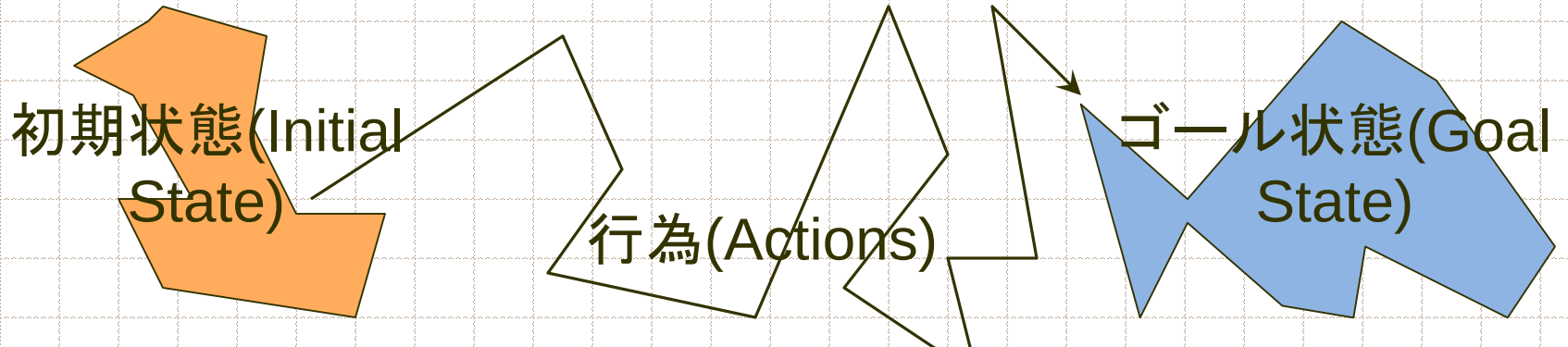
人工知能基礎
第13回

探索による問題解決(6)、情報を持つ探索

ソフトウェア情報学部
David Ramamonjisoa

ゴールを用いるエージェントの構築

- エージェントの環境を状態で表現する(**state**)?
- エージェントのゴールは何か(**goal** to be achieved?)
- 可能な行為は何かある(What are the **actions**?)
- 問題を解決するためにどのような情報が状態と状態遷移に記述するべきか



探索戦略

- 探索戦略選択のための四つの基準
  - 完全性： 解が存在するとき，それを見つけることが保証されているか？
  - 最適性： いくつか異なる解があるとき，戦略は最も良い解を見つけるか？
  - 時間計算量： 解を見つけるまでにどれくらい時間が掛かるか？
  - 空間計算量： 探索を行うためにどのくらいメモリを必要とするか？
- 情報を持つ探索(informed search)，ある探索
  - 現在の状態からゴールに至る順路の中で、最小コストの順路（最適順路）などを考えて効率的に行う.

ゲームプレイング

- 将棋、チェス、チェッカー、囲碁、オセロのような、二人ゲームを考える （ボードゲーム）
- そのようなゲームのコンピュータプログラムは、どうつくれば良いのか。
  - 探索問題の一種。
  - 良さそうな手を打つ。明らかに無駄な手は打たない。
  - 相手がこちらを妨害する。そのような状況で、最善の手を考える。
- どれくらい強いプログラムが作られたか。

情報をもつ探索戦略(Informed search strategies)

- 情報をもつ探索戦略は問題定式化の使用可能情報のみ利用する。
- 敵対的探索(adversarial search)
  - 相手の手番が予測不可能ため、可能な手番をすべて探索する
  - ゲームの時間制限があるため、手番(ゴール)を近似する

目次

- ゲームプレイング
  - ミニマックス法
  - α-β法

人口知能とゲームプログラミングのパイオニア

1949 Shannon paper

1951 Turing paper

1958 Bernstein program

55-60 Simon-Newell program
(α-β McCarthy?)

61 Soviet program

66 – 67 MacHack 6 (MIT AI)

70's NW Chess 4.5

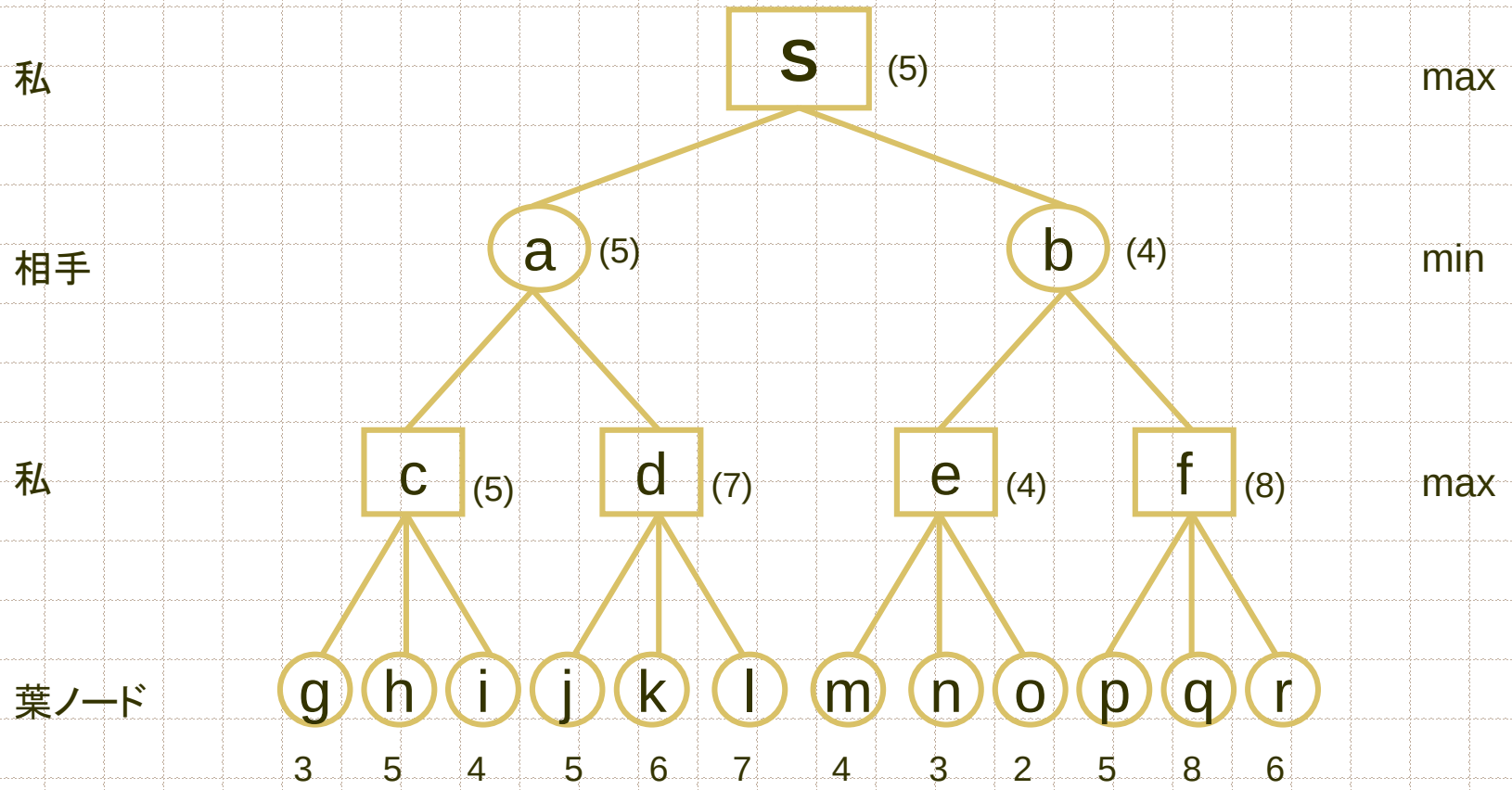
80's Cray Blitz

90's Belle, Hitech, Deep Thought,
Deep Blue

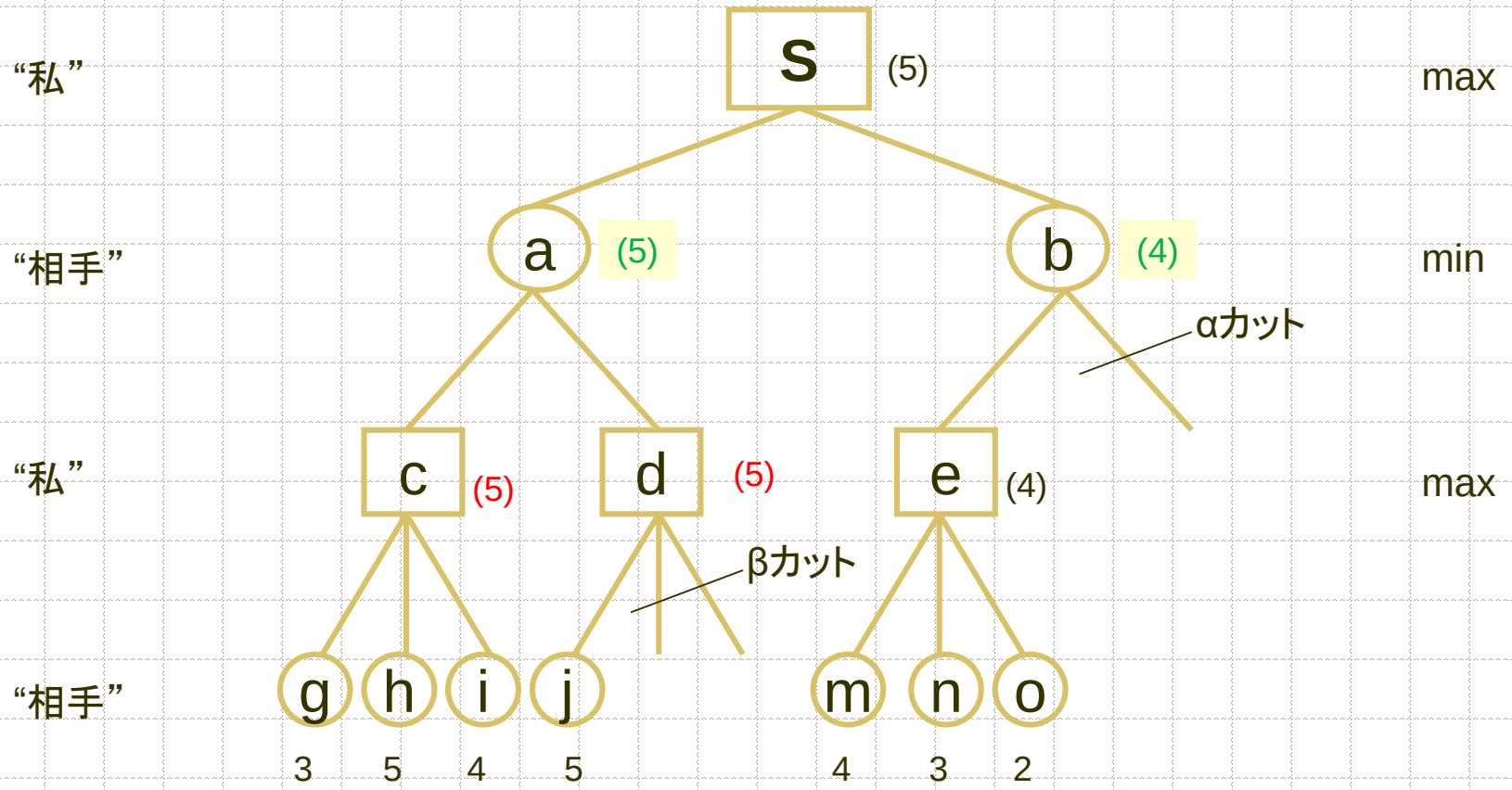
α-β法

- ミニマックス法を改良して、不必要な手の探索を削除する。
  - チェスで、1秒間に1000の局面を評価できるとして、一手の時間を150秒とすると、150000局面読める。
  - チェスの平均分岐数は36なので、このプログラムは、3手あるいは4手しか先読みできない。(36^3=46656, 36^4=1679616)
  - 平均的な人間のプレイヤでも、6から8手は読める。
  - 抜本的な効率の改善策が必要！ α-β法は、それを叶える！
- ゲーム木を深さ優先探索で調べながら、ミニマックス値の上限値、下限値を求め、その範囲を超える探索枝を刈り込む。

ゲーム木(ミニマックス法)



ゲーム木, α-β法による枝刈り



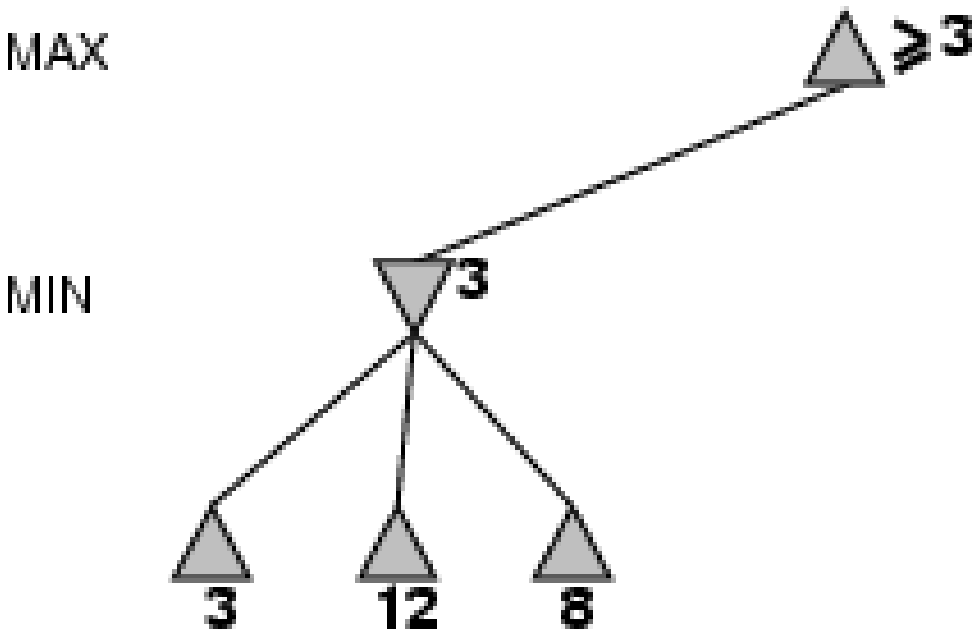
α-β法による枝刈り

- 私が**a**、相手が**c**の手を打ったとすると、私は次に手**h**を打って、評価値**5**を達成する。
- 節点**a**では、私は、評価値の最も低い手を選ぶので、それは**5**を超えることはない（評価値の上限値）。
- 相手は、**c**ではなくて、**d**の手を打つかもしれない。そのとき、私は**j**の手を打つことによって評価値**5**を達成できる。その時点で、**d**の評価値が**5**以上になることがわかる（私は、**d**で最強の手を打つから）。すると、相手は、**d**ではなく、**c**を選んだ方が良いことになる。
- そのため、**d**の下の枝は、それ以上調べなくてよい。
- この枝刈りを**βカット**という。

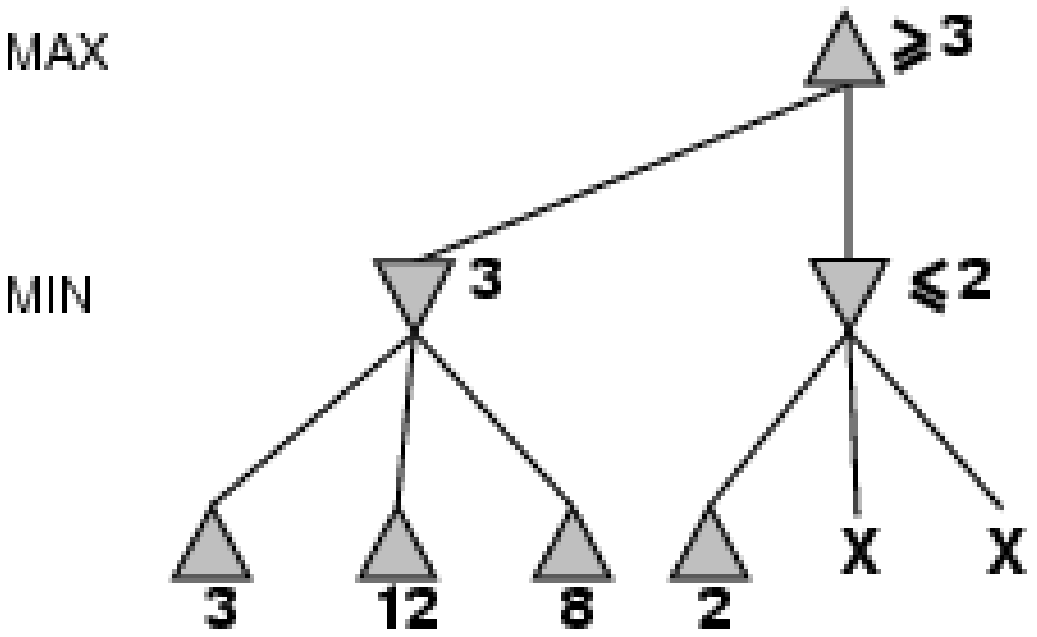
α-β法による枝刈り（つづき）

- 節点**a**の評価値***f(a)***は、**5**に決まり、節点**S**では、私は評価値の最も高い手を選ぶので、それは**5**を下回ることはない（評価値の下限値）。***f(S)≧5***
- つぎに、私が**b**、相手が**e**の手を打ったとすると、私は次に手**m**を打って、評価値**4**を達成する。
- 節点**b**では、相手は、評価値の最も低い手を選ぶので、それは**4**を超えることはない（評価値の上限値）。
- そのとき、私は**a**の手を打つことによって評価値**5**を達成できるので、**b**を選ぶことはない。
- そのため、**b**の下の枝は、それ以上調べなくてよい。
- この枝刈りを**αカット**という。

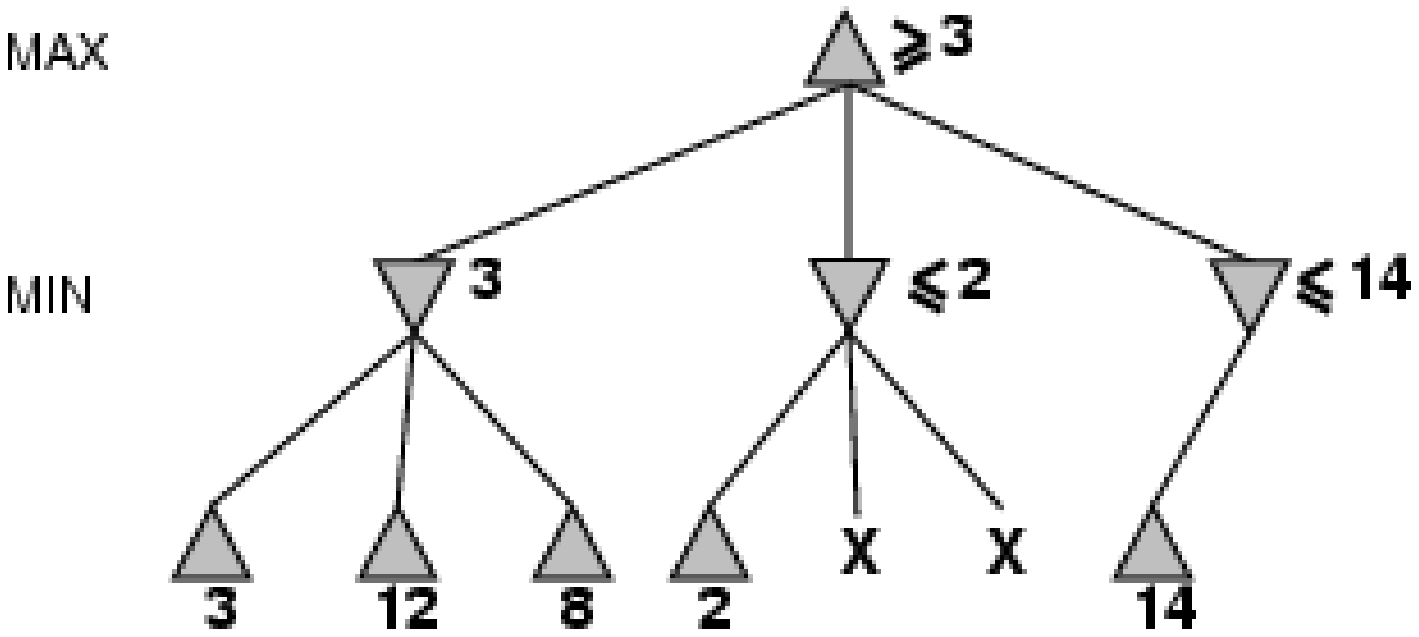
α-β 枝刈りの例2(pruning example)



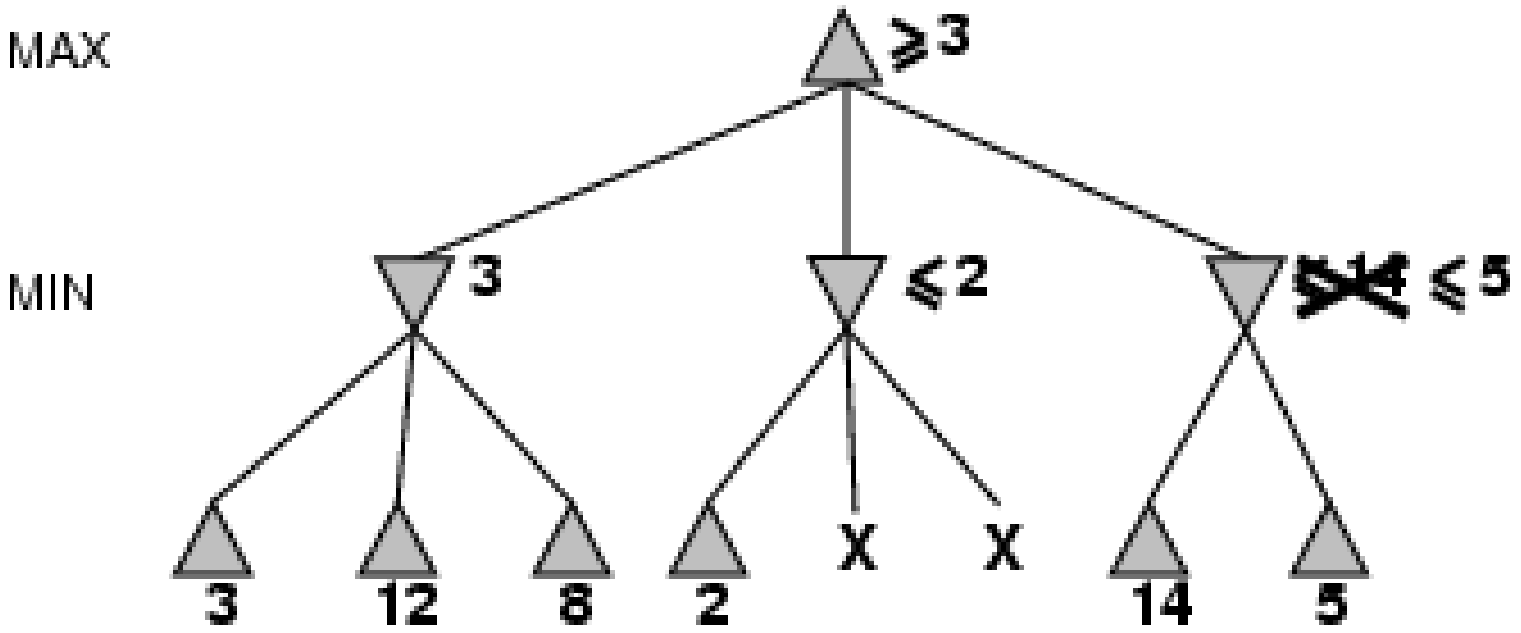
α-β枝刈りの例2(pruning example)



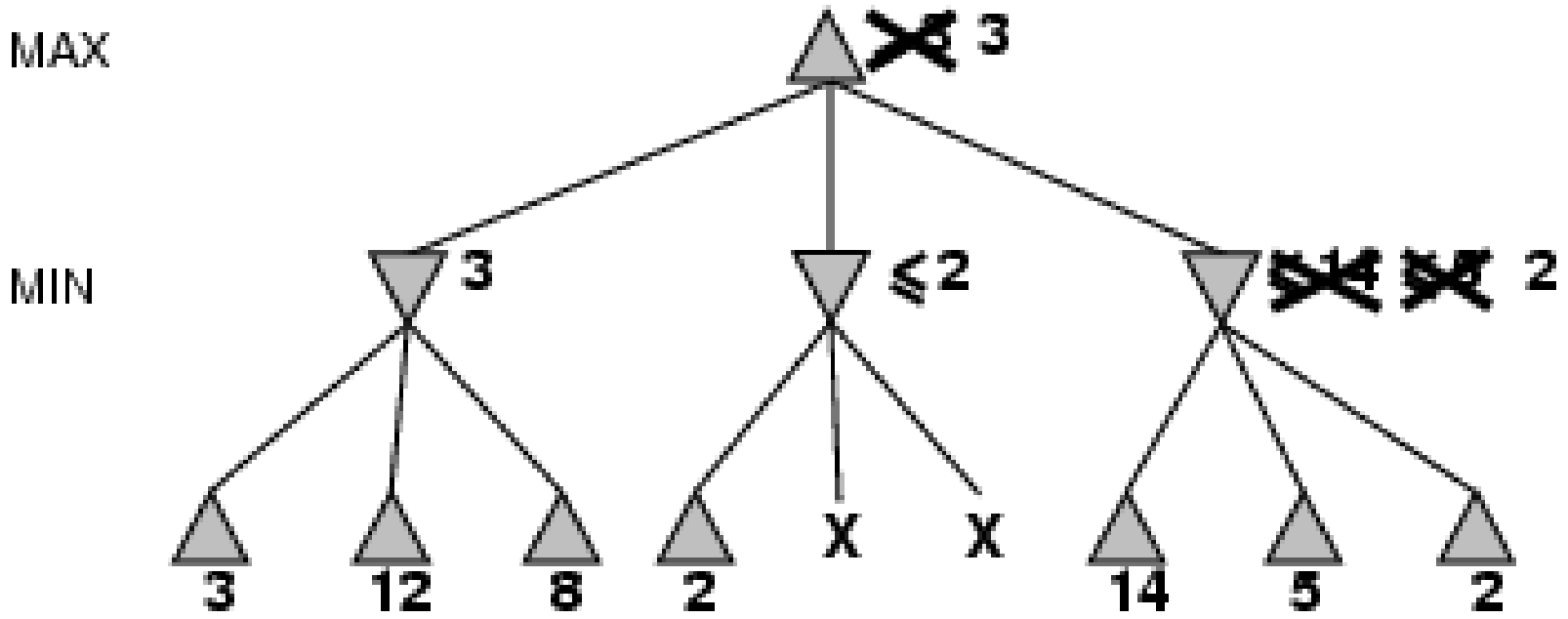
α-β枝刈りの例2(pruning example)



α-β枝刈りの例2(pruning example)



α-β枝刈りの例2(pruning example)



α-β法による効率の改善度

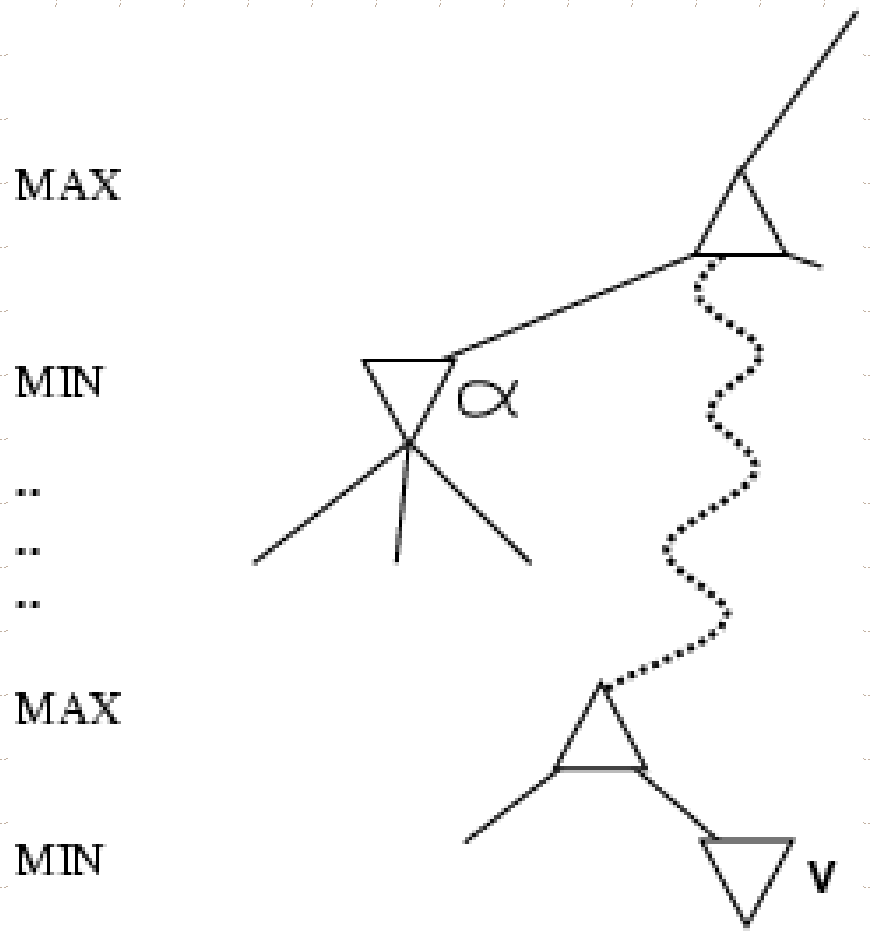
- α-β法による効率の改善は、探索木の子節点の並び方に依存する。
  - 評価値の上下限から外れる枝を先に調べることができれば、その分早く枝刈りできる。
  - 必ず最初の子節点で枝刈りができるとすると、そのときの効率の改善によって、今までの2倍先読みができることが知られている。
  - すなわち、チェスの場合、6から8手の先読みが可能となる。

α-βの性質(Properties)

- 枝刈りは最後の結果に影響ない。(Pruning does not affect final result)
- 先手後手の順序をうまく行えば、枝刈りの効率は良くなる。
- 完璧な順序であるとき、時間計算量は　$O(b^{m/2})$
  → 深さの探索は2倍になる。
- 計算の戦略は重要となる。

α-βと呼ぶのはなぜ?

- α はMAXの経路で最大値である。
-
- もし ν が αより低い値であれば、νを避ける。
  →
  その枝を刈る。
- β はMINにたいして同じ仕組みである。
-



The α-βアルゴリズム

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

  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE($state, \alpha, \beta$) **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*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

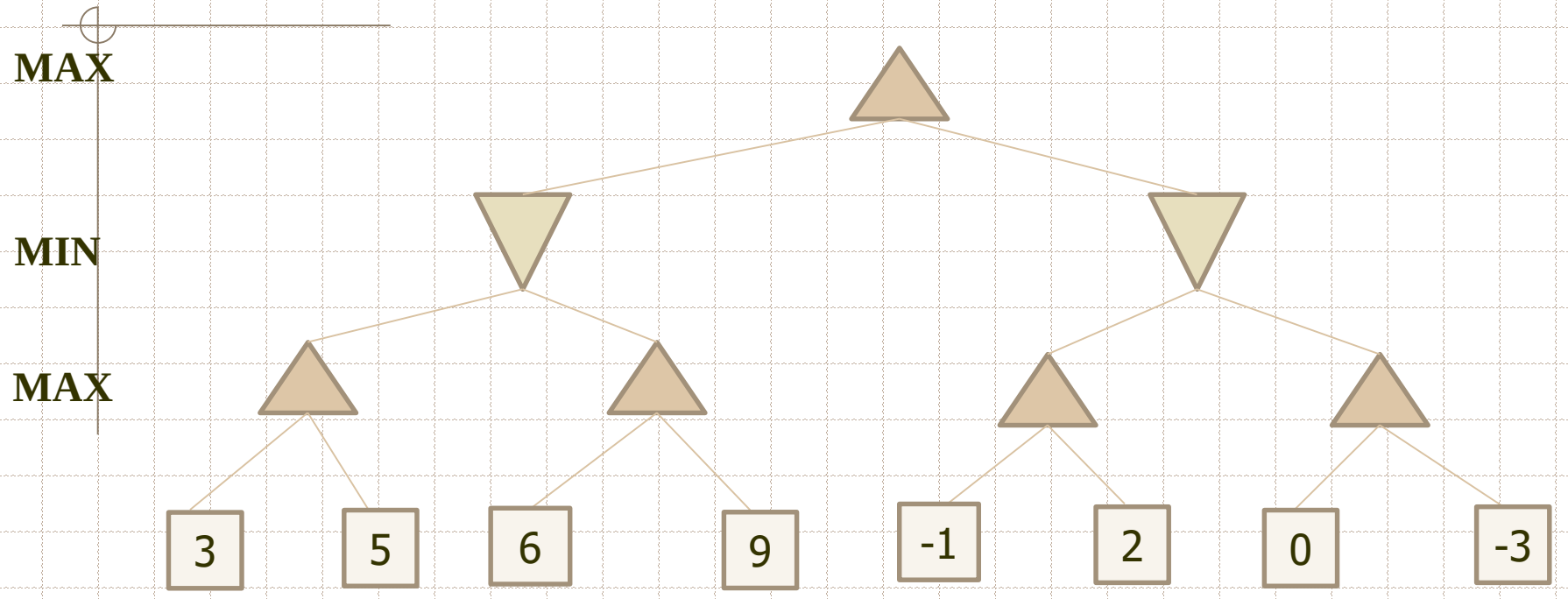
The α-βアルゴリズム

```
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
```

ゲーム探索のまとめ

- 相手がある場合には、自分、敵とも、最強の手を打とうとする。
- 自分の評価値を基準にすれば、自分は評価値を最も高くし、相手は、それを最も低くするように振舞う。
- このような手順をミニマックス法と呼ぶ。
  - ミニマックス法では、実用的な性能が得られない。
- ミニマックス法を改良して、不必要な探索枝を刈り込むアルゴリズムにα-β法がある。
  - 必ず最初の子節点で枝刈りができるとすると、そのときの効率の改善によって、今までの2倍先読みができることが知られている。


MAX
MIN
MAX
3
5
6
9
-1
2
0
-3

アルファ･ベータ法の性質 (Properties of minimax)

- 完全(Complete)? Yes (if tree is finite)
-
- 最適(Optimal)? Yes (against an optimal opponent)
-
- 時間計算量(Time complexity)? $O(2b)^{m/2}$)
-
- 空間計算量(Space complexity)? O(bm) (depth-first exploration)

- チェスのゲーム：b ≈ 35, m ≈100 マスターレベル
  → 正解の解を求めるのは不可能である。
- ミニマックス法の2倍くらいの深さを探索できる

参考ページ

- https://ja.wikipedia.org/wiki/ゲーム木
- https://ja.wikipedia.org/wiki/ミニマックス法
- https://ja.wikipedia.org/wiki/アルファ・ベータ法