هوش مصنوعی

فصل پنجم

جستجوی خصمانه

فهرست

- بازیها چیستند و چرا مطالعه میشوند؟
- انواع بازیها
- الگوریتم minimax
- بازیهای چند نفره
- هرس آلفا-بتا
- بازیهای قطعی با اطلاعات ناقص
- بازیهایی که حاوی عنصر شانس هستند

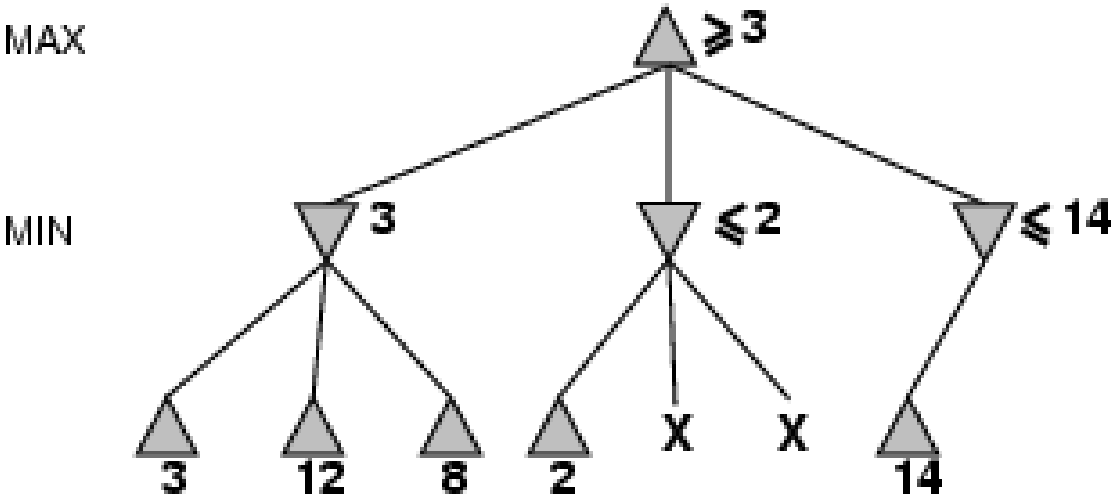
جستجوی خصمانه

بازی ها چیستند و چرا مطالعه میشوند؟

- بازیها حالتی از محیطهای چند عاملی هستند
  - هر عامل نیاز به در نظر گرفتن سایر عاملها و چگونگی تأثیر آنها دارد
  - تمایز بین محیطهای چند عامل رقابتی و همکار
  - محیطهای رقابتی، که در آنها اهداف عاملها با یکدیگر برخورد دارند، منجر به مسئله های خصمانه میشود که به عنوان بازی شناخته میشوند
- چرا مطالعه میشوند؟
  - قابلیتهای هوشمندی انسانها را به کار میگیرند
  - ماهیت انتزاعی بازی ها
  - 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 شده اند

جستجوی خصمانه

یک نمونه بازی



جستجوی خصمانه

انواع بازی ها

	قطعی	تصادفی
اطلاعات کامل	شطرنج ریورسی	تخته نرد
اطلاعات ناقص		پوکر

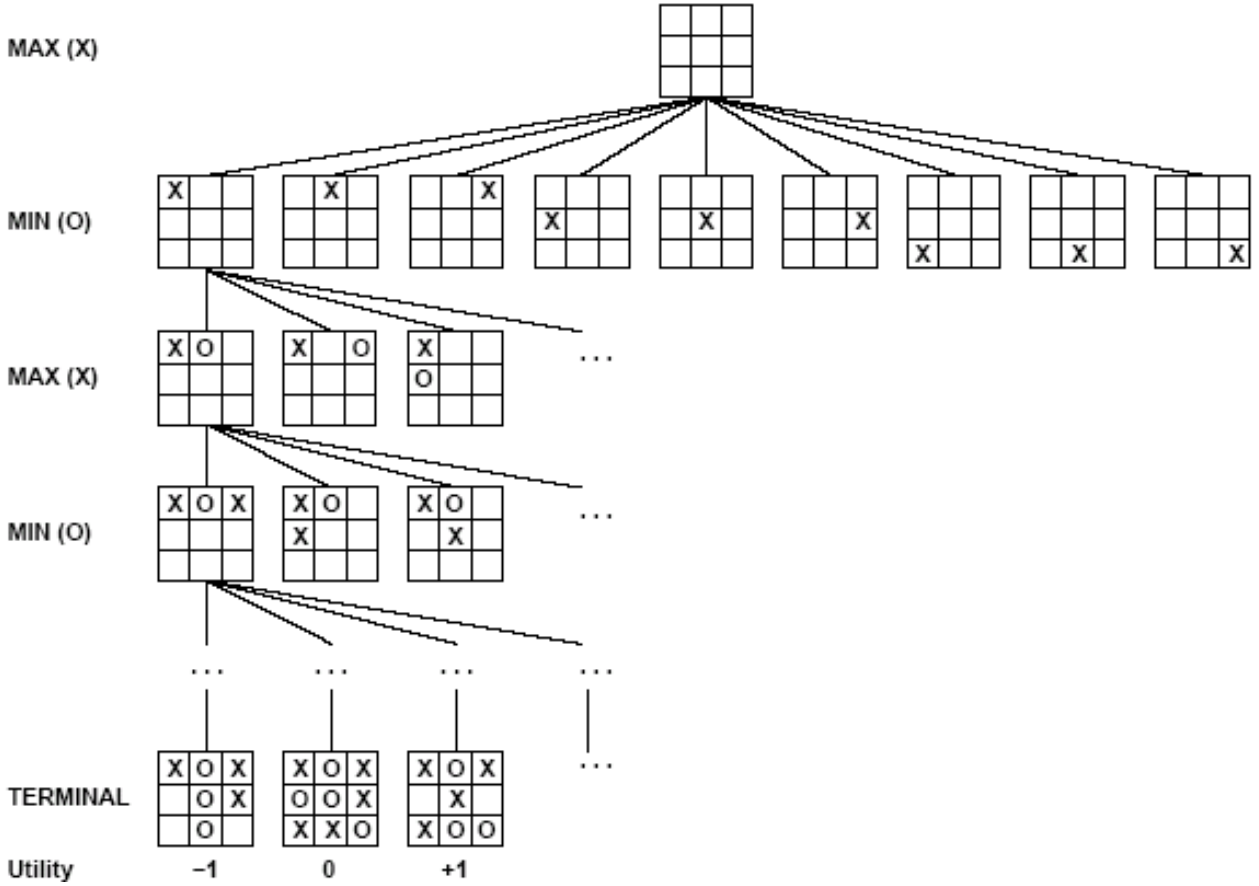
جستجوی خصمانه

یک نمونه بازی

- بازی دو نفره: Min و Max
  - اول Max حرکت میکند و سپس به نوبت بازی میکنند تا بازی تمام شود
  - در پایان بازی، برنده جایزه و بازنده جریمه میشود
- بازی به عنوان یک جستجو:
  - حالت اولیه: موقعیت صفحه و شناسه های قابل حرکت
  - تابع جانشین:لیستی از (حالت،حرکت) که معرف یک حرکت معتبر است
  - آزمون هدف:پایان بازی چه موقع است؟(حالتهای پایانه)
  - تابع سودمندی: برای هر حالت پایانه یک مقدار عددی را ارائه میکند. مثلا برنده(۱+) و بازنده(۱-)
- حالت اولیه و حرکات معتبر برای هر بازیکن، **درخت بازی** را برای آن بازی ایجاد میکند

جستجوی خصمانه

یک نمونه بازی



الگوریتم:

- **بازیکن:** انتخاب **بهترین حالت**
- **حریف:** انتخاب بهتـرین موقعیت برای خودش یا **بـدترین وضـعیت** بـرای بازیکن

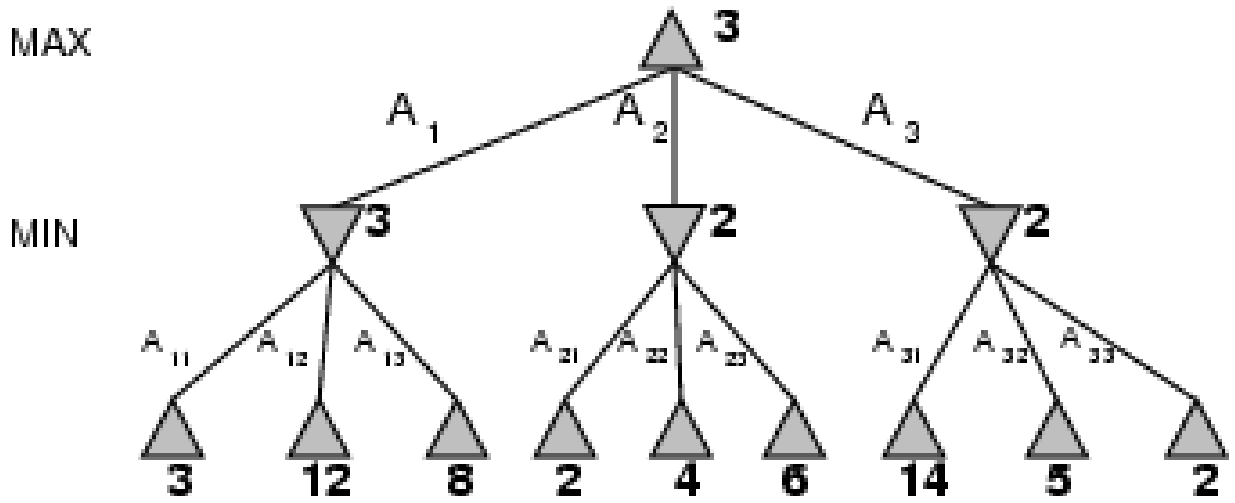
بازیکن: ماکزیمم حالت

حریف: مینیمم حالت

$$\text{MiniMax-value(n)} = \begin{cases} \text{UTILITY}(n) & \text{if } n \text{ is a terminal state} \\ \max_{s \in Successors(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MAX node} \\ \min_{s \in Successors(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MIN node.} \end{cases}$$

جستجوی خصمانه

الگوریتم minimax



```
function MINIMAX-DECISION(state) returns an action
  inputs: state, current state in game

  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
```

جستجوی خصمانه

الگوریتم minimax

کامل بودن: بله (اگر درخت محدود باشد)

بهینگی: بله

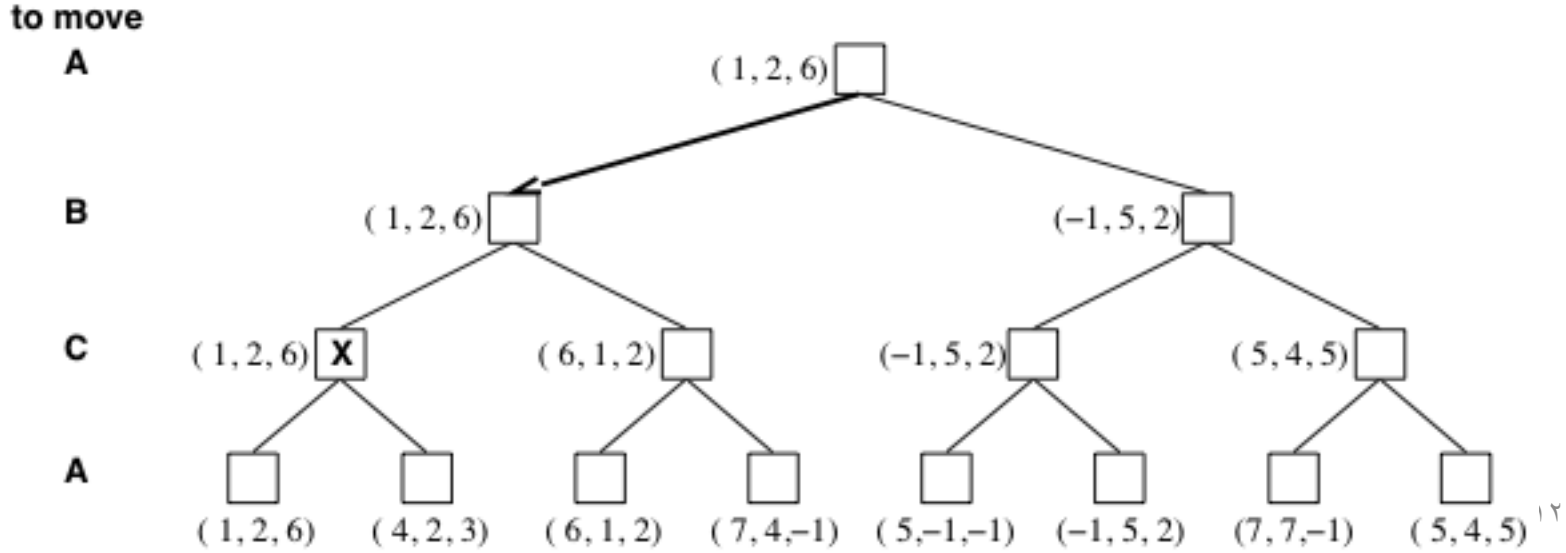
پیچیدگی زمانی: $O(b^m)$

پیچیدگی فضا: $O(bm)$

جستجوی خصمانه

بازیهای چند نفره

- تخصیص یک **بردار** به هر گره، به جای یک مقدار
- بازیهای چند نفره معمولاً شامل **اتحاد** رسمی یا غیر رسمی بین بازیکنان است
  - اتحاد با پیشروی بازی ایجاد و از بین میرود
  - بازیکنان بطور خودکار همکاری میکنند، تا به هدف مطلوب انحصاری برسند



جستجوی خصمانه

هرس آلفا-بتا

در الگوریتم MaxMin:

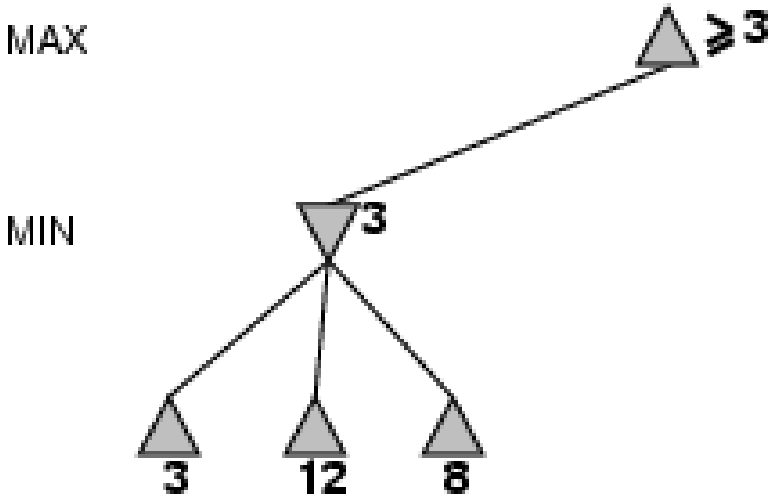
- تعداد حالتهای بازی که باید بررسی شوند، بر حسب تعداد حرکتها، توانی است
  - راه حل: محاسبه تصمیم الگوریتم، بدون دیدن همه گره ها امکانپذیر است

هرس آلفا-بتا:

- انشعابهایی که در تصمیم نهایی تأثیر ندارند را حذف میکند
  - آلفا: مقدار بهترین انتخاب در هر نقطه انتخاب در مسیر Max تاکنون
  - بتا: مقدار بهترین انتخاب در هر نقطه انتخاب در مسیر Min تاکنون
  - تعداد گره هایی که باید بررسی شوند به $O(b^{d/2})$ تقلیل میابد
  - فاکتور انشعاب مؤثر به جای b برابر با جذرb خواهد بود
  - پیش بینی آن نسبت به minimax دو برابر است

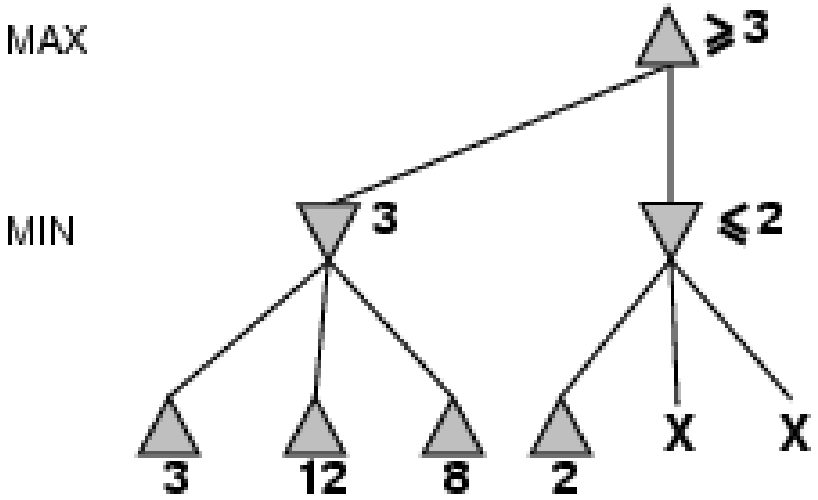
جستجوی خصمانه

یک نمونه بازی



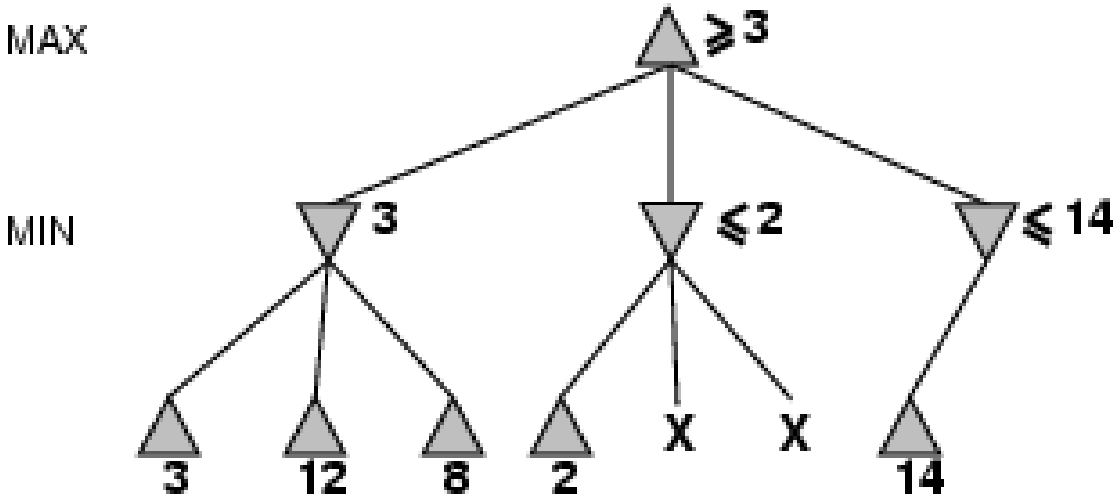
جستجوی خصمانه

یک نمونه بازی



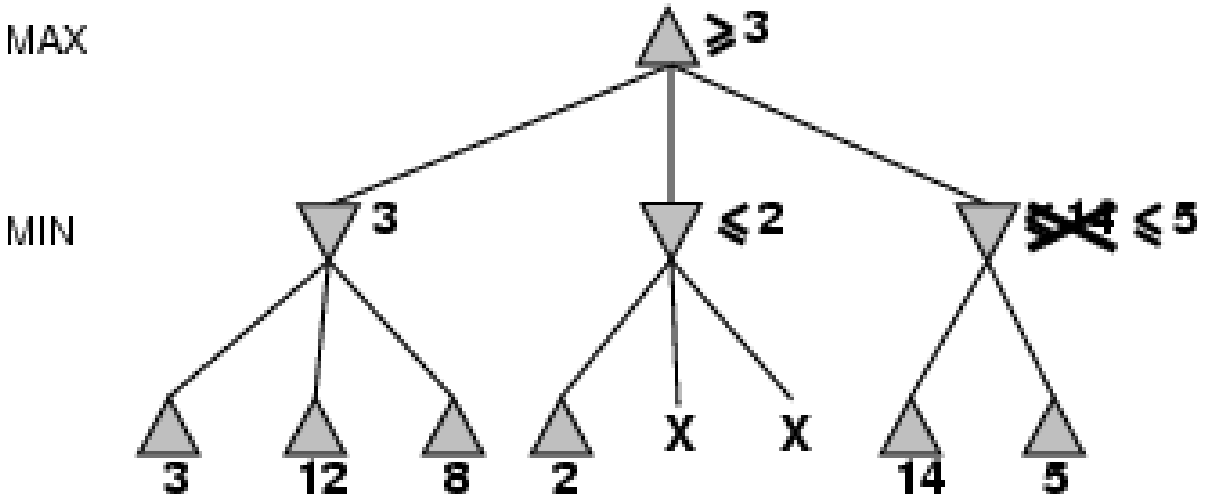
جستجوی خصمانه

یک نمونه بازی



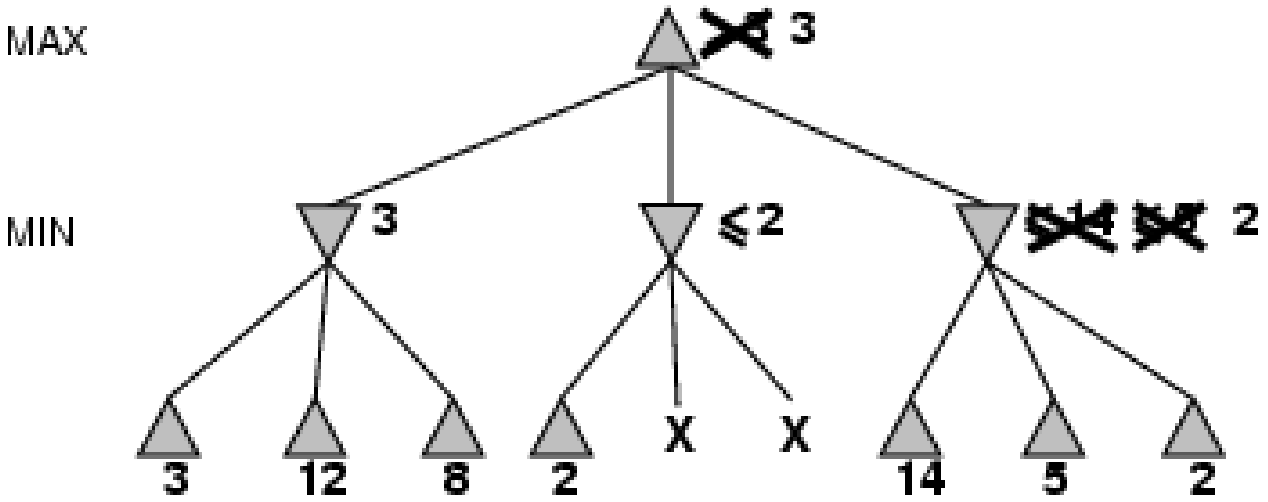
جستجوی خصمانه

یک نمونه بازی



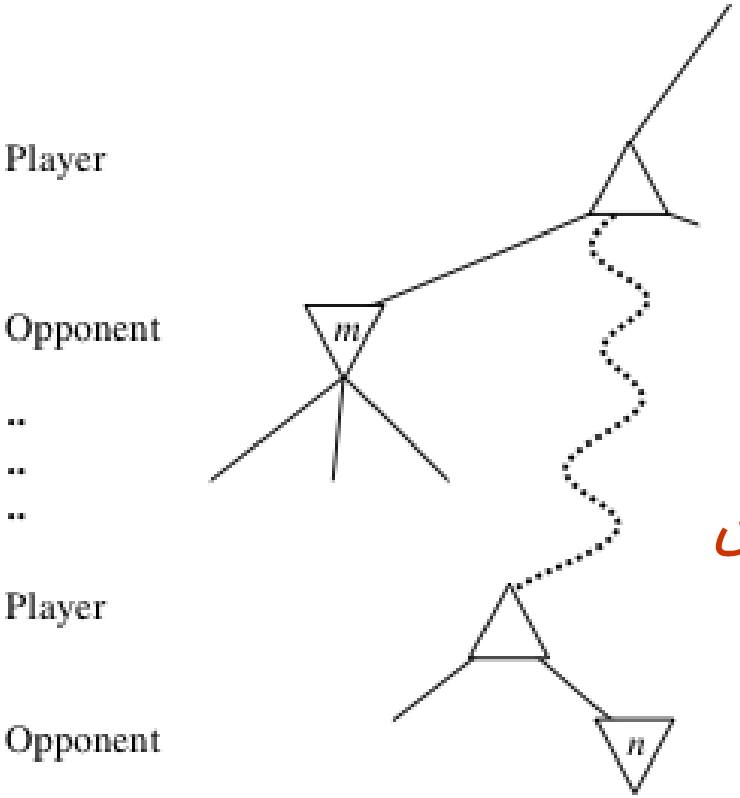
جستجوی خصمانه

یک نمونه بازی



جستجوی خصمانه

هرس آلفا-بتا



- گره n که هر جای درخت میتواند باشد، بررسی میشود
- اگر بازیکن انتخاب بهتری داشته باشد
  - در گره والد n
  - یا هر انتخاب بهتری تا کنون
- n هیچوقت در بازی واقعی قابل دسترس نخواهد بود
- در نتیجه n هرس میشود

- α : مقدار بهترین(بزرگترین مقدار) انتخاب پیدا شده تاکنون، درهر نقطه انتخاب درطول مسیر برایMAX.

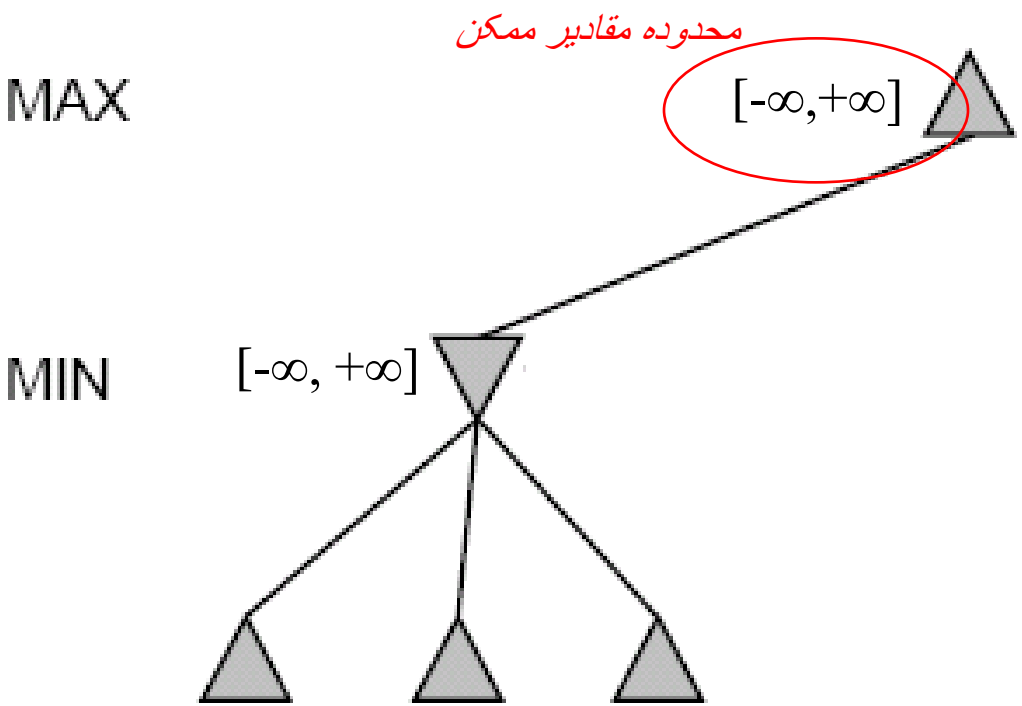
- β: مقدار بهترین(کوچکترین مقدار) انتخاب پیدا شده تاکنون، درهر نقطه انتخاب درطول مسیر برایMIN.

- قانون اول: چنانچه مقدار آلفای یک گره از بتای پدرش یا پدرانش بیشتر شد نیازی به بررسی سایر فرزندان آن گره نیست. آن گره را هرس کنید.

- قانون دوم: چنانچه بتای یک گره از آلفای پدرش یا پدرانش کمتر شد نیازی به بررسی آن گره نیست. آن گره را هرس کنید.

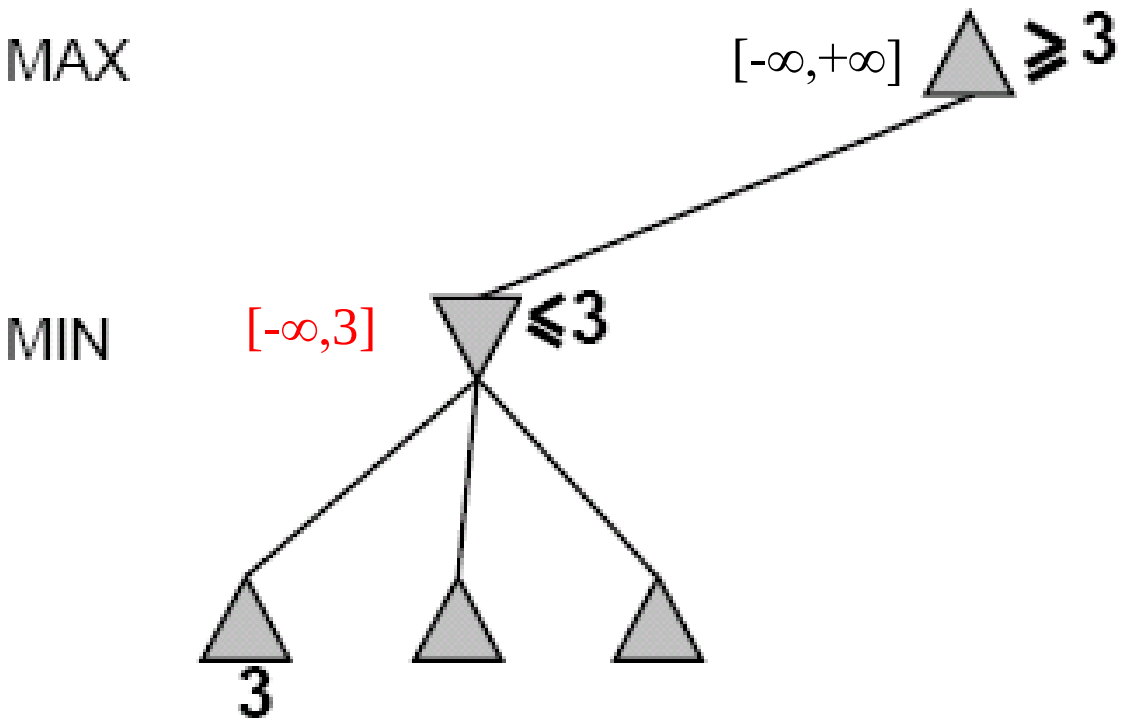
جستجوی خصمانه

مثال: هرس آلفا-بتا



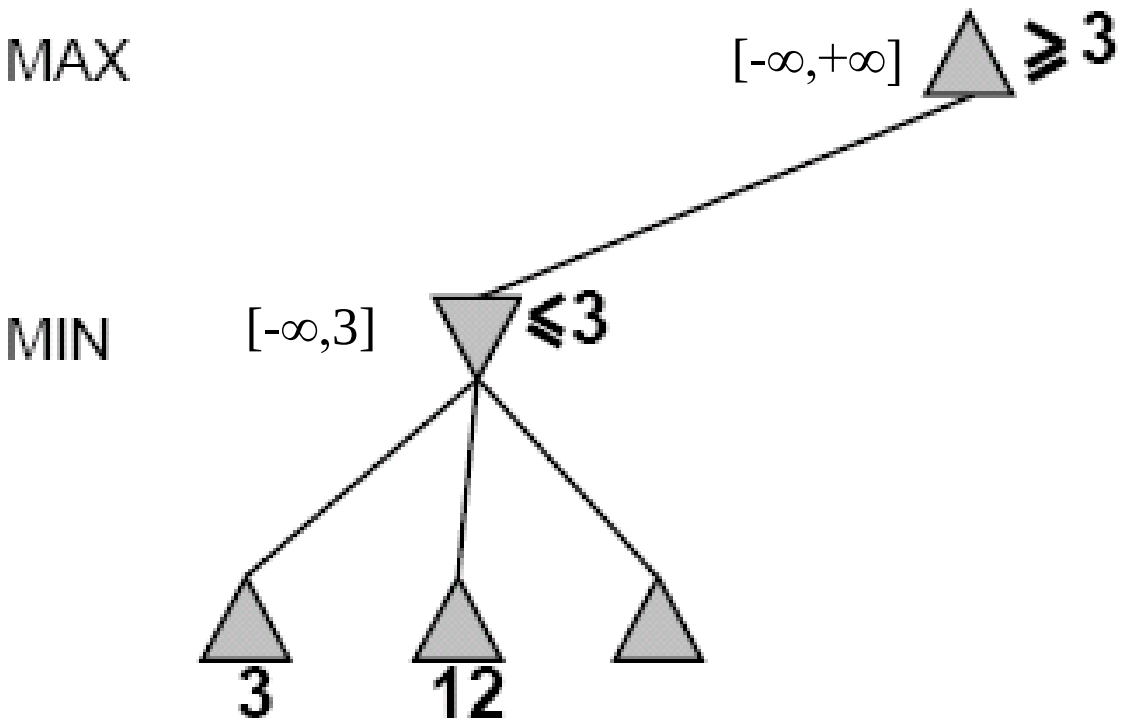
جستجوی خصمانه

مثال: هرس آلفا-بتا

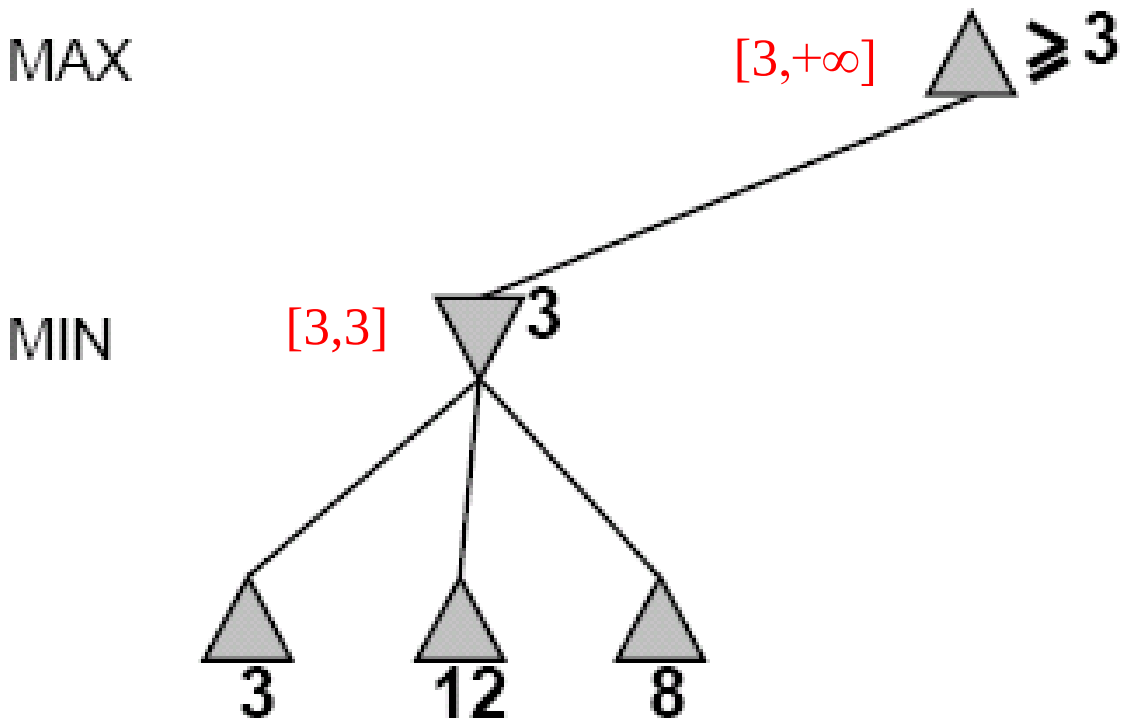


جستجوی خصمانه

مثال: هرس آلفا-بتا

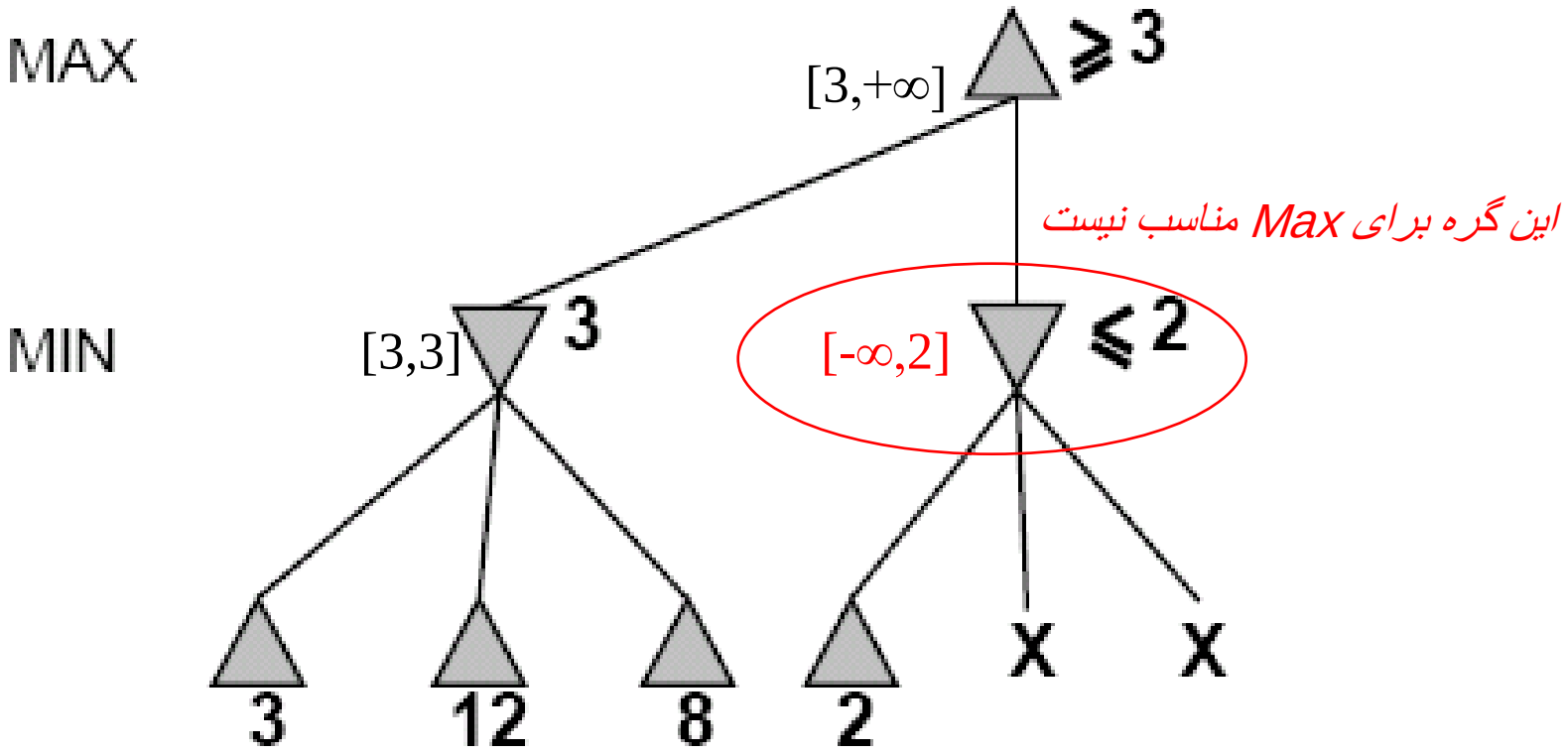


جستجوی خصمانه



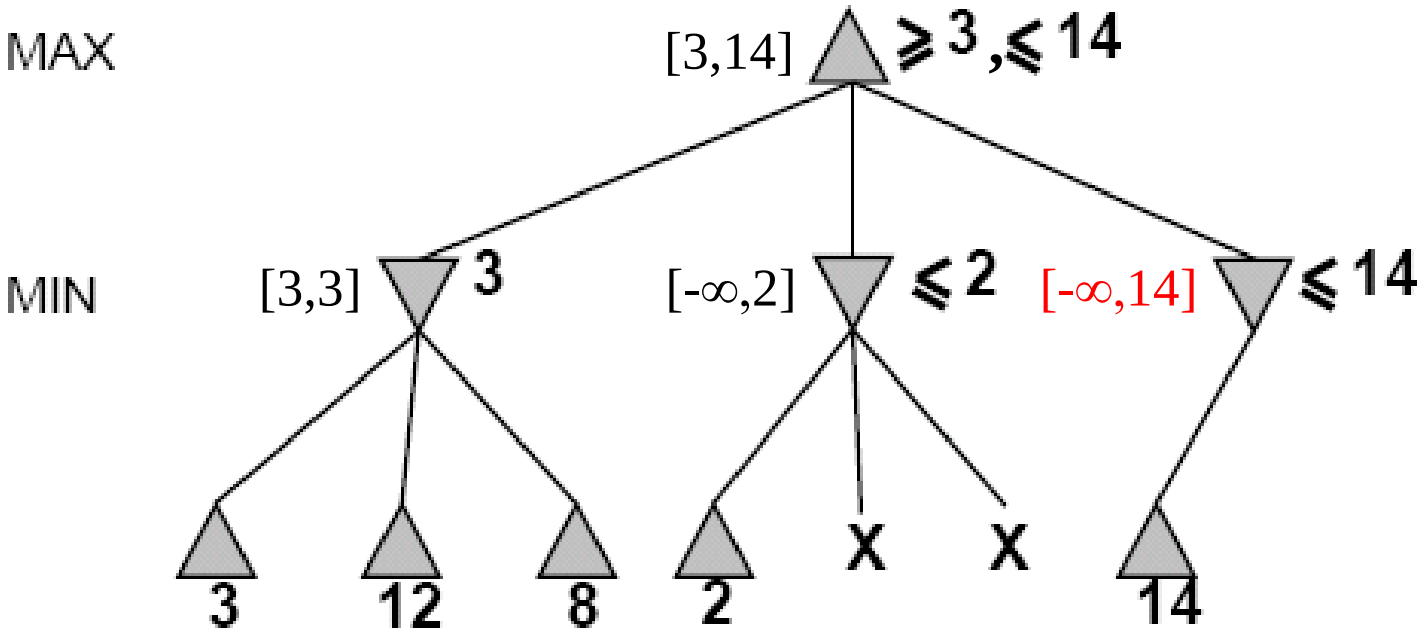
جستجوی خصمانه

مثال: هرس آلفا-بتا



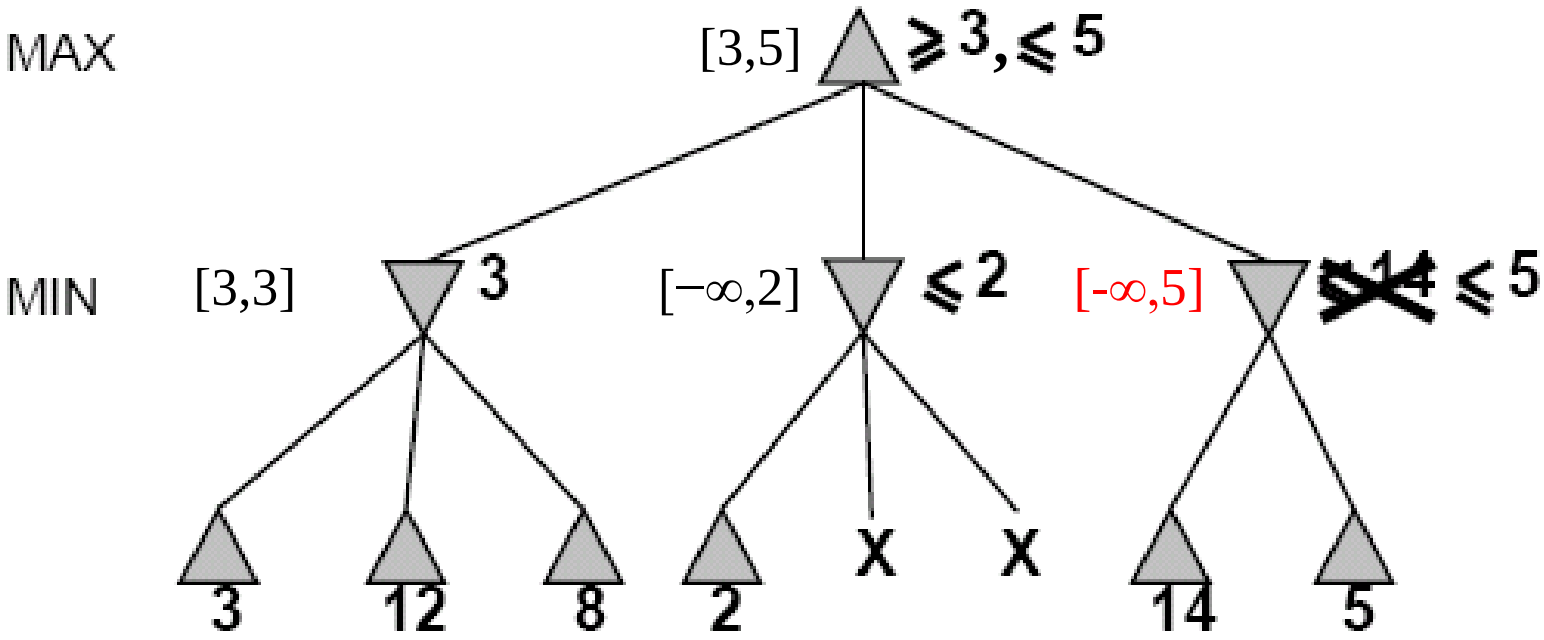
جستجوی خصمانه

مثال: هرس آلفا-بتا



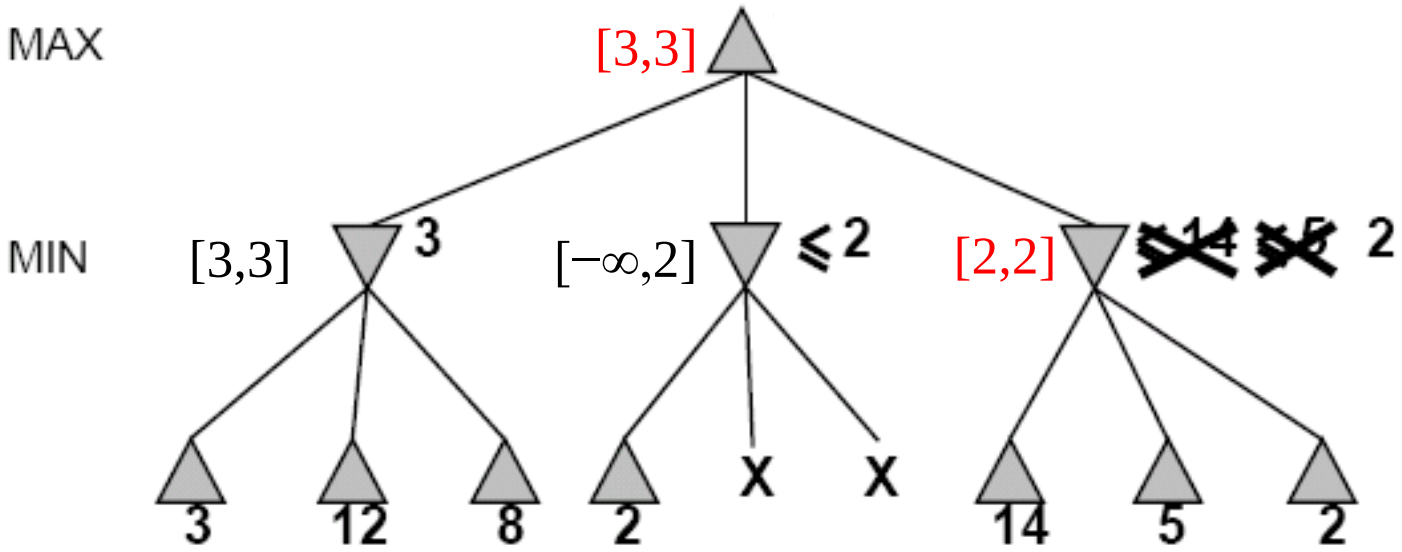
جستجوی خصمانه

مثال: هرس آلفا-بتا



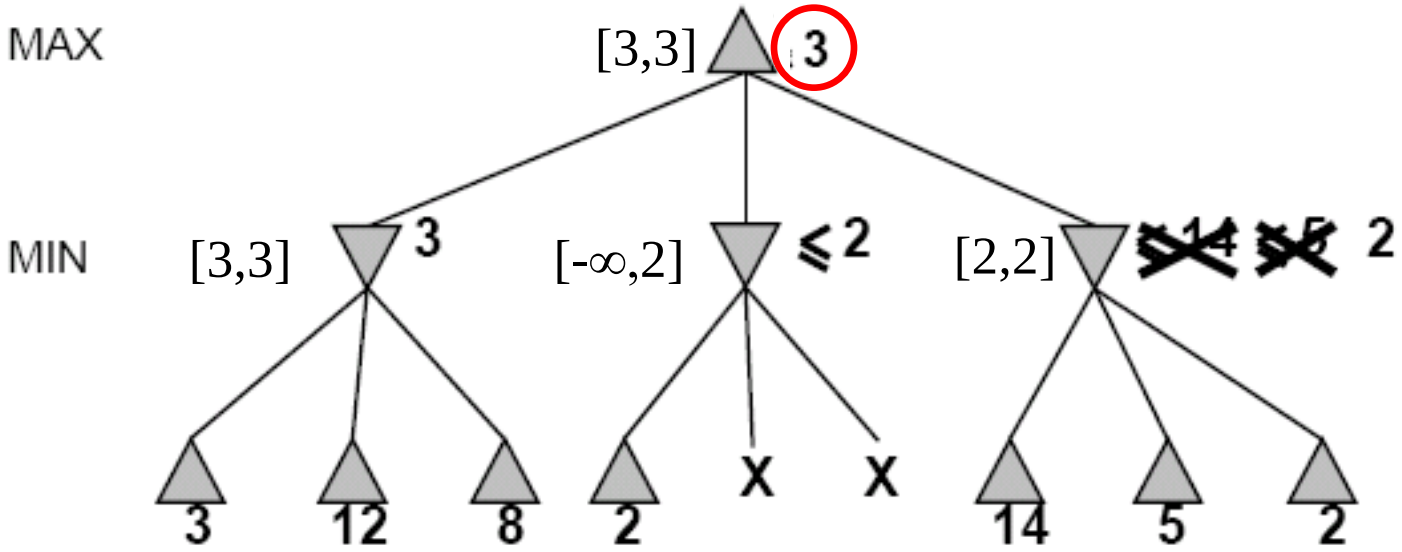
جستجوی خصمانه

مثال: هرس آلفا-بتا



جستجوی خصمانه

مثال: هرس آلفا-بتا



```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game

  v ← MAX-VALUE(state, −∞, +∞)
  return the action in SUCCESSORS(state) with value v
```

```
function MAX-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s, α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

- اگرچه که هرس آلفا- بتا باعث می شود که تنها بخشی از فضای حالت تا حالت پایانی توسعه یابد، اما این عمق هنوز هم عملی نیست.
- چون حرکات بازی باید در زمان منطقی انجام شوند, جستجو می بایست زودتر قطع شده و یک تابع ارزیابی اکتشافی در هنگام جستجو به حالتها اعمال شود تا از این طریق گره های غیرپایانی به صورت مؤثر گره پایانی تصور شوند.
- به بیان دیگر، پیشنهاد این است که minimax یا آلفا- بتا از دو طریق تغییر یابند:
- ۱. تابع سودمندی با یک تابع ارزیابی اکتشافی EVAL جایگزین شود که تخمینی از سودمندی وضعیت می دهد.
- ۲. آزمون پایانی با **آزمون برش** (cutofftest) جابه جا شود که تصمیم می گیرد چه زمان EVAL را اعمال کند.

جستجوی خصمانه

بازیهای قطعی با اطلاعات ناقص

معایب الگوریتم های پیشین

- الگوریتم minimax کل فضای جست و جوی بازی را تولید میکند
- الگوریتم آلفا-بتا با وجود هرس درخت، اما کل مسیر حالتهای پایانه، حداقل برای بخشی از فضای حالت، باید جست و جو شود
  - این عمق عملی نیست، زیرا حرکات باید در زمانی معقول انجام شود

شانون(۱۹۵۰)

برای کمتر شدن زمان جست و جو و اعمال تابع ارزیابی اکتشافی به حالتهای جستجو، بهتر است از گره های غیر پایانه به گره های پایانه پرداخته شود

جستجوی خصمانه

بازیهای قطعی با اطلاعات ناقص

- در شانون، minimax و آلفا-بتا به دو روش بطور متناوب عمل میکنند
- جایگزینی تابع سودمندی با تابع ارزیابی اکتشافی بنام EVAL
  - تخمینی از سودمندی موقعیت ارائه میکند
- جایگزین تست پایانه با تست توقف
  - تصمیم میگیرد EVAL چه موقع اعمال شود

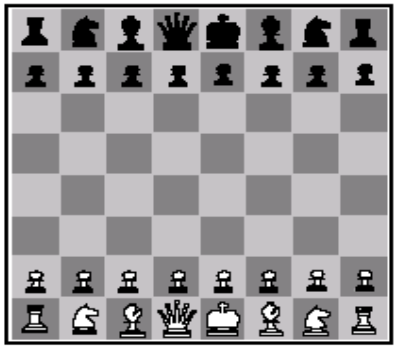
جستجوی خصمانه

تابع ارزیابی اکتشافی EVAL

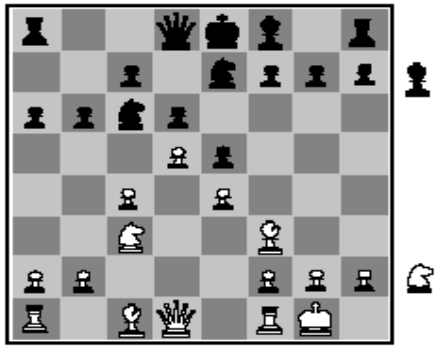
- تابع ارزیابی، ارائه تخمینی از سودمندی مورد انتظار بازی از یک موقعیت خاص
  - توابع اکتشافی، تخمینی از فاصله تا هدف را بر میگرداندند
- اغلب توابع ارزیابی، **خواص** گوناگونی از حالتها را محاسبه میکنند
  - خواص روی هم رفته، کلاسهای هم ارزی یا دسته های مختلفی از حالتها را تعریف میکنند
  - حالتهای هر دسته، برای تمام خواص مقدار یکسانی دارند
- هر دسته حاوی چند حالت است که
  - موجب برنده شدن
  - موجب رسم شدن
  - منجر به باختن
- تابع ارزیابی نمیداند کدام حالت منجر به چه چیزی میشود، اما میتواند مقداری برگرداند که تناسب حالتها را با هر نتیجه نشان دهد

جستجوی خصمانه

مثال: تابع EVAL

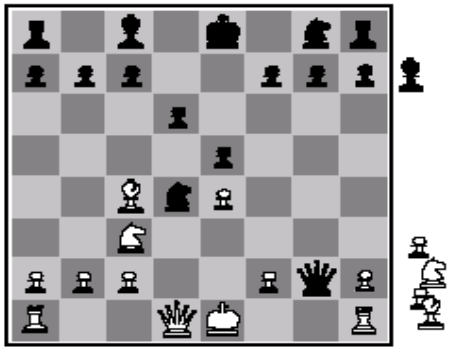


(a) White to move
Fairly even

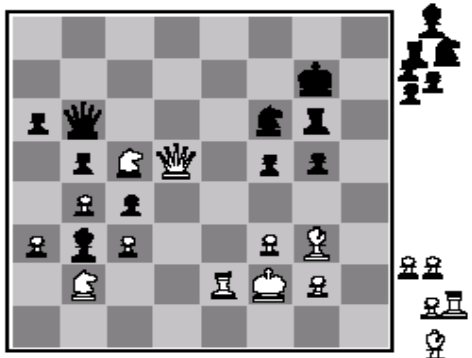


(b) Black to move
White slightly better

f_i



(c) White to move
Black winning



(d) Black to move
White about to lose

- اغلب توابع ارزیابی، مقدار عددی جداگانه ای برای هر خاصیت محاسبه، سپس آنها را ترکیب میکنند تا مقدار کل بدست آید

مثال در تابع بازی شطرنج:

f_i تعداد هر نوع قطعه در صفحه

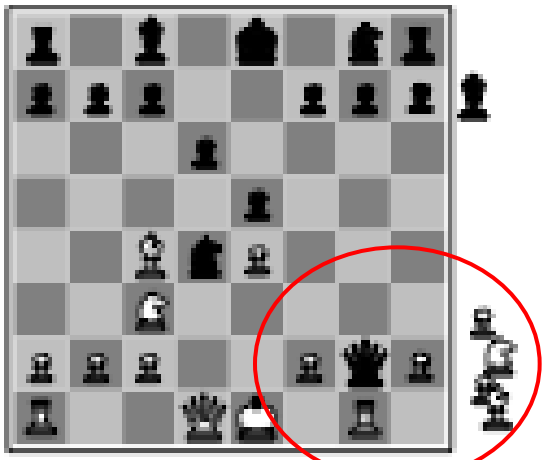
w_i مقادیر آن قطعات(۱ برای پیاده، ۳ برای اسب یا فیل،۵ برای رخ و ...)

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

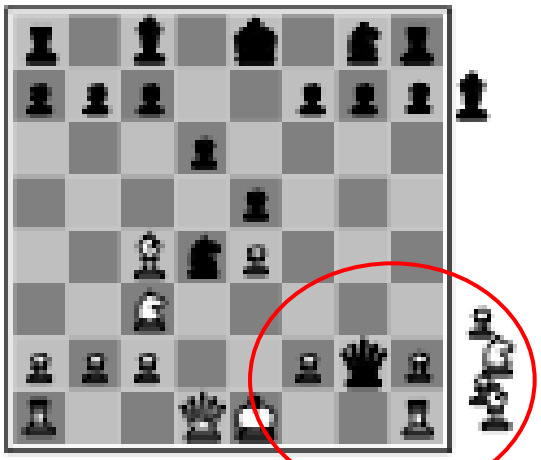
جستجوی خصمانه

مثال: تابع EVAL

ارزیابی تابع EVAL از مقدار پیروزی در دو موقعیت کاملا متفاوت



ب) سفید حرکت میکند



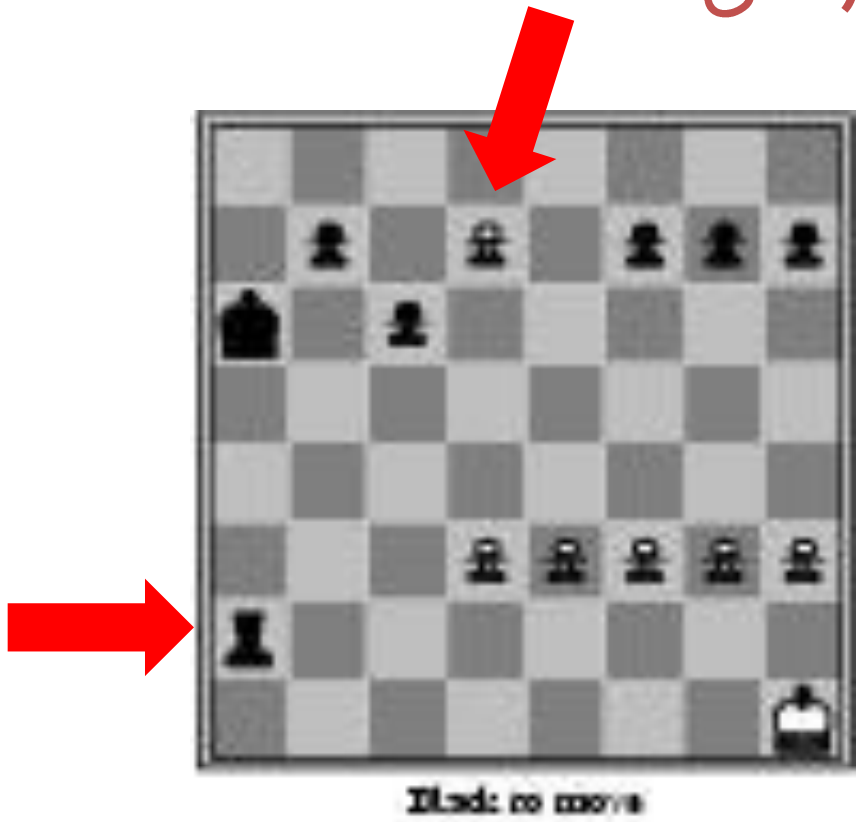
الف) سفید حرکت میکند

الف) سیاه، مزیت اسب و دو پیاده دارد و بازی را میبرد

ب) پس از اینکه سفید، وزیر را در اختیار میگیرد، سیاه میبازد

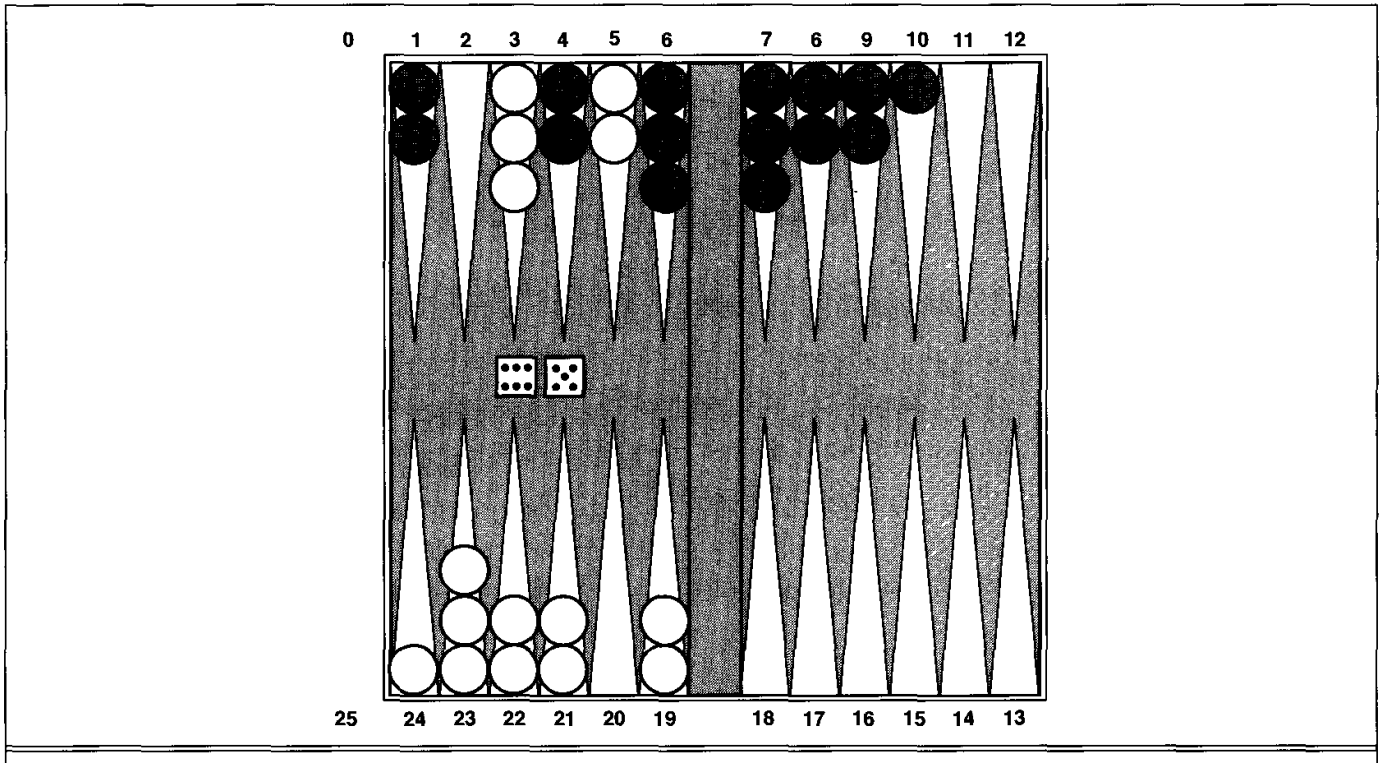
جستجوی خصمانه

اثر افق



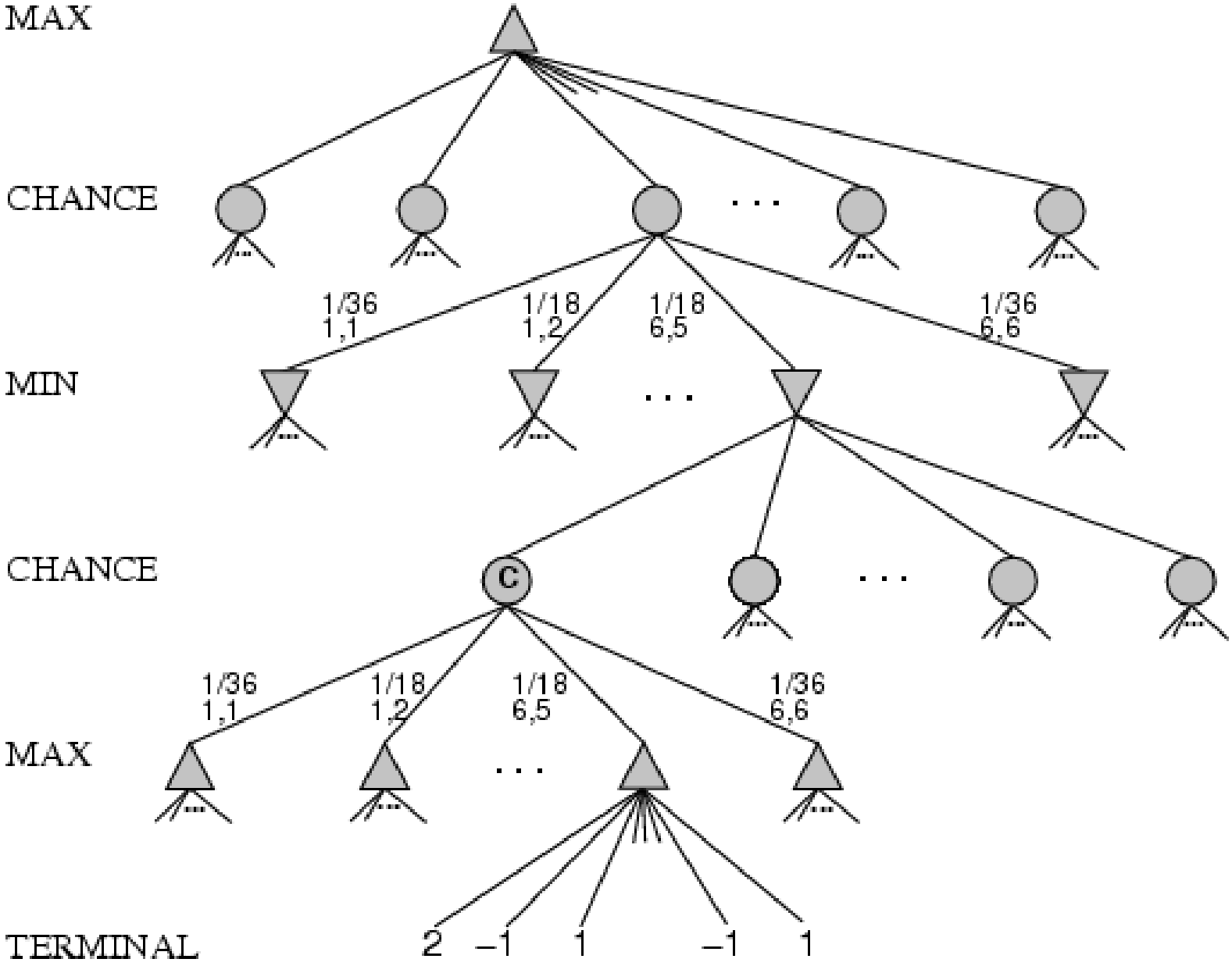
- وقتی بوجود می آید که برنامه با اثری از رقیب مواجه شود که منجر به خرابی جدی گشته و اجتناب پذیر است

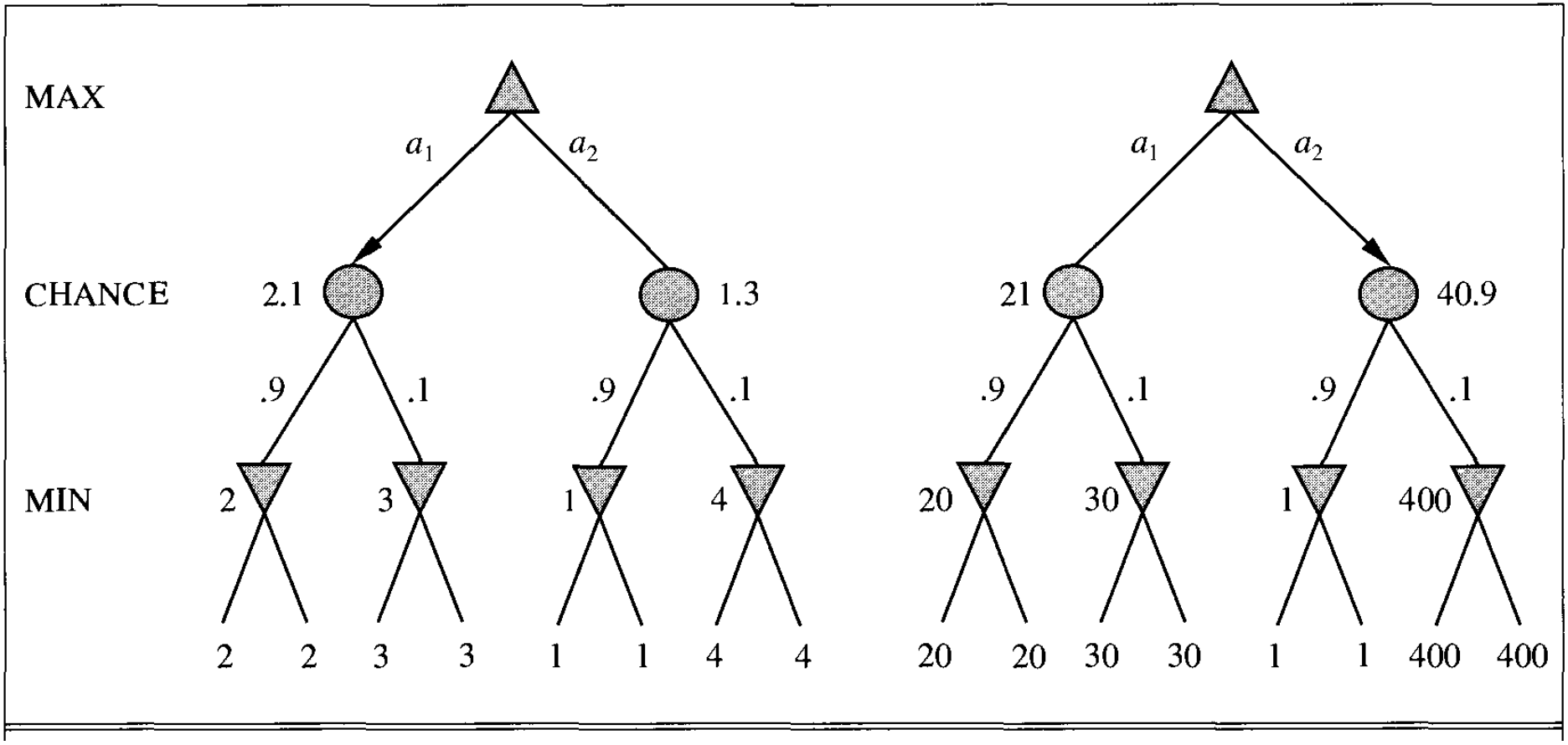
- مثال: شکل مقابل؛
  سیاه در اصل جلوست، اما اگر سفید پیاده اش را از سطر هفتم به هشتم ببرد، پیاده به وزیر تبدیل میشود و موقعیت برد برای سفید بوجود می آید



- چهار حرکت مجاز (۱۱-۵ و ۱۰-۵)، (۲۴-۱۹ و ۱۱-۵)، (۱۶-۱۰ و ۱۰-۵) و (۱۶-۱۱ و ۱۱-۵)

$$\text{EXPECTIMINIMAX}(n) = \begin{cases} \text{UTILITY}(n) & \text{if } n \text{ is a terminal state} \\ \max_{s \in Successors(n)} \text{EXPECTIMINIMAX}(s) & \text{if } n \text{ is a MAX node} \\ \min_{s \in Successors(n)} \text{EXPECTIMINIMAX}(s) & \text{if } n \text{ is a MIN node} \\ \sum_{s \in Successors(n)} P(s) \cdot \text{EXPECTIMINIMAX}(s) & \text{if } n \text{ is a chance node} \end{cases}$$


MAX
CHANCE
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MIN
CHANCE
C
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MAX
TERMINAL
2
−1
1
−1
1


MAX
CHANCE
MIN
a_1
a_2
2.1
1.3
.9
.1
2
3
1
4
2
2
3
3
1
1
4
4
a_1
a_2
21
40.9
.9
.1
20
30
1
400
20
20
30
30
1
1
400
400