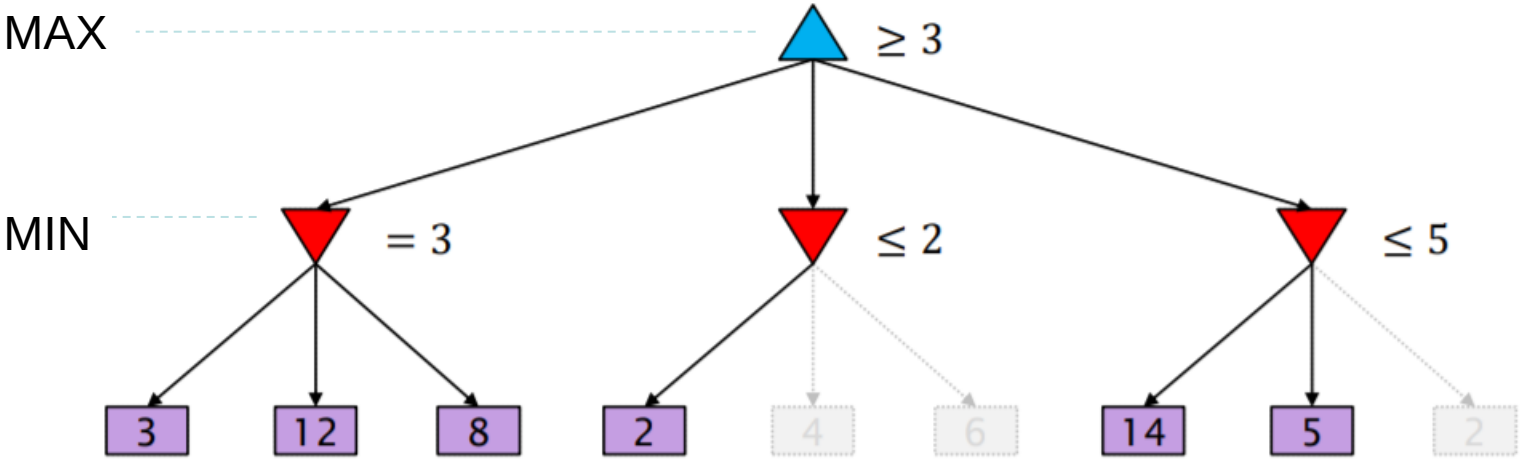
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



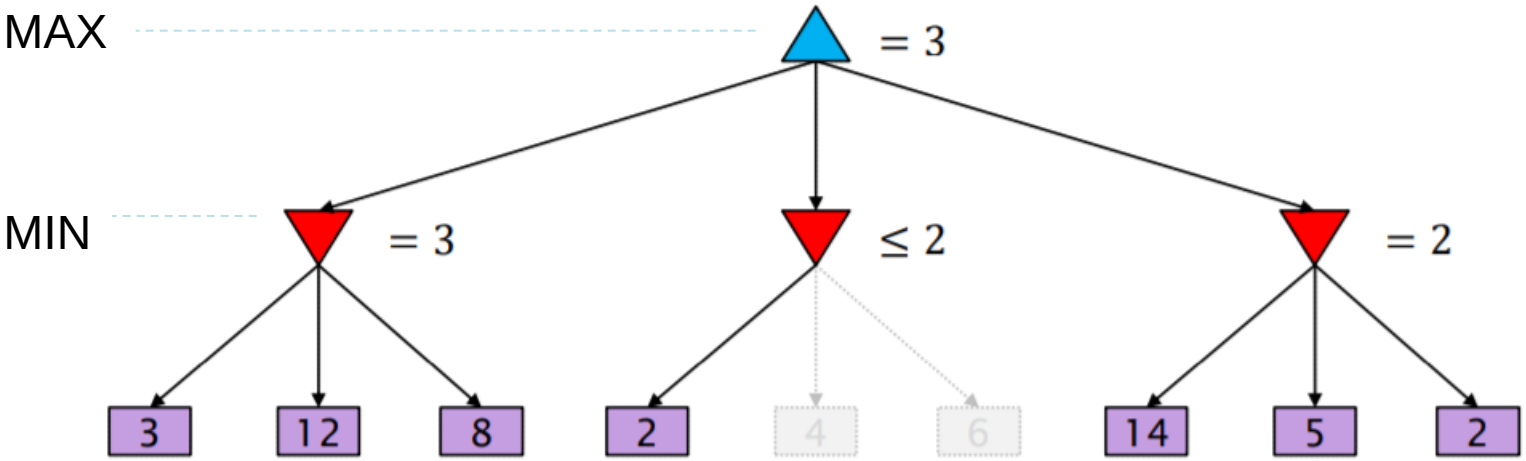
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



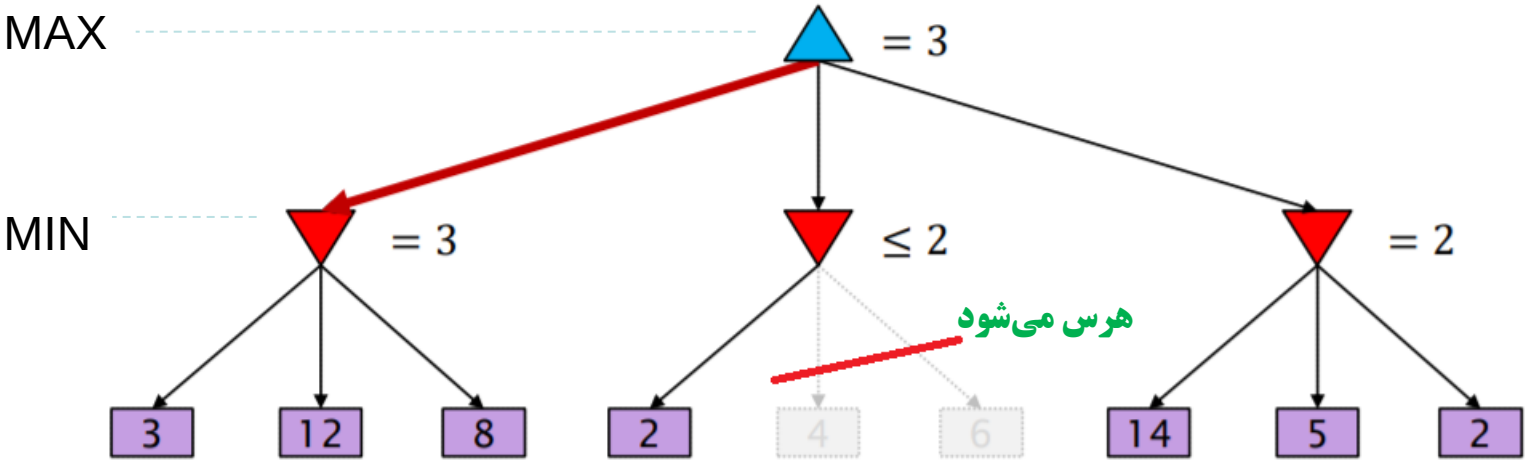
بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



بازی‌ها

الگوریتم هرس (آلفا– بتا) بازی فرضی دو نفره



بازی‌ها

نکته:

- درصورتیکه عمق درخت جستجو برای رسیدن به یک حالت پایانی زیاد باشد،آنگاه با استفاده از هرس نیز در یک زمان معقول اینکار امکانپذیرنخواهد بود.
- برای حل مشکل،می‌توان از یک تست قطع(Cutoff test) استفاده نمود،ودر محل قطع درخت جستجو میزان سودمندی حالت را محاسبه نمود.

بازی‌ها

(پیاده سازی الگوریتم هرس آلفا – بتا)

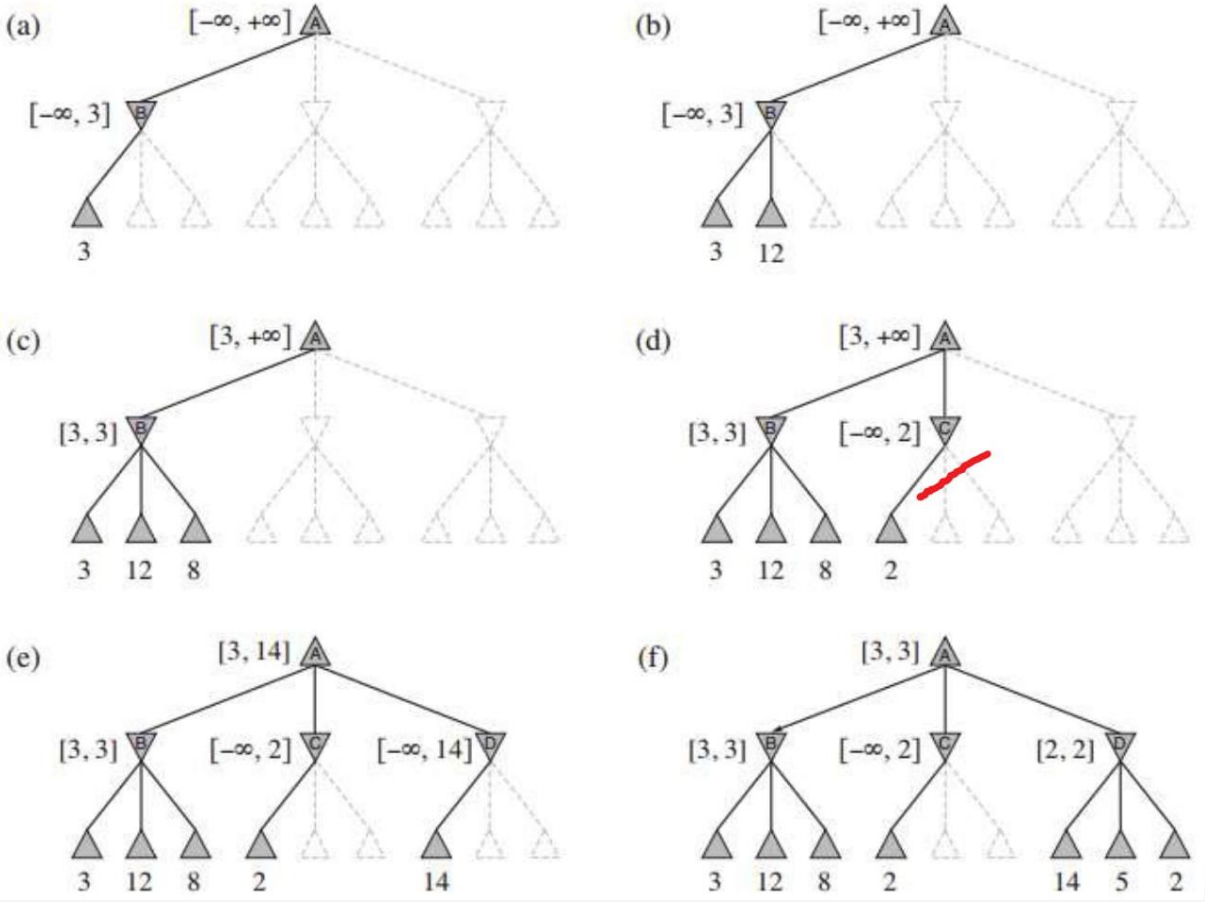
آلفا: بهترین گزینه برای بازیکن MAX روی مسیر تا ریشه
بتا: بهترین گزینه برای بازیکن MIN روی مسیر تا ریشه

```
def max-value(state, α, β):
  initialize v = −∞
  for each successor of state:
     v = max(v, value(successor, α, β))
     if v ≥ β return v
     α = max(α, v)
  return v
```

```
def min-value(state , α, β):
  initialize v = +∞
  for each successor of state:
     v = min(v, value(successor, α, β))
     if v ≤ α return v
     β = min(β, v)
  return v
```

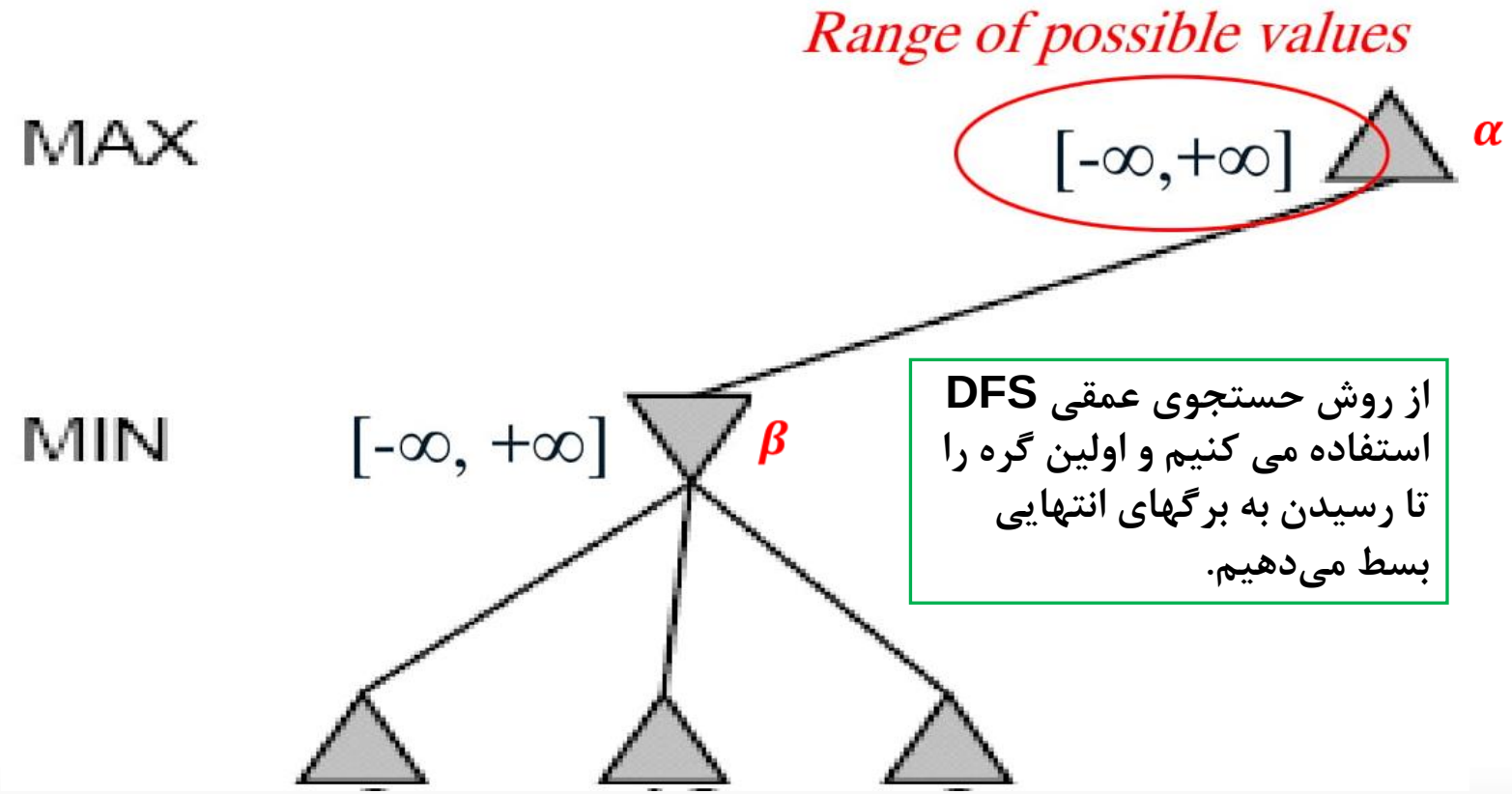
بازی‌ها

(الگوریتم آلفا – بتا در یک نگاه)



بازی‌ها

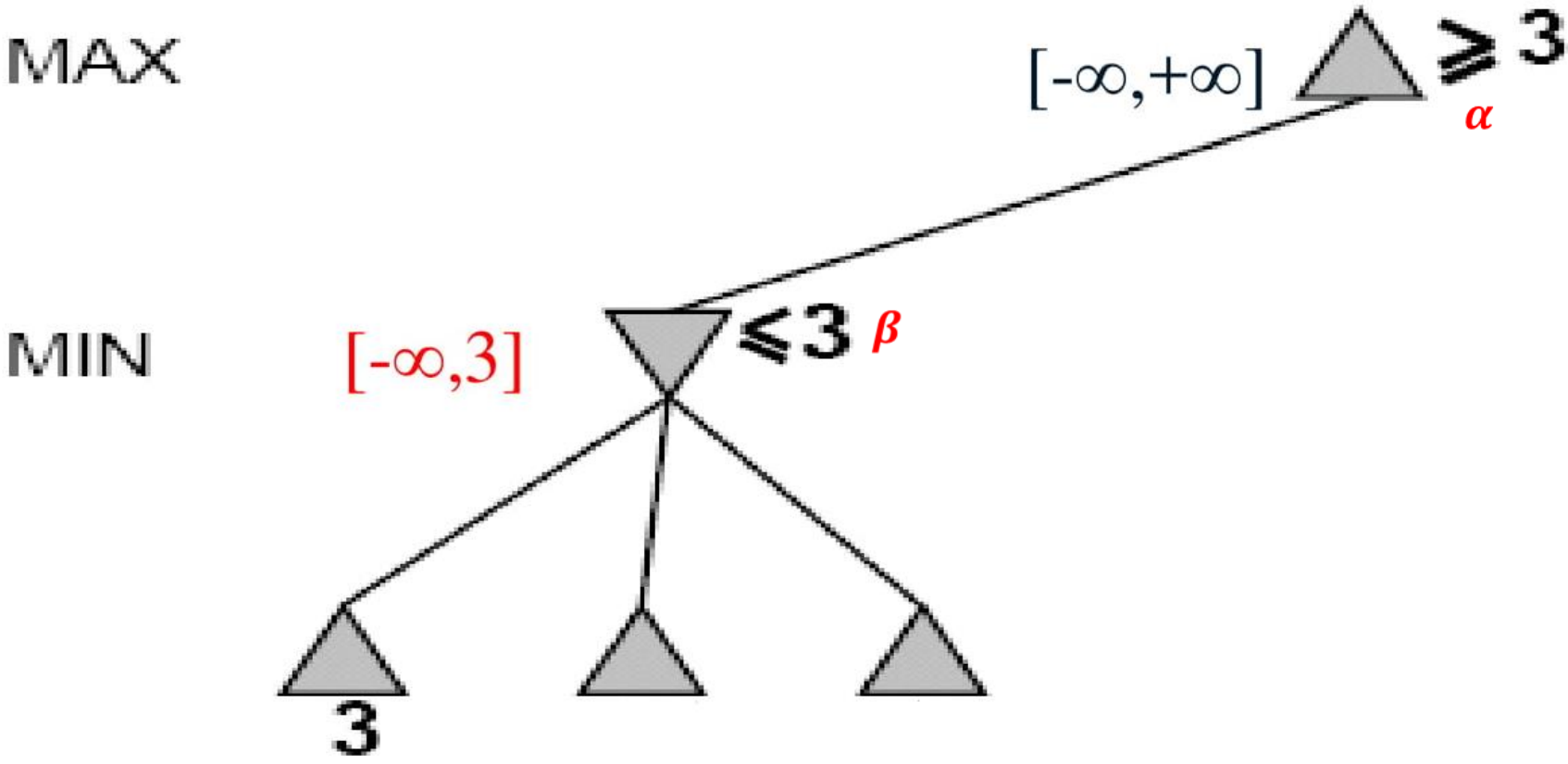
در ادامه روش استفاده از الگوریتم هرس آلفا – بتا در این مسئله را بصورت تفضیلی بررسی می کنیم.



از روش حستجوی عمقی DFS استفاده می کنیم و اولین گره را تا رسیدن به برگهای انتهایی بسط می‌دهیم.

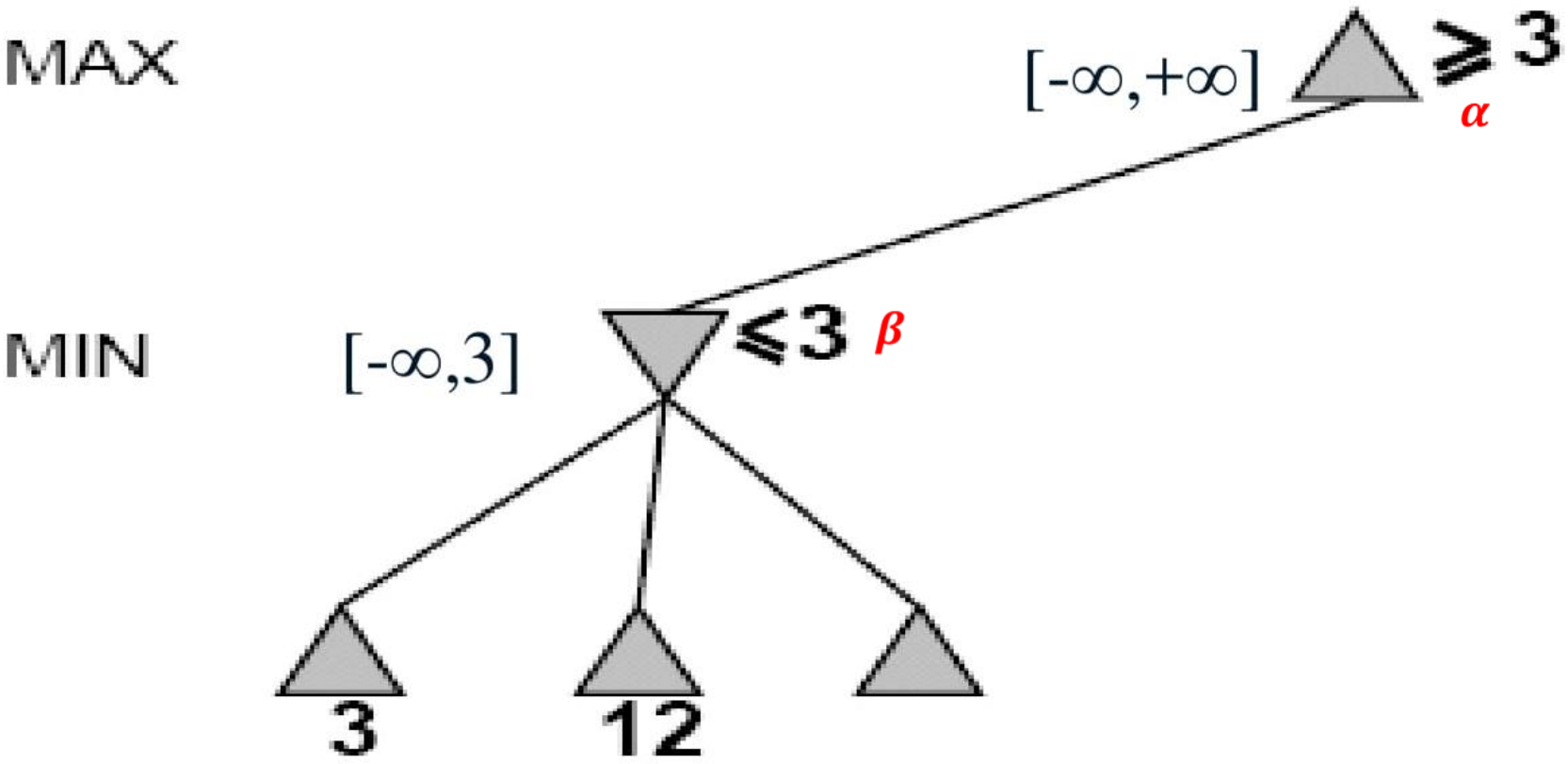
بازی‌ها

(الگوریتم آلفا – بتا بصورت مشروح)



بازی‌ها

(الگوریتم آلفا – بتا بصورت مشروح)



بازی‌ها

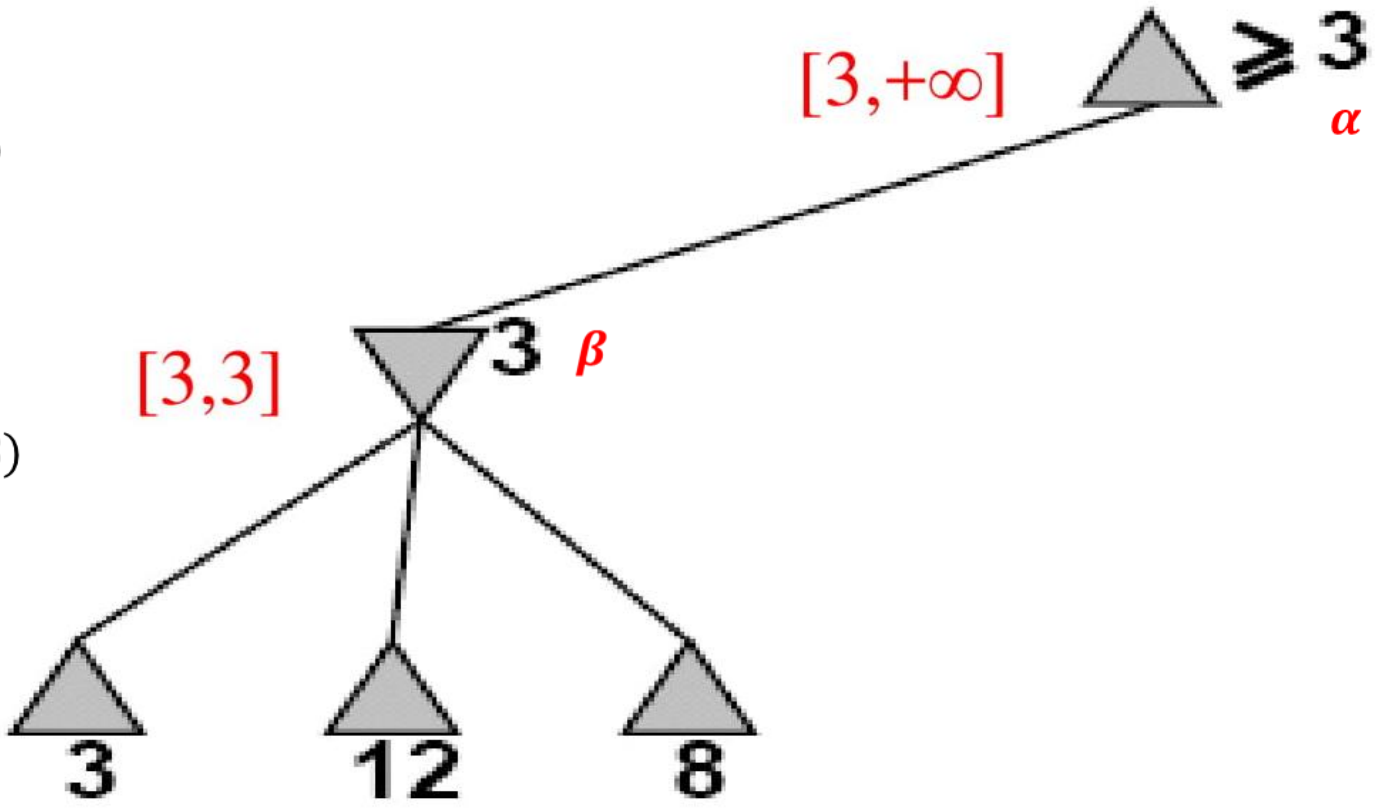
(الگوریتم آلفا – بتا بصورت مشروح)

MAX

$\alpha = \max(3, +\infty)$

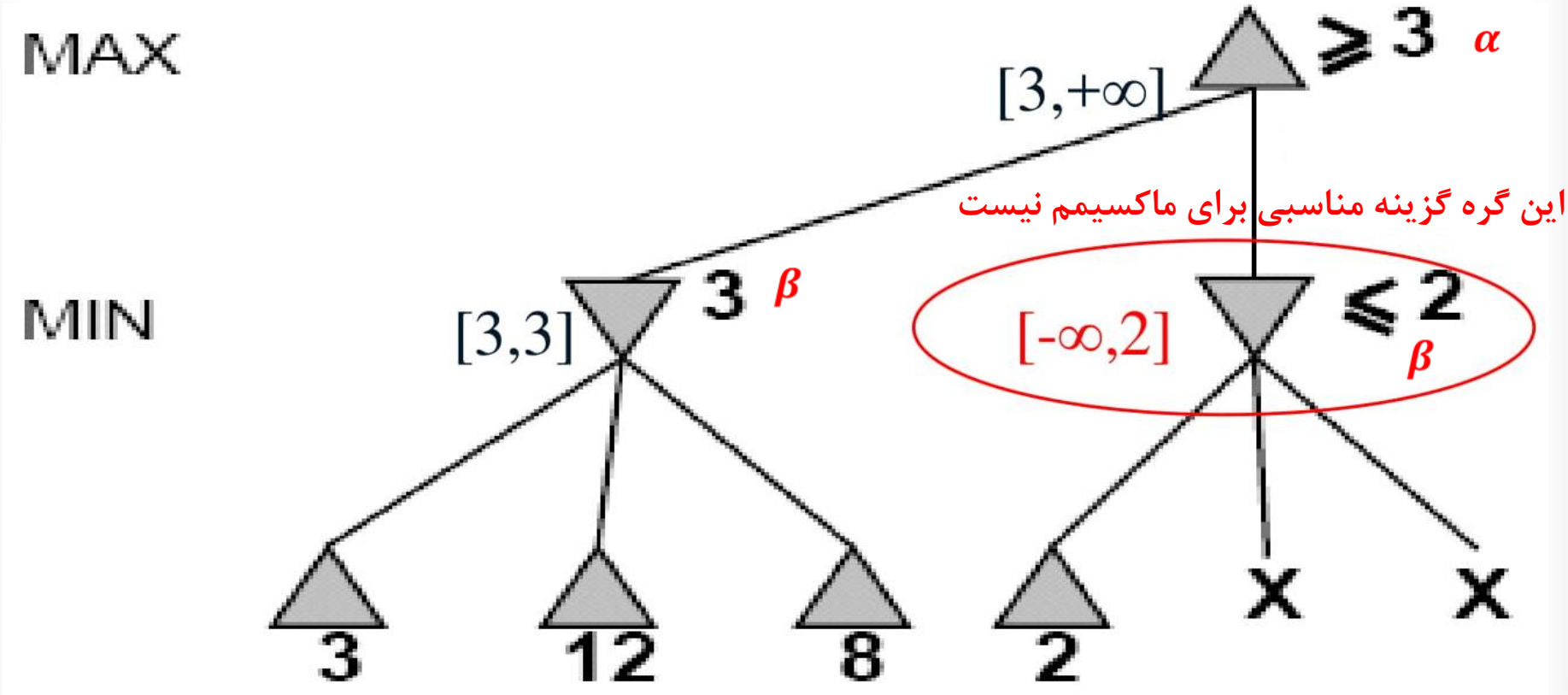
MIN

$\beta = \min(3, 12, 8)$



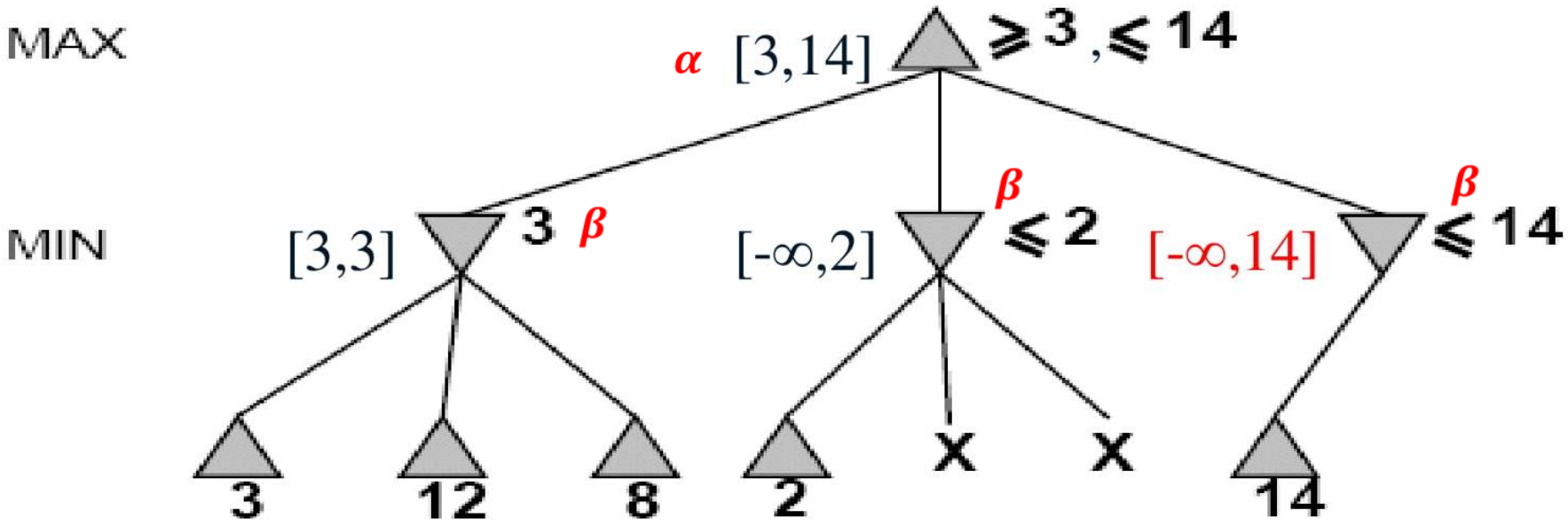
بازی‌ها

(الگوریتم آلفا – بتا بصورت مشروح)



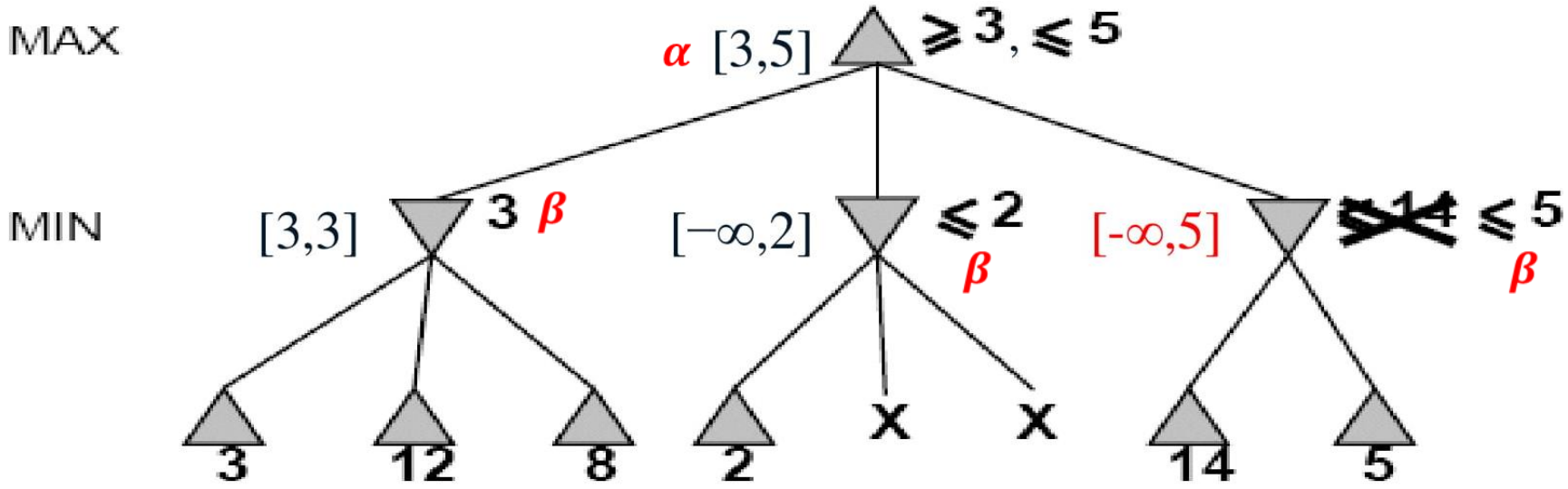
بازی‌ها

(الگوریتم آلفا – بتا بصورت مشروح)



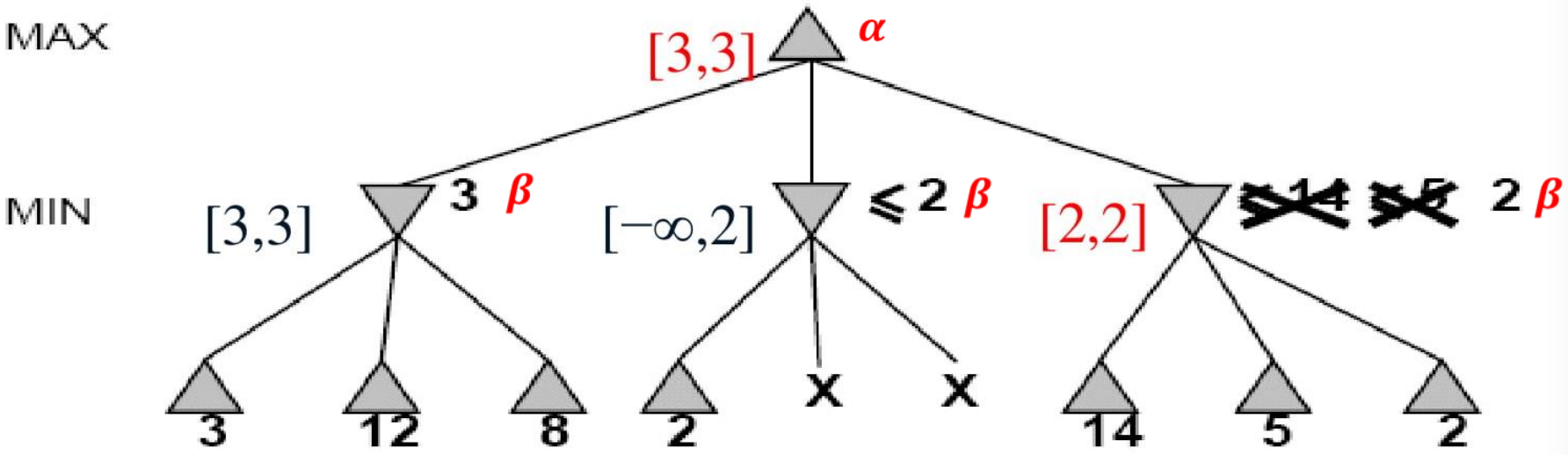
بازی‌ها

(الگوریتم آلفا – بتا بصورت مشروح)



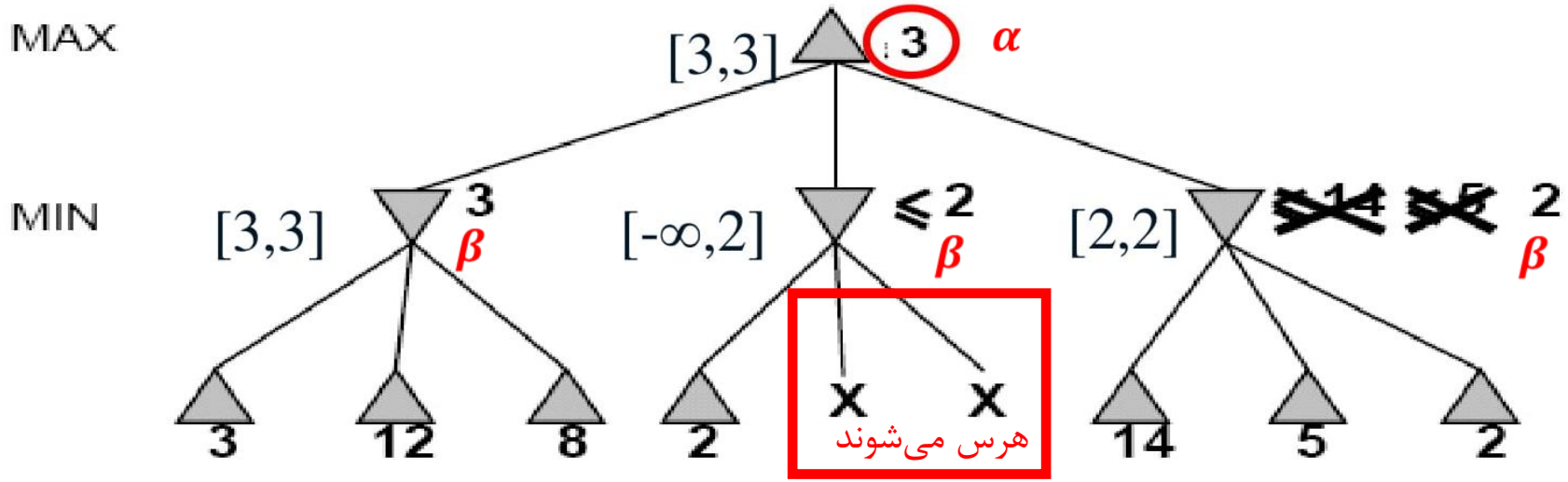
بازی‌ها

(الگوریتم آلفا – بتا بصورت مشروح)



بازی‌ها

(الگوریتم آلفا – بتا بصورت مشروح)



$$MINIMAX\ -VALUE\ (\text{root}) = \max(\min(۳,۱۲,۸), \min(۲, x, y), \min(۱۴,۵,۲)$$

$$= \max(۳, \min(۲, x, y), ۲)$$

$$= \max(۳, z, ۲) \qquad z \leq ۲$$

$$= ۳$$

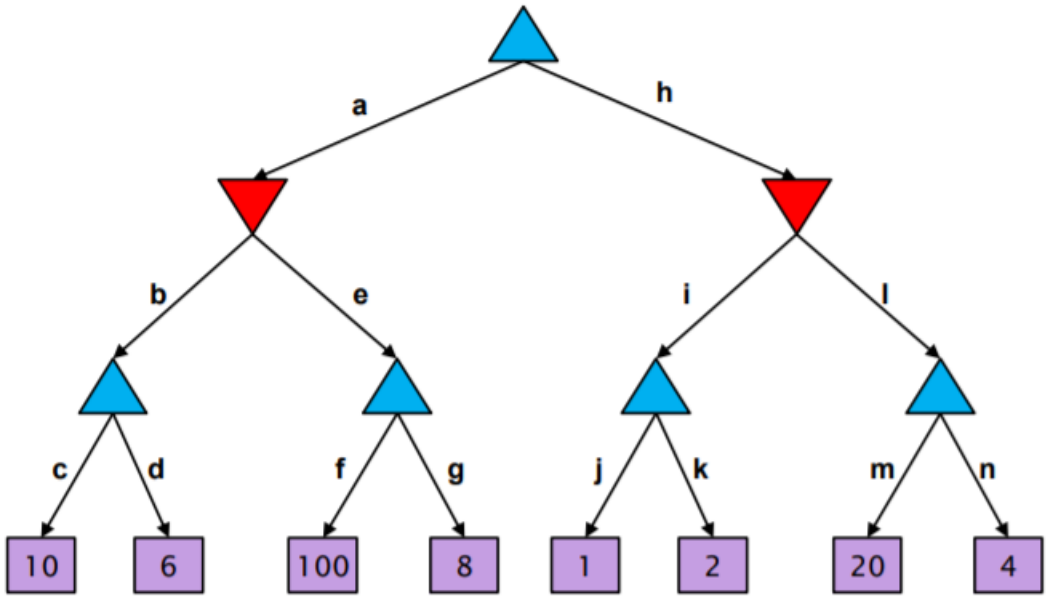
بازی‌ها

Alpha-Beta Algorithm

- هرس تاثیری بر روی نتیجه نهایی ندارد.
- کل زیر درخت هرس خواهد شد.
- مرتب سازی (Perfect Ordering) می تواند بهبود چمشگیری داشته باشد.
  - ترتیب فرزندان یک گره بر روی کارایی هرس کردن تأثیر می‌گذارد.
  - با یک «ترتیب عالی»:
    - پیچیدگی زمانی به $O(b^{m/2})$ کاهش می‌یابد.
    - بنابراین عمق قابل جستجو دو برابر می‌شود.
    - با این حال هنوز هم نمی‌توان به عنوان مثال درخت شطرنج را به طور کامل تا انتها جستجو کرد.

بازی‌ها

تمرین: کدام یک از شاخه‌های درخت زیر با استفاده از هرس آلفا-بتا هرس می‌شوند؟



بازی‌ها

پایان فصل هفتم