《算法设计与分析》

第二章 图与遍历算法

马丙鹏

2023年09月25日

中国科学院大学
University of Chinese Academy of Sciences

第二章 图与遍历算法

- 2.1 图的基本概念和性质
- 2.2 图的遍历算法
- 2.3 双联通图与网络可靠性
- 2.4 对策树

2.4 对策树

- 1. 博弈问题描述
  - 古语有云，世事如棋。
  - 生活中每个人如同棋手，其每一个行为如同在一张看不见的棋盘上布一个子，精明慎重的棋手们相互揣摩、相互牵制，人人争赢，下出诸多精彩纷呈、变化多端的棋局。
  - 博弈论就是研究“棋手们”“出棋”着数中理性化、逻辑化的部分，并将其系统化为一门科学。
  - 换句话说，就是研究个体如何在错综复杂的相互影响中做出最合理的策略。

2.4 对策树

- **1. 博弈问题描述**
  - 博弈论(Game Theory)，亦名对策论、赛局理论，属应用数学的一个分支, 目前在生物学、经济学、国际关系、计算机科学、政治学、军事战略和其他很多学科都有广泛的应用。
  - 博弈论衍生于古老的博弈游戏而得名，如象棋、扑克等。数学家们将具体的问题抽象化，通过建立自完备的逻辑框架、体系研究其规律及变化。

2.4 对策树

- 1. 博弈问题描述
  - 双人完备信息博弈：
    - 一人一步：
      - 两位选手对垒，轮流走步，
    - 双方信息完备：
      - 每一方不仅知道对方已经走过的棋步，而且还能估计出对方未来的走步。
    - 零和：
      - 对弈结果是一方赢，另一方输；或者双方和局。
    - 例如，有象棋、围棋等。
    - 只讨论双人完备信息博弈问题
  - 用对策树的模型来描述对弈局势。

2.4 对策树

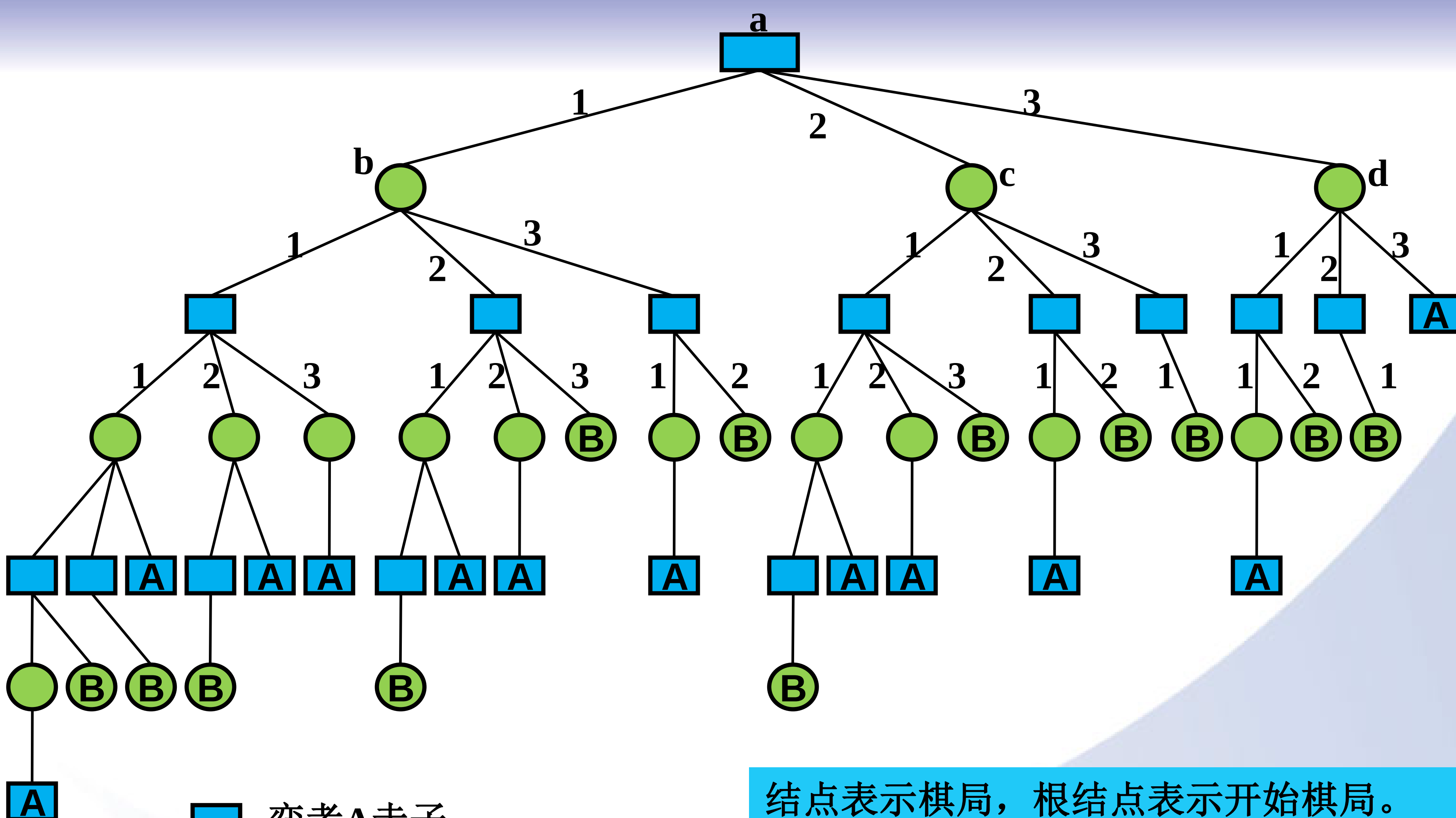
- **1. 博弈问题描述**
  - 拾火柴棍游戏
  - 在盘面上放n支火柴，由弈者A和B两人参加比赛。
  - 规则：两名弈者轮流从盘上取走火柴，每次从盘中取走1，2 或3支火柴为合法着，否则为非法着。
  - 胜负：拿走盘中最后一支火柴的弈者为负，而对方赢。
  - 棋局：以盘中剩下的火柴数来表示当前时刻的棋局。
  - 状态：拾火柴棍游戏在任一时刻的状态由该时刻的棋局和轮到走下一着的弈者一起决定。
  - 终局：表示胜局、负局或和局情况的棋局。
  - 非终止棋局：非终局的其它棋局。
  - 在本游戏中只有一种终局形式，即盘中没有火柴棍子，必有一人胜，另一人负，不会出现和局。

2.4 对策树

■ 2. 数学模型

- 棋局序列C_1，C_2，…，C_m称为有效(棋局)序列，如果：
  - ① C_1是开始棋局；
  - ② C_i不是终止棋局，$i=1$，2，…，m-1；
  - ③ 由C_i按下述规则走到C_{i+1}：
    - ✓若i是奇数，则A走一合法步骤；
    - ✓若i是偶数，则B走一合法步骤。
- 以C_m为终局的一个有效棋局序列C_1，C_2，…，C_m是该游戏的一盘“战例”。
- 有限次博弈游戏的所有可能的实际战例可以用一棵对策树来表示。

一个n=6情况下的拾火柴棍游戏的对策树如下：



弈者A走子

弈者B走子

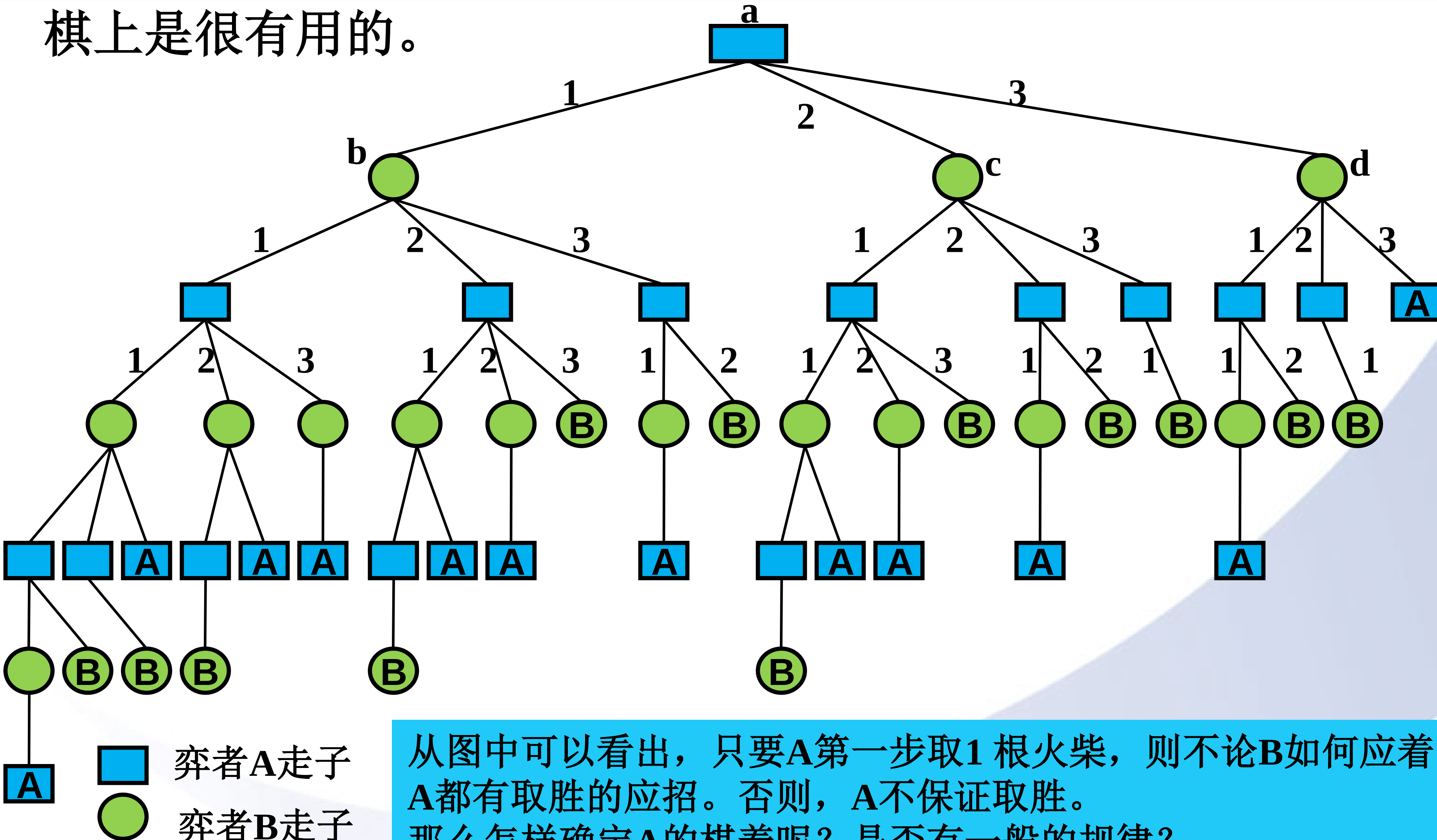
结点表示棋局，根结点表示开始棋局。
方形结点表示轮到A走子的棋局，
圆形结点表示轮到B走子的棋局。
终局用叶子结点表示。
叶子结点中标出在此终局获胜者的名字

2.4 对策树

■ 2. 数学模型

- 本例中，每个弈者至多可以拾取3根火柴，所以对策树中每个结点的度不大于3。
- 对策树的深度表示博弈游戏中最长战例的长度。如上面的树深度为7，代表从开始到结束，每个游戏至多走6步就可结束。
- 棋局变换是通过A或B的作出走子的选择并从上一级走到下一级的完成的。
- 奇数级A走棋着，偶数级B走棋着。

对策树在决定采取什么对策，即确定弈者下一步应走哪步棋上是很有用的。



从图中可以看出，只要A第一步取1 根火柴，则不论B如何应着，A都有取胜的应招。否则，A不保证取胜。
那么怎样确定A的棋着呢？是否有一般的规律？

2.4 对策树

- **3. 估价函数**
  - 定义一个估价函数E(X)，反映弈者在棋局X下获胜机会的大小。
  - 设E(X)是弈者A的估价函数
    - 若棋局X能使A有较大的获胜机会,则E(X)的值就高;
    - 若棋局X使得A有较大的失败可能,则E(X)的值就低。
    - 能使A获胜的终止棋局或不管B如何应着都保证A能获胜的棋局, E(X)取最大值。
    - 对于能保证B取胜的棋局, E(X)取最小值。

2.4 对策树

■ 4. 棋局价值函数

- □对终局定义E(X)，
- □对其它棋局定义棋局价值函数V(X)，
- □V(X)、E(X)给出A在各步走棋参考。
- □如，n=6的拾火柴棍游戏，终止棋局在对策树中是叶子结点，定义终局的估值E(X)如下：

$$E(X)=\begin{cases}1 & X\text{对于}A\text{是胜局}\\ -1 & X\text{对于}A\text{是负局}\end{cases}$$

- □叶子结点的价值函数V(X)=E(X)。

2.4 对策树

■ 4. 棋局价值函数

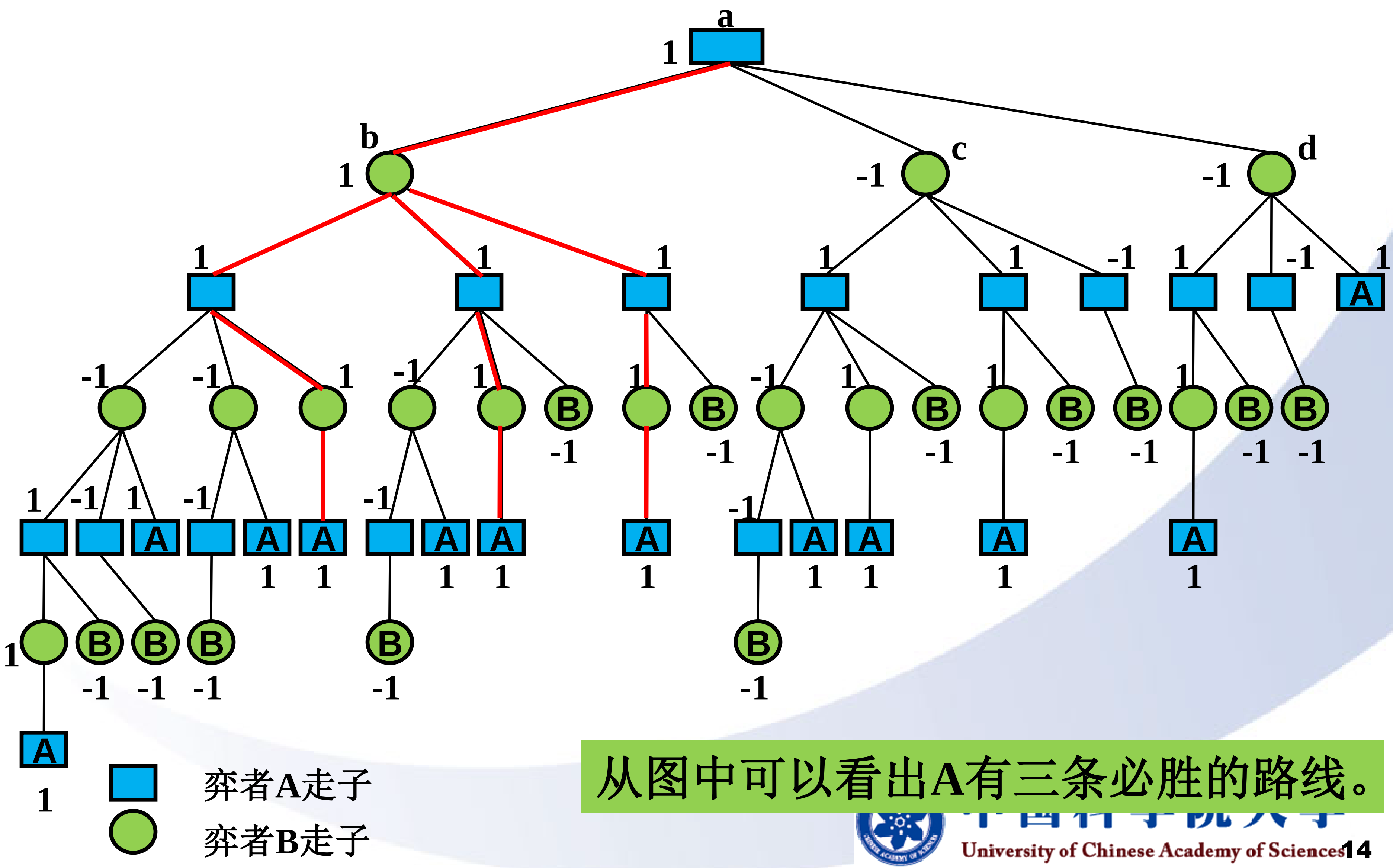
- □对于其它结点，给出相对于A来说能够取胜的价值。一般地，若X不是叶点，且有儿子X_1，X_2，…，X_d，则定义X的价值为：

$$V(X)=\begin{cases}\max\limits_{1\le i\le d}\{V(X_i)\} & \text{若X是方形结点}\\ \min\limits_{1\le i\le d}\{V(X_i)\} & \text{若X是圆形结点}\end{cases}$$

- □这种计算结点价值的方式称为最大最小过程。
- □在n=6的拾火柴棍游戏中，若已经知道结点b, c, d的价值，则a的价值应当取b, c, d价值的最大值。
- □这是因为从棋局a出发，A下一着棋的走法应当是导致其得胜可能性最大的下一步棋局。

中国科学院大学
University of Chinese Academy of Sciences

在计算出对策树各结点的价值后，就很容易看出A(要想取胜)在其所处的各个棋局上应该采取的对策了。



从图中可以看出A有三条必胜的路线。

2.4 对策树

- **4. 棋局价值函数**
  - 对于简单的博弈游戏，可以很容易地画出对策树中的所有结点，给出所有可能的棋局，然后通过计算E(X)、V(X)就可以估算出每个棋局的价值，并很好地走棋。
  - 如上面n=6的拾火柴棍游戏，对策树给出了所有可能的棋局。
  - 从对策树根结点出发到达叶结点的每条路径恰是一个战例。
  - 对于较大规模的博弈，一般很难列出所有可能的棋局。
    - 国际象棋的完整对策树的结点数据估计将达到10^{100}，即每秒能生成10^{11}个结点，也需要10^{80}年以上的时间才能生成完整的对策树。

2.4 对策树

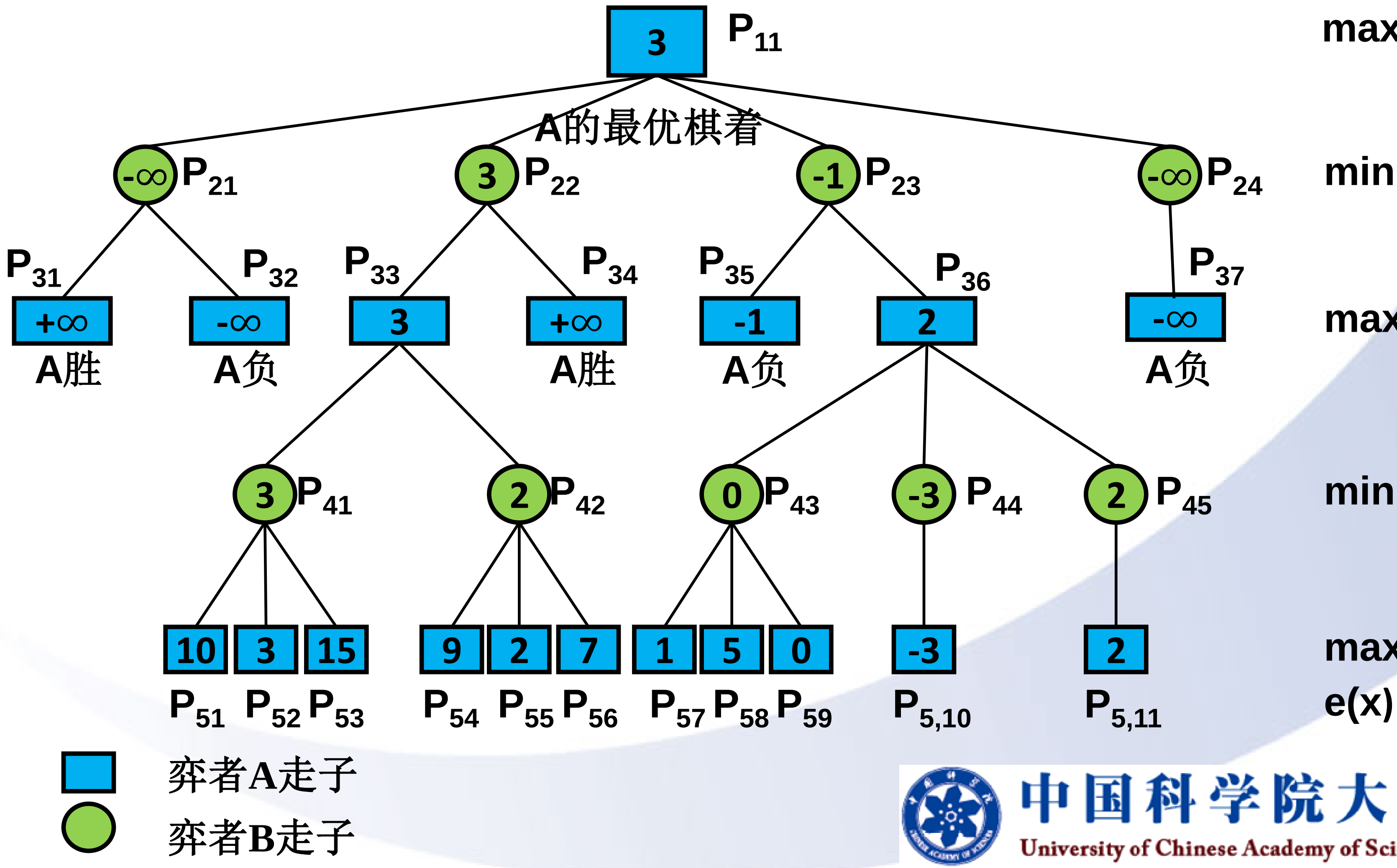
■ 4. 棋局价值函数

- 需要对搜索空间进行两遍分析，第一遍生成全部搜索树，第二遍估计代价，效率低。
- 对于具有大规模对策树的博弈，该如何有效求解？
  - 一般不采取考察其完整对策树的办法来确定弈者的对策，
  - 通常采用向前预测几步，然后决定下一步的策略。而向前预测几步的局面可以用部分对策树表示出来。

2.4 对策树

■ 4. 棋局价值函数

一盘假想游戏
的部分对策树



2.4 对策树

■ 4. 棋局价值函数

- 用估价函数E(X)估算对策树(实际是子树)的叶子结点的值，
- 再根据价值函数V(X)的最大最小计算过程逐一确定其它结点的价值。
- 最后确定下一步该走什么样的棋着。
- 注：使用产生部分对策树确定下一步棋着的方法所导致的棋局的质量将取决于这两名弈者所采用的估价函数的性能和通过最大最小过程来确定当前棋局的价值V(X)所使用算法的好坏。

2.4 对策树

■ **4. 棋局价值函数**

□ 在V(X)的表达式中，

$$V(X)=\begin{cases}\max\limits_{1\le i\le d}\{V(X_i)\} & \text{若X是方形结点}\\ \min\limits_{1\le i\le d}\{V(X_i)\} & \text{若X是圆形结点}\end{cases}$$

其中， X_1， X_2， …， X_d是X的儿子结点。

□ 需要区分弈者A、B，以确定是取最大值还是取最小值。

□ 通过改变弈者B价值的符号的方法，将V(X)公式中取最小值的计算改成取最大值(max)的计算，从而可以以一种简单的递归过程完成V(X)的计算。

2.4 对策树

■ 4. 棋局价值函数

- 改写后的V(X)为V'(X)：

$$V'(X) = \begin{cases} e(x) & \text{若X是所生成子树的叶子结点} \\ \max\limits_{1\le i\le d}\{-V'(X_i)\} & \text{若X不是所生成子树的叶子结点} \end{cases}$$

- 当X是叶子结点时，
  - ➢若X是A走棋的位置，则e(X)=E(X);
  - ➢若是B走棋的位置，则e(X)=-E(X)。
  - ➢ 注：E(X)是相对于A的估价函数。
- 上式在奇数级结点和偶数级结点之间转换价值的符号，从而可以方便地求的各结点的价值。

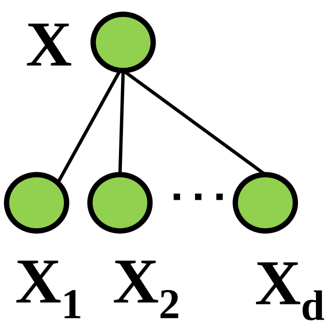
2.4 对策树

■ 5. 求取V'(X)的递归算法

- 通过对以X为根、高为h的对策(子)树的后根次序遍历，可以产生求取V'(X)的递归算法。

算法2.12 对策树的后根次序求值算法

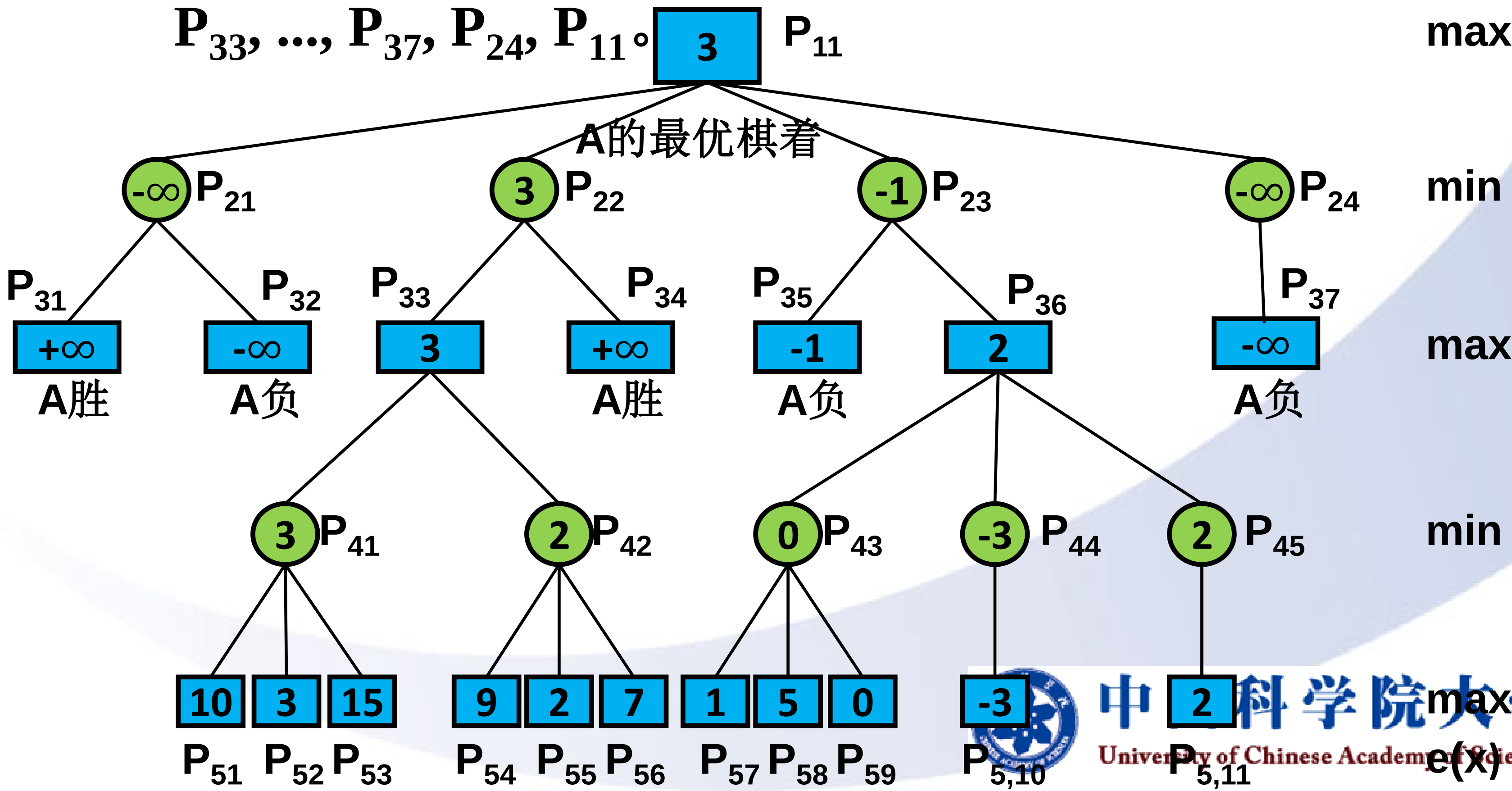
```
procedure VE(X, h)
//通过至多向前看h着棋计算V'(X)，弈者A的估价函数是e(X)。假定由任一不是终
  局的棋局X开始，此棋局的合法棋着只允许将棋局X转换成棋局X1, X2, …, Xd.
  if X 是终局或h=0 then return e(X) endif
  ans← -VE(X1, h-1); //遍历第一棵子树
  for i←2 to d do
     ans←max(ans, -VE(Xi, h-1));
  repeat
  return ans;
end VE
```



2.4 对策树

■ 5. 求取V'(X)的递归算法

在上述部分对策树上调用VE(P_{11}, 4),各棋局价值值的确定次序为: P_{31}, P_{32}, P_{21}, P_{51}, P_{52}, P_{53}, P_{41}, P_{54}, P_{55}, P_{56}, P_{42}, P_{33}, ..., P_{37}, P_{24}, P_{11}。



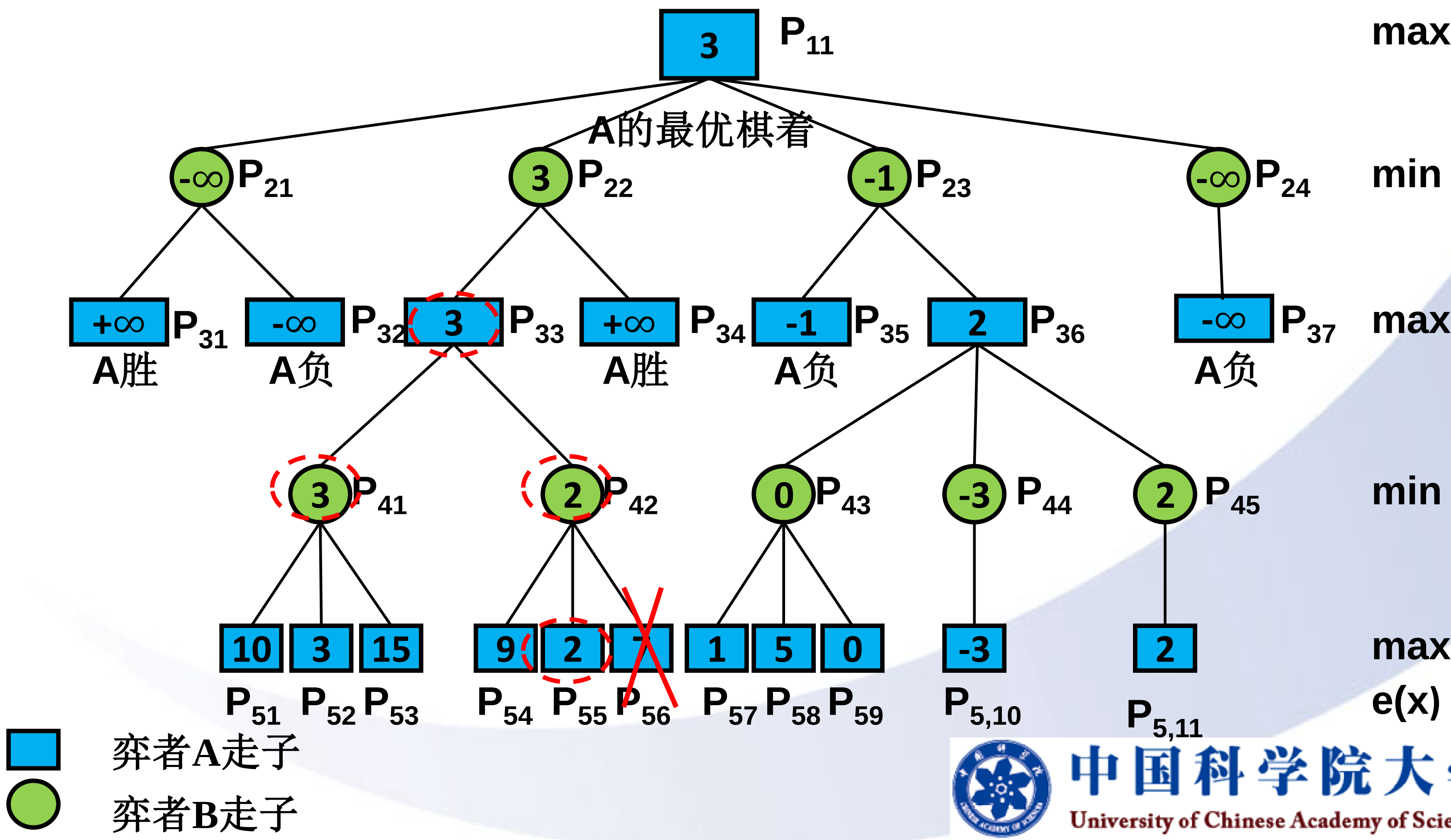
2.4 对策树

- **5. 求取V'(X)的递归算法**
  - 求取棋局P_{11}的$V'(P_{11})$值的目的在于让A决定下一步走棋的对策。上述计算过程对从P_{11}出发的、向前展望h步的所有节点都计算V'(X)，然后求出$V'(P_{11})$的值。
  - 即使不生成所有的结点也可以精确地计算出$V'(P_{11})$:
    - 在算出$V(P_{41})=3$ 后，就知道$V(P_{33})$至少是3，因为P_{33}是求最大值的位置。
    - 在计算进行到P_{55}时，算出$V(P_{55})=2$，则$V(P_{42})$的值必小于3，因为P_{42}是求最小值的位置，
    - 一旦$V(P_{55})=2$，不管P_{42}的其余儿子(如P_{56})的值等于多少，$V(P_{42})$都不可能大于3。
    - 所以$V(P_{56})$根本没必要计算，分枝P_{56}可以被“剪去”。

2.4 对策树

■ 5. 求取V'(X)的递归算法：

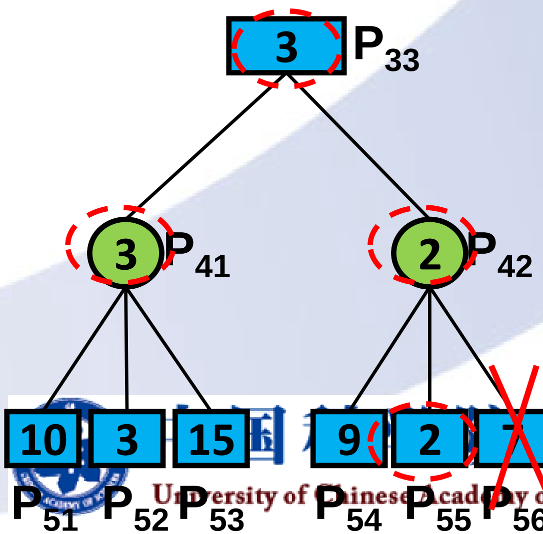
一盘假想游戏
的部分对策树



2.4 对策树

■ 6. α截断规则：

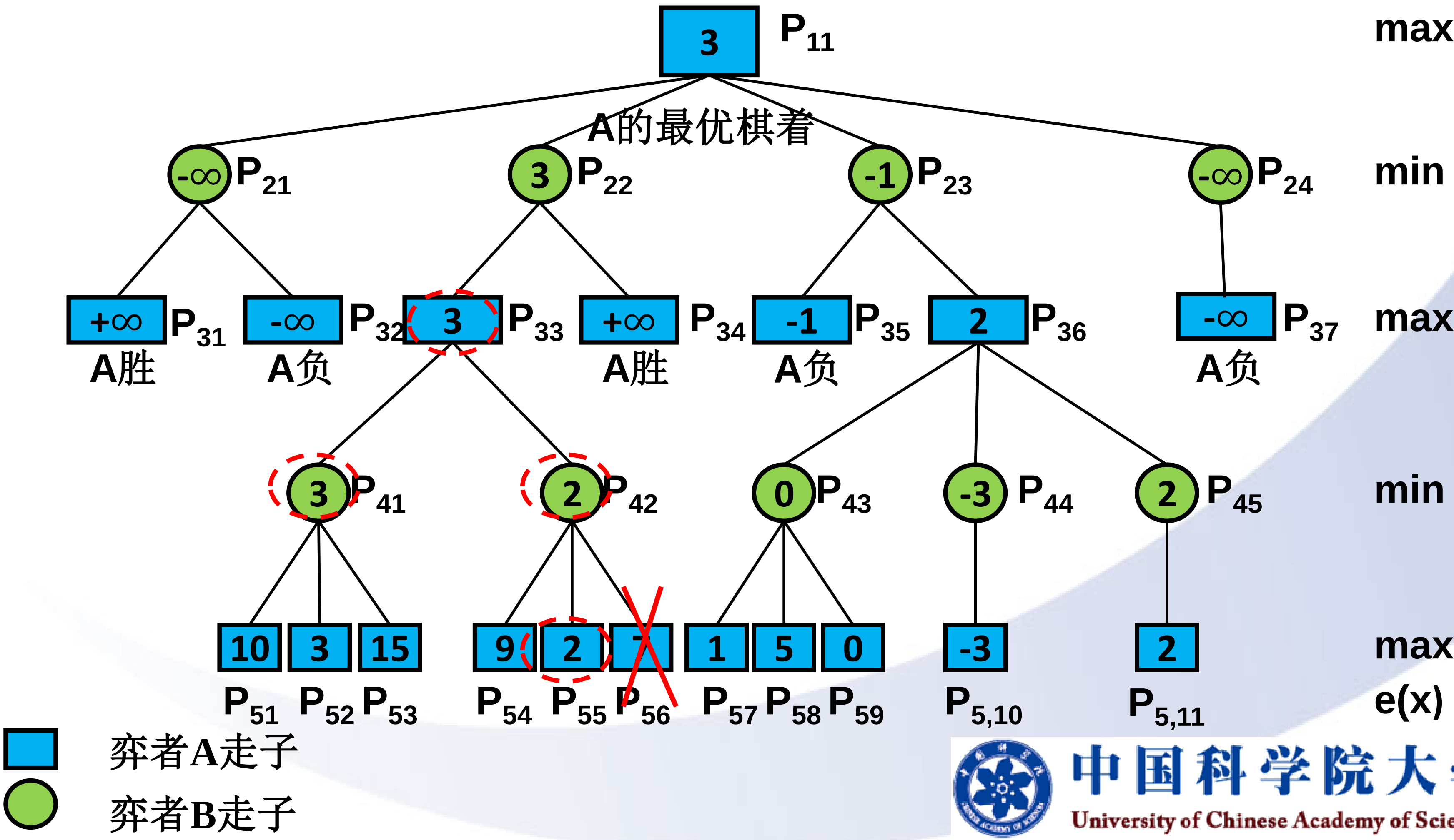
- □为求最大值的位置定义一个α值，它是该位置迄今为止最大的可能值。则α截断规则如下：
  - ➢如果一个求最小值位置的值被判断为小于或等于它的父亲的α值，那么可以停止生成这个求最小值位置其余儿子的值。
  - ➢在这种规则下终止生成结点值的行为称为α截断。
- □如上例，一旦确定$V(P_{41})=3$，则P_{33}的α值就变成3，而$V(P_{55}) \leq P_{33}$的α值，所以就不用再去生成P_{56}的值。



2.4 对策树

■ 6. α截断规则：

一盘假想游戏
的部分对策树



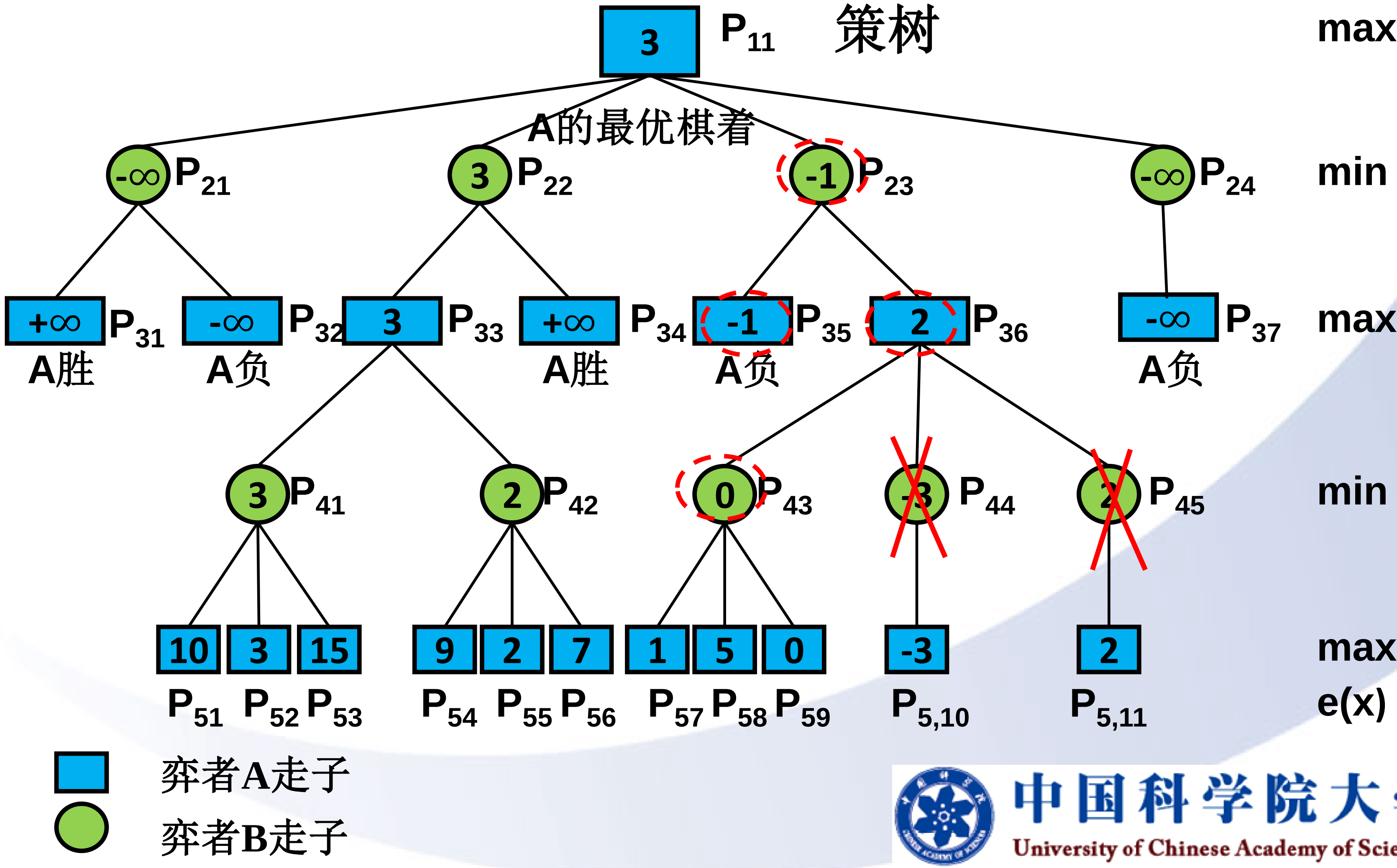
2.4 对策树

- 7. β截断规则：
  - 同理，存在一个对最小值的截断规则，称为β截断
  - 实例分析：
    - 在算出$V(P_{35})=-1$ 后，就知道$V(P_{23})$至多是-1，因为P_{23}是求最小值的位置。
    - 在计算进行到P_{43}时，算出$V(P_{43})=0$，则$V(P_{36})$的值必大于-1，因为P_{36}是求最大值的位置。
    - 因此，一旦$V(P_{43})=0$，不管P_{36}的其余儿子(如P_{44}、P_{45})的值等于多少，$V(P_{36})$都不可能小于-1。
    - 所以$V(P_{44})$、$V(P_{45})$根本没必要计算，分枝P_{44}、P_{45}可以被“剪去”。

2.4 对策树

■ 7. β截断规则：

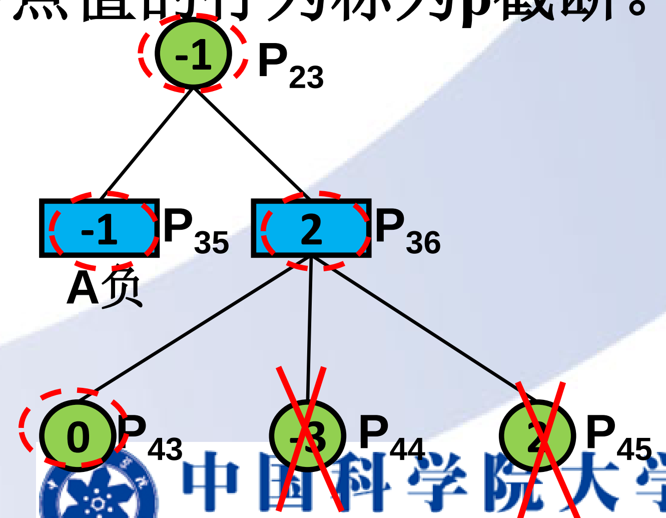
一盘假想游戏的部分2.4 对策树



2.4 对策树

■ **7. β截断规则：**

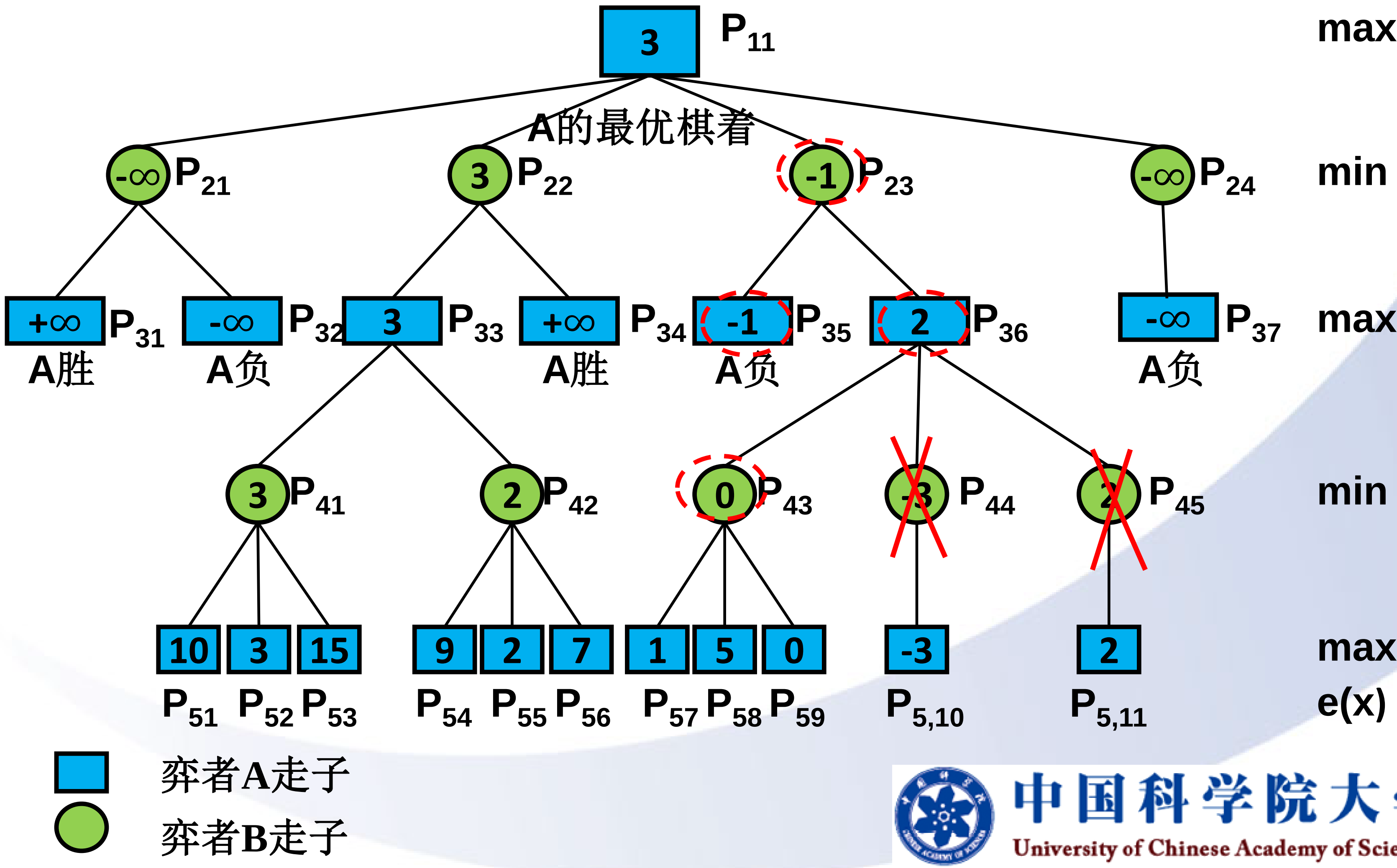
- 为求最小值的位置定义一个β值，它是该位置迄今为止最小的可能值。则β截断规则如下：
  - 如果一个求最大值位置的值被判断为大于或等于它的父亲的β值，那么可以停止生成这个求最大值位置其余儿子的值。
  - 在这种规则下终止生成结点值的行为称为β截断。
- 如上例，一旦确定$V(P_{35})=-1$，则P_{23}的β值就变成-1，而$V(P_{43}) \geq P_{23}$的β值，所以就不用再去生成P_{44}、P_{45}的值。



2.4 对策树

■ 7. β截断规则：

一盘假想游戏
的部分对策树



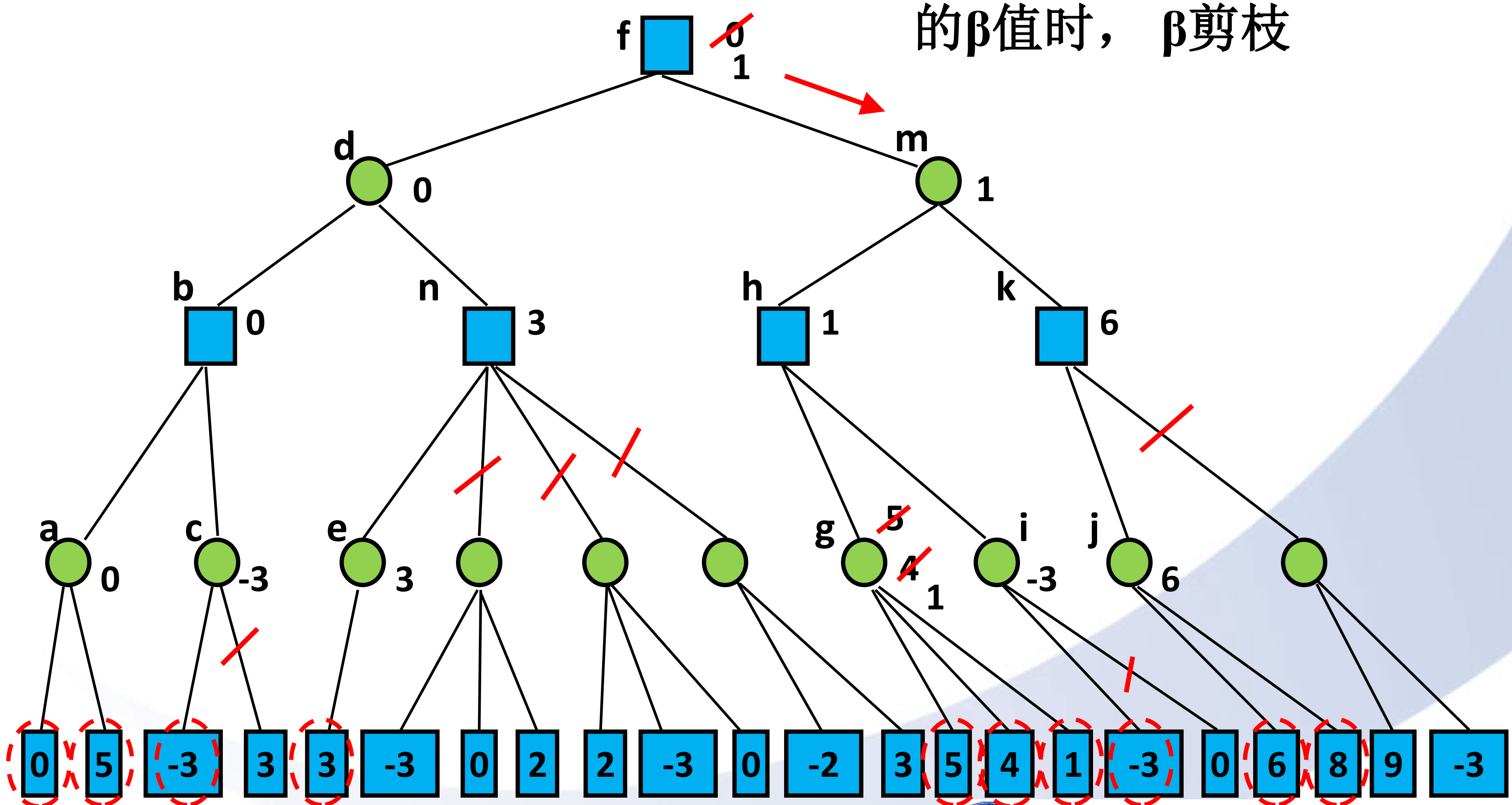
2.4 对策树

- **8. α-β截断规则：**
  - 上述两条规则合在一起称为α-β截断规则。
  - 极大节点的下界为α。
  - 极小节点的上界为β。
  - 剪枝的条件：
    - 后辈节点的β值≤祖先节点的α值时，α剪枝
    - 后辈节点的α值≥祖先节点的β值时， β剪枝
  - 简记为：
    - 极小≤极大， α剪枝
    - 极大≥极小， β剪枝

2.4 对策树

后辈节点的β值≤祖先节点的α值时，α剪枝

后辈节点的α值≥祖先节点的β值时，β剪枝

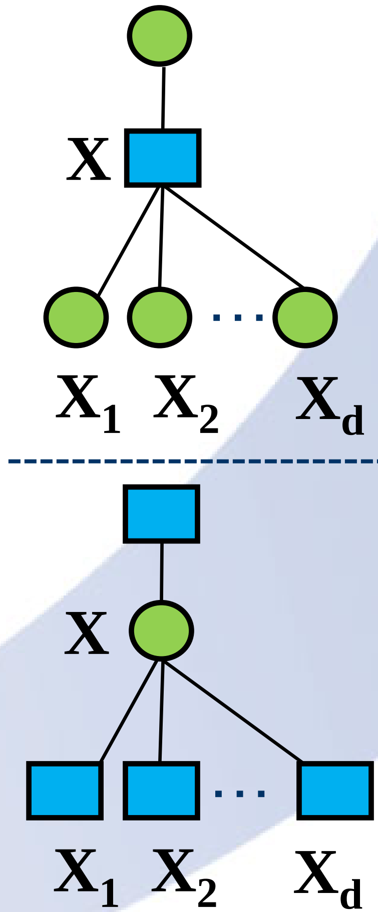


中国科学院大学
University of Chinese Academy of Sciences

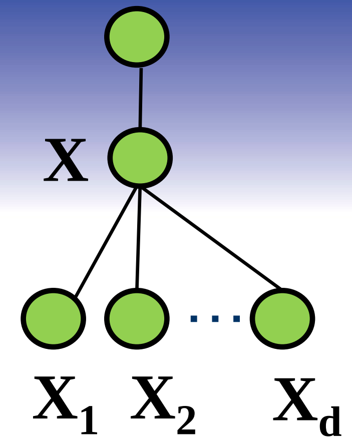
2.4 对策树

■ 8. α-β截断规则：

- 按照V'(X)的定义，由于在最小值位置采用改变符号的方法把最小值计算转变成最大值计算，所以α-β截断规则改写如下：
  - 为每个位置定义一个B值，它是该位置迄今为止最大的可能值，则α-β截断规则如下：
  - 对于任一结点X，设B是该结点父亲的B值且D=-B，那么如果X的值被判断为大于或等于D，则可以停止生成X的其它儿子。



2.4 对策树



■ 8. α-β截断规则：

算法2.12 使用α−β截断规则的后根次序求值算法

procedure VEB(X, h, D)

//至多向前看h着棋，使用α−β截断规则和V'(X)公式计算。弈者A的估价函数是e(X)。

//假定由任一不是终局的棋局X 开始，合法棋着只允许将棋局X转换成棋局X_1，X_2，…，X_d.

//D是in型变量，代表X的父亲目前所知的最大可能值的相反数，即D=-B。

if X是终局或 h =0 **then return** e(X) **endif**

ans ← -VEB(X_1, h-1, ∞)　　X还没有B值

for i ← 2 **to** d **do**

if ans ≥ D **then return** ans **endif** //使用α−β截断规则

ans←max(ans, -VEB(X_i, h-1, -ans));　　X已有B值

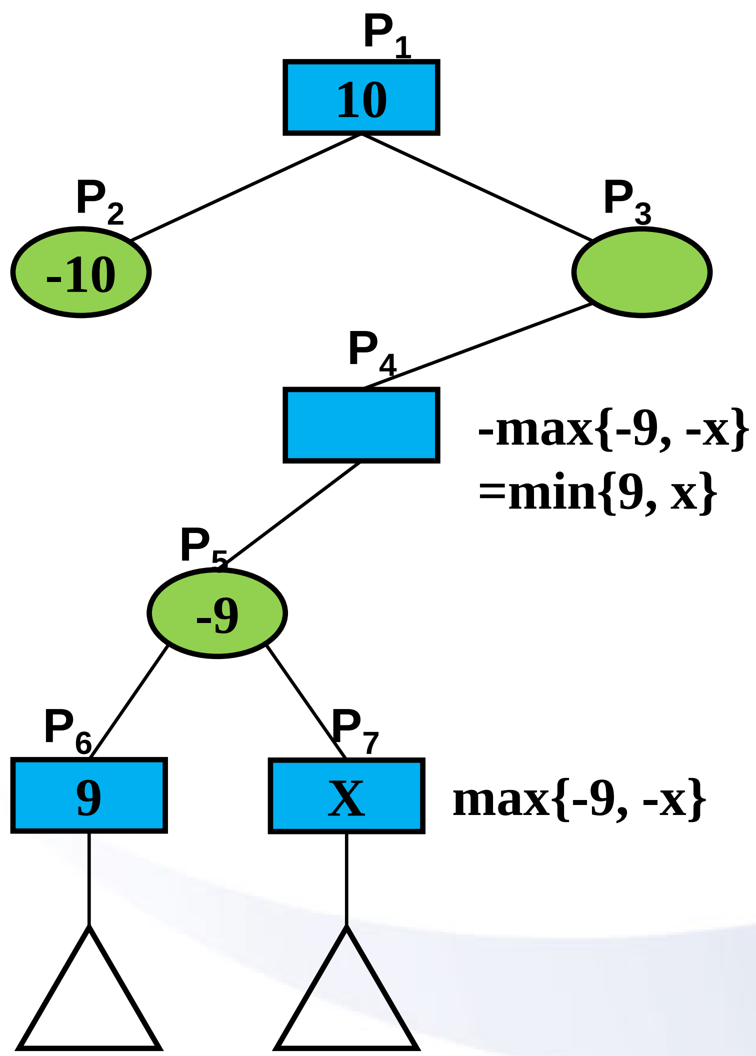
repeat

return ans;

end VEB

如果A从Y处走子，则初次调用的方式为VEB(Y, h, ∞)

2.4 对策树



实例分析

分析VEB的执行过程：

初始调用VEB(P_1，h_{P1}，∞)，h是树的深度。其后调用依次为：

VEB(P_2, h_{p1}-1, ∞);//P_1的左分枝

//ans_{P2}=-10

//B_{P1}=10

VEB(P_3, h_{p1}-1, 10);

VEB(P_4, h_{P3}-1, ∞);//P_3的左分枝

VEB(P_5, h_{P4}-1, ∞);//P_4的左分枝

VEB(P_6, h_{P5}-1, ∞);//P_5的左分枝

//ans_{P6}=9

//B_{P5}=-9

VEB(P_7, h_{P5}-1, -9);

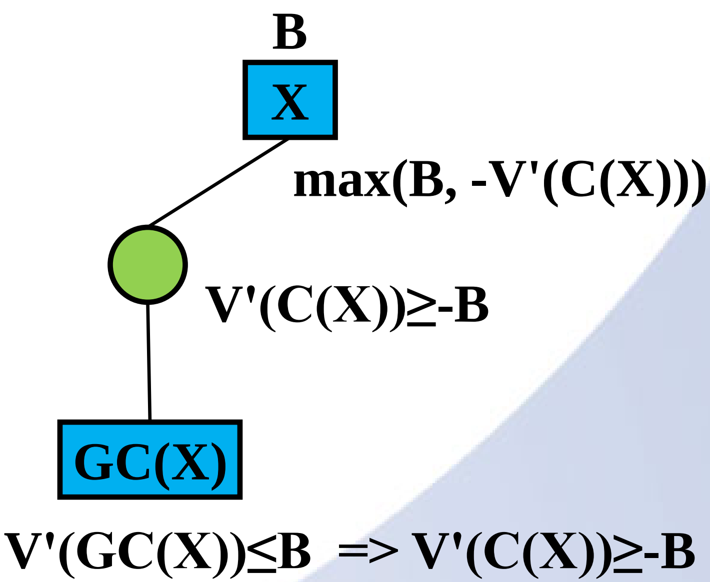
...

VEB算法的改进：

对结点X而言，为了影响结点X的值，可以利用X的B值估计X的孙子们的值所应有的下界。如图所示：

对X的值B，如果V'(GC(X))≤B，那么V'(C(X))≥-B，而若V'(C(X))≥-B，X值的计算是max(B，-V'(C(X)))=B。不会受到任何影响。

所以，除非V'(GC(X))＞B，否则它就不能影响到V'(X)。故B可以作为GC(X)值应有的下界。



把下界的概念加入算法VEB得到算法AB，其中参数LB是X值应有的下界。

2.4 对策树

■ 8. α-β截断规则：

procedure AB(X, h, LB, D)

// LB是X应有的价值下界。

//通过至多向前看h着棋，使用α−β 截断规则和V'(X)公式计算。弈者A的估价函数是e(X)。

//假定由任一不是终局的棋局X开始，合法棋着只允许将棋局X转换成棋局$X_1, X_2, \ldots, X_d$.

if X 是终局或 h=0 **then return** e(X) **endif**

ans = LB // X当前的价值下界。

for i ← 1 **to** d **do**

if ans ≥ D **then return** ans **endif** //使用α−β截断规则

ans = max(ans, -AB(X_i, h-1, -D, -ans))

repeat

return ans;

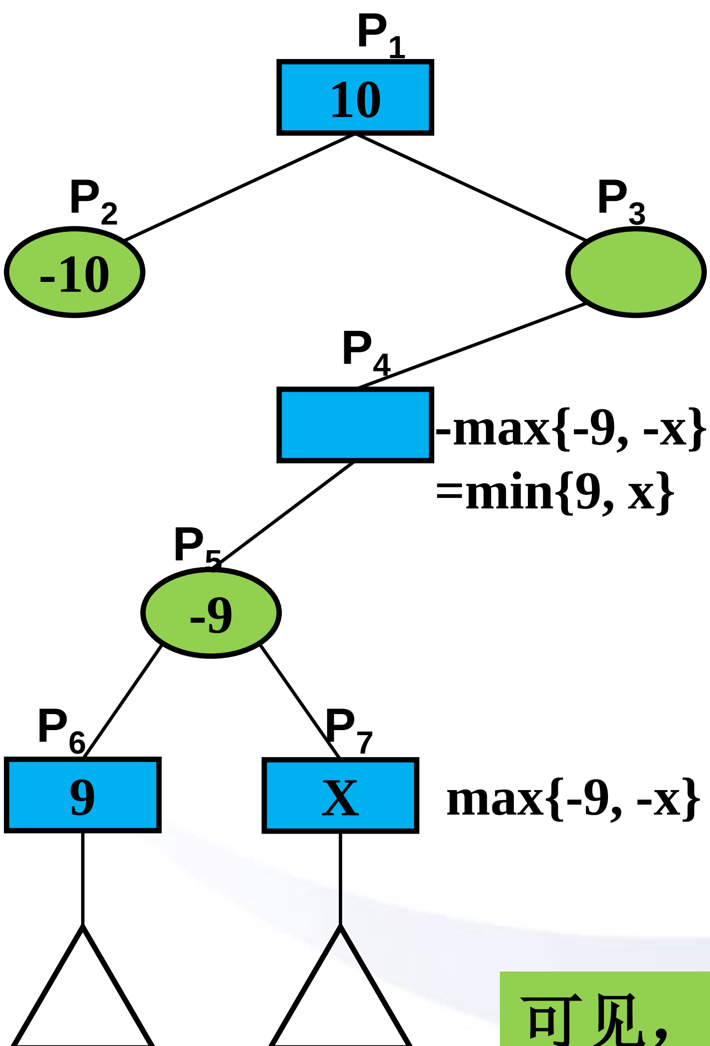
end AB

初次调用的方式：

VEB(Y, h, -∞, ∞)

2.4 对策树

实例分析



分析AB的执行过程：

初始调用AB(P_1, h_{P1}, $-\infty$, ∞)，h是树的深度。其后调用依次为：

//ans=$-\infty$, D=∞

P_1的左分枝：AB(P_2, h_{P1}-1, $-\infty$, ∞);

//ans=$-\infty$, D=∞

//返回时有，ans=max($-\infty$, 10)=10

P_1的右分枝：AB(P_3, h_{P1}-1, $-\infty$, -10);

//ans=$-\infty$, D=-10

P_3的左分枝：AB(P_4, h_{P3}-1, 10, ∞);

//ans=10, D=∞

P_4的左分枝：AB(P_5, h_{P4}-1, $-\infty$, -10);

//ans=$-\infty$, D=-10

P_5的左分枝：AB(P_6, h_{P5}-1, 10, ∞)=9

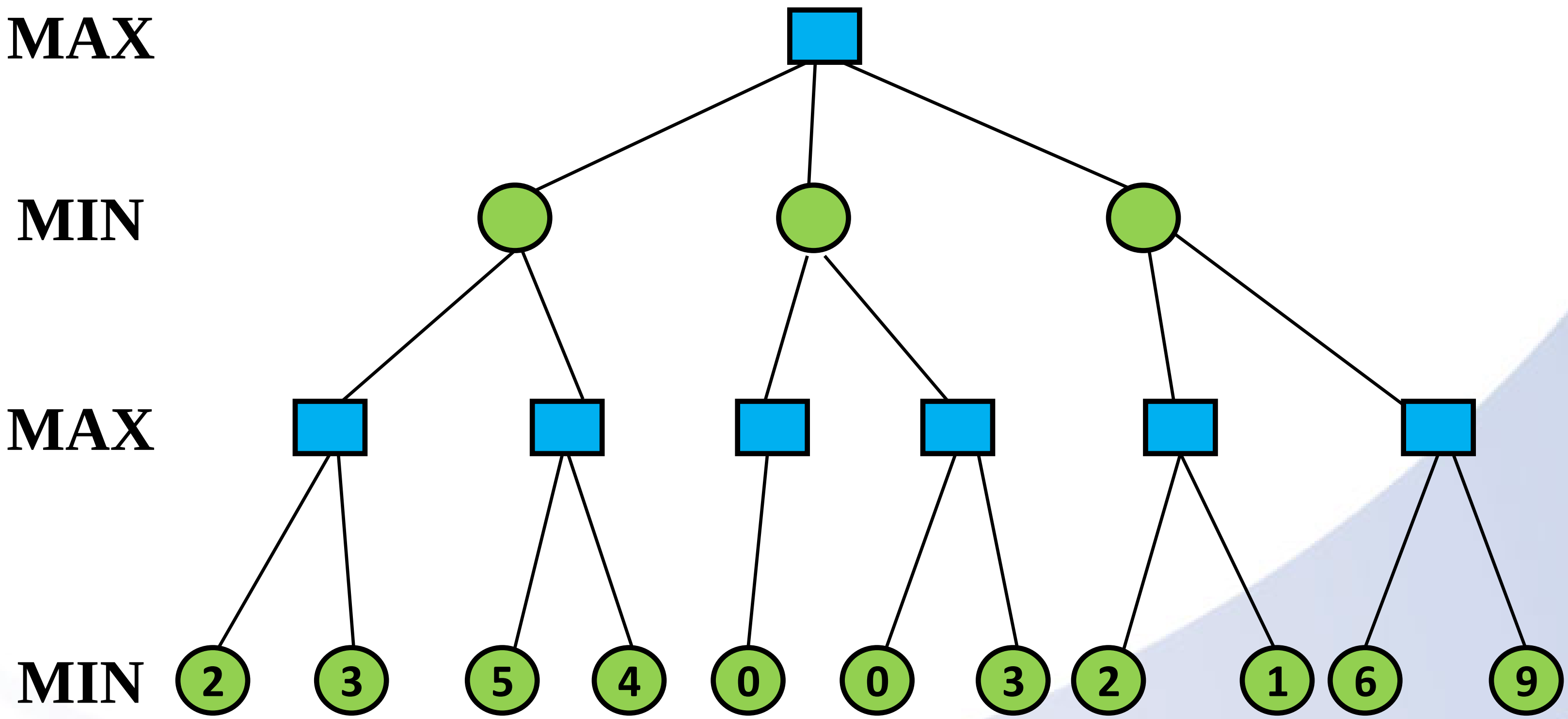
//ans_{P5}=max($-\infty$, -9)=-9

之后，因为ans_{P5}=-9≥D，所以直接return(ans_{P5})，而不会计算到右分枝结点P_7。

可见，AB可以比VEB有更大的截断。

作业-课后练习3

- 按α- β过程剪枝，并给出α- β值



End