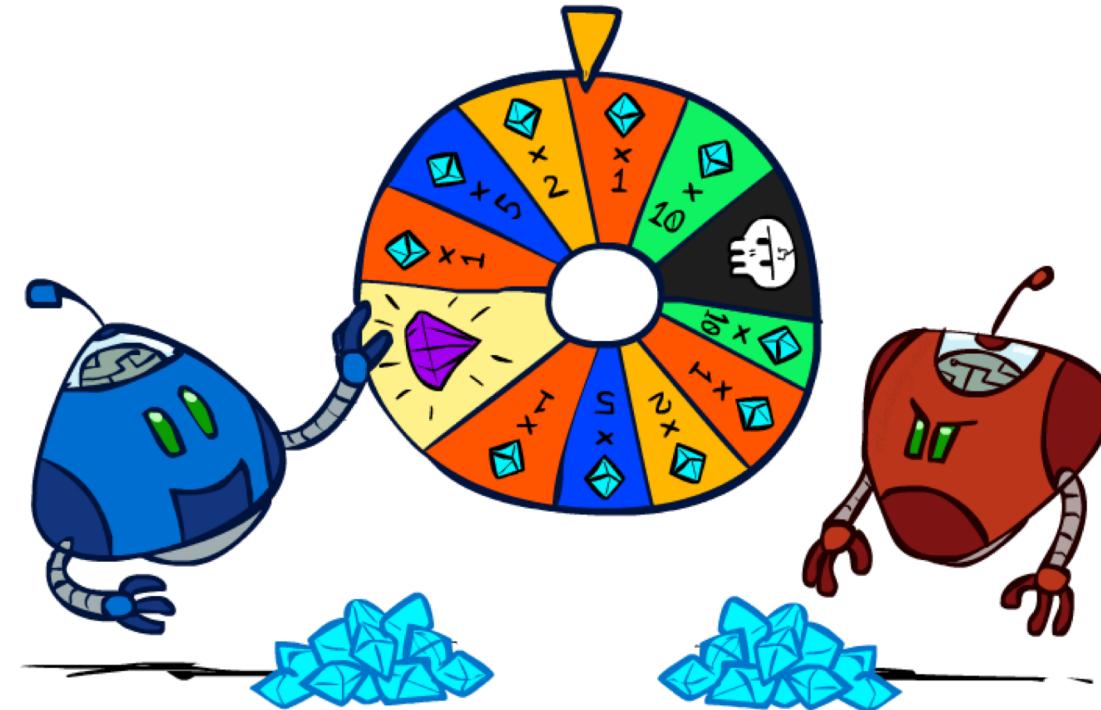


بنام حَرَادْخانَه وَ

هوش مصنوعی

عدم قطعیت و سودمندی



محمد طاهر پیلهور

[Most slides from CS188 Intro to AI at UC Berkeley]



جلسه‌ی قبل – بازی‌های رقابتی

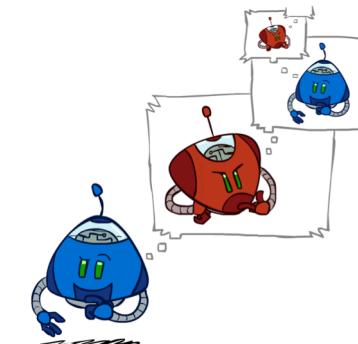
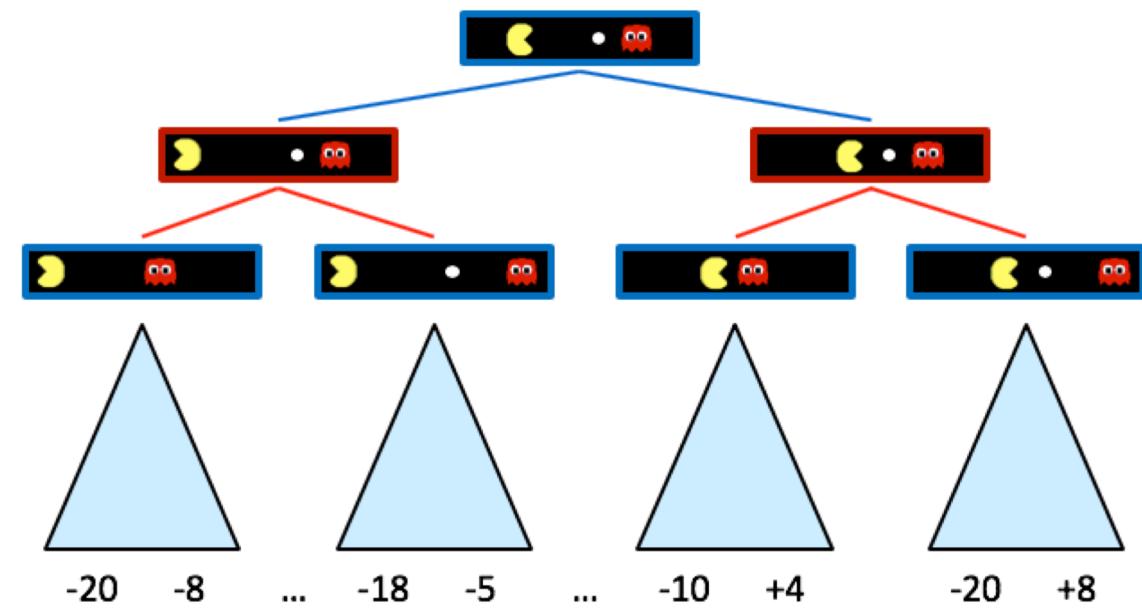
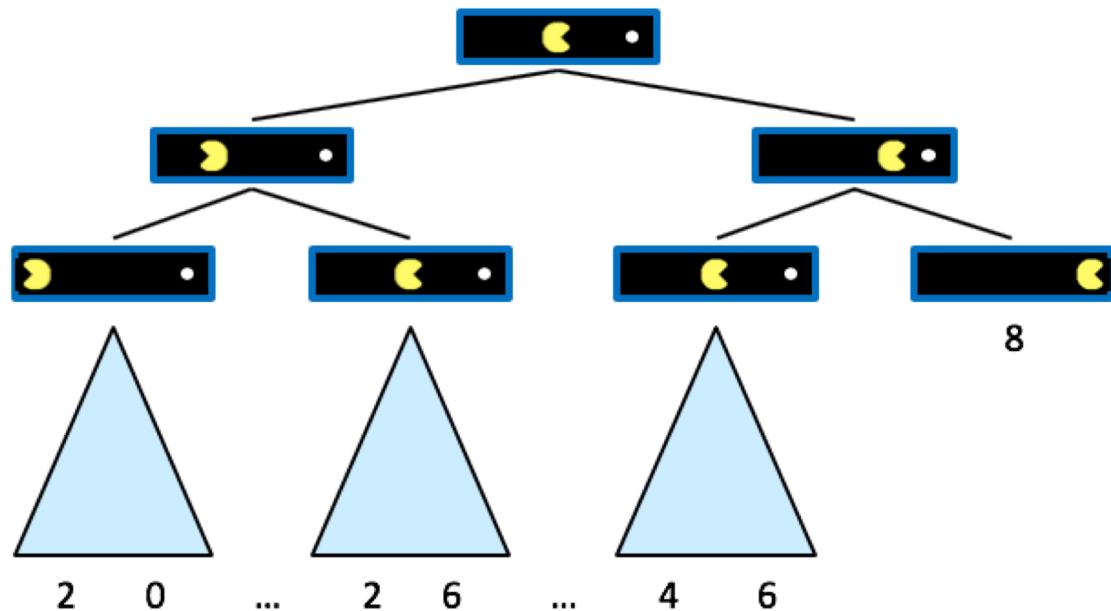


بازی‌های Zero-Sum

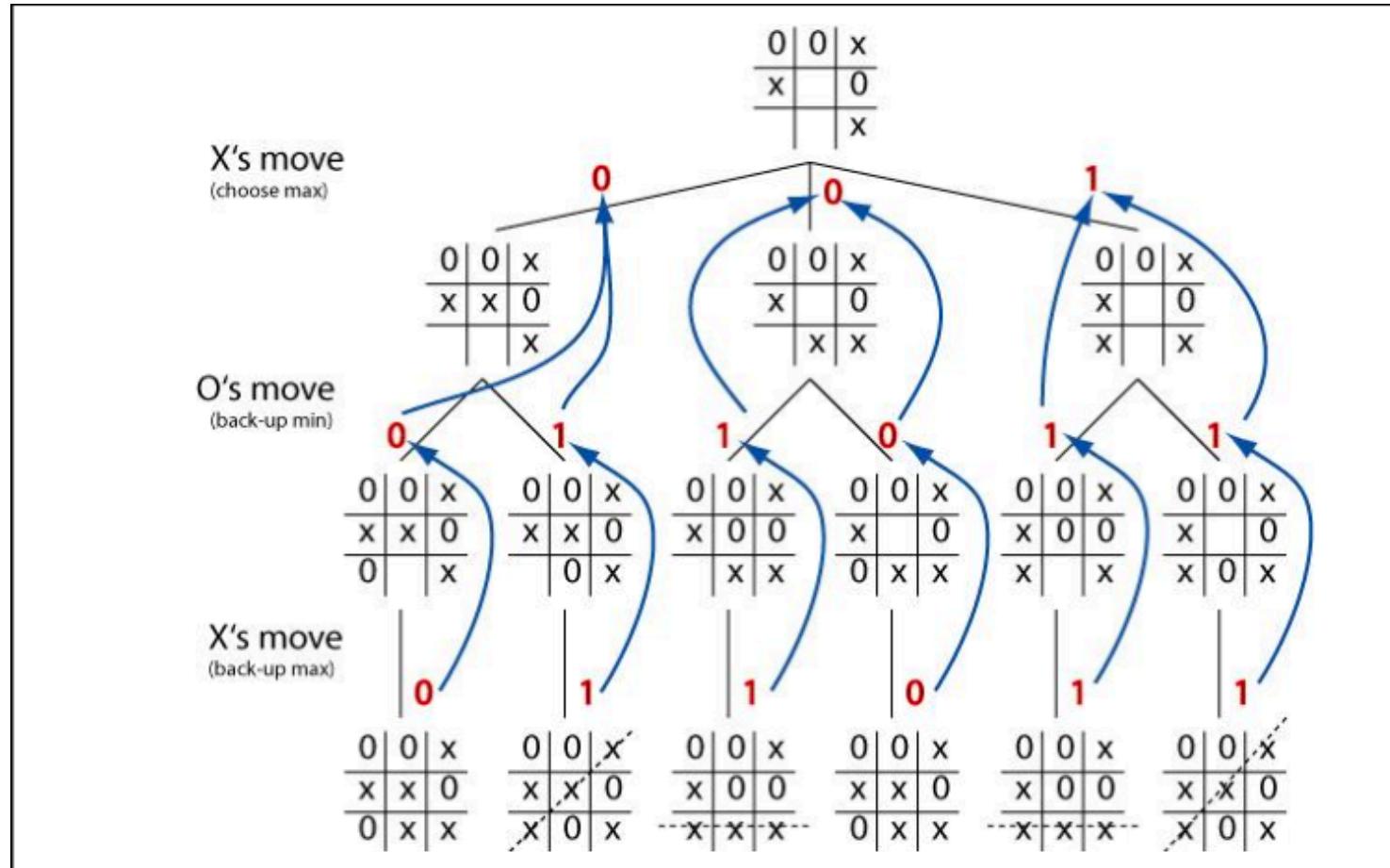
عامل‌ها منافع متضاد دارند

یک مقدار کلی وجود دارد که یکی قصد دارد آن را زیاد کند و دیگری کم
کاملاً رقابتی

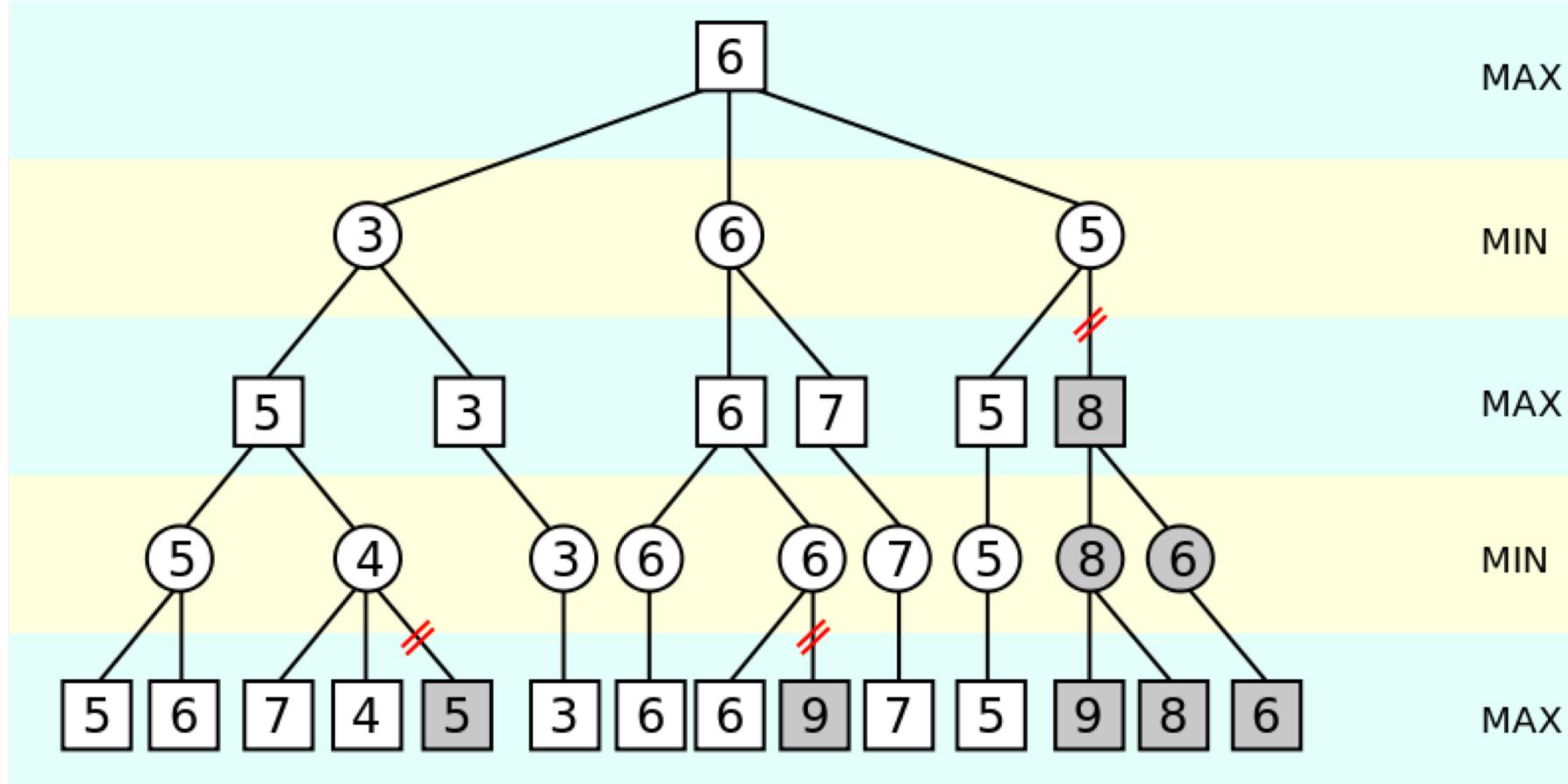
جلسه‌ی قبل – ارزش حالات



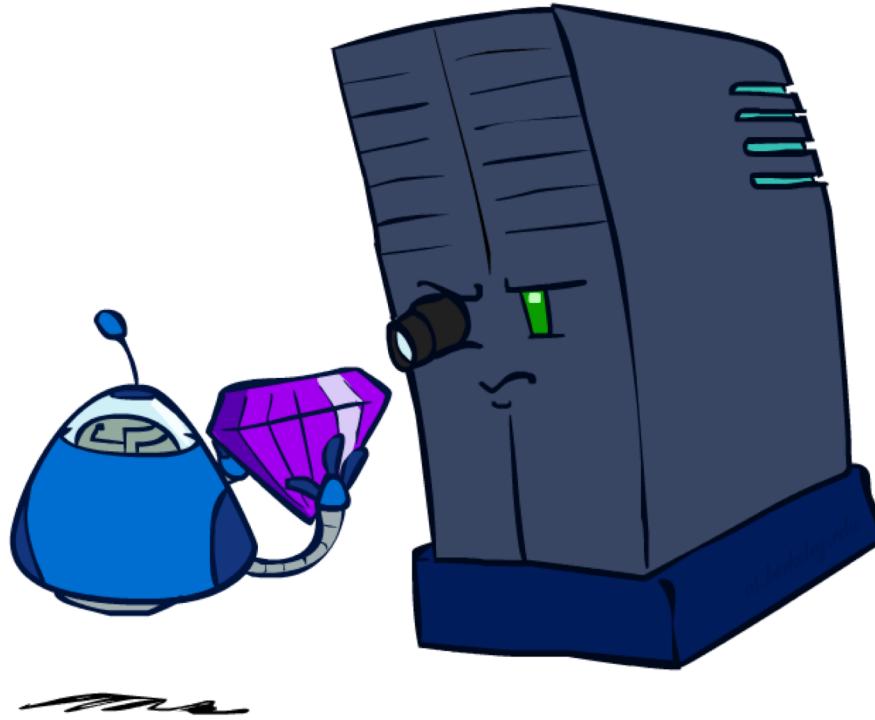
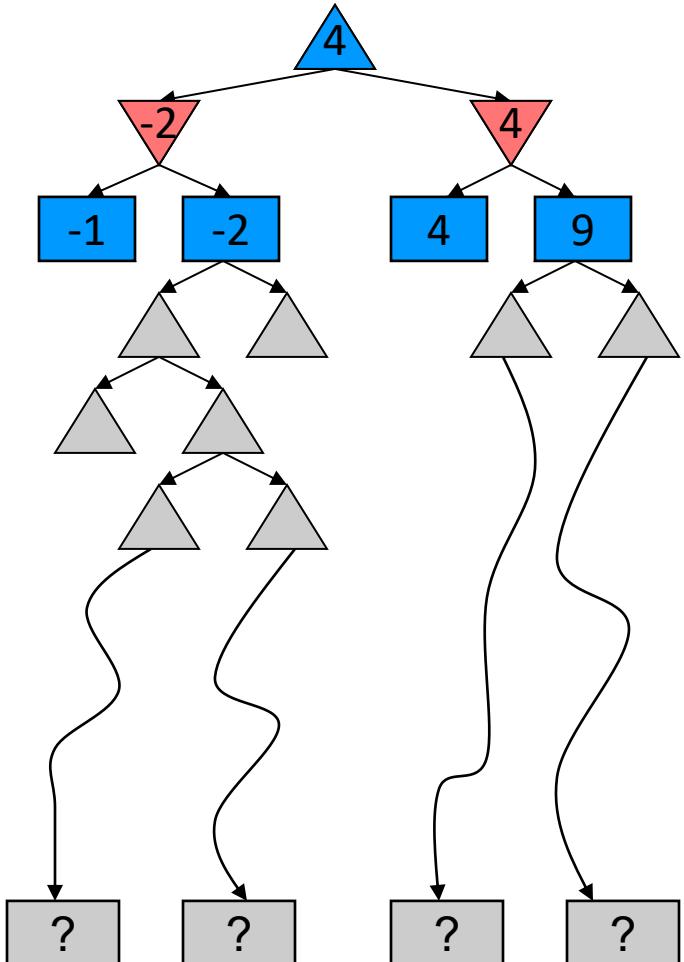
جلسهی قبل - درخت minimax



جلسه‌ی قبل - هرس کردن آلفابتا



جلسه‌ی قبل – توابع ارزیابی



برنامه شطرنج بسازید!



Lauri Hartikka

[Follow](#)

I like programming and Careless Whisper. <http://bit.ly/JMRmkU>

Mar 30, 2017 · 5 min read

A step-by-step guide to building a simple chess AI



<https://medium.freecodecamp.org/simple-chess-ai-step-by-step-1d55a9266977>

برنامه شطرنج بسازید!

Step 2 : Position evaluation

Now let's try to understand which side is stronger in a certain position. The simplest way to achieve this is to count the relative strength of the pieces on the board using the following table:



10



-10



30



-30



30



-30



50



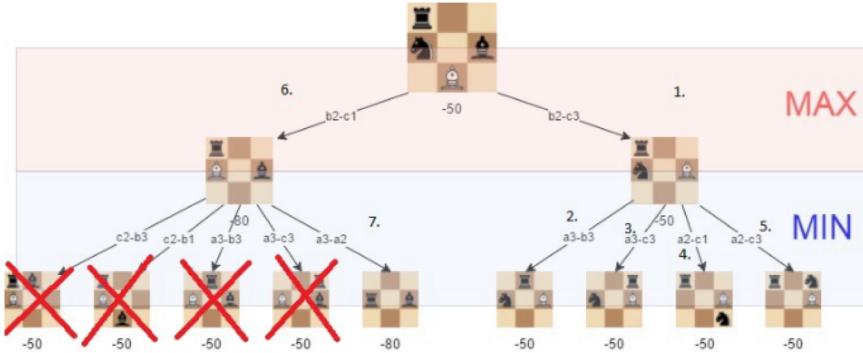
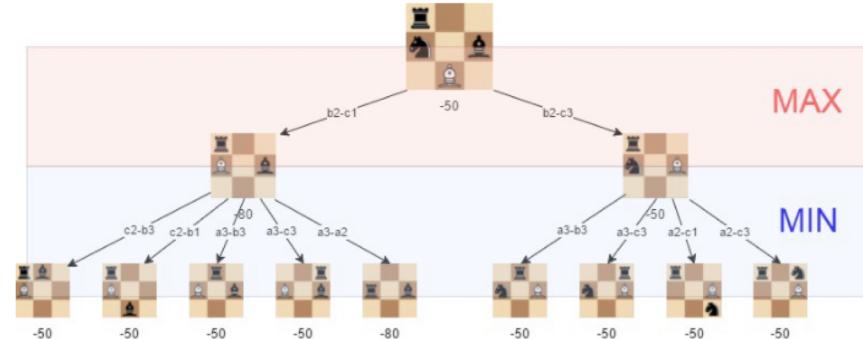
-50



90



-90



برنامه شطرنج بسازید!

HTML ▾

```
11 <option value="1">1</option>
12 <option value="2">2</option>
13 <option value="3" selected>3</option>
14 <option value="4">4</option>
15 <option value="5">5</option>
16 </select>
17
18 <br>
19 <span>Positions evaluated: <span id="position-count"></span></span>
20 <br>
21 <span>Time: <span id="time"></span></span>
22 <br>
23 <span>Positions/s: <span id="positions-per-s"></span> </span>
24 <br>
25 <br>
26 <div id="move-history" class="move-history">
27 </div>
28 </div>
29
```

JavaScript + jQuery 2.2.4 ▾

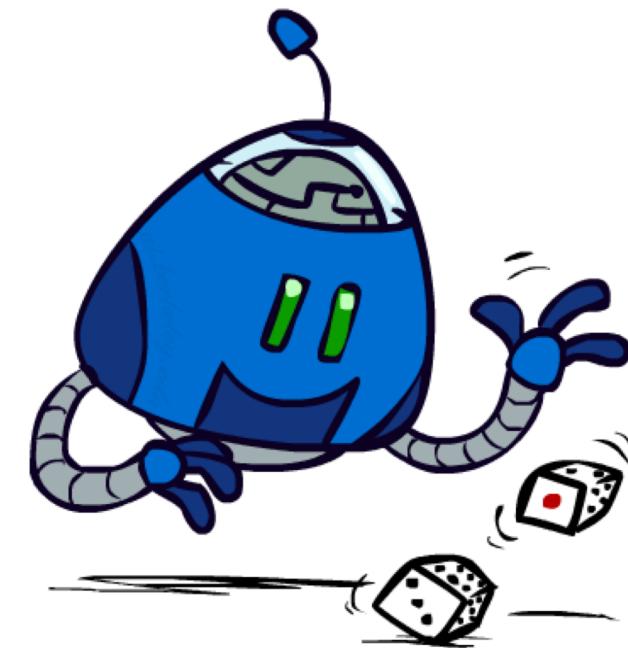
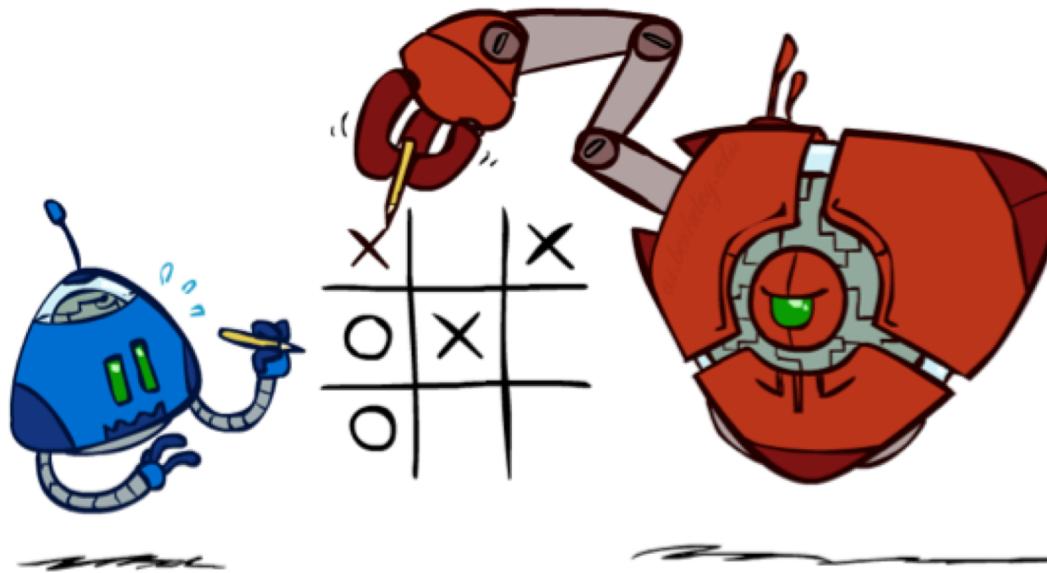
```
1 var board,
2     game = new Chess();
3
4 /*The "AI" part starts here */
5
6 var minimaxRoot =function(depth, game, isMaximisingPlayer) {
7
8     var newGameMoves = game.ugly_moves();
9     var bestMove = -9999;
10    var bestMoveFound;
11
12    for(var i = 0; i < newGameMoves.length; i++) {
13        var newGameMove = newGameMoves[i]
14        game.ugly_move(newGameMove);
15        var value = minimax(depth - 1, game, -10000, 10000,
16        !isMaximisingPlayer);
17        game.undo();
18        if(value >= bestMove) {
19            bestMove = value;
20            bestMoveFound = newGameMove;
21        }
22    }
23
24    return bestMoveFound;
25}
```

CSS ▾

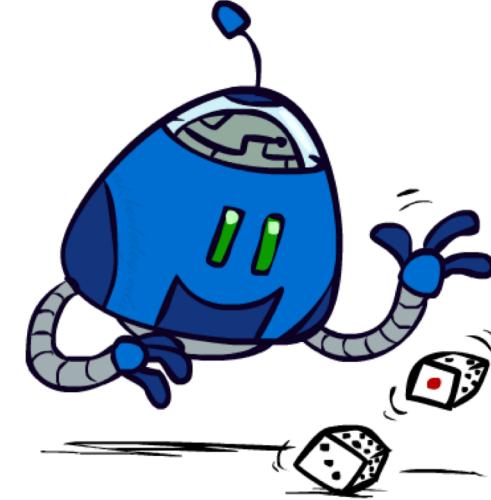
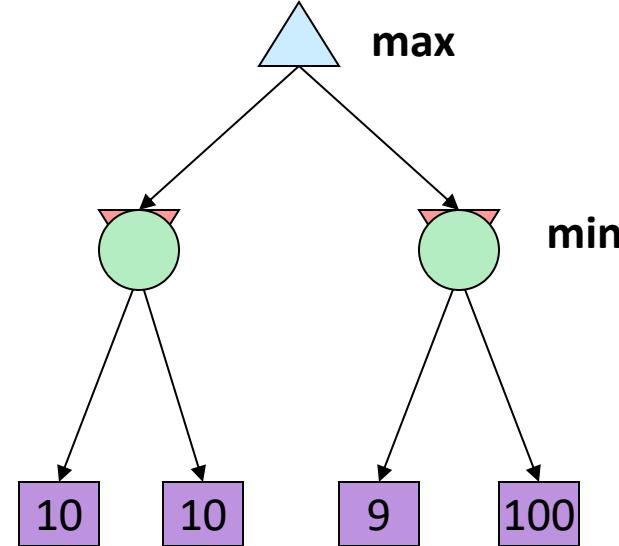
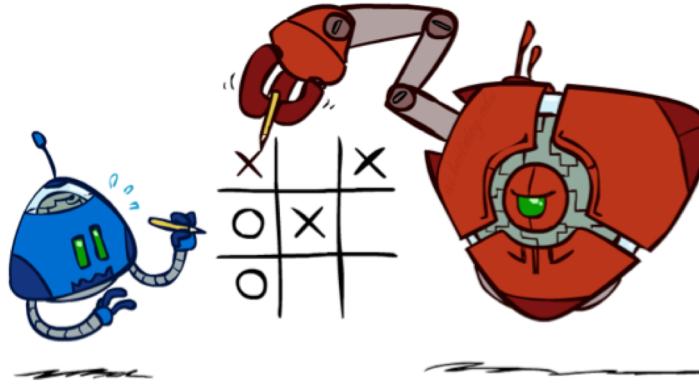
```
1 .board {
2     width: 400px;
3     margin: auto
4 }
5
6 .info {
7     width: 400px;
8     margin: auto;
9 }
10
11 .move-history {
12     max-height: 100px;
13     overflow-y: scroll;
14 }
15
16 /*!
17 * chessboard.js $version$
18 *
19 * Copyright 2013 Chris Oakman
```

<https://jsfiddle.net/Laa0p1mh/3/>

خروجی‌های غیرقطعی



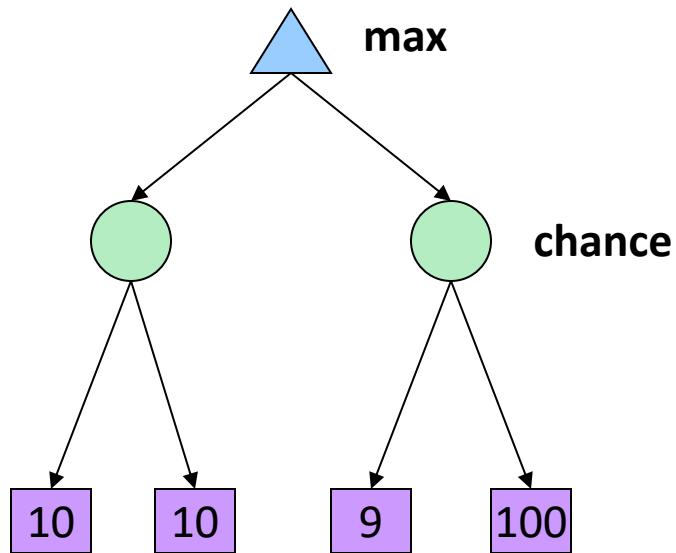
بدترین حالت و حالت متوسط



عدم قطعیتی که اتفاقی سنت نه محصول بازی با یک رقیب

جستجوی Expectimax

در چه شرایطی خروجی تصمیم ما دچار عدم قطعیت می‌شود؟



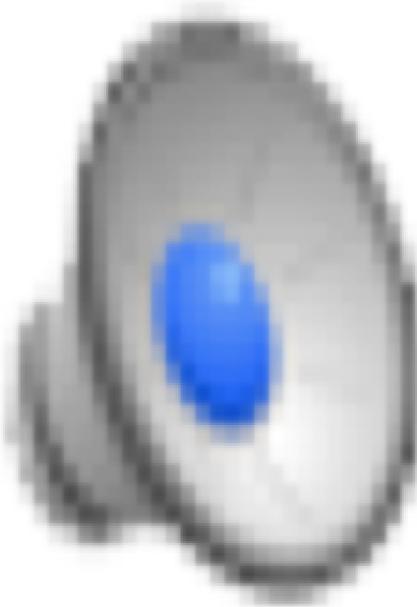
اگر مساله تصادفی باشد (بازی‌های تاسی!)
حرکت حریف غیرقابل پیش‌بینی باشد (شطرنج)
در اجرای تصمیم دچار مشکل شویم (اگر چرخ روبات لیز بخورد و نتواند به جایی که
می‌خواهد برسد)

بدترین خروجی (مینی‌مکس) - خروجی قابل انتظار (اکسپکتیمکس!)

جستجوی اکسپکتیمکس: برای حریف، سودمندی متوسط (مورد انتظار) را
در نظر بگیر!

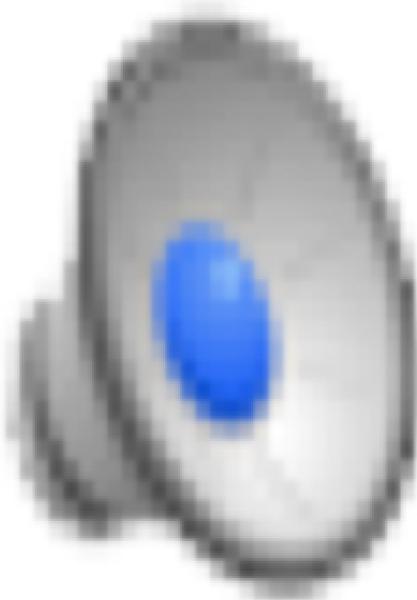
برای حریف، به جای در نظر گرفتن تصمیم مینی‌مکس (کمترین سودمندی)، احتمال انجام
هر عمل (توسط حریف) را با سودمندی آن ضرب و «سودمندی متوسط» را حساب کن.

Minimax در مقابل Expectimax



expectimax-vs-minimax-1.mp4 ویدئو

Minimax در مقابل Expectimax



expectimax-vs-minimax-2.mp4 ویدئو

الگوريتم Expectimax

```
def value(state):
```

 if the state is a terminal state: return the state's utility

 if the next agent is MAX: return max-value(state)

 if the next agent is EXP: return exp-value(state)

```
def max-value(state):
```

 initialize v = $-\infty$

 for each successor of state:

 v = max(v, value(successor))

 return v

```
def exp-value(state):
```

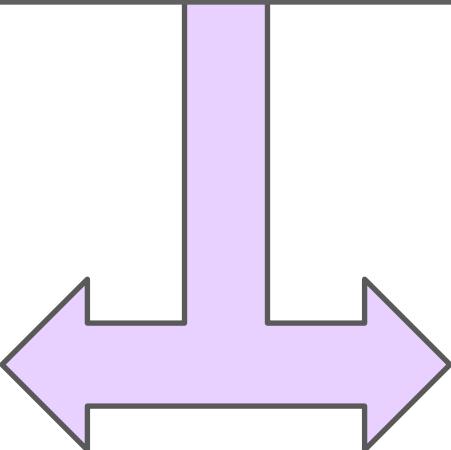
 initialize v = 0

 for each successor of state:

 p = probability(successor)

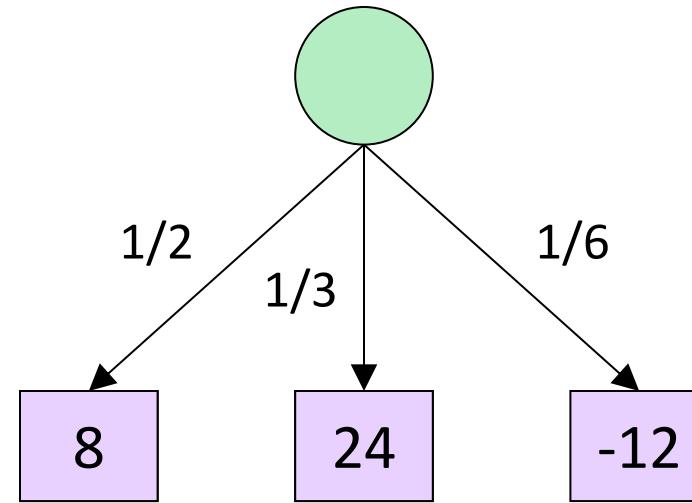
 v += p * value(successor)

 return v



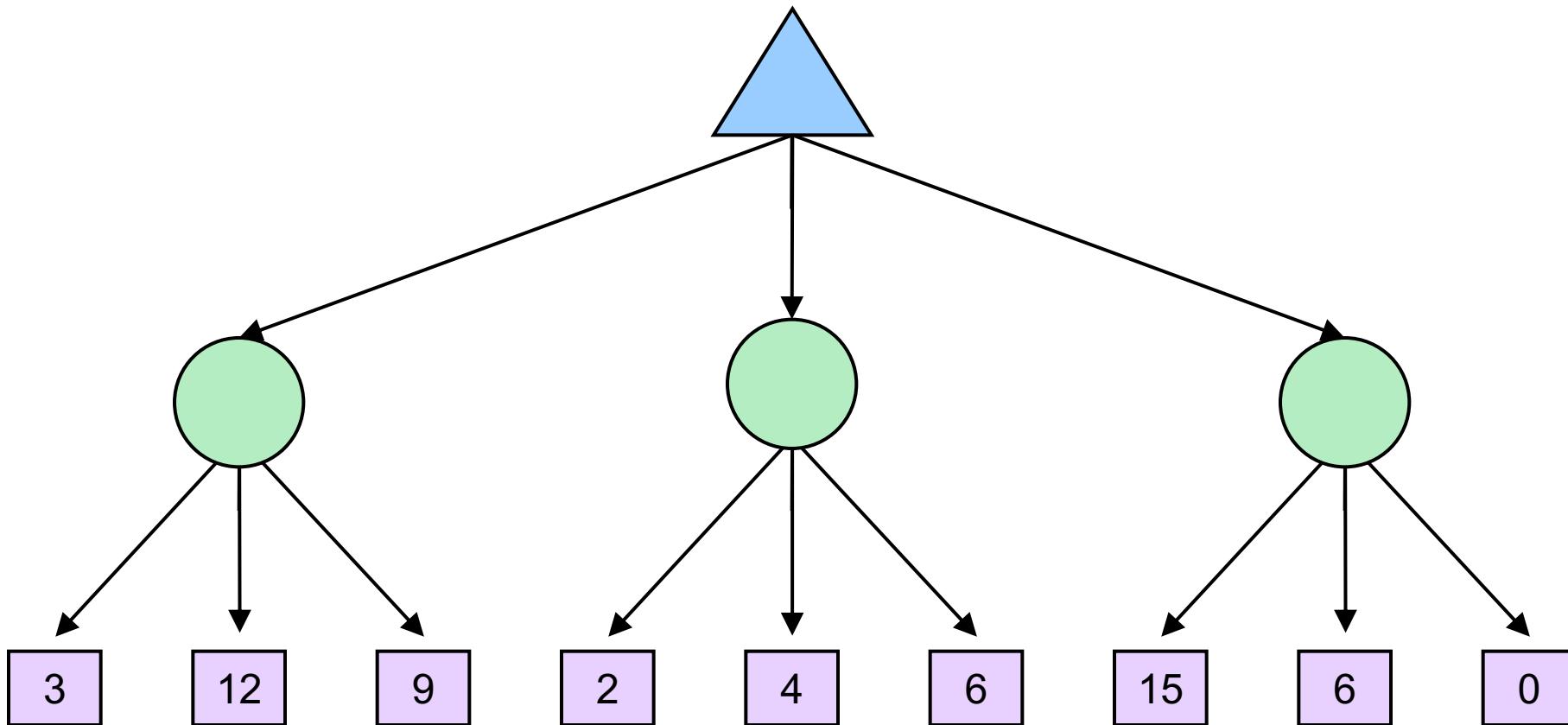
الگوريتم Expectimax

```
def exp-value(state):  
    initialize v = 0  
    for each successor of state:  
        p = probability(successor)  
        v += p * value(successor)  
    return v
```

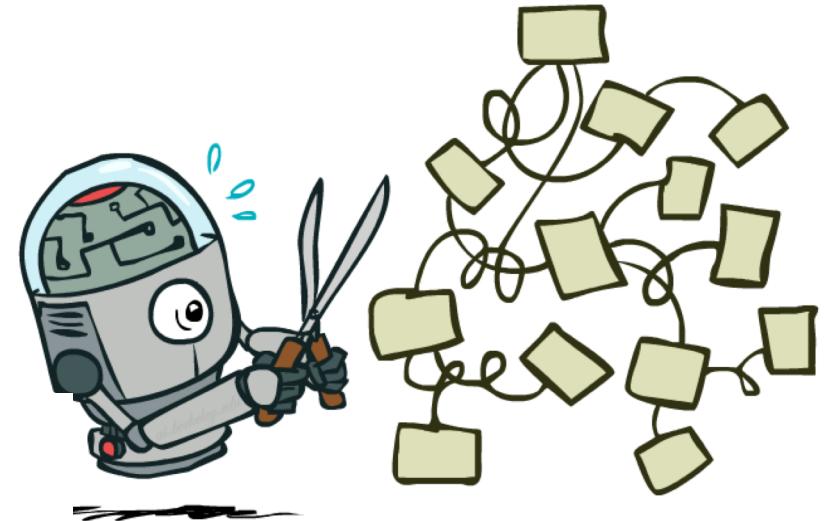
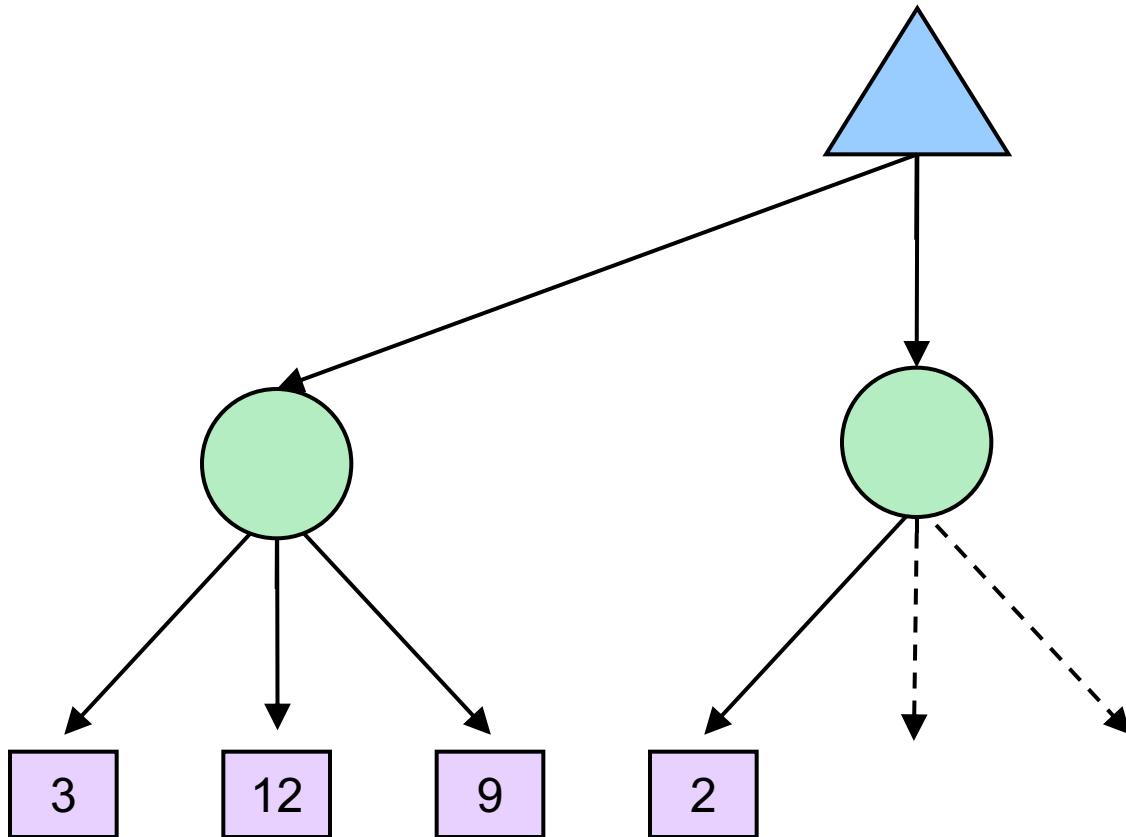


$$v = (1/2)(8) + (1/3)(24) + (1/6)(-12) = 10$$

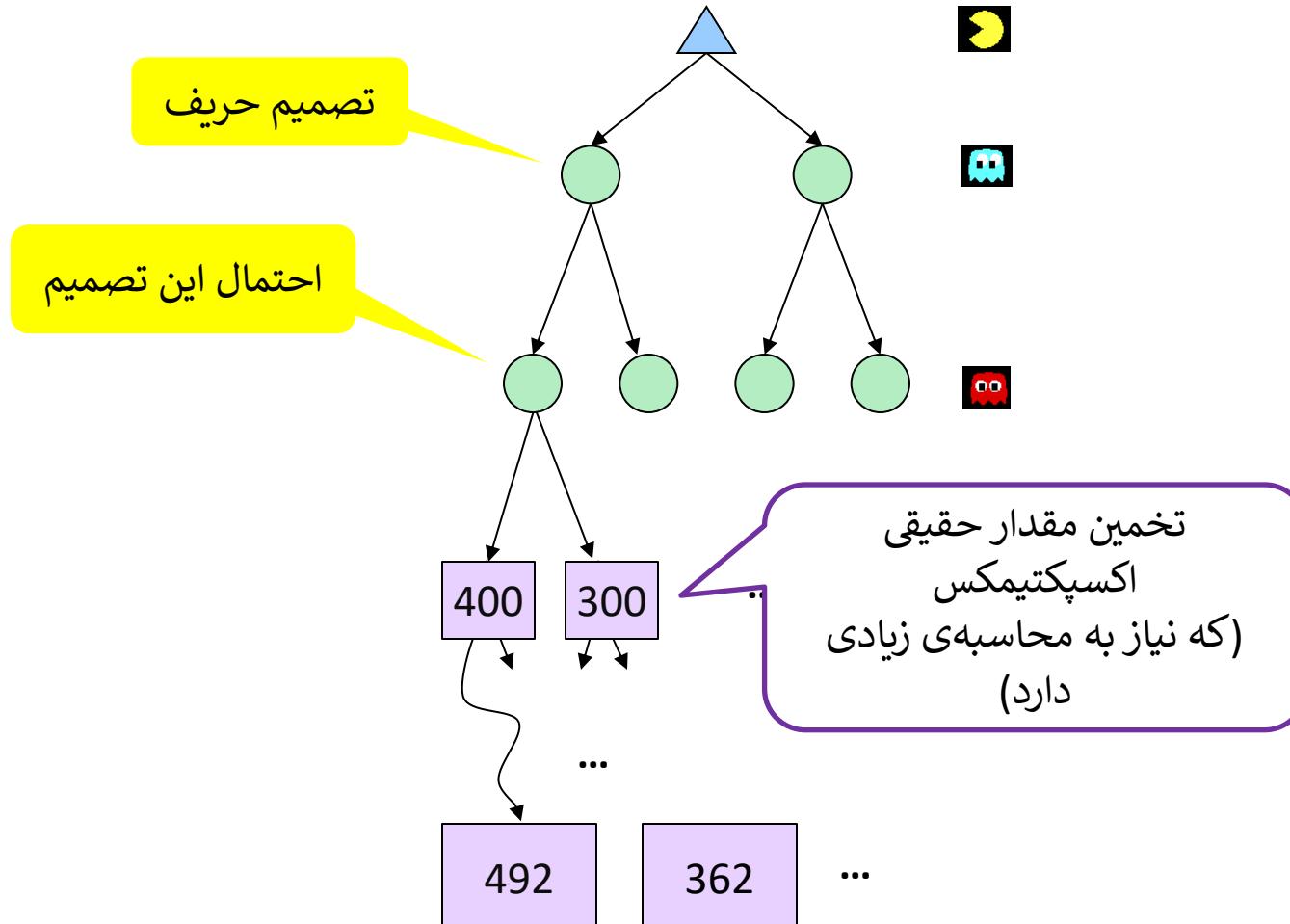
مثالExpectimax



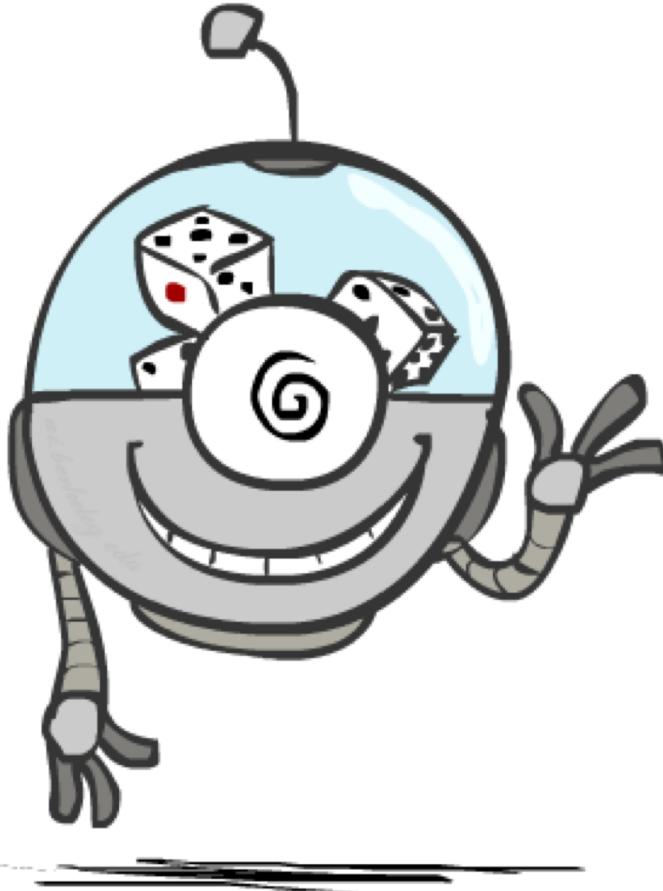
هرس کردن Expectimax



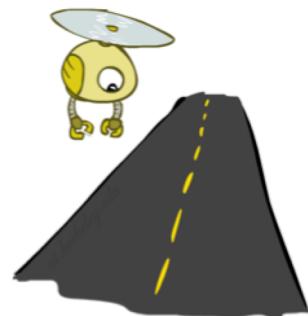
با عمق محدود Expectimax



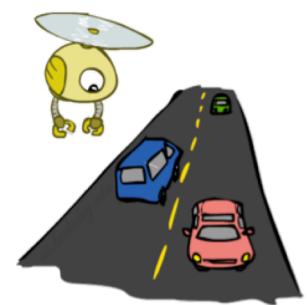
احتمالات



احتمالات - یادآوری



0.25



0.50



0.25

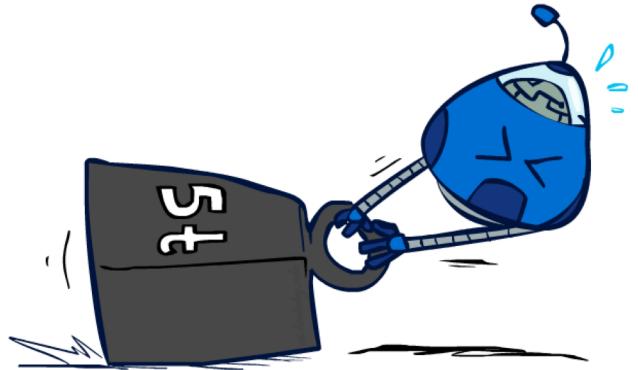
متغیر تصادفی - رویدادی که خروجیش مشخص نیست
توزیع احتمالاتی - به هر رویداد احتمالی می‌دهد

مثال: ترافیک!

متغیر تصادفی: T - مقدار ترافیک چقدر است?
خروجی‌های ممکن: هیچ، کم، سنگین
توزیع: $P(T=0.25) = 0.25, P(T=0.50) = 0.50, P(T=1.00) = 0.25$ (هیچ = 0.25, سنگین = 0.50, کم = 0.25)

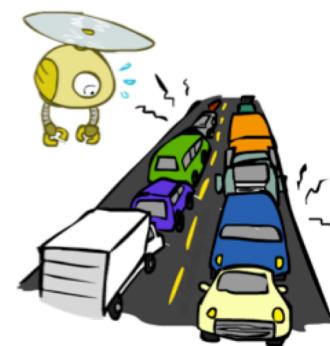
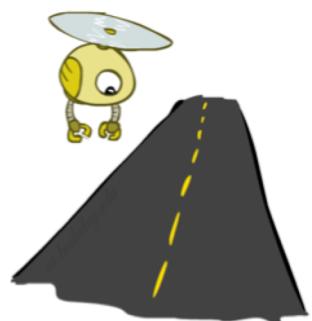
با شواهد و اطلاعات جدید، احتمالات ممکن است عوض شوند
 $P(T=0.25 | \text{Hour}=8\text{am}) = 0.60$

مثال

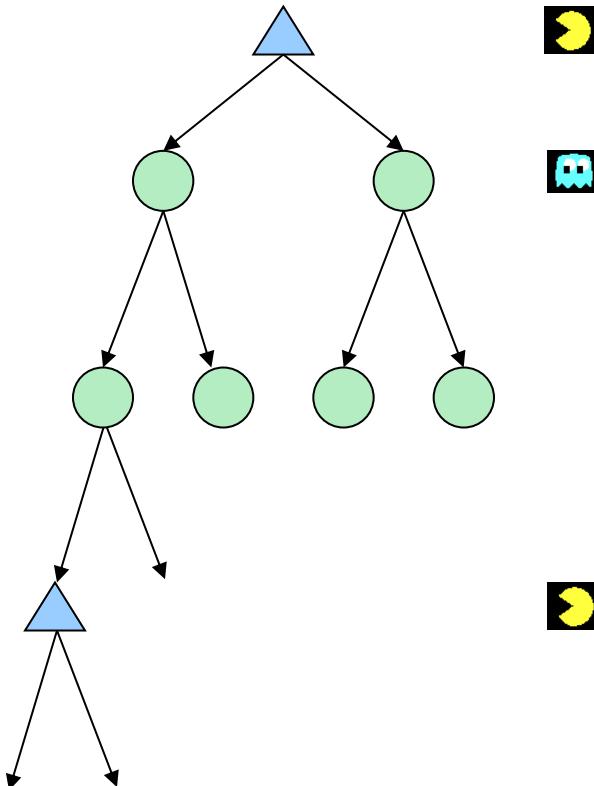


مثال: چقدر طول میکشد به فروشگاه برسیم؟

زمان:	20 min	x	30 min	x	60 min	x	35 min
احتمال:	0.25		0.50		0.25		



چطور پیش بینی کنیم؟



مدل احتمالاتی که ما از عملکرد حریف بدست می آوریم به این معنی نیست که حریف ما احتمالاتی (تصادفی) عمل میکند (مثلما تاس میندازد)، بلکه تصور ما از عملکردش اینگونه است

به یک مدل (احتمالاتی) نیاز داریم که به ما بگوید که واکنش حریف (محیط) در ازای هر یک از تصمیمات ما چگونه خواهد بود
* با چه احتمالی یک واکنش خاص را انجام خواهد داد

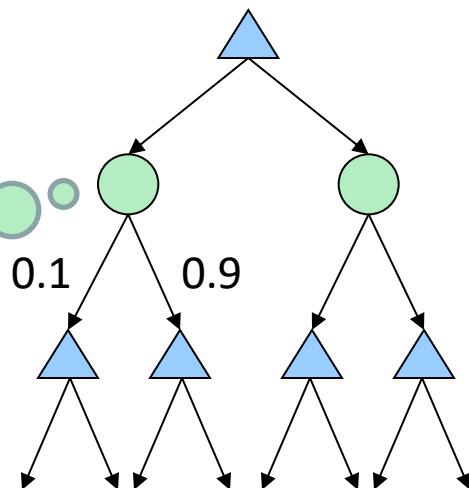
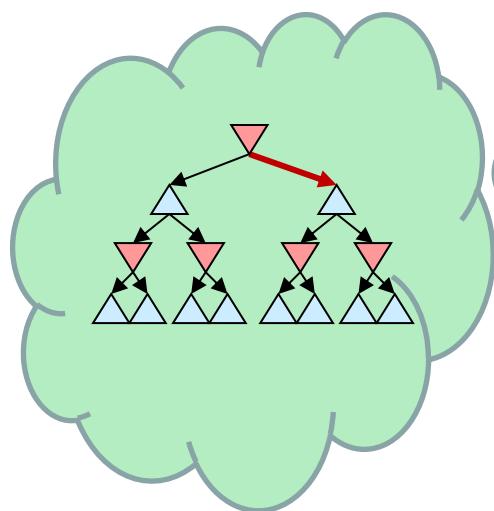
مدل می تواند به سادگی توزیع یونیفرم باشد (تصادفی، انداختن تاس!)

یا اینکه پیچیده باشد و نیازمند محاسبه‌ی سنگین!

فعلا فرض میکنیم که این احتمالات را (با تعریف مساله) داریم!

سوال

فرض کنیم استراتژی بازی حریف را می‌دانیم: ۸۰ درصد اوقات مینیممکس با عمق ۲، ولی در ۲۰ درصد اوقات تصادفی (که مثلاً پیش‌بینی استراتژی را برای ما سخت کند). از چه جستجویی استفاده کنیم؟



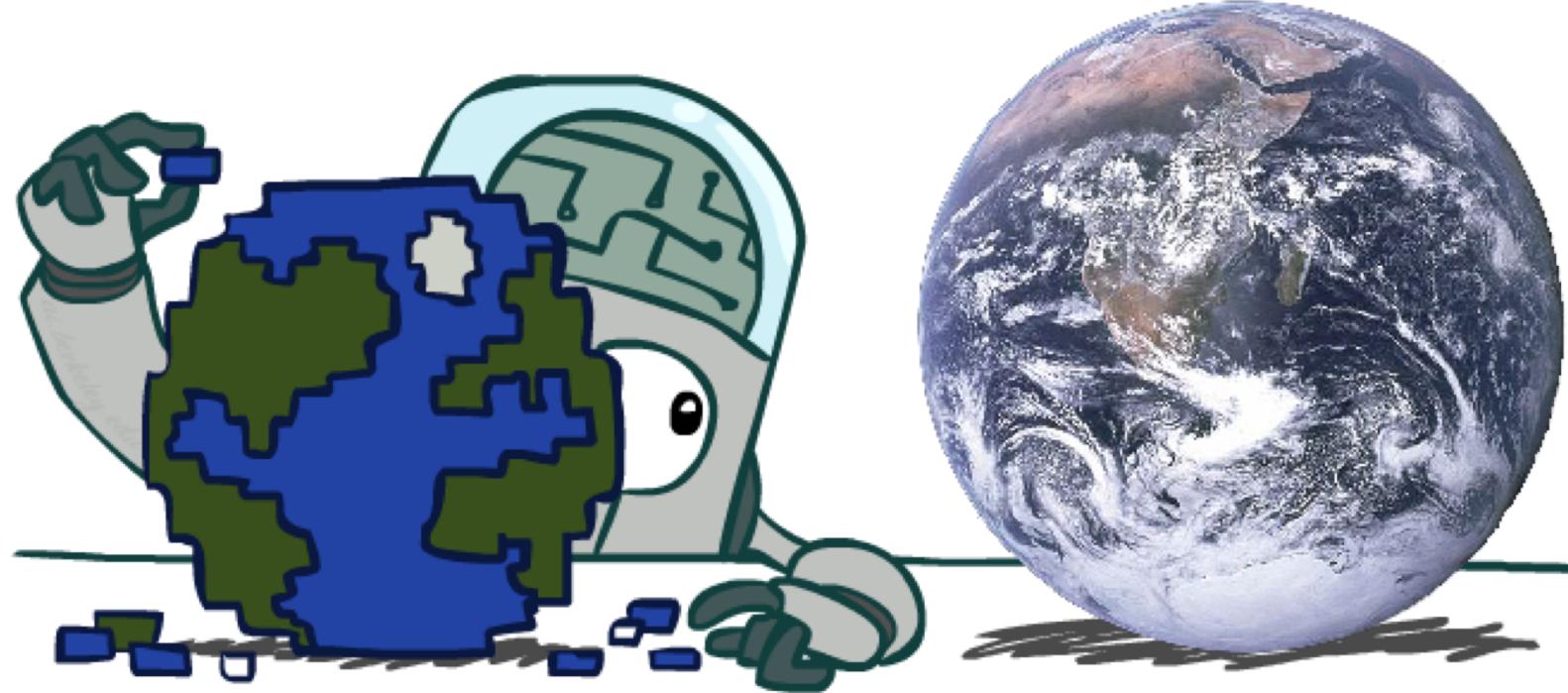
Expectimax!

احتمال تصمیمات حریف؟
با شبیه‌سازی رفتار حریف می‌توانیم محاسبه کنیم

اگر بخواهیم حریفی که ما را شبیه‌سازی می‌کند
شبیه‌سازی کنیم چه؟!

مزیت مینی‌مکس اینست که کلاً یک درخت داریم که همه چیز را نشان می‌دهد (بنابراین از نظر محاسباتی، خیلی ساده‌تر)

فرضها در مدلسازی



مشکلات خوشبینی و بدبینی

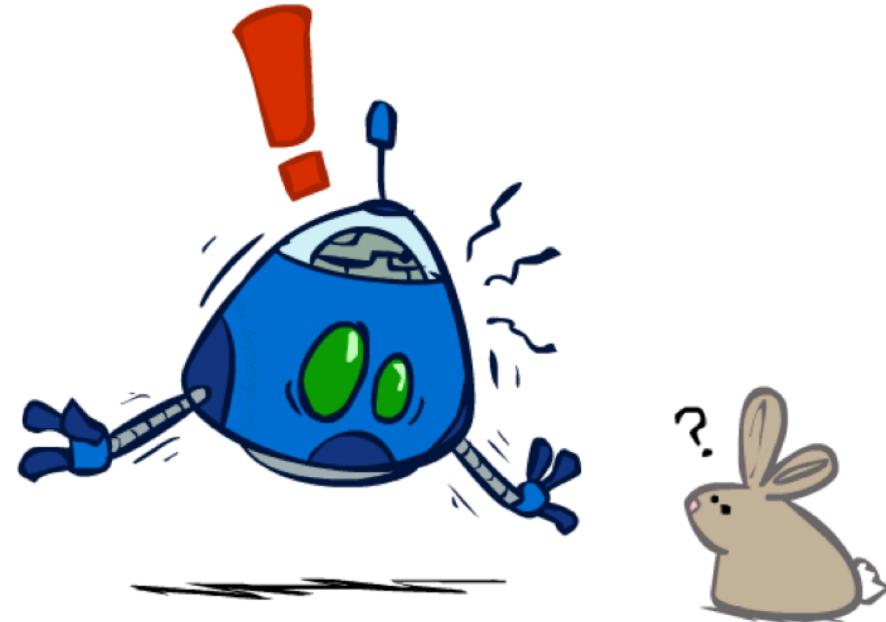
خوشبینی

فرض کنیم محیط تصادفی است در حالی که در حقیقت رقابتی است

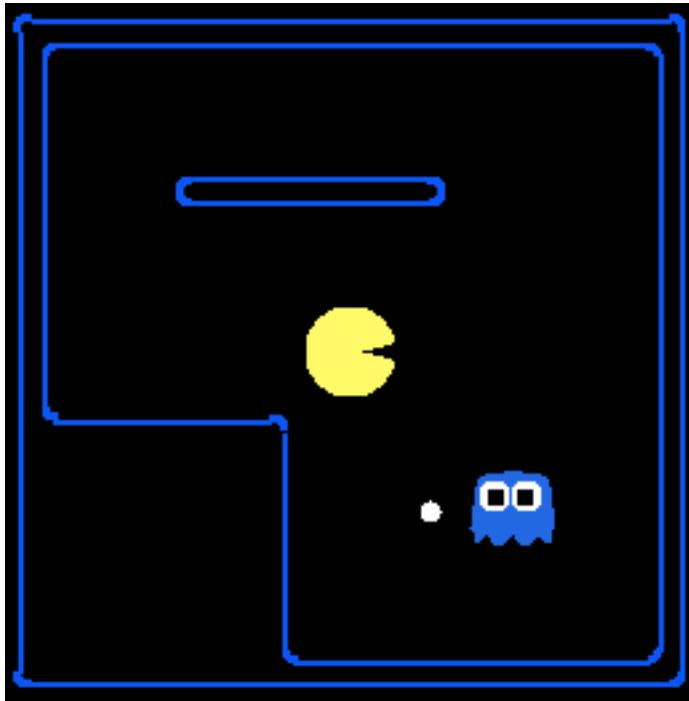


بدبینی

بدترین حالت را فرض کنیم (رقابتی)، در صورتی که محیط تصادفی است



فرض در مقابل حقیقت

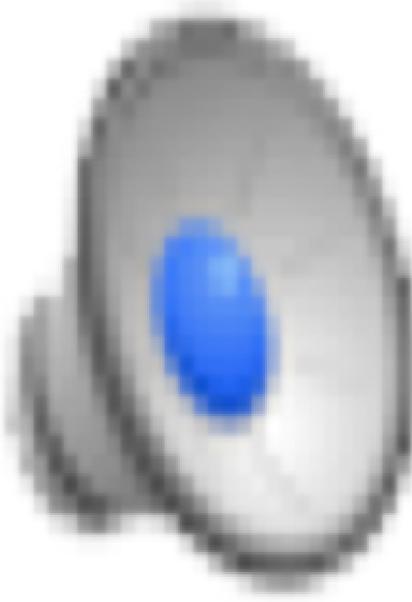


		رقابتی	تصادفی
Minimax			
Expectimax			

برای ارزیابی این حالات، می‌توانیم شبیه‌سازی کنیم!

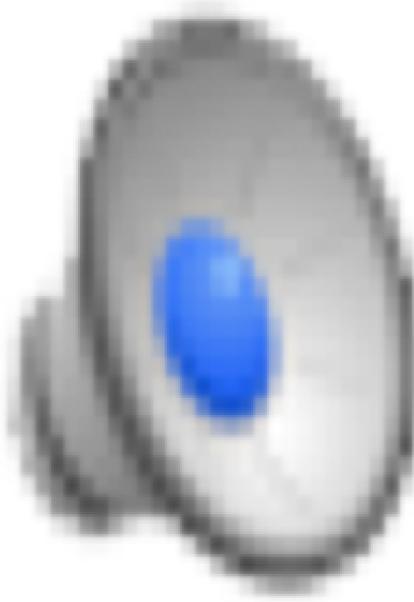
پکمن: جستجوی عمق ۴ – تابع هدف (هزینه): از دردسر دوری کن
حریف: جستجوی عمق ۲ – تابع هدف (هزینه): برو دنبال پکمن

پکمن: اکسپکتیمکس – حریف: تصادفی



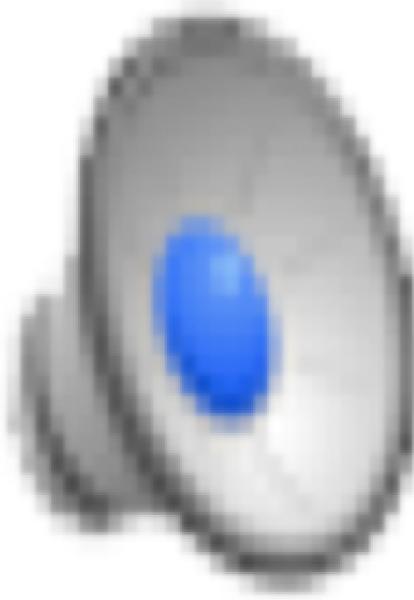
ویدئو expectimax-1.mp4

پکمن: مینیمکس – حریف: رقابتی



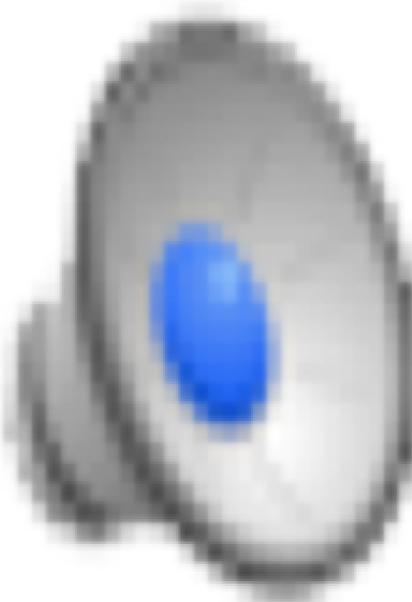
minimax-1.mp4
ویدئو

پکمن: اکسپکتیمکس – حریف: رقابتی



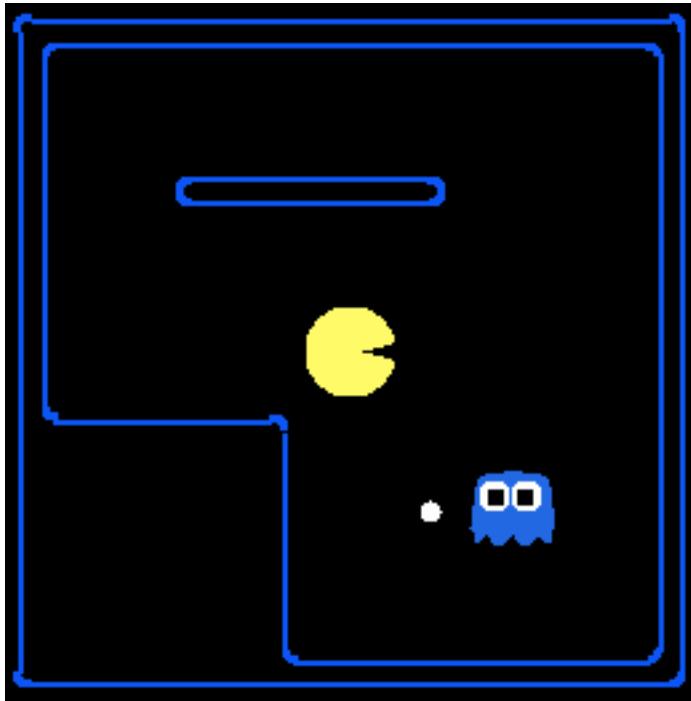
ویدئو expectimax-2.mp4

پکمن: مینیمکس – حریف: تصادفی



ويdeo expectimax-3.mp4

فرض در مقابل حقیقت

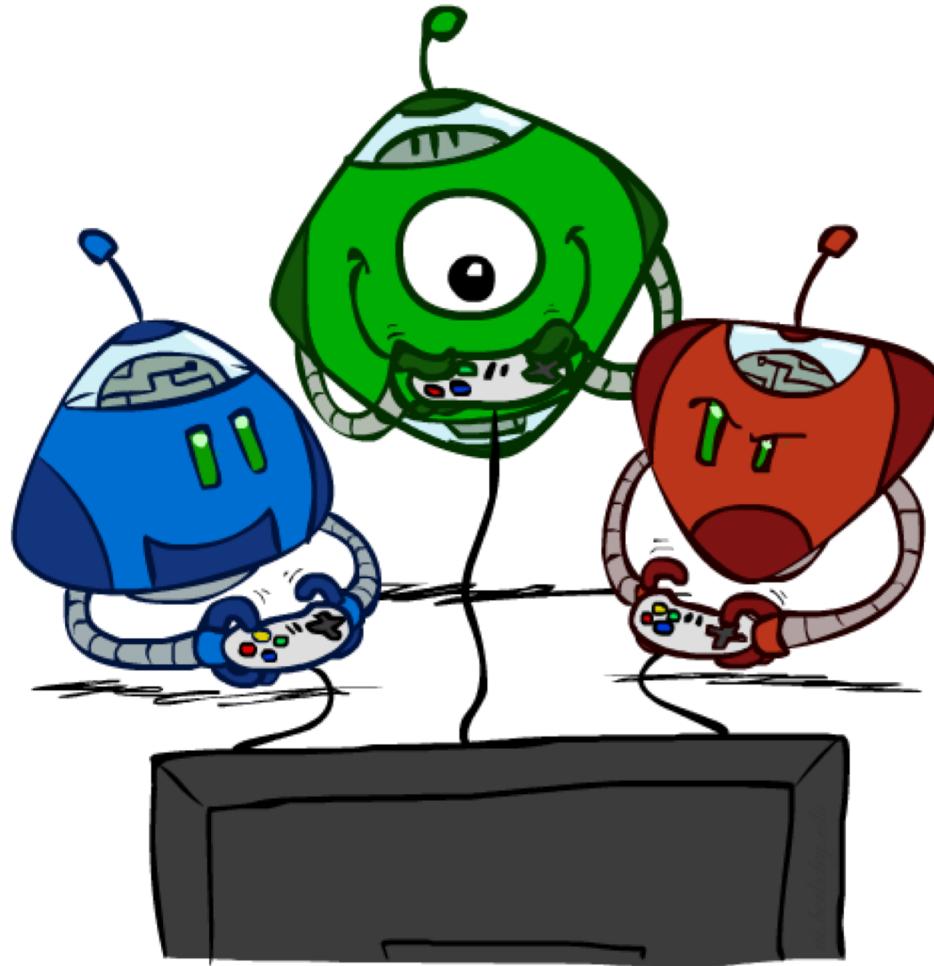


	رقبتی	تصادفی
Minimax	برنده: ۵/۵ متوسط امتیاز: ۴۸۳	برنده: ۵/۵ متوسط امتیاز: ۴۹۳
Expectimax	برنده: ۱/۵ متوسط امتیاز: -۳۰۳	برنده: ۵/۵ متوسط امتیاز: ۵۰۳

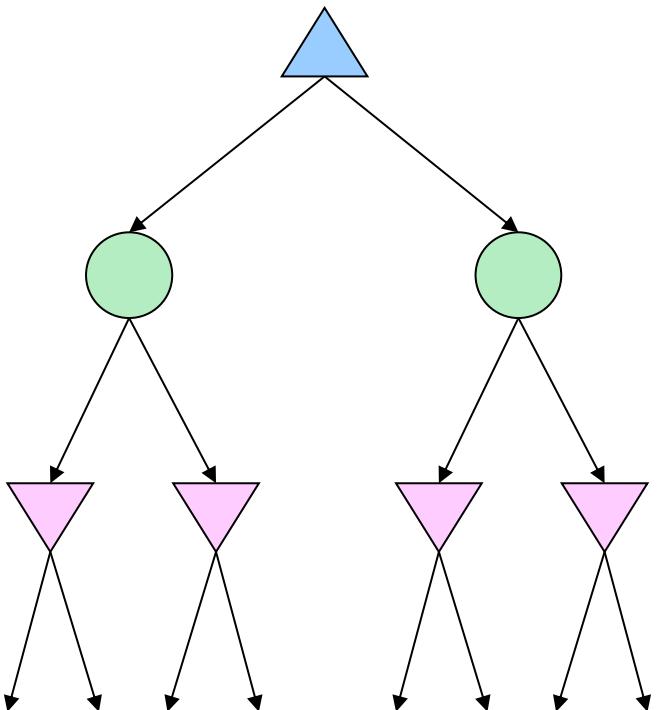
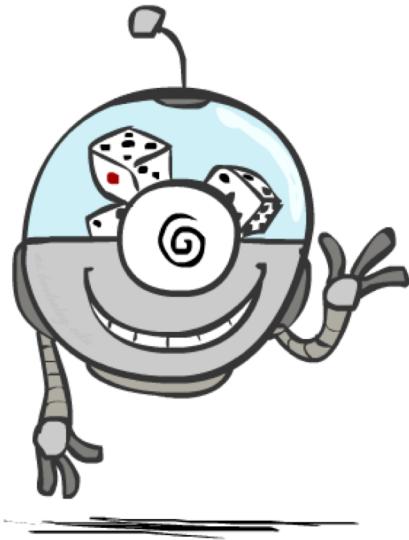
میانگین ۵ بار بازی

پکمن: جستجوی عمق ۴ – تابع هدف (هزینه): از دردسر دوری کن
حریف: جستجوی عمق ۲ – تابع هدف (هزینه): برو دنبال پکمن

انواع دیگر بازی



لایه های متفاوت



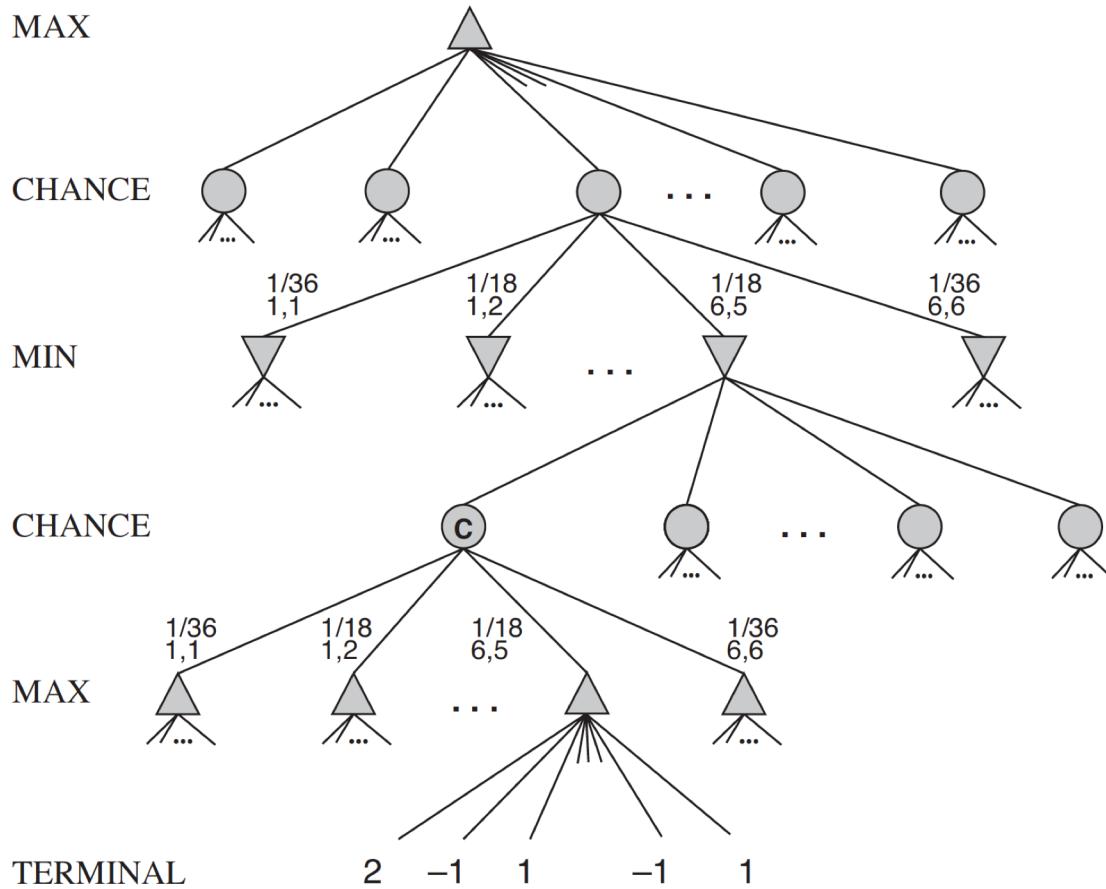
خته نرد:

Expectiminimax

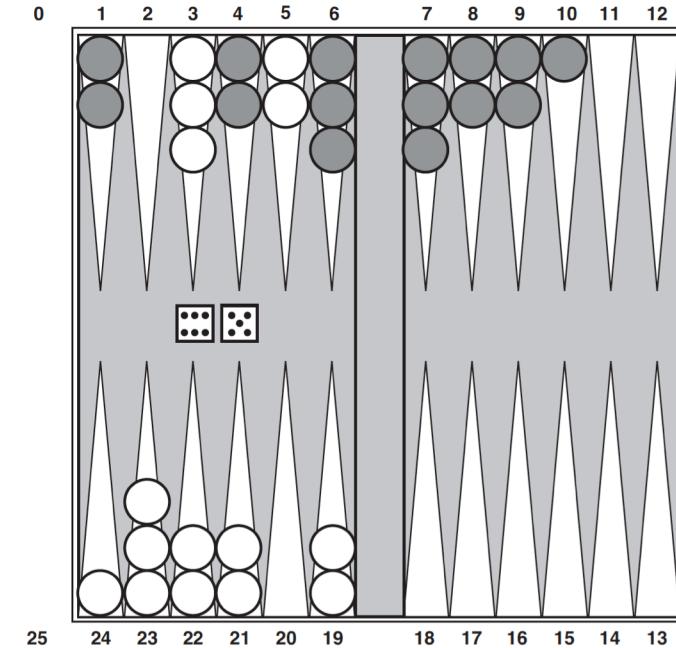
الگوریتم مینی‌مکس، ولی:

محیط (تاس): یک عامل
تصادفی اضافه

تخته نرد



تاس، ضریب انشعاب را زیاد می‌کند



تخته نرد



تاس، ضریب انشعاب را زیاد می‌کند

تعداد حالات ۲ تاس: ۲۱

متوسط تعداد بازی‌های ممکن: ۲۰

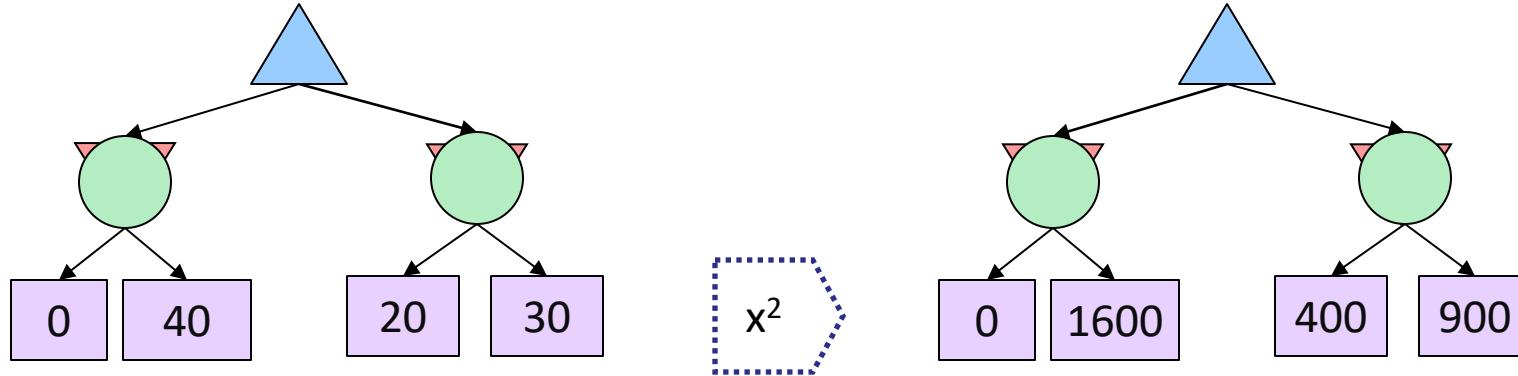
$$\text{عمق} = 20 \times (21 \times 20)^3 = 1.2 \times 10^9$$

با زیاد شدن عمق، به دلیل رشد خیلی زیاد حالات، احتمال رسیدن به یک حالت خاص خیلی کم می‌شود

می‌توان خیلی در عمق فرو نرفت

قهرمان دنیا TDGammon
تابع هزینه مناسب + عمق ۲ + یادگیری تقویتی

سودمندی



برای مینیممکس، اندازه‌ی مقادیر نهایی اهمیتی نداشتند

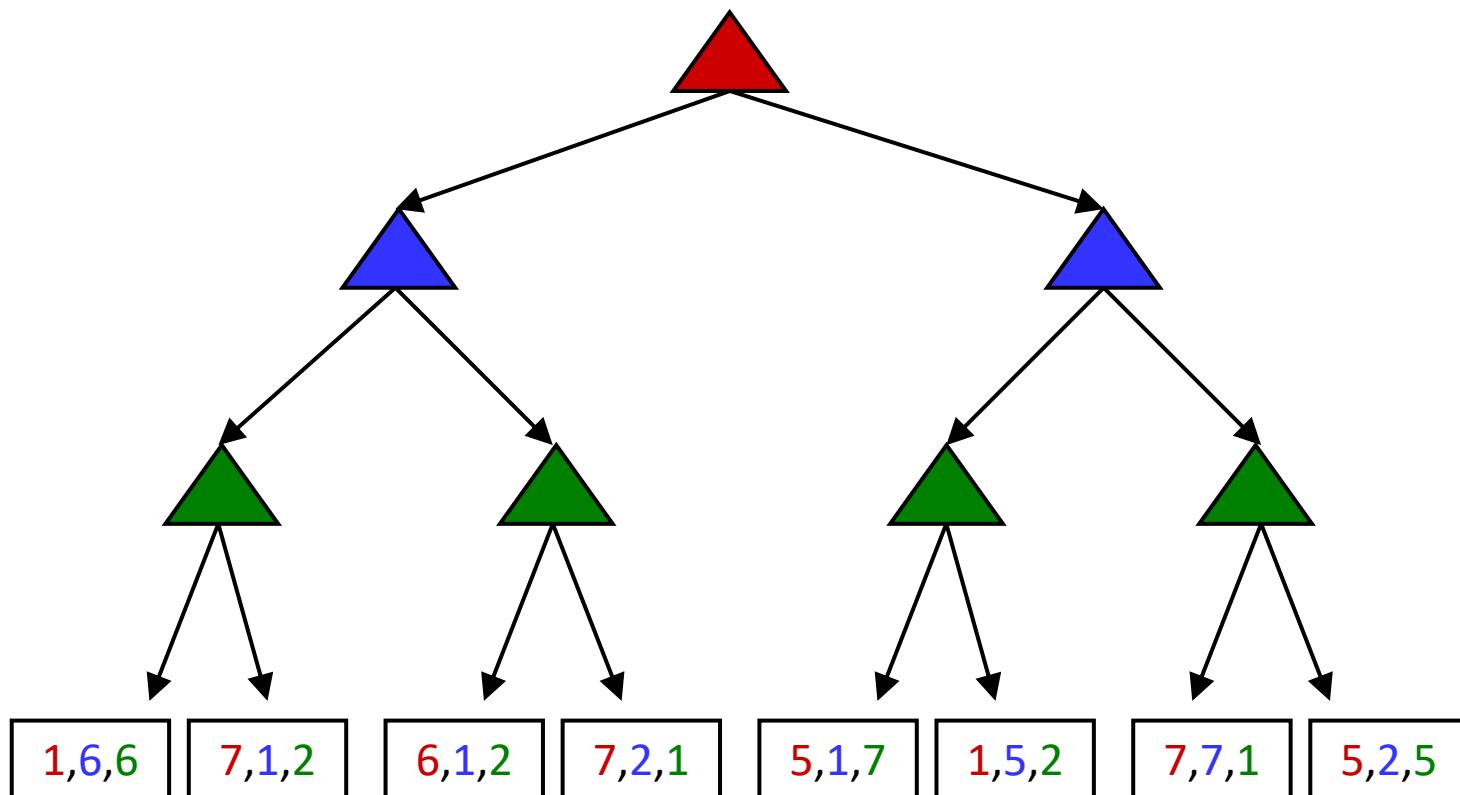
فقط می‌خواستیم انتخاب‌های بهتر را شناسایی کنیم

عدم حساسیت به *monotonic transformations*

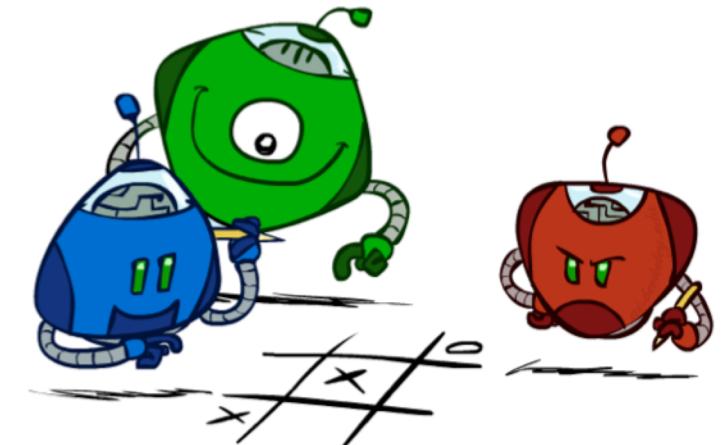
برای اکسپکتیمکس، مقادیر اهمیت دارند!

بازی‌های چند عاملی

در صورت حضور چند بازیکن چطور؟ یا بازی‌هایی که Zero-sum نیستند



برگ‌ها چند مقداری می‌شوند
هر بازیکن سعی در بیشینه کردن امتیاز خود دارد
احتمال مشارکت یا رقابت به وجود می‌آید



State-of-the-art in Games

Chess ($b \approx 35$)

Depth d_1	Strength (Elo)
20	2894
19	2828
18	2761
17	2695
16	2629
15	2563
14	2496
13	2430
12	2364
11	2298
10	2231
9	2165
8	2099
7	2033
6	1966

Houdini



Komodo



Stockfish



State-of-the-art in Games

Checkers ($b < 10$)

Chinook

Plays perfectly (no one can beat it; a perfect player would draw)

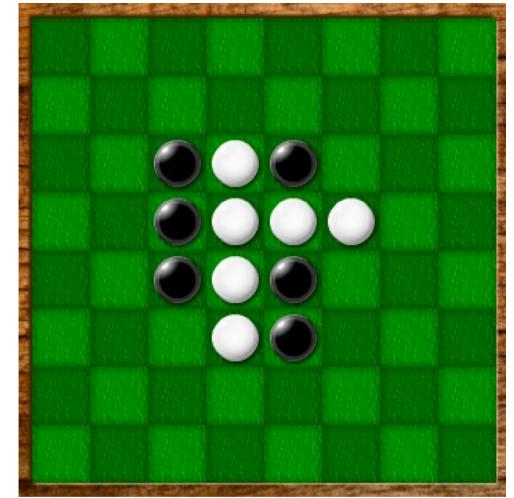
Makes use of a huge endgame database (you can download them!)

# PIECES	# POSITIONS
1	120
2	6,972
3	261,224
4	7,092,774
5	148,688,232
6	2,503,611,964
7	34,779,531,480
8	406,309,208,481
9	4,048,627,642,976
10	34,778,882,769,216

State-of-the-art in Games

Othello (b around 8)

Human champions refuse to compete against computers, which are too good.



Go (b around 300)

AlphaGo (Google)



سؤال؟

