هوش مصنوعی

درس نهم: جستجوی خصمانه (رقابتی)

Adversarial Search

سید کاوه احمدی

The Prisoner's Dilemma

		Player 2	
		Cooperate	*Defect*
Player 1	*Cooperate*	3, 3	0, 5
	Defect	5, 0	1, 1

If you are one of them, what will **YOU** do?

محیط‌های چند عامله

همکاری کننده
- هدف عامل‌ها یک چیز است
- یک تیم فوتبال

رقابت کننده
- عامل‌ها رقیب هم هستند
- شطرنج

بازی‌ها

- بازی‌ها یکی از مهمترین مسائل در محیط‌های چند عامله رقابتی هستند.
  - هر عامل نیاز به در نظر گرفتن سایر عامل‌ها و چگونگی تأثیر آنها بر عملکرد خود دارد.
  - اهداف عامل‌ها با یکدیگر برخورد دارند. مسئله خصمانه است!
- چرا مطالعه می‌شوند؟
  - قابلیت‌های هوشمندی انسان‌ها را به کار می‌گیرند
  - ماهیت انتزاعی بازی‌ها
  - حالت بازی را به راحتی می‌توان نمایش داد و عامل‌ها معمولا به مجموعه کوچکی از فعالیت‌ها محدود هستند که نتایج آنها با قوانین دقیقی تعریف شده‌اند

انواع بازی

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble nuclear war

جستجوی خصمانه

- بخشی از تئوری مشهور بازی است.
  - ایده اصلی از رشته اقتصاد می‌آید (تعادل نَش).
- تفاوتش با بازی معمای زندانی این است که بازیکنان به نوبت بازی می‌کنند.
- در مقوله جستجو قرار می‌گیرد.
- تفاوت عمده با جستجوهای کلاسیک این است که همزمان چند رقیب (opponent) به دنبال رسیدن به هدف هستند.
  - مسیر رسیدن (کوتاه‌ترین مسیر) به هدف مهم نیست.
  - مهم این است که قبل از دیگران به هدف برسیم!
    - برای این منظور بازی حریف را خراب می‌کنیم!

خصوصیات بازی‌های مورد مطالعه در این بخش

- بازی دو نفره است
  - بازی‌ها تیمی (گروهی) نیستند.
- بازی جمع-صفر (zero sum) است.
  - به میزان برد یک نفر، طرف مقابل ضرر می‌کند.
- نوبت بازی چرخشی است.
  - نه لزوما یک در میان
  - اگر نحوه چرخش مشخص نباشد، امکان ترسیم درخت بازی وجود نخواهد داشت.
- تعریف دقیق برد، باخت و تساوی در مسئله عنوان شده است.
  - در حالت کلی همه بازی‌ها برنده ندارند.

خصوصیات بازی‌های مورد مطالعه در این بخش

- صفحه بازی کاملا قابل مشاهده است.
  - داشتن اطلاعات کافی (کامل) از بازی ضروری است.
- عامل شانس حذف شده (تصادفی نیست).
- در این بازی‌ها میزان عقلانیت حریف مطرح نیست.
- بازیکن خودی Max و حریف min است.
  - Max می‌خواهد به حداکثر امتیاز برسد و min می‌خواهد Max به حداقل امتیاز برسد.

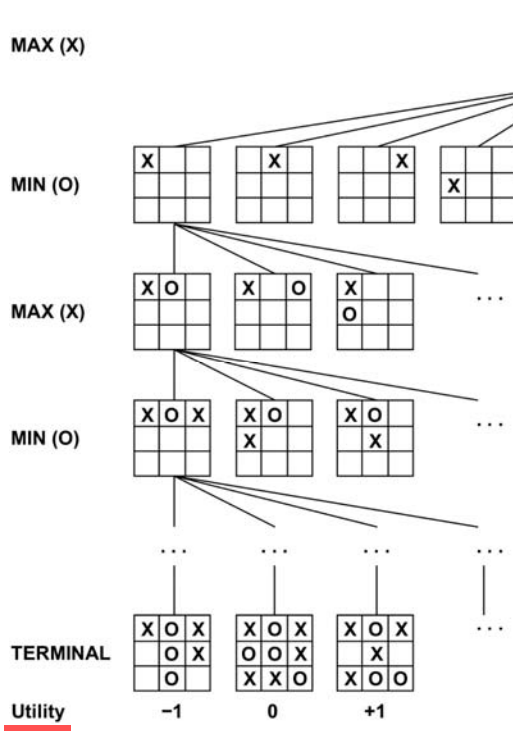
بازی‌های دو نفره بازی به عنوان یک جستجو

- حالت اولیه: موقعیت صفحه و شناسه‌های قابل حرکت
- تابع مابعد: لیستی از زوج مرتب‌های (action, state) که معرف یک حرکت معتبر و وضعیتی که با آن حرکت به آن می‌رسیم است.
- تابع هدف: پایان بازی چه موقع است؟ (حالت‌های پایانی)
- تابع سودمندی: برای هر حالت پایانی یک مقدار عددی را ارائه می‌کند.
  - Utility(S) = Max احتمال برد - min احتمال برد
    - مثلاً برنده (+۱) و بازنده (−۱) و در حالت مساوی هر دو صفر
- حالت اولیه و حرکات معتبر برای هر بازیکن، **درخت بازی** را برای آن بازی ایجاد می‌کند

بازی دو نفره Min و Max

- اول Max حرکت می‌کند سپس min و به همین ترتیب به نوبت بازی می‌کنند تا بازی تمام شود.
- در پایان بازی، برنده جایزه می‌گیرد و بازنده جریمه می‌شود.
- همه‌ی تصمیم گیری‌ها دست Max نیست و حرکت min نیز در درخت بازی موثر است. به همین دلیل درخت خیلی وسیع می شود و قابل رسم نیست.
  - Max: چون سعی در حداکثر کردن امتیاز ما دارد
  - min: چون سعی در حداکثر کردن امتیاز ما دارد

بازی دوز (Tic-Tac-Toe) – درخت بازی



- x متعلق به Max و o متعلق به min است.
- اگر بازیکنی سه مهره خود را در یک ردیف سه‌تایی (افقی، عمودی یا قطری) قرار دهد، یا اینکه صفحه بازی پر شود، بازی به اتمام رسیده است.
- در حالت اولیه Max می‌تواند ۹ حرکت انجام دهد و بعد از آن به نوبت هر کدام یک حرکت انجام می‌دهند.
- عدد نسبت داده شده به هر برگ درخت بازی، میزان سودمندی آن حالت پایانی از دید بازیکن Max را نشان می‌دهد.

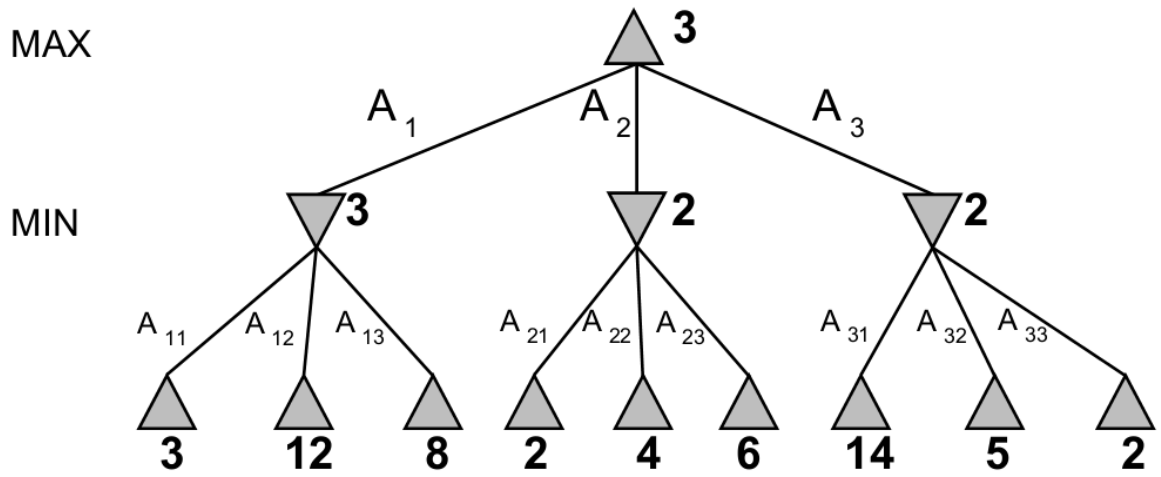
درخت بازی

- ۹! حالت و ارتفاع درخت حداکثر ۹ است.
- در بازی شطرنج، ارتفادع متوسط درخت ۱۰۰ و فاکتور انشعاب ۳۵ است!

الگوریتم Minimax

- بازی چگونه انجام می‌شود؟!
  - الگوریتم Minimax در واقع یک جستجوی اول عمق در درخت بازی است.
  - در هر مرحله یک نود را به عنوان حالت بهینه انتخاب می‌کنیم و اگر حریف حرکت مورد نظر ما را انجام نداده باشد از سایر مسیرها باید استفاده شود:
    - Max نودی را انتخاب می‌کند که بیشترین فایده را برایش داشته باشد.
    - min نودی را انتخاب می‌کند که کمترین بازده را برای Max داشته باشد (بیشرین بازده برای خودش است).
  - این استراتژی با فرض اینکه min بهترین عمل برای خود (که بدترین عمل برای Max است) را انتخاب می‌کند و zero sum بودن بازی، تصمیم بهینه گرفته می‌شود.

مثال الگوریتم Minimax برای یک بازی دونفره



- این الگوریتم مقادیر minimax هرحالت جانشین را با استفاده از محاسبه‌ی بازگشتی بدست می‌آورد.
- ابتدا به سمت برگ‌ها حرکت می‌کند و سپس هنگام بازگشت مقادیرminimax ذخیره می‌شوند.

الگوریتم Minimax

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

بازی را Max شروع می‌کند: فرزندان min هستند.

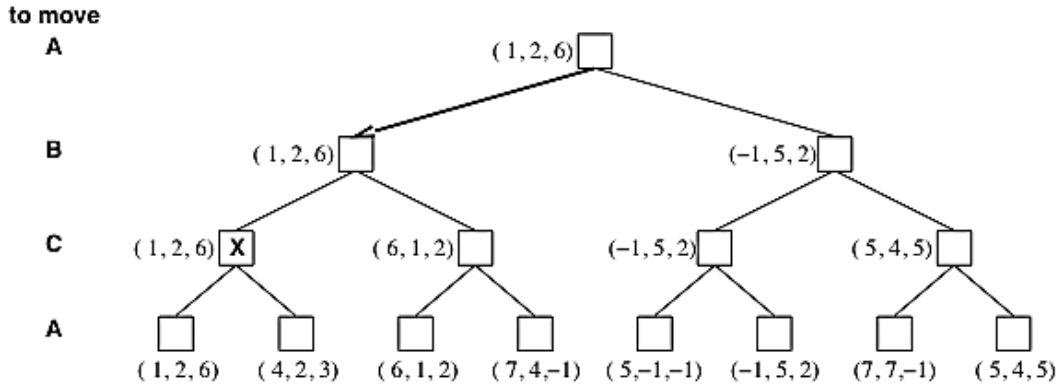
بعد از Max، minب ازی می‌کند

الگوریتم Minimax

- کامل بودن: بله (اگر درخت محدود باشد)
- بهینگی: بله
- Minimax درخت بازی را به صورت عمقی اکتشاف می‌کند اگر حداکثر عمق درخت m باشد و در هر نقطه b حرکت معتبر وجود داشته باشد:
  - پیچیدگی زمانی: $O(b^m)$
  - پیچیدگی فضا: $O(bm)$

بازی‌های چند نفره

- در بازی‌های چند نفره، هر بازیکن سعی می‌کند با توجه به حرکت سایر بازیکنام، برداری را انتخاب کند که برای خودش بیشترین سودمندی را داشته باشد.
  - در هر بردار، اعداد به ترتیب سودمندی A، B و C را نشان می‌دهند.



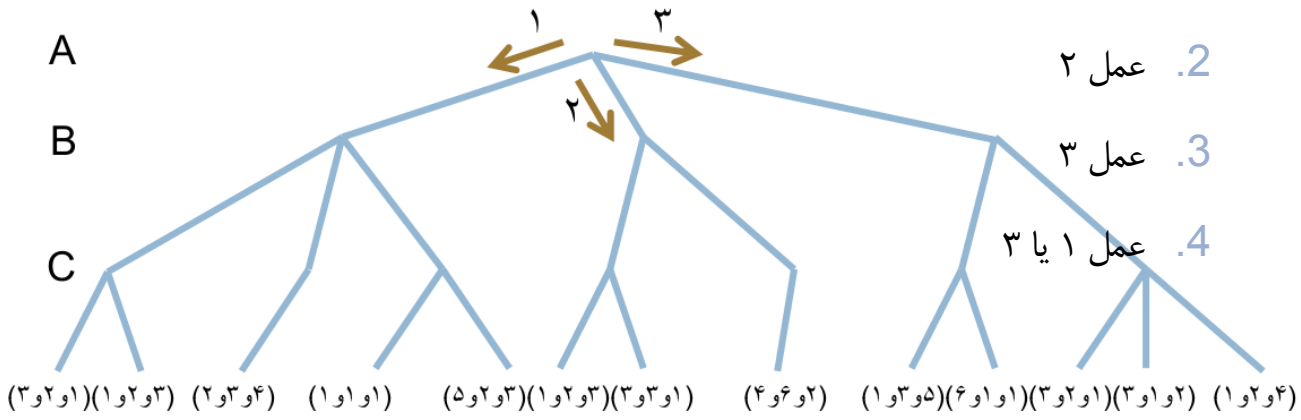
بازی‌های چند نفره

- بازی‌های چند نفره (non-zero sum) معولا شامل اتحاد رسمی یا غیر رسمی بین بازیکنان است.
  - اتحاد بر مبنای منافع و نه تبانی
  - اتحاد (Alliance) در ابتدای بازی ایجاد می‌شود و سپس با پیشروی بازی از بین می‌رود.
  - بازیکنان بطور خودکار همکاری می‌کنند، تا به هدف مطلوب انحصاری برسند.
  - مثال: A و B ضعیفتر از C هستند و هر کدام به تنهایی در مقابل آن شکست می‌خورند، پس در ابتدای بازی به جای حمله به یکدیگر با هم علیه C متهد می‌شوند و با پیشروی بازی و ضعیف شدن C بطور خودکار اتحاد آنها شکسته می‌شود.

آی تی ۸۵

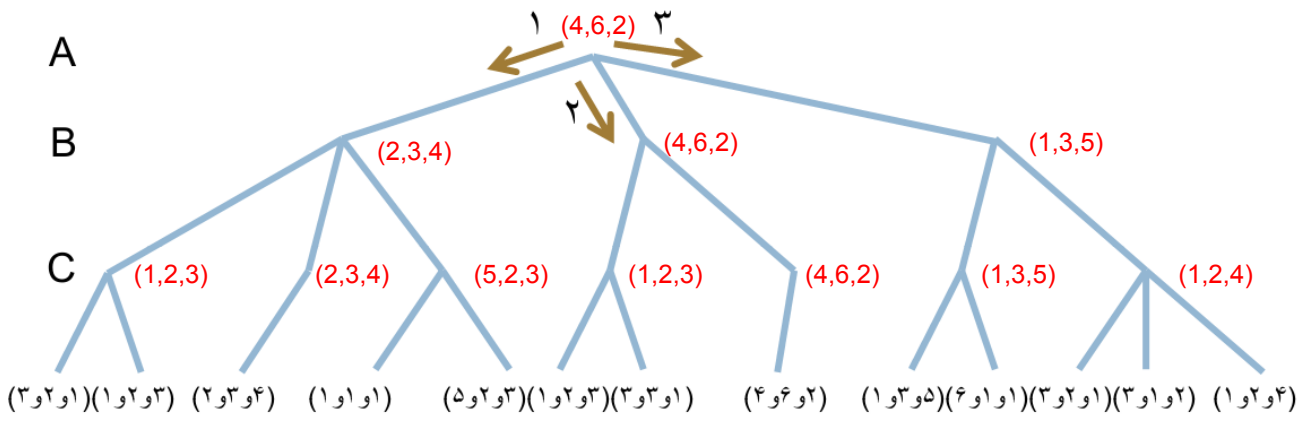
- در یک بازی سه نفره سه بازیکن A و B و C به ترتیب یک بار حق انتخاب عمل دارند. اگر درخت بازی به شکل زیر باشد بازیکن A با استفاده از الگوریتم Minimax چه عملی را انتخاب خواهد کرد؟

1. عمل ۱
2. عمل ۲
3. عمل ۳
4. عمل ۱ یا ۳



آی تی ۸۵

- پاسخ گزینه ۲ است.

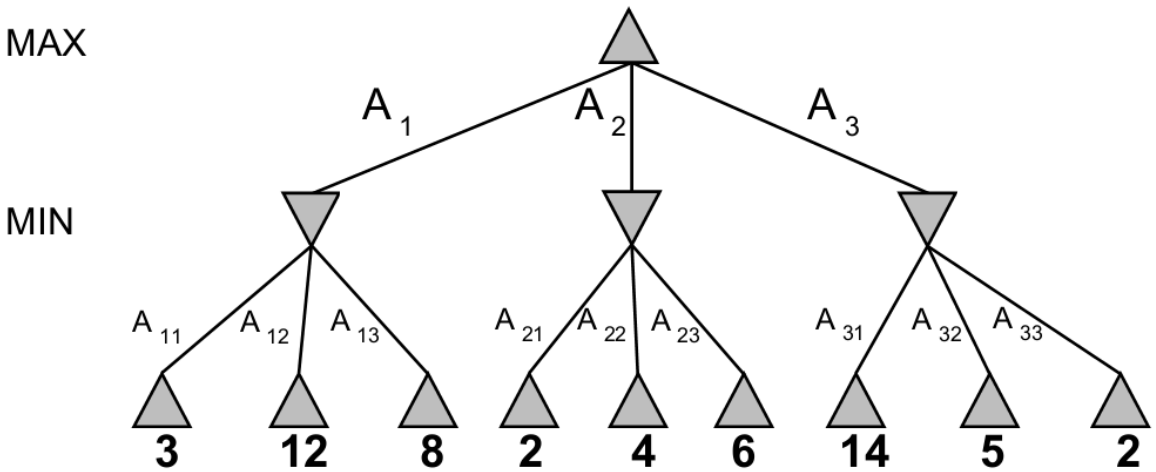


هرس (Pruning)

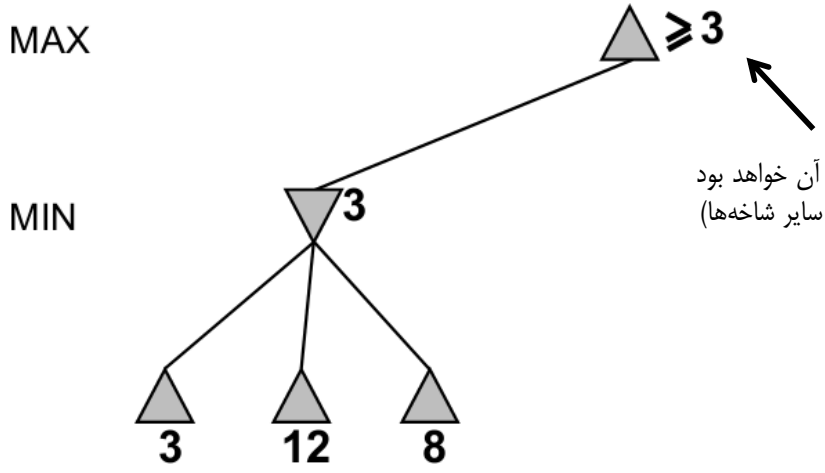
- در الگوریتم Minimax:
  - تعداد حالت‌های بازی که باید بررسی شوند، بر حسب تعداد حرکت‌ها، نمایی است.
    - راه حل: محاسبه تصمیم الگوریتم، بدون دیدن همه گره‌ها امکانپذیر است
    - می‌توان قسمت‌هایی از درخت را که هیچگاه جواب مسأله نیستند، بدون دیدن و قبل از تشکیل حذف کرد (Pruning).
- روش‌های هرس همیشه منجر به بهبود حافظه می‌شوند.
  - هدف اصلی هرس بهبود حافظه و توانایی پیمایش تا عمق بیشتر است.
  - در صورتیکه یک زیر درخت بزرگ هرس شود ممکن است در زمان نیز صرفه جویی شود.

هرس α-β (α−β Pruning)

- هرس α-β انشعاب‌هایی که در تصمیم نهایی تاثیر ندارند را حذف می‌کند.

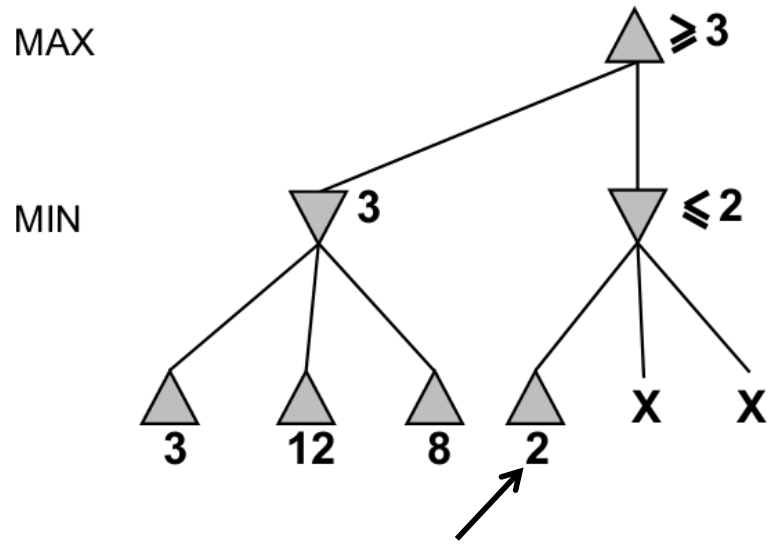


هرس α-β (α−β Pruning)



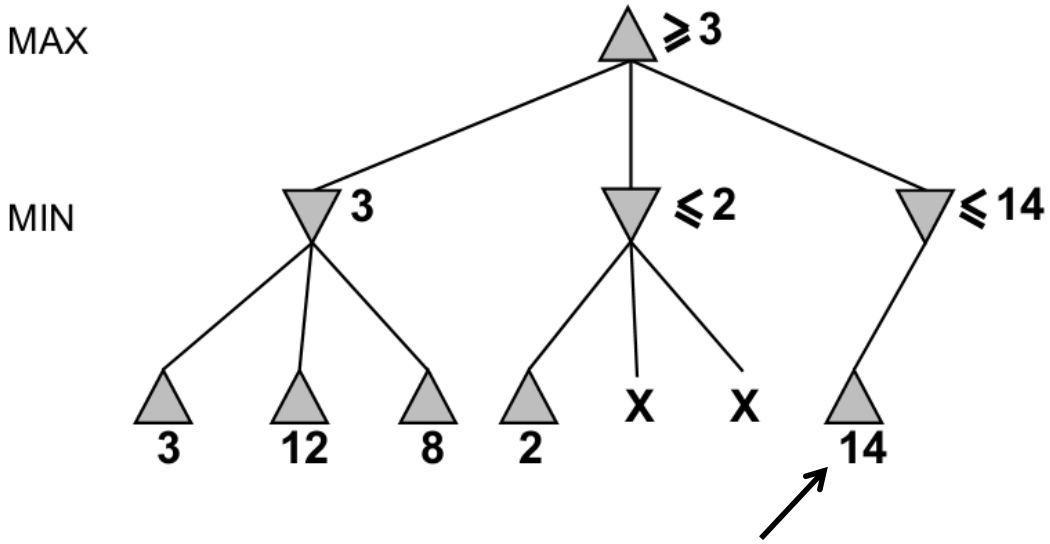
انتخاب Max، ۳ یا عددی بزرگتر از آن خواهد بود
(در صورت پیدا کردن چنین عددی در سایر شاخه‌ها)

هرس α-β (α−β Pruning)



مطابق آنچه گفته شد، min در این شاخه عدد ۲ یا عددی کوچکتر از آن را انتخاب خواهد کرد. Max در شاخه قبلی می‌تواند به مقدار ۳ برسد بنابراین هیچگاه وارد این شاخه نمی‌شود. با دیدن ۲ و با دانستن اینکه این عدد برای Max مطلوب نیست، می‌توان سایر اعداد این شاخه را هرس کرد.

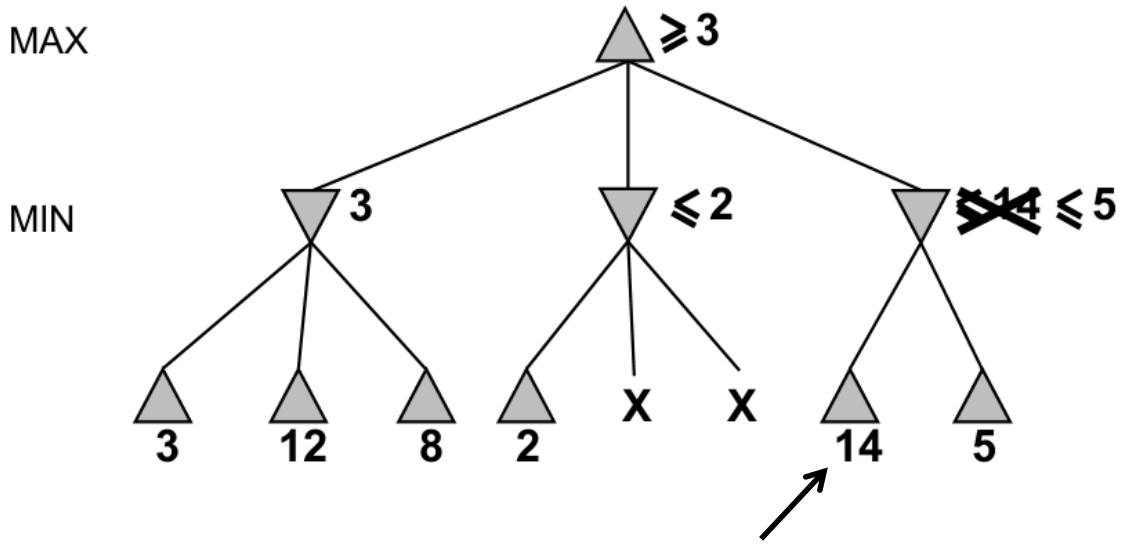
هرس α-β (α−β Pruning)



در این حالت می‌دانیم که min در این شاخه عدد ۱۴ یا عددی کوچکتر از آن را انتخاب خواهد کرد.
این انتخاب می‌تواند برای Max مطلوب باشد.

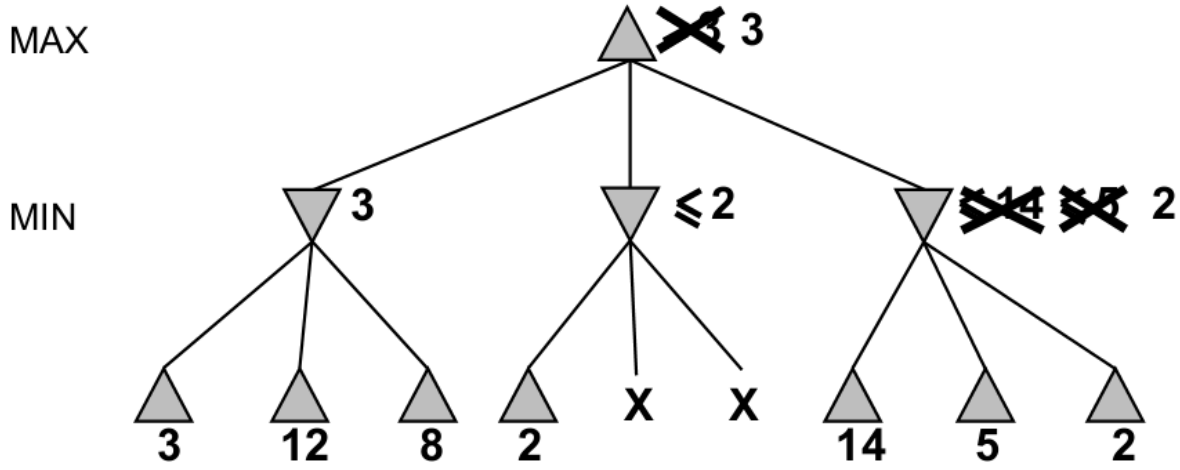
هرس α-β (α−β Pruning)

- انشعاب‌هایی که در تصمیم نهایی تاثیر ندارند را حذف می‌کند.



با دیدن ۵ می‌دانیم که min در این شاخه عدد ۵ یا عددی کوچکتر از آن را انتخاب خواهد کرد. این انتخاب کماکان برای Max مطلوب است.

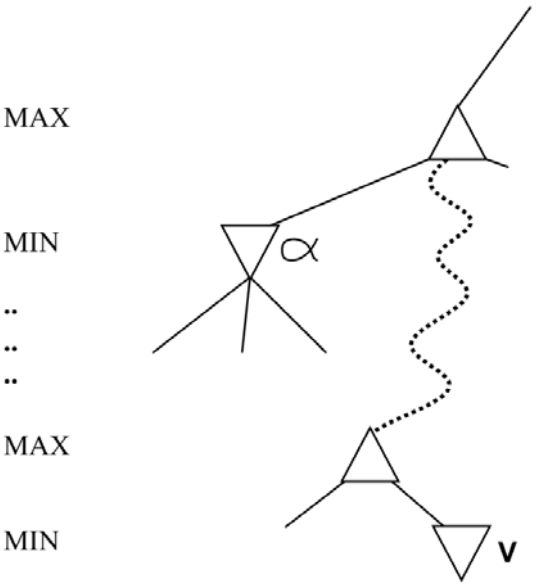
هرس α-β (α−β Pruning)



هرس α-β (α−β Pruning)

- به وسیله هرس α-β می‌توان این ایده را پیاده‌سازی کرد.
- این هرس نام خود را از دو پارامتر تصمیم گیری روی کران‌های خود گرفته است:
  - α: مقدار بهترین (بیشترین) انتخاب که تا کنون در طول مسیر برای Max داشتیم.
  - β: مقدار بهترین (کمترین) که تا کنون در طول مسیر برای min داشتیم.
- جستجو تحت هر گره min که مقدار β آن کوچکتر یا مساوی مقدار α هر گره Max پدر باشد متوقف خواهد شد.
- جستجو تحت هر گره Max که مقدار α آن بزرگتر یا مساوی مقدار β هر گره min پدر باشد متوقف خواهد شد.

هرس α-β (α–β Pruning)



- گره n که هر جای درخت می‌تواند باشد، بررسی می‌شود.
  - اگر بازیکن انتخاب بهتری مانند m داشته باشد (در گره پدر n یا اجداد آن)، n هیچوقت در بازی واقعی قابل دسترس نخواهد بود، در نتیجه هرس می‌شود.
- هرس α-β می‌تواند به درخت با هر عمقی اعمال شود و به جای یک نود ممکن است یک زیر درخت بزرگ را هرس کند.

آی تی ۸۳

- اگر در روش هرس α-β که به طور عادی از روش جستجوی اول عمق بهره می‌گیرد، از روش جستجوی اول سطح استفاده نماییم، کدام یک از گزینه‌های زیر صحیح خواهد بود؟

1. جستجو پاسخ بهینه مسئله را می‌یابد.
2. تغییری در عملکرد روش صورت نمی‌گیرد.
3. کارایی جستجو در حد روش MINIMAX کاهش می‌یابد.
4. کارایی روش افزایش می‌یابد، اما حافظه بیشتری نیز مصرف می‌گردد.

آی تی ۸۳

- گزینه ۳ صحیح است.
- در سطح پیش می‌رویم همه نودها تشکیل می‌شوند و هرس نداریم

آی تی ۸۹

- کدام عبارت در مورد جستجوی min-max و هرس α-β غلط است؟

1. در جستجوی min-max فقط بهترین راه حل با بیشترین امتیاز برای بازیکن max تولید می‌شود.
2. هرس α باعث حذف برخی زیر شاخه‌ها می‌شود.
3. در جستجوی min-max بازیکن رقیب باید بهترین بازی خود را انجام دهد.
4. هرس α-β باعث افزایش سرعت جستجو می‌شود.

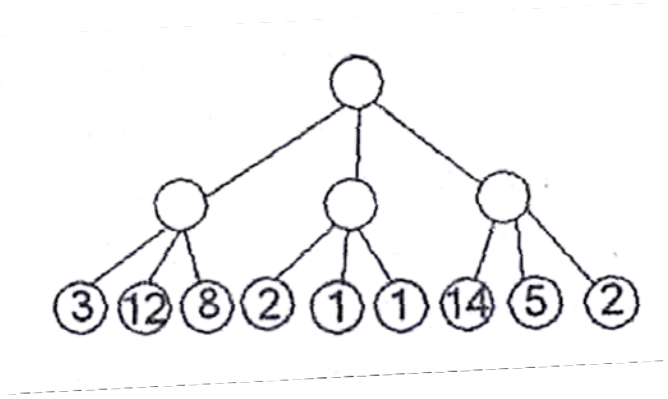
آی تی ۸۹

- گزینه ۴ صحیح است.
  - هرس α-β باعث بهبود مصرف حافظه و توانایی پیمایش تا عمق بیشتر می‌شود.

آی تی ۸۹

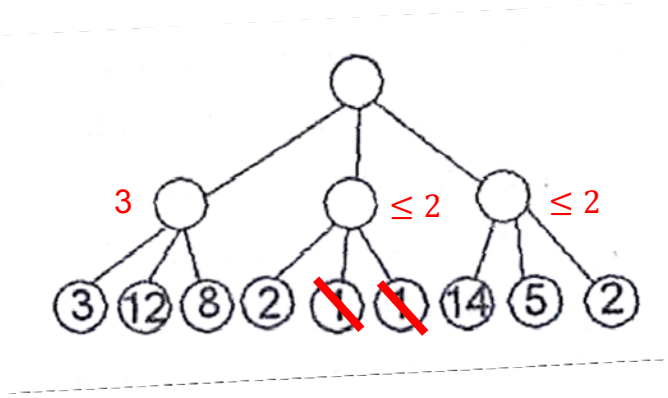
- در صورت استفاده از هرس α-β در درخت زیر چند گره هرس می شوند؟

1. 2
2. 1
3. 0
4. 3



آی تی ۸۹

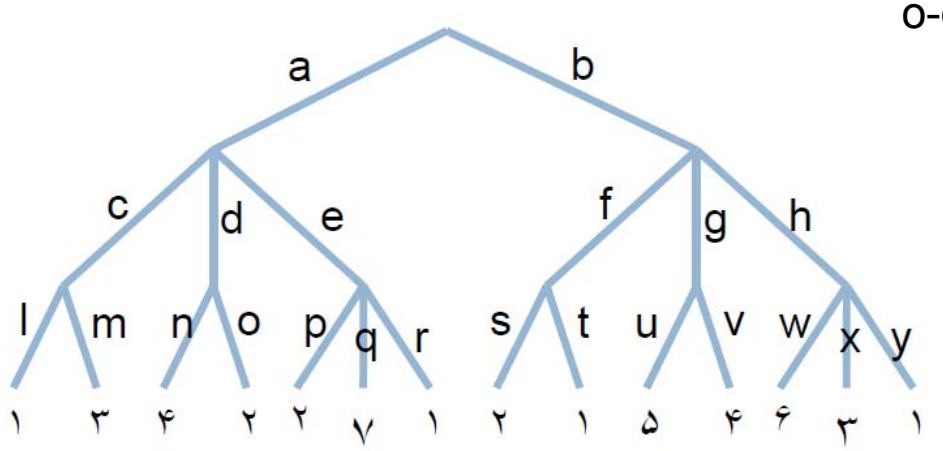
- گزینه ۱.
- در صورتی که ترتیب بازی بازیکنان مشخص نشده باشد، اول Max بازی می‌کند.



مهندسی ۸۶

- در صورت استفاده از هرس α-β در درخت زیر چند گره هرس می شوند؟

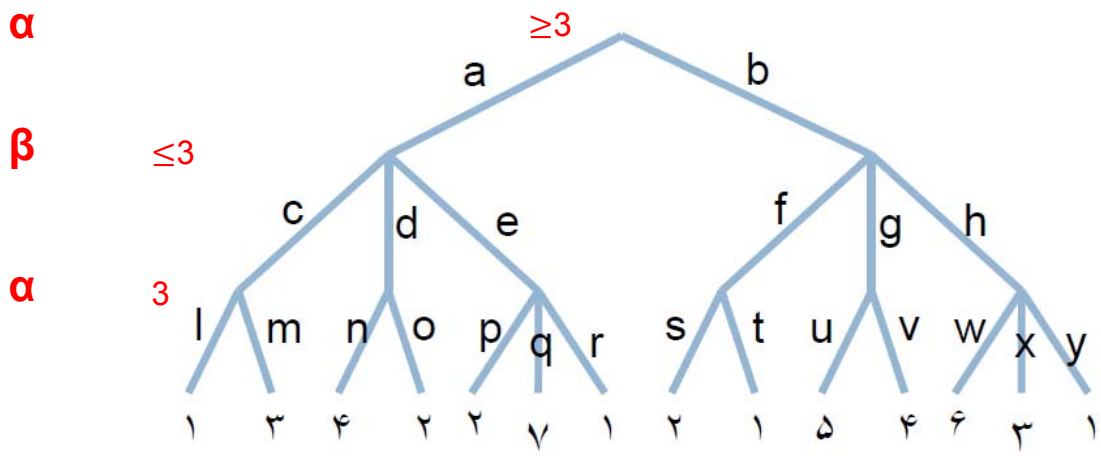
1. e-g-h
2. o-r-v-h
3. o-q-r-v-x-y
4. o-r-g-h



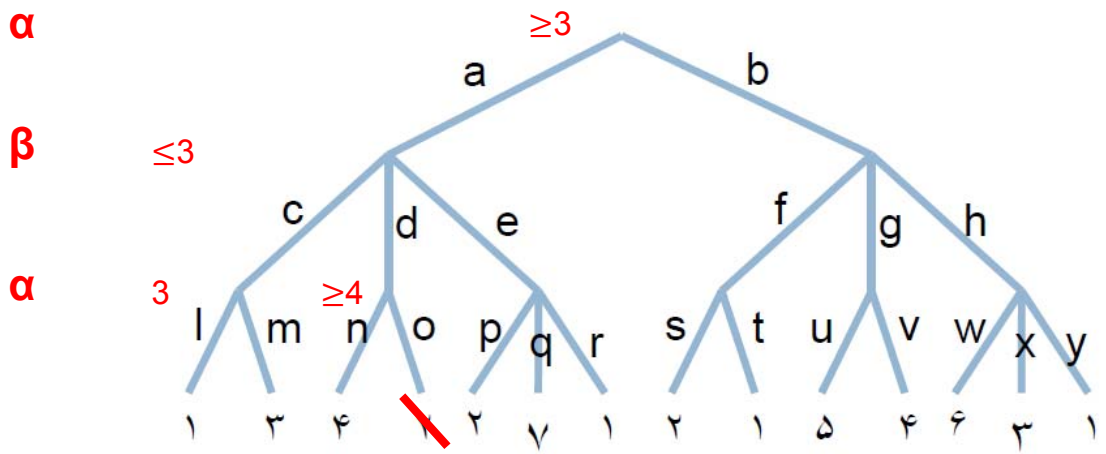
مهندسی ۸۶

- گزینه ۴ پاسخ است.
- روش تستی:
  - α: بهترین انتخاب Max در شاخه (بیشتر بهتر)
  - β: بهترین انتخاب min در شاخه (کمتر بهتر)
  - اگر $\alpha_{\text{والد}} \geq \beta$ یا $\beta \leq \alpha_{\text{والد}}$ هرس کن
- با یادگیری مفهوم احتمال اشتباه کمتر خواهد بود.

مهندسی ۸۶

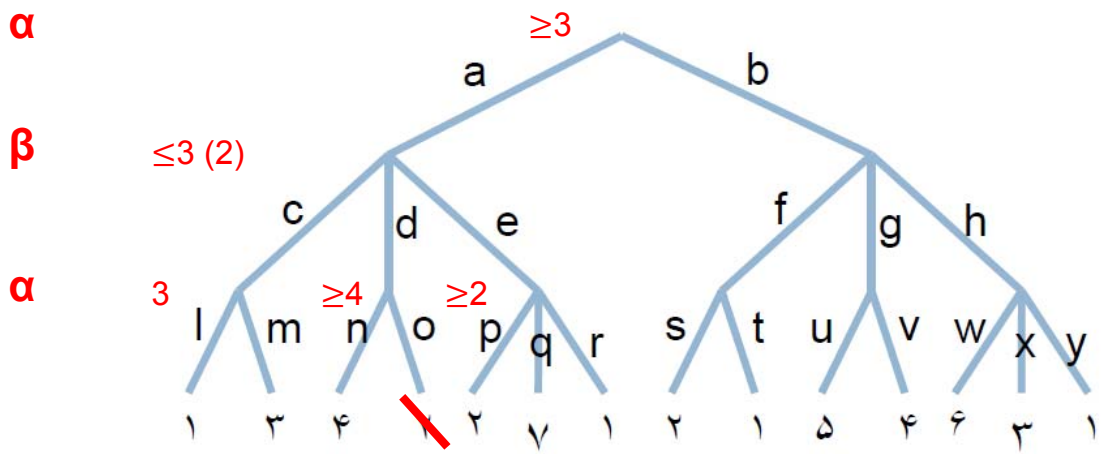


مهندسی ۸۶



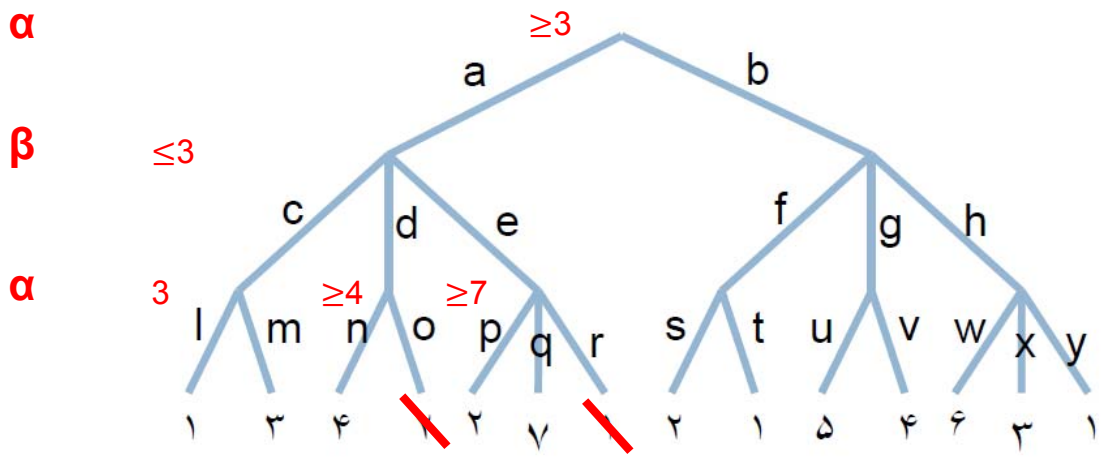
پس هرس صورت می‌گیرد $\alpha \geq \beta_{\text{والد}}$

مهندسی ۸۶



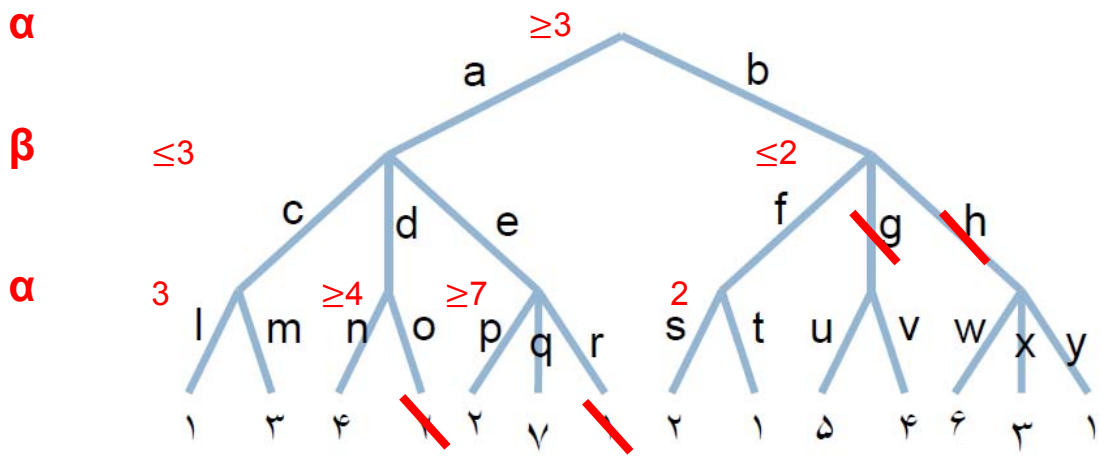
β با مقدار کمتر پیدا شده به روز می‌شود

مهندسی ۸۶



پس هرس صورت می‌گیرد $\alpha \geq \beta$ والد

مهندسی ۸۶



پس هرس صورت می‌گیرد $\beta \leq \alpha$ والد

آی تی ۹۲

- در صورتی که در جستجوی درخت‌های بازی، ترتیب پیمایش گره‌ها از چپ به راست، به راست به چپ تغییر کند، کدام گزاره زیر صحیح است؟

1. مقدار بیشینه برای گره ریشه ثابت می‌ماند، تعداد گره‌های هرس شده توسط آلفا–بتا ثابت می‌ماند.
2. مقدار بیشینه برای گره ریشه ثابت می‌ماند، تعداد گره‌های هرس شده توسط آلفا–بتا ممکن است تغییر نماید.
3. مقدار بیشینه برای گره ریشه ممکن است تغییر نماید، تعداد گره‌های هرس شده توسط آلفا–بتا ممکن است تغییر نماید.
4. مقدار بیشینه برای گره ریشه ممکن است تغییر نماید، تعداد گره‌های هرس شده توسط آلفا–بتا ثابت می‌ماند.

آی تی ۹۲

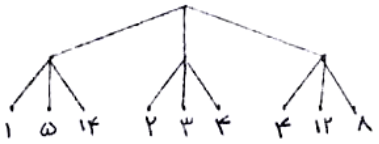
- گزینه ۲ پاسخ است.
  - ترتیب فرزندان فقط در نحوه هرس شدن درخت تاثیر می‌گذارد ولی نتیجه نهایی را تغییر نمی‌دهد.

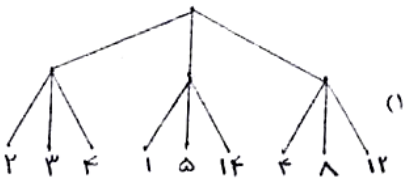
نکته

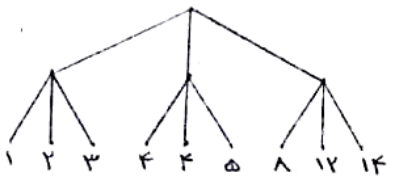
- زمانی که محدودیت امتیاز داشته باشیم، علاوه بر سایر شرایط هرس α-β ممکن است در مرزها نیز هرس اتفاق بیفتد.
- هنگامی که Max می‌خواهد انتخاب کند، اگر حداکثر امتیاز انتخاب شده باشد، سایر شاخه‌ها هرس می‌شوند.
- هنگامی که min می‌خواهد انتخاب کند، اگر حداقل امتیاز انتخاب شده باشد، سایر شاخه‌ها هرس می‌شوند.

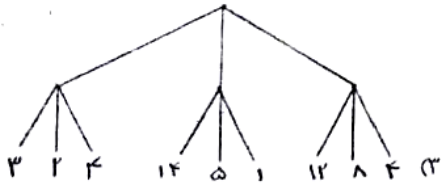
مهندسی ۸۹

در درخت بازی زیر به فرض اینکه امتیازات بتوانند ما بین ۱ تا ۱۵ باشند کدام یک از ترتیب‌دهی‌های زیر باعث می‌شود هرس آلفا ـ بتا هیچ حذفی انجام ندهد؟



۱)


۲)


۳)


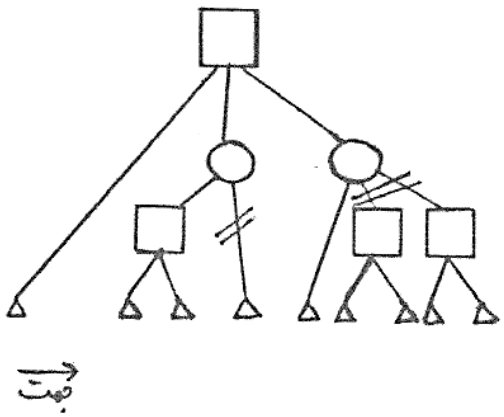
۴) در درخت به همین شکل فعلی نیز شاخه‌ای هرس نخواهد شد

مهندسی ۸۹

- گزینه ۳ پاسخ است.
  - مطابق سوال حداقل امتیاز ۱ است. بنابراین در گزینه‌های ۱ و ۲، min پس از دیدن ۱ در ابتدای شاخه‌ها، ادامه آنرا هرس می‌کند.
  - در گزینه ۳ هیچ هرسی انجام نمی‌شود.

مهندسی ۹۰

در گراف مقابل مربع نشانه بازیکن **Max**، دایره نشانه بازیکن **Min** و مثلث نشانه حالت پایانی است. اگر مقادیر ارزیابی بتوانند در فاصله بسته [۰, ۱۰] باشند و با هرس آلفا ـ بتا فقط یال‌های علامت زده شده با //حذف شوند، ترتیب گره‌های پایانی به ترتیب از چپ به راست در شکل کدام یک از گزینه‌های زیر خواهد بود؟



۱) ۴، ۱، ۰، ۱۰، ۹، ۳، ۵، ۲، ۸ ← جهت

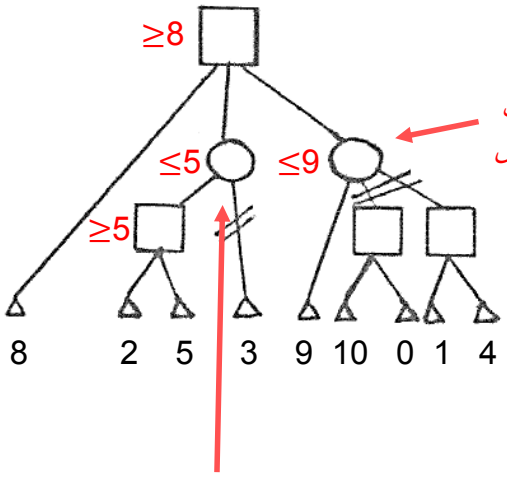
۲) ۰، ۱، ۲، ۳، ۴، ۵، ۸، ۹، ۱۰ ← جهت

۳) ۴، ۲، ۱، ۰، ۸، ۱۰، ۳، ۵، ۹ ← جهت

۴) ۱۰، ۹، ۸، ۹، ۴، ۳، ۲، ۱، ۰ ← جهت

مهندسی ۹۰

- بررسی گزینه ۱:

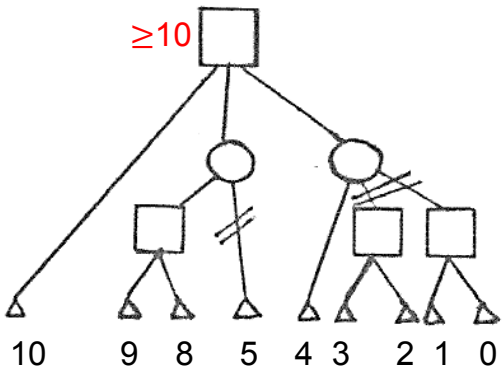


ممکن است Max در این شاخه انتخاب بهتری از ۸ داشته باشد و نمی‌توان هرس را انجام داد.

Max انتخاب بهتری دارد (۸) و هیچگاه به این شاخه نمی‌آید

مهندسی ۹۰

- بررسی گزینه ۲:

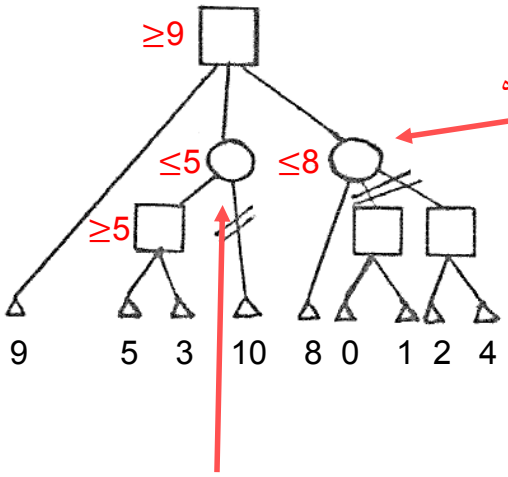


۱۰ حداکثر مقداری است که Max می‌تواند به آن برسد (مطابق صورت سوال). با اعمال آن به Max در این درخت، عملا سایر درخت (به جز برگ‌های اول هر والد) هرس خواهد شد.

مهندسی ۹۰

- بررسی گزینه ۳:

گزینه صحیح

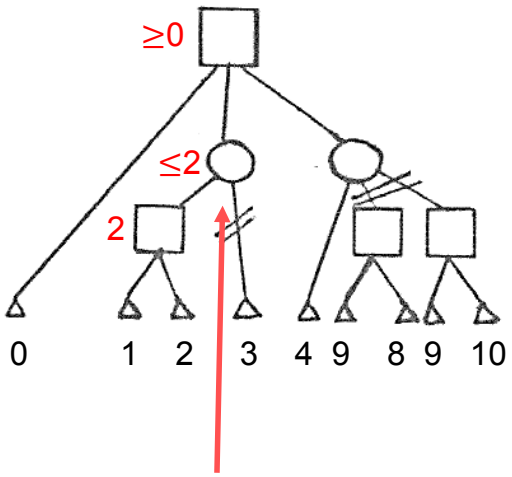


Max انتخاب بهتری دارد (۹) و هیچگاه به این شاخه نمی‌آید.

Max انتخاب بهتری دارد (۹) و هیچگاه به این شاخه نمی‌آید.

مهندسی ۹۰

- بررسی گزینه ۴:

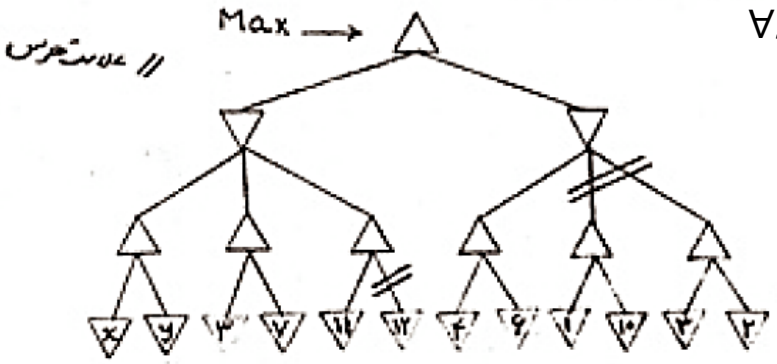


min ممکن است در این شاخه انتخاب کوچکتری از ۲ داشته باشد و با توجه به بهترین انتخاب Max که ۰ است، نمی‌توان این شاخه را هرس کرد.

مهندسی ۹۱

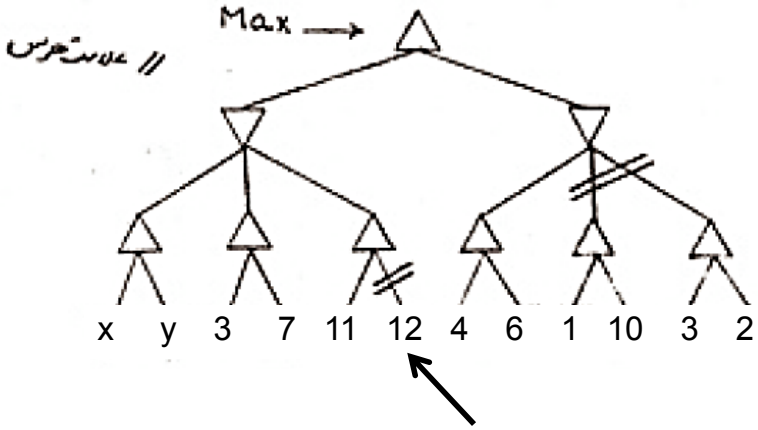
- در درخت بازی زیر کدام یک از گزینه‌ها محدوده مناسبی برای مقادیر x و y تعیین می‌کند به طوریکه شاخه‌های علامت زده شده در هرس آلفا-بتا، هرس شوند؟

1. به ازای هیچ مقداری از x و y شاخه‌های علامت زده شده حذف نخواهند شد.
2. $\forall x,y \quad 0 \le x \le 3, 0 \le y \le 3$
3. $\forall x,y \quad 6 - x - y > 0$
4. $\forall x,y \quad x < 5, y > 7$



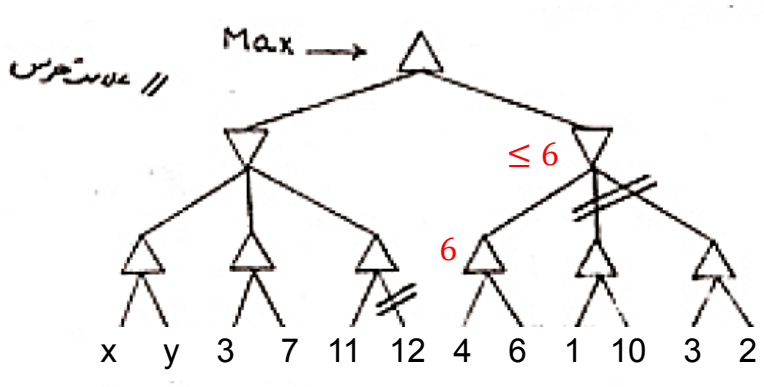
مهندسی ۹۱

- گزینه ۴ صحیح است.



برای اینکه این شاخه هرس شود، min باید انتخاب‌های کمتر از ۱۱ در شاخه‌های دیگر داشته باشد (که دارد). از طرفی اگر حداقل یکی از x یا y از ۳ کوچکتر باشند، به دلیل وجود انتخاب کوچکتر از ۳ در شاخه اول، شاخه ۷ هرس خواهد شد. بنابراین گزینه ۲ غلط است.

$$x||y \ge 3$$



برای آنکه شاخه‌های سمت راست حذف شوند، با توجه به وجود βی ۶ در این شاخه، Max باید انتخاب بهتری از ۶ در شاخه سمت چپ داشته باشد. یعنی β در شاخه سمت راست باید از ۶ بزرگتر باشد. در صورتی که هر دوی x و y از ۷ کوچکتر باشند (max(x,y)<7)، min این عدد را انتخاب کرده، شاخه‌های سمت راست هرس نخواهند شد. بنابراین یکی از x یا y باید بزرگتر از ۷ باشد. تنها گزینه ۴ این مورد را تظمین می‌کند.

ترتیب دهی (Ordering)

- تأثیر هرس α-β به ترتیب بررسی فرزندان (مابعدها) بستگی دارد، بنابراین اگر ابتدا فرزندی که بهتر است بررسی شود نتیجه مطلوب تر است.
  - به این حالت مرتب کردن (Ordering) گفته می‌شود.
  - تعداد گره‌هایی که باید بررسی شوند به $O\left(b^{m/2}\right)$ تقلیل می‌یابد.
  - فاکتور انشعاب مؤثر به جای b برابر با جذر b خواهد بود.
- اگر آخرین سطح را Max بازی می‌کند مرتب سازی نزولی و اگر min بازی می‌کند صعودی.
- همچنین برای جلوگیری از تکرار نودها می‌توان آنها را در یک جدول هش ذخیره کرد.

آی تی ۸۸

- در یک درخت بازی اگر ترتیب ملاقات گره‌ها عوض شود:

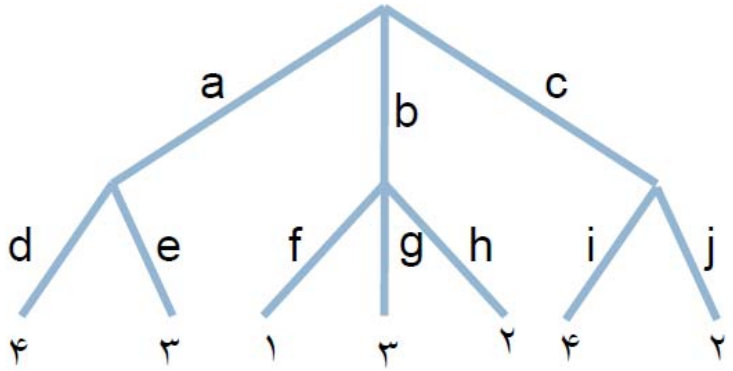
1. هیچ تغییری رخ نخواهد داد
2. احتمال یافتن جواب بهینه تغییر خواهد کرد
3. شاخه‌هایی که با هرس α-β حذف می‌شوند تغییر خواهد کرد.
4. مسیری که با استفاده از الگوریتم min-max انتخاب می‌شود تغییر خواهد کرد

آی تی ۸۸

- پاسخ گزینه ۳ است

مهندسی ۸۵

- در درخت بازی زیر اگر از هرس α-β با ترتیب دهی استفاده شود، چه شاخه هایی حذف خواهند شد؟

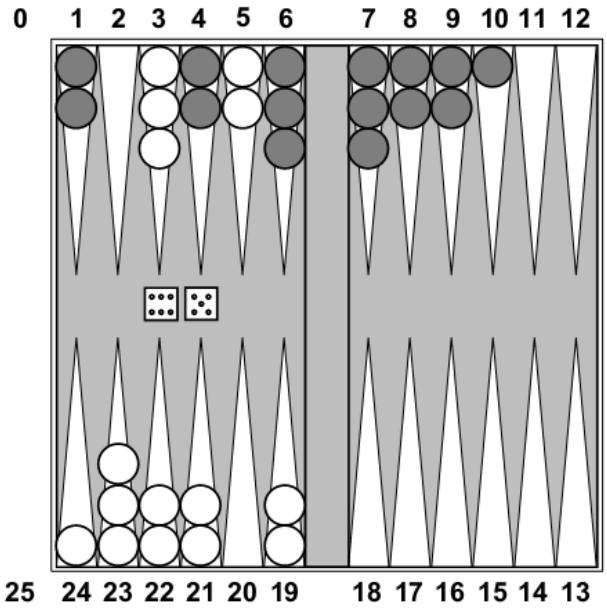


1. g-h
2. b-c
3. g-h-j
4. e-g-h-j

مهندسی ۸۵

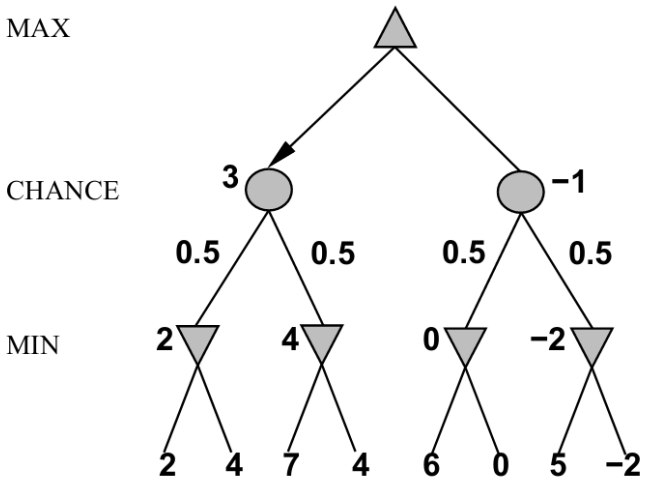
- گزینه ۳ صحیح است.
  - در ترتیب دهی، یال‌ها بدون جابجا کردن حروف آنها مرتب می‌شوند.
  - با توجه به اینکه در سطح آخر min بازی می‌کند، برگ‌های هر والد به ترتیب صعودی مرتب می‌شوند.

بازی‌هایی که حاوی عنصر شانس هستند



- تخته نرد یک بازی عمومی است که شانس و مهارت را با هم ترکیب می‌کند.
- مثال: مقادیر تاس برای بازیکن سفید ۵ و ۶ است. بنابراین ۴ حرکت ممکن خواهد داشت.
- اگرچه بازیکن سفید از اعمال مجاز خود باخبر است اما از مقادیر حاصل از پرتاب تاس توسط رقیب و در نتیجه اعمال قانونی او اطلاعی ندارد.

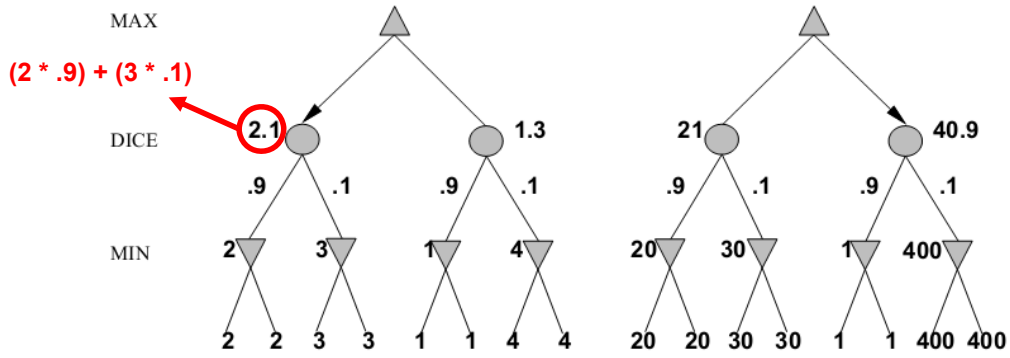
ارزیابی موقعیت در بازی‌های حاوی عنصر شانس



- تخته نرد نمی‌تواند درخت کاملی از نوع درخت بازی دوز و شطرنج داشته باشد.
- درخت این بازی باید علاوه بر گره‌های Max و min شامل گره‌های شانس نیز باشد.
- دوایری که در شکل آمده‌اند گره‌های شانس هستند.
- شاخه‌هایی که از هر گره شانس خارج شده‌اند، مقادیر ممکن تاس‌ها را مشخص نموده و هرکدام با شانسی که دارند برچسب خورده‌اند

ارزیابی موقعیت در بازی‌های حاوی عنصر شانس

- می‌خواهیم حرکتی را انتخاب کنیم که به بهترین موقعیت منجر شود.
  - چون در اینجا مقدار Minimax قطعی وجود ندارد، فقط می‌توانیم میانگین یا مقدار مورد انتظار را محاسبه کنیم.
  - این مقدار، کلیه ترکیبات ممکن تاس‌ها را در نظر می‌گیرد.
  - پس مقدار Minimax در بازی‌های قطعی را به مقدار مورد انتظار در بازی‌های شامل عنصر شانس عمومیت می‌دهیم.
- در این مورد باید در مورد مقادیر ارزیابی دقیقتر عمل نمود



مهندسی ۸۴

- بیشترین تاثیر افزودن شانس (مثل ریختن تاس) در بازی‌ها، بر روی درخت جستجوی تولید شده چیست؟

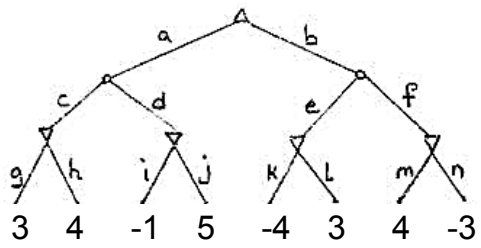
1. هرس کردن شاخه‌ها مشکل‌تر می‌شود.
2. روش‌هایی مثل minimax و آلفا–بتا نمی‌توانند با عنصر شانس کار کنند.
3. برای محاسبه تابع ارزیابی باید لبه‌های مرزی که بازتاب عنصر شانس هستند را اعمال نمود.
4. برای هر حرکت بازیکن، سطح دیگری از گره‌ها تولید می‌شوند که احتمالات معرفی شده بوسیله عنصر شانس را در بر می‌گیرند.

مهندسی ۸۴

- گزینه ۴ صحیح است.

مهندسی ۸۸

فرض کنید در یک بازی، هر بازیکن ابتدا یک سکّه سالم پرتاب می‌کند. اگر شیر آمد شاخه سمت چپ و اگر خط آمد شاخه سمت راست درخت بازی مقابل را انجام می‌دهد. در درخت بازی داده شده به فرض اینکه امتیاز بازیکن‌ها در بازه [−5,5] است، اگر هرس آلفا ـ بتا اجرا شود، کدام شاخه‌ها حذف خواهند شد؟



۱) l − f

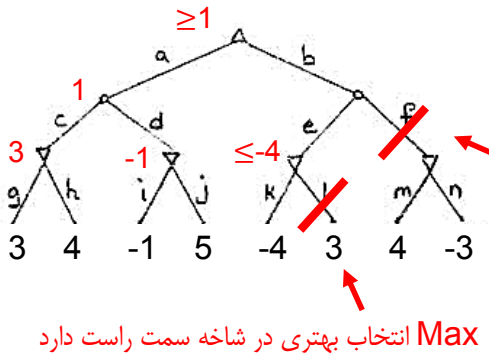
۲) l − n

۳) j − l − n

۴) هیچ هرسی رخ نخواهد داد.

مهندسی ۸۸

- گزینه ۱ صحیح است.



با توجه به وجود ۴-، حتی اگر در شاخه f به حداکثر امتیاز (۵) برسیم، این شاخه ما را به مقدار (-4*.5)+(5*.5)=.5 خواهد رساند که از مقداری که Max در a می‌تواند با آن برسد کمتر است. بنابراین با توجه به بازه مشخص شده برای امتیازات، می‌توان این شاخه را نیز حذف کرد.

تصمیمات بلادرنگ (Realtime Decisions)

- Minimax باید کل فضای جستجوی بازی را در اختیار داشته باشد تا بتواند تصمیم گیری کند.
- الگوریتم α-β با وجود هرس درخت، باز هم باید تا رسیدن به حالات پایانی درخت را بسط دهد، در نتیجه قسمت وسیعی از فضای حالت باید جستجو شود.
- بررسی این عمق از درخت عملی نیست، زیرا حرکات باید در زمانی معقول انجام شوند (شانون ۱۹۵۰).
  - نمی‌توان منتظر بسط کل درخت ماند!
- عمق درخت بازی باید محدود شود و بجای تابع سودمندی در پایان بازی، از تابع ارزیابی (Eval) برای نودهای برگ درخت محدود شده (که پایانی نیستند) استفاده شود.

بازی‌های قطعی با اطلاعات ناقص

- اعمال تغییرات زیر در الگوریتم Minimax و هرس آلفا – بتا
  - جایگزینی تست پایانه با تست برش (Cutting Test)
    - تصمیم می‌گیرد Eval چه موقع اعمال شود.
  - جایگزینی تابع سودمندی (Payoff) با تابع ارزیابی (Eval)
    - تابع ارزیابی تخمینی از سودمندی یک موقعیت خاص را ارائه می‌کند.
      - توابع اکتشافی، تخمینی از فاصله تا هدف را بر می‌گرداندند
    - باید مقادیری که تابع ارزیابی به حالات پایانی نسبت می‌دهد با مقادیری که تابع سودمندی به آنها نسبت می‌دهد متناسب باشد.
    - در نودهای غیر پایانی، این تابع باید به درستی شانس‌های واقعی برد را منعکس کند. (لازم نیست بازی شانسی باشد، همین که هنوز به پایان نرسیده‌ایم عدم قطعیت وجود دارد.)

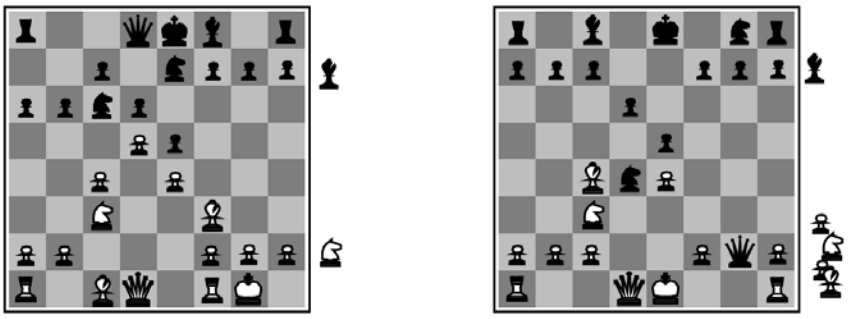
تابع ارزیابی اکتشافی Eval

- اغلب توابع ارزیابی، خواص گوناگونی از حالت‌ها را محاسبه می‌کنند و برای یک خصوصیت در یک حالت، ترکیبات عددی متفاوتی بر می‌گردانند.
  - مثال: در بازی شطرنج، ارزش سرباز ۱، اسب یا فیل ۳، رخ ۵ و وزیر ۹ است. موقعیت قرار داشتن آنها در صفحه نیز امتیازهای خاص خود را دارد. این مقادیر با یکدیگر جمع شده و مقدار ارزیابی (Eval) یک حالت را مشخص می‌کند.

تابع ارزیابی اکتشافی Eval

- در بازی شطرنج:
  - w_i تعداد از هر نوع مهره در صفحه
  - f_i ارزش آن مهره (۱ برای پیاده، ۳ برای اسب و فیل ۵ برای رخ و...)

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

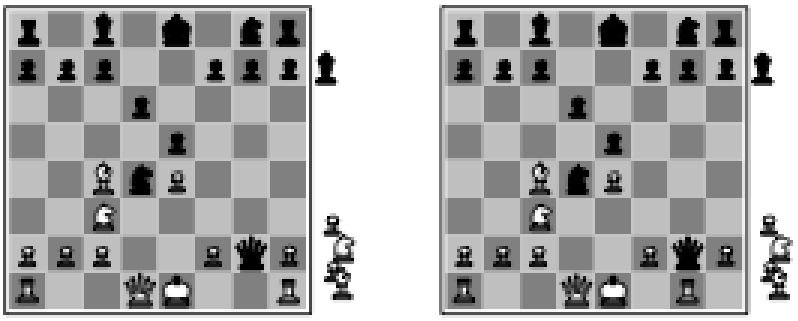


حالت ساکن

- حالت ساکن حالتی است که از آن تا حرکت بعدی، تابع ارزیابی تفاوت خیلی زیادی نکند (تسخیرهای [Capture] خیلی سودمند انجام نشود).
  - مثال: موقعیتی در بازی شطرنج که یک حرکت منجر به حذف وزیر می‌شود، یک حالت ساکن نمی‌باشد
- تابع ارزیابی باید فقط برای حالاتی به کار برده شود که ساکن هستند.
  - موقعیت‌های غیرساکن باید بسط داده شوند تا به موقعیت ساکن برسیم.
  - این جستجو، جستجوی ساکن (Quiescent Search) نام دارد.

مثال: تابع EVAL

- ارزیابی تابع EVAL از مقدار پیروزی در دو موقعیت کاملا متفاوت

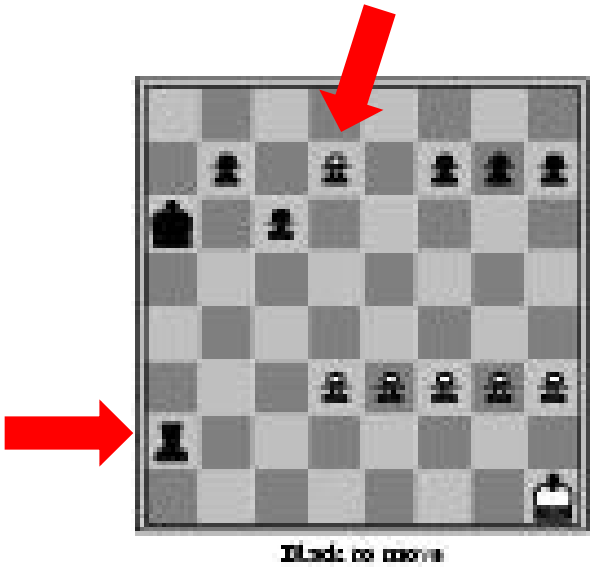


ب) سفید حرکت میکند　　الف) سفید حرکت میکند

الف) سیاه، مزیت اسب و دو پیاده دارد و بازی را می‌برد
ب) پس از اینکه سفید، وزیر را در اختیار می‌گیرد، سیاه می‌بازد

اثر افق

- وقتی بوجود می آید که برنامه با اثری از رقیب مواجه شود که منجر به خرابی جدی گشته و اجتناب پذیر است



- مثال: شکل مقابل؛
- سیاه در اصل جلوست، اما اگر سفید پیاده اش را از سطر هفتم به هشتم ببرد، پیاده به وزیر تبدیل میشود و موقعیت برد برای سفید بوجود می آید