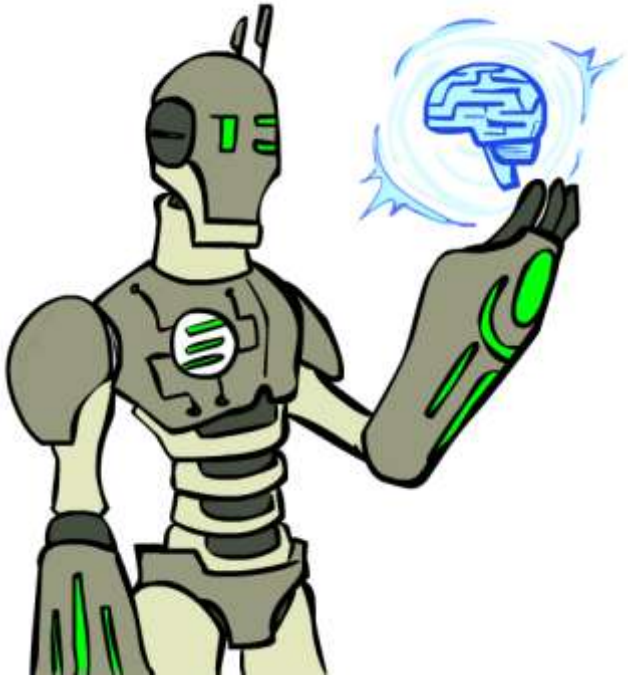
人工智能 I

本节课安排

- 博弈搜索问题
  Game Search Problem
  Adversarial Search Problem
- 极小极大搜索
  Minimax Search
- α-β 剪枝
- 练习

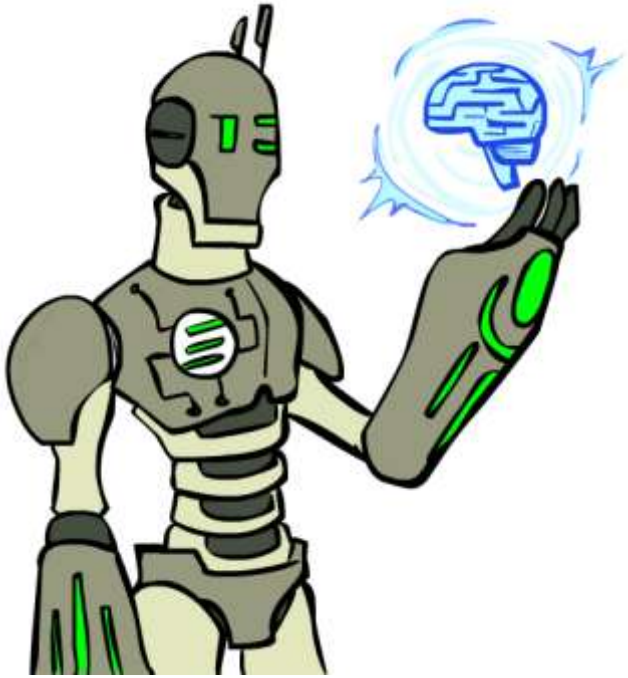


本周五实验课安排

周次	5
日期	2025/10/24
星期	星期五
时间	8:00-12:15
机房地点	五五楼435-软实七
实验任务	有信息搜索

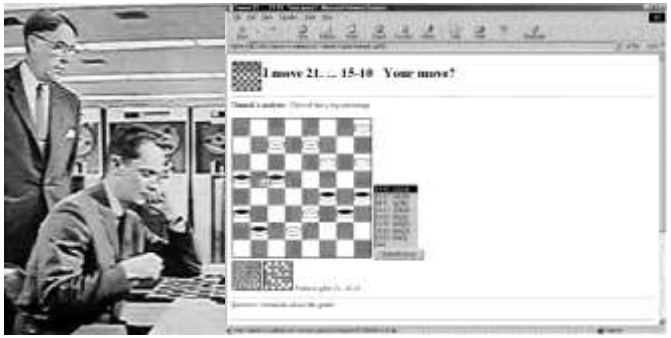
本节课安排

- 博弈搜索问题
  Game Search Problem
  Adversarial Search Problem
- 极小极大搜索
  Minimax Search
- α-β 剪枝
- 练习



例子：棋类游戏

- 西洋跳棋 (1962, IBM)
  - Chinook VS. Marion Tinsley
  - 提出机器学习、强化学习



- 国际象棋 (1997, IBM)
  - DeepBlue VS.卡斯帕罗夫
  - 硬件加速搜索
  - 超级专家系统

- 围棋 (2016, DeepMind)
  - AlphaGo VS. 李世石
  - DL+ RL + 蒙特卡洛方法

任务环境的属性

- 完全可观测 ***V.S.*** 部分可观测
- 单体 ***V.S.*** 多体
- 静态 ***V.S.*** 动态
- 确定性 ***V.S.*** 随机性
- 周期性***V.S.*** 序贯性
- 离散 ***V.S.*** 连续

博弈

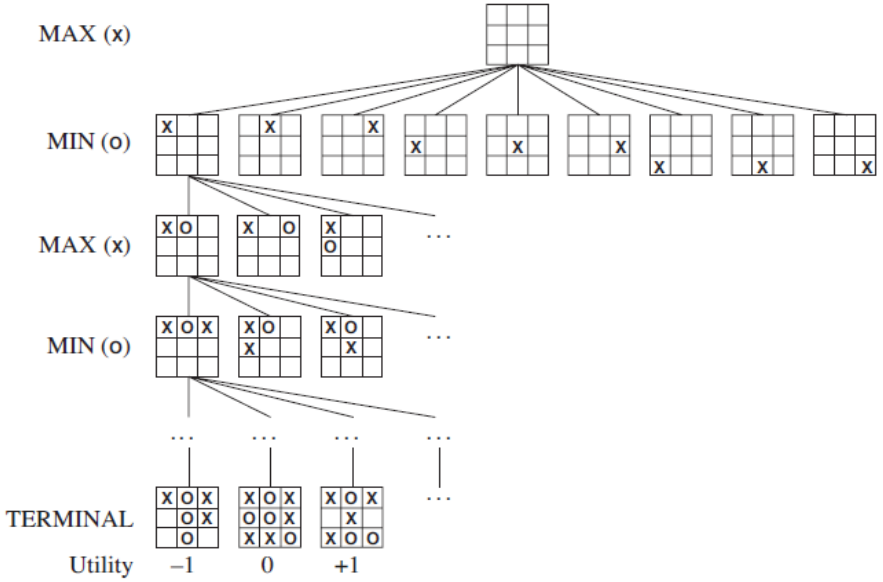
人工智能中“博弈”通常专指博弈论专家们称为**有完整信息的、确定性的、轮流行动的、两个游戏者**的零和（Zero-Sum）游戏。

博弈问题(Game)的描述

- S_0: 初始状态
- Player(s): 状态s下，由哪个玩家行动
  - MAX （先行）
  - MIN
- Actions(s): 状态s下的动作
- Result(s, a): 后继函数
- Terminal_Test(s): 判断s是否为终止状态
- Utility(s, p): 终止状态s对于玩家p的效用（得分）
  - p=MAX
  - Utility(s)

博弈树（Game Tree）

- 根节点：初始状态，行为决策节点
- 叶子节点：终止状态（终局）
- 中间节点
- 与搜索树类似，主要不同点：
  - 博弈树：有MAX与MIN两位玩家，深度有限
  - 搜索树：一位玩家，动态、可无限延伸



三子棋的博弈树

MAX (X)

MIN (O)

MAX (X)

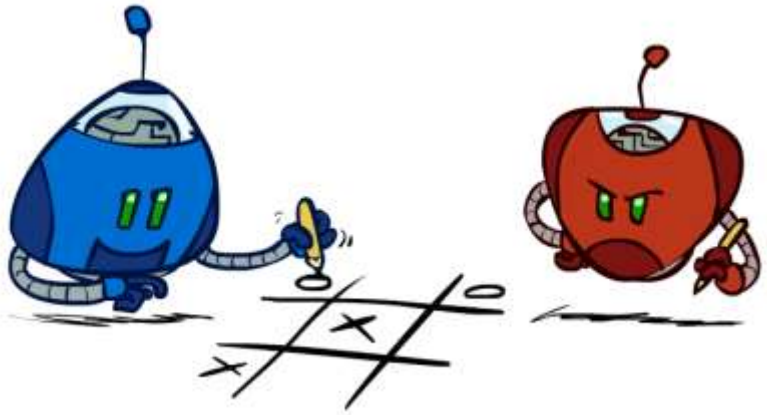
MIN (O)

TERMINAL

Utility

−1 0 +1

初始状态

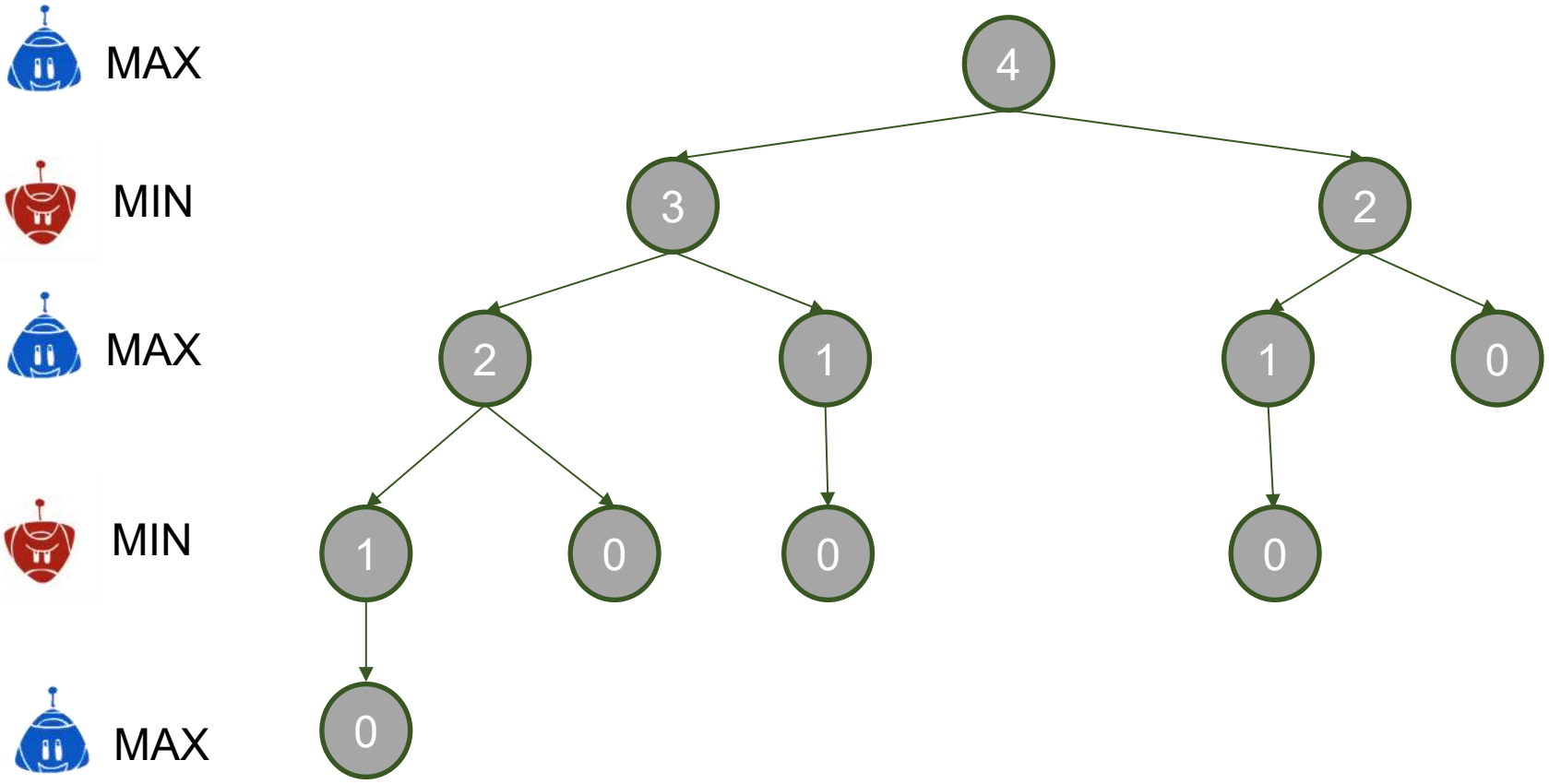


博弈问题（Game）的求解

- 问题：初始状态的行为决策
- 基本思路：
  - 玩家根据当前状态，向对自己有利的终局采取行动（为最大效用而行动）
  - 终局的效用：已知
  - 中间节点的效用：难点

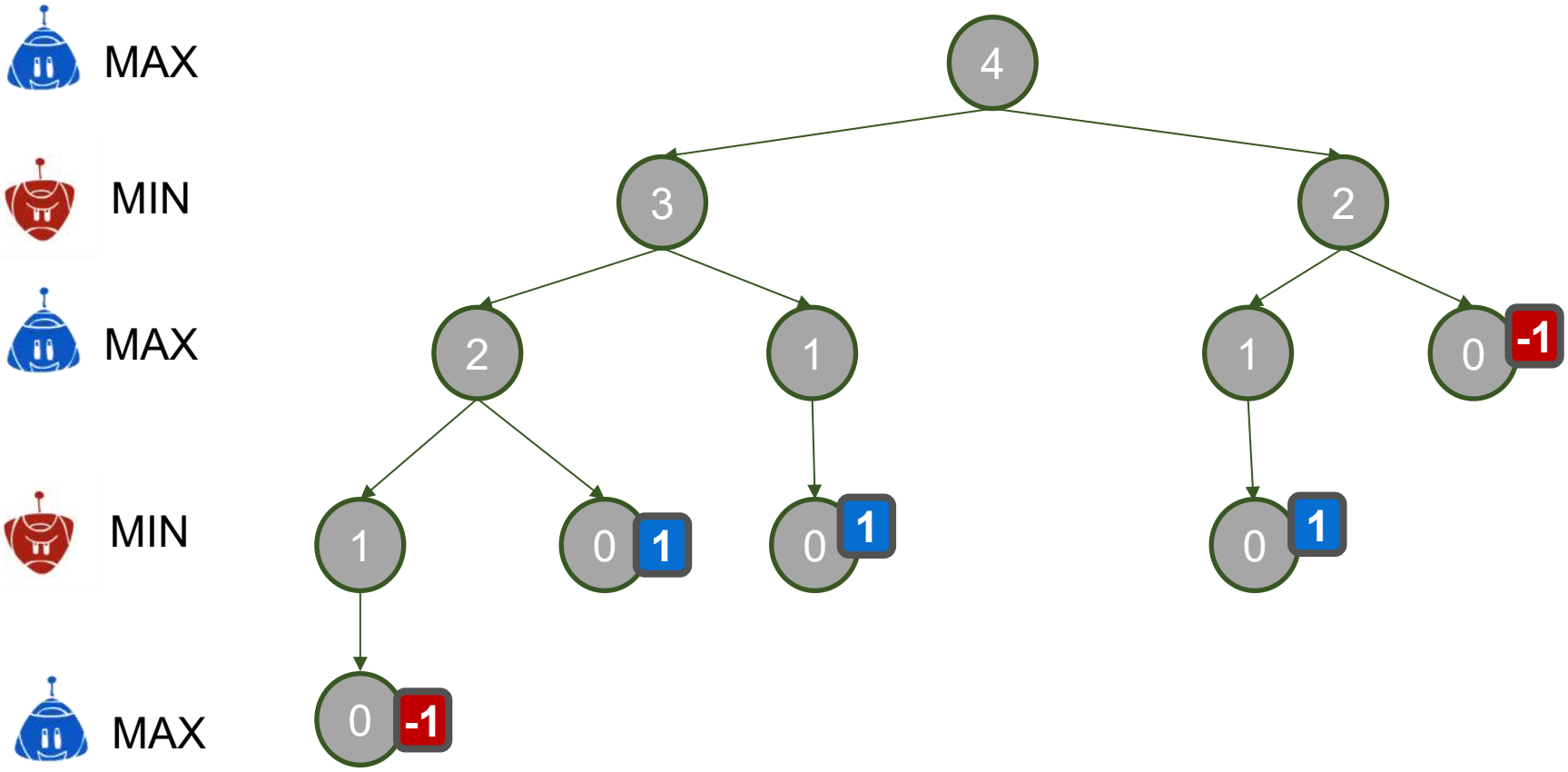
例子：取石子

- 盘子中有4颗石子，两个玩家轮流从中取出。至少取1颗，至多取2颗。不能继续操作的玩家输
  - MAX、MIN两个玩家，MAX先走，MAX赢的效用为1，输为-1



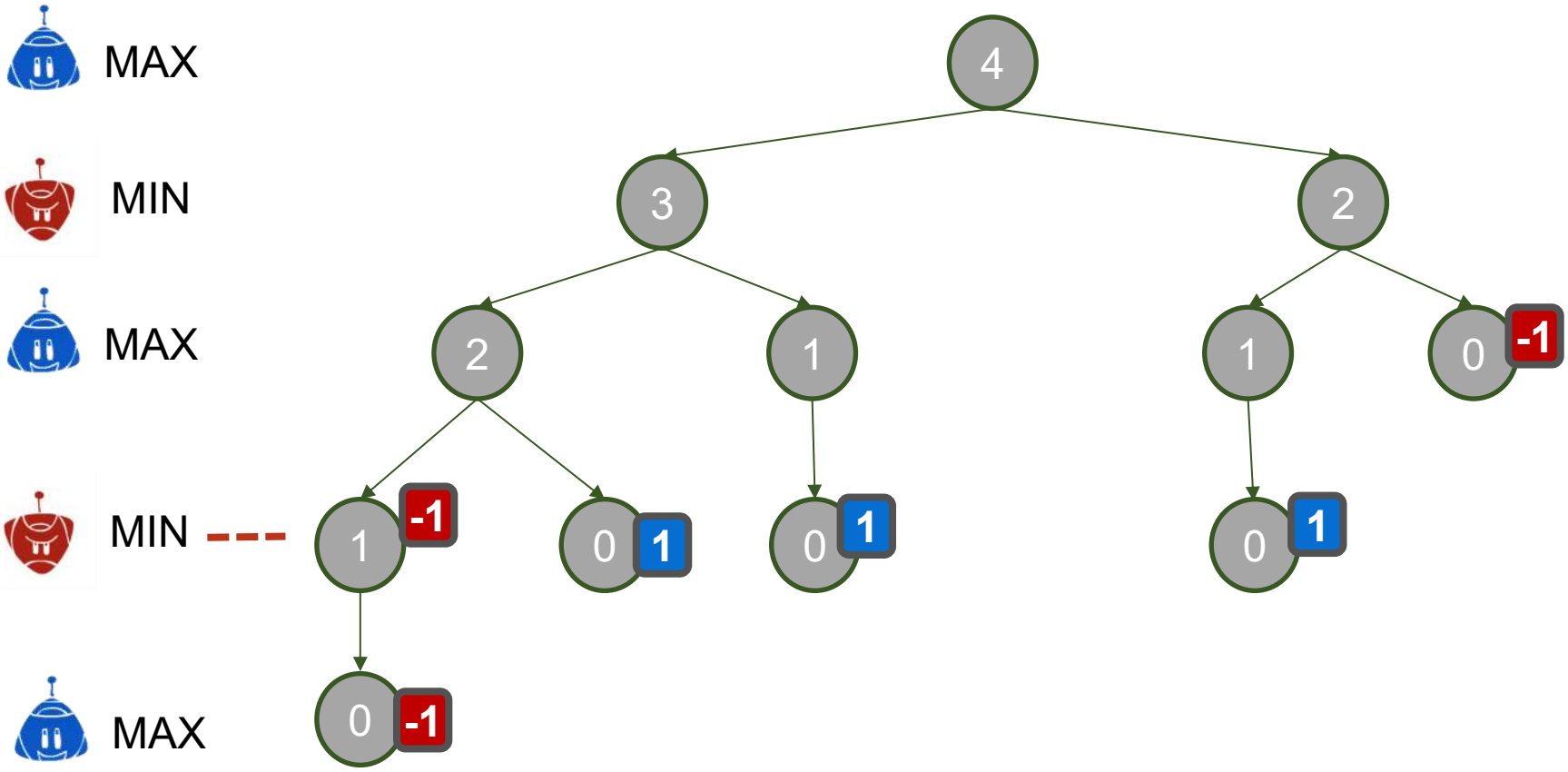
例子：取石子

- 盘子中有4颗石子，两个玩家轮流从中取出。至少取1颗，至多取2颗。不能继续操作的玩家输
  - MAX、MIN两个玩家，MAX先走，MAX赢的效用为1，输为-1



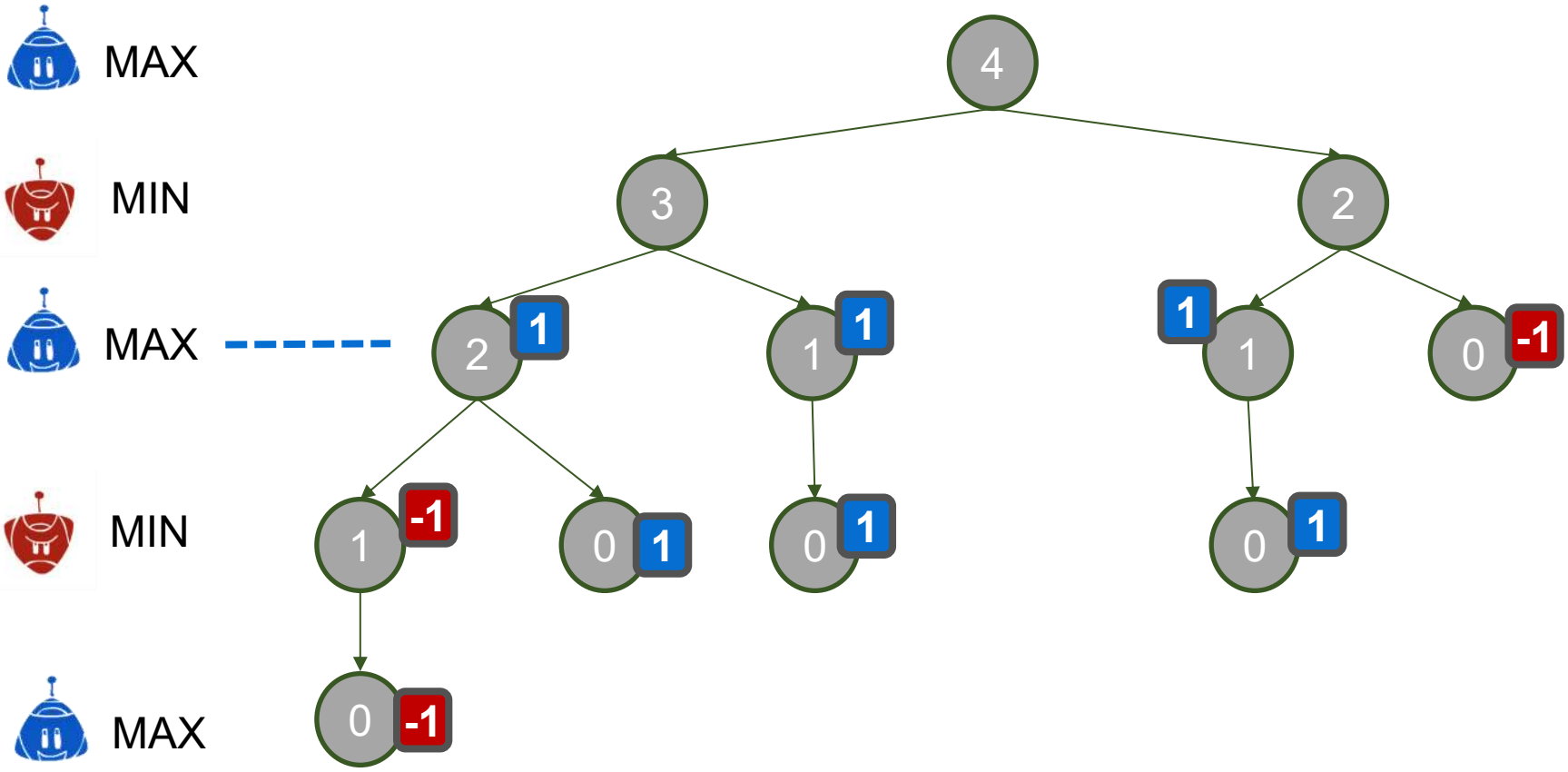
例子：取石子

- 盘子中有4颗石子，两个玩家轮流从中取出。至少取1颗，至多取2颗。不能继续操作的玩家输
  - MAX、MIN两个玩家，MAX先走，MAX赢的效用为1，输为-1



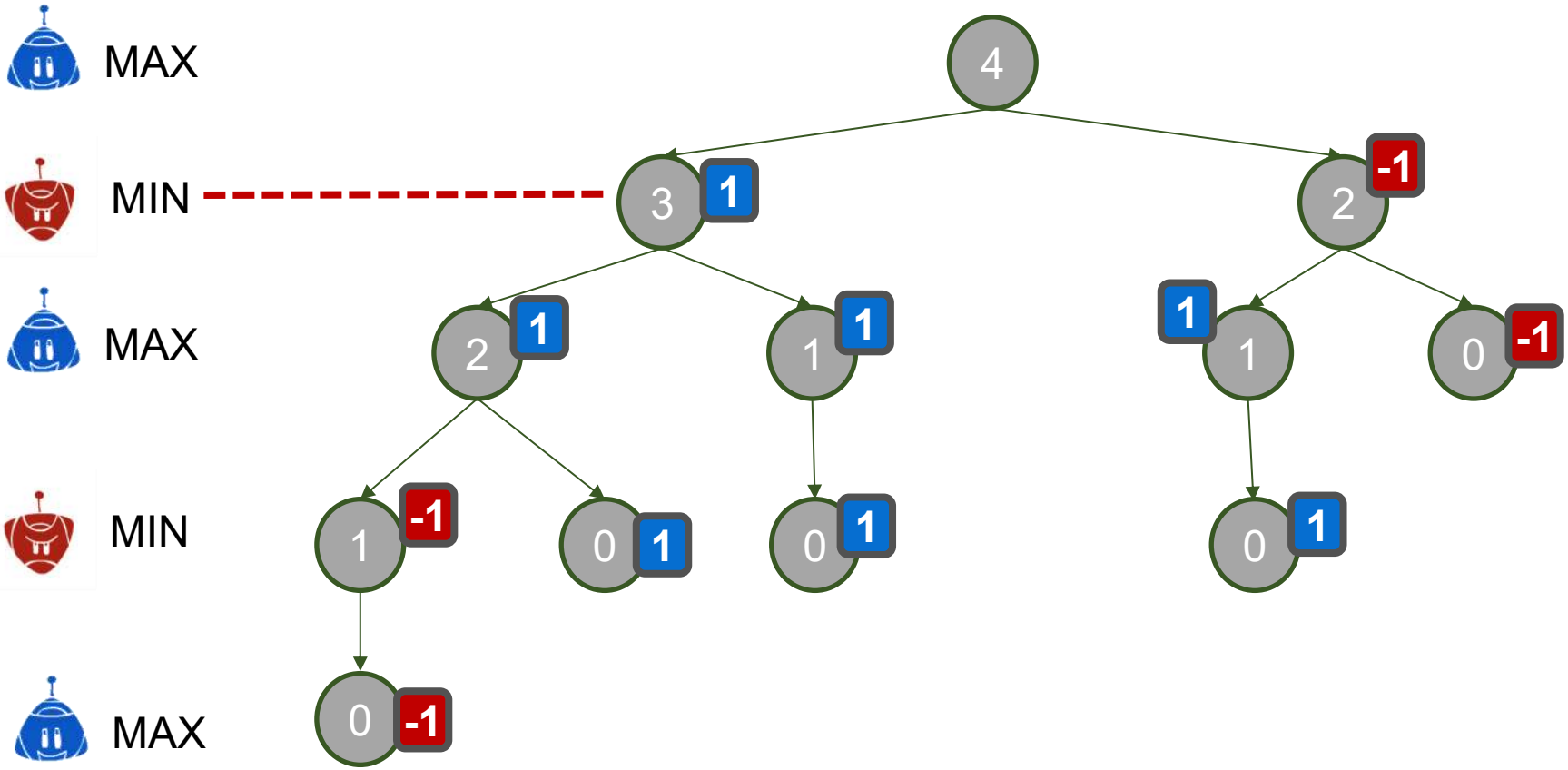
例子：取石子

- 盘子中有4颗石子，两个玩家轮流从中取出。至少取1颗，至多取2颗。不能继续操作的玩家输
  - MAX、MIN两个玩家，MAX先走，MAX赢的效用为1，输为-1



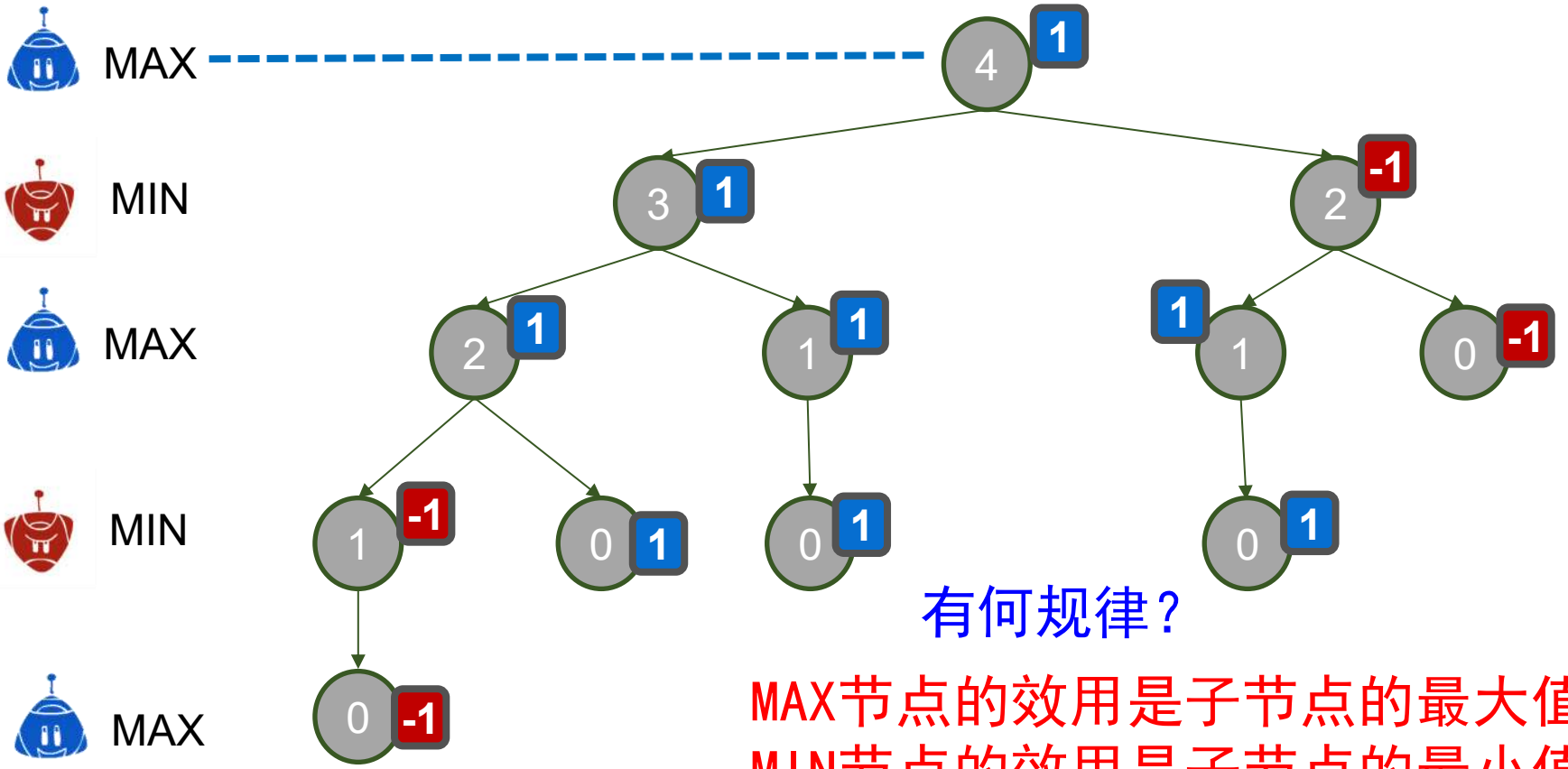
例子：取石子

- 盘子中有4颗石子，两个玩家轮流从中取出。至少取1颗，至多取2颗。不能继续操作的玩家输
  - MAX、MIN两个玩家，MAX先走，MAX赢的效用为1，输为-1



例子：取石子

- 盘子中有4颗石子，两个玩家轮流从中取出。至少取1颗，至多取2颗。不能继续操作的玩家输
  - MAX、MIN两个玩家，MAX先走，MAX赢的效用为1，输为-1



有何规律？

MAX节点的效用是子节点的最大值

MIN节点的效用是子节点的最小值


Garry Kasparov
Deep Blue

本节课安排

- 博弈搜索问题
  Game Search Problem
  Adversarial Search Problem

- 极小极大搜索
  Minimax Search

- α-β 剪枝

- 练习

极小极大搜索

- **MiniMax值:**

$$\text{MiniMax}(s) = \begin{cases} \text{Utility}(s), & \text{if Terminal\_Test}(s), \\ \max_a \text{MiniMax}\big(Result(s,a)\big), & \text{if Player}(s) = \text{MAX}, \\ \min_a \text{MiniMax}\big(Result(s,a)\big), & \text{if Player}(s) = \text{MIN}. \end{cases}$$

- **MiniMax决策（最优决策）：**

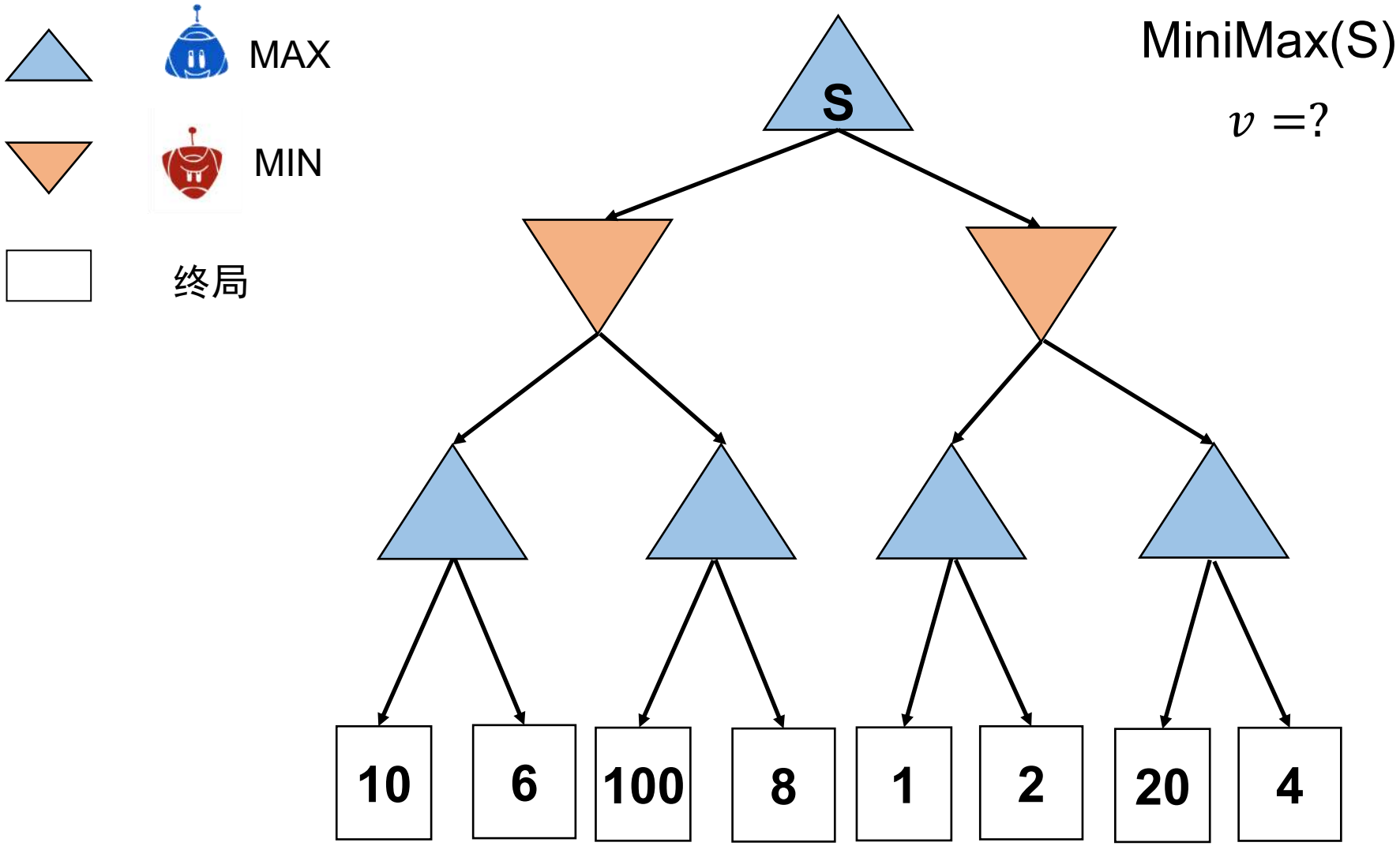
```
function MiniMax-Decision(state)
    If player(state) = MAX
        return arg max_a MiniMax(Result(state, a))
    If player(state) = MIN
        return arg min_a MiniMax(Result(state, a))
```

极小极大搜索：算法

```
function MiniMax(state)
If Terminal-Test(state) return Utility(state)
If player(state) = MAX {
        v = -∞
        for each child {
            v = max(v, MiniMax(child))
        }
 else {
        v = ∞
        for each child {
            v = min(v, MiniMax(child))
         }
return v//返回值
```

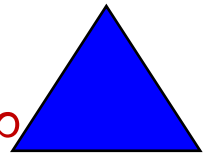

SOUTHEAST UNIVERSITY
東南大學

极小极大搜索：算法测试

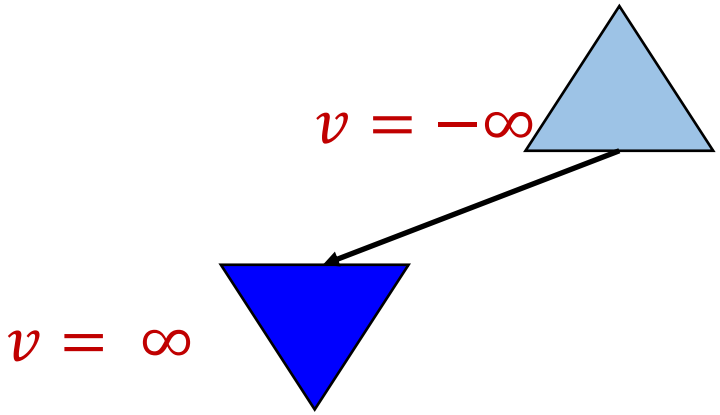


极小极大搜索：算法测试

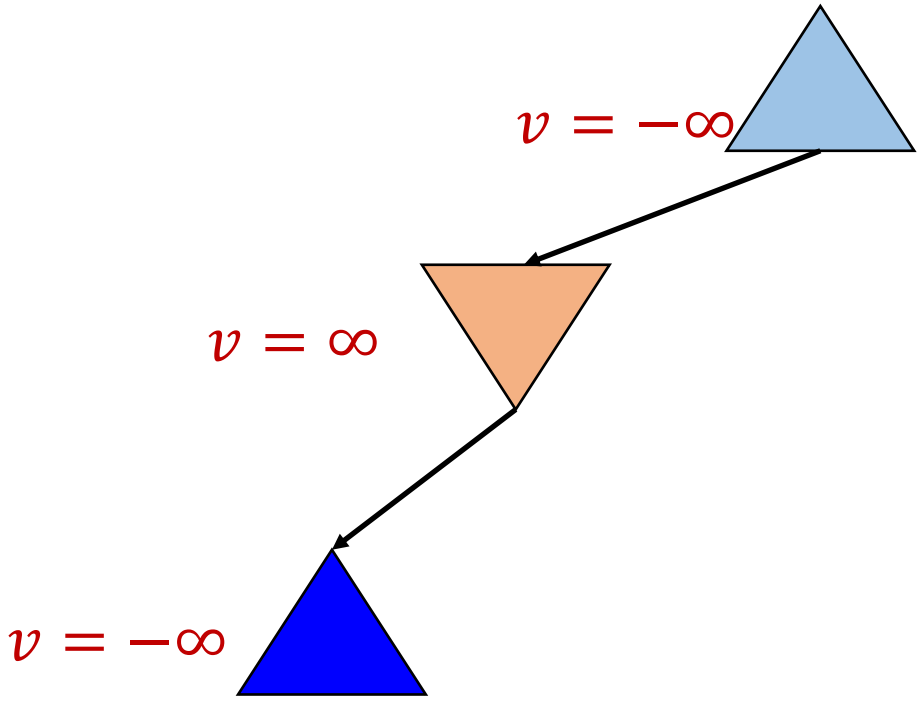
$v = -\infty$



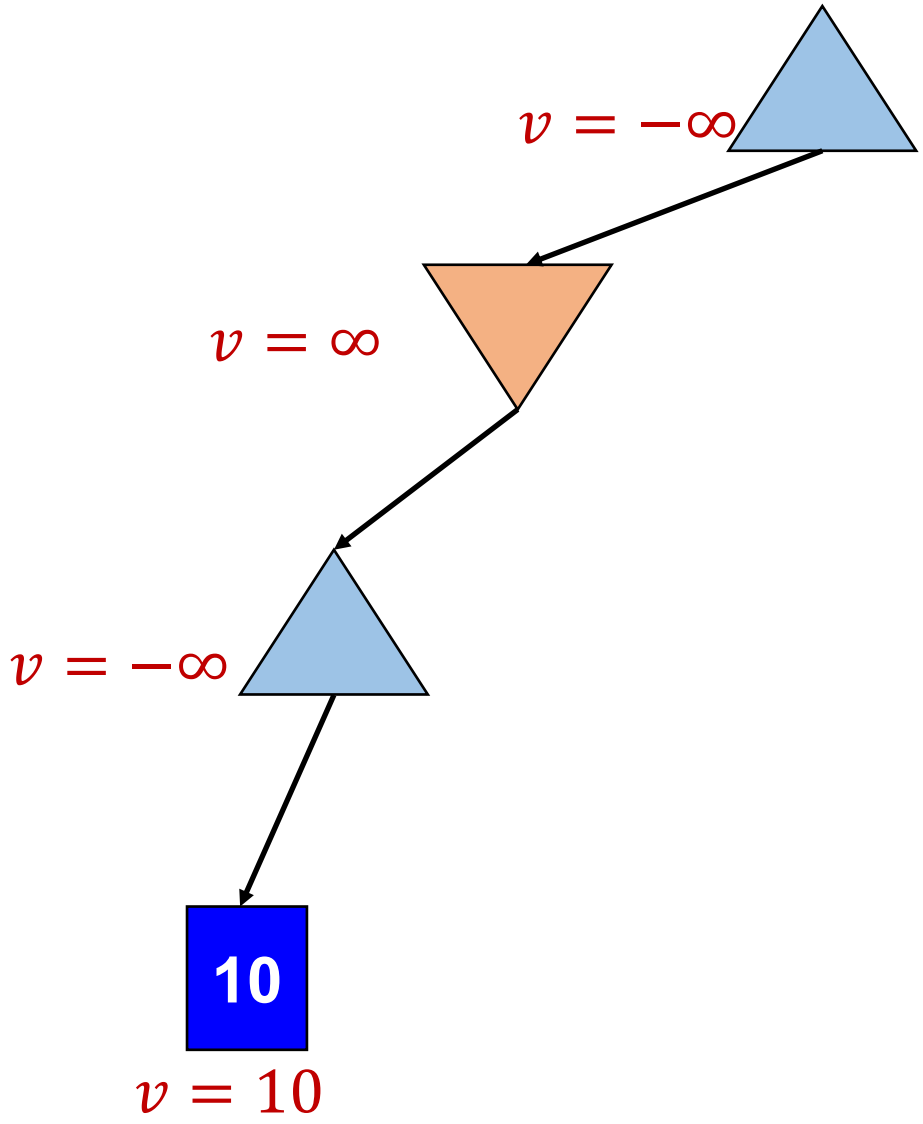
极小极大搜索：算法测试



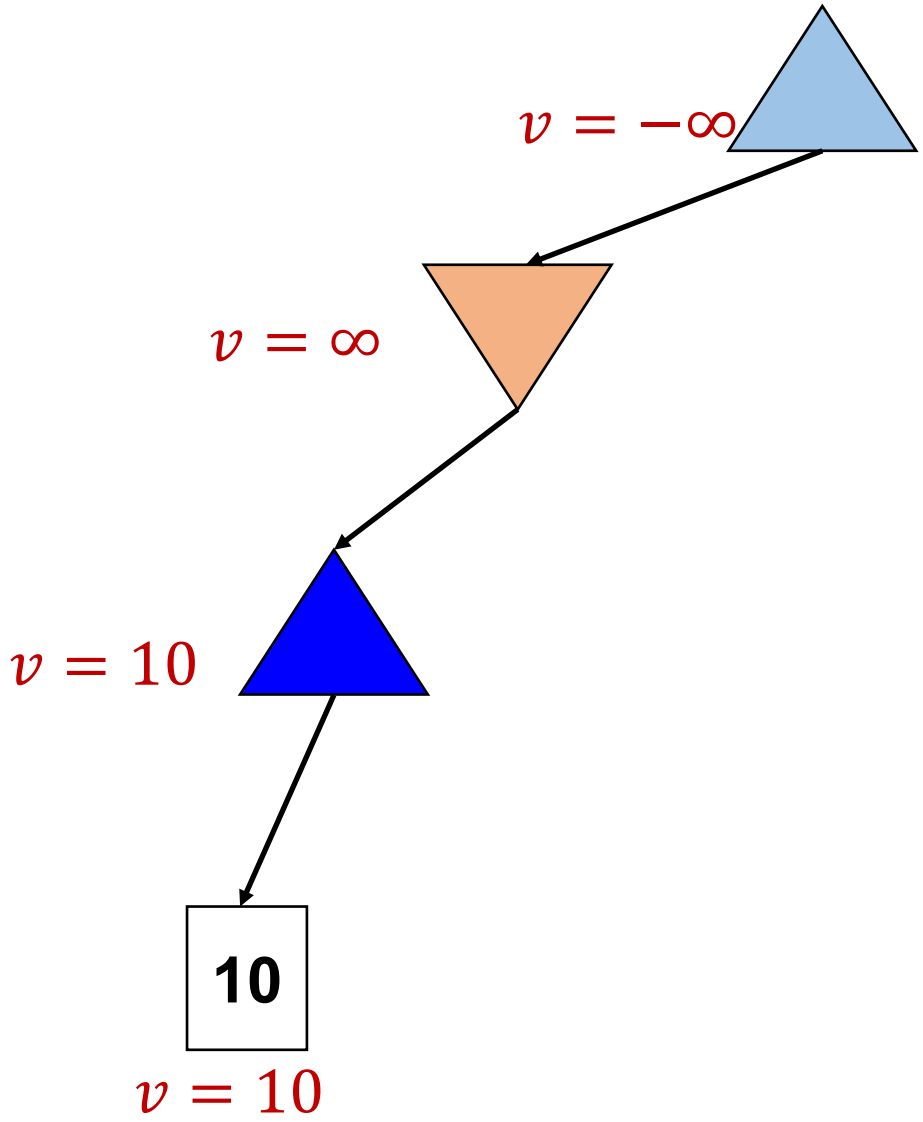
极小极大搜索：算法测试



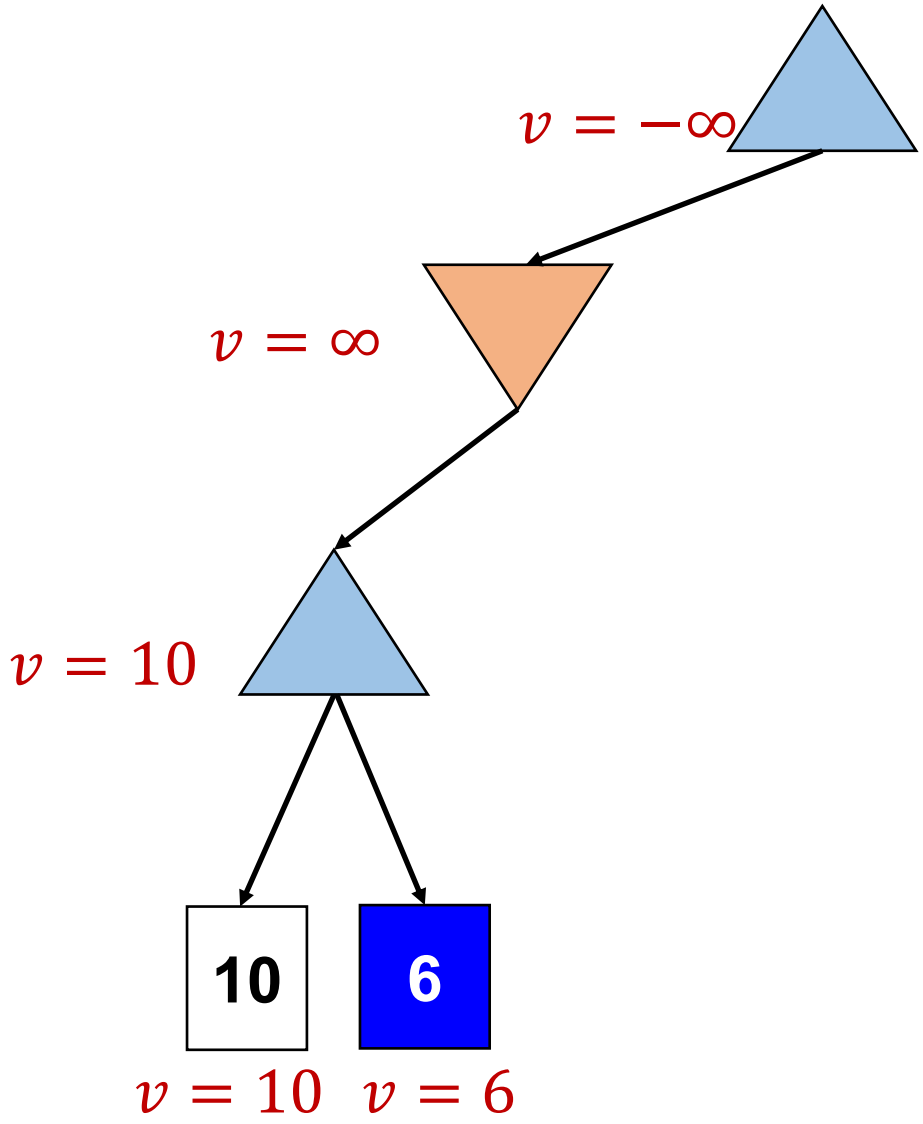
极小极大搜索：算法测试



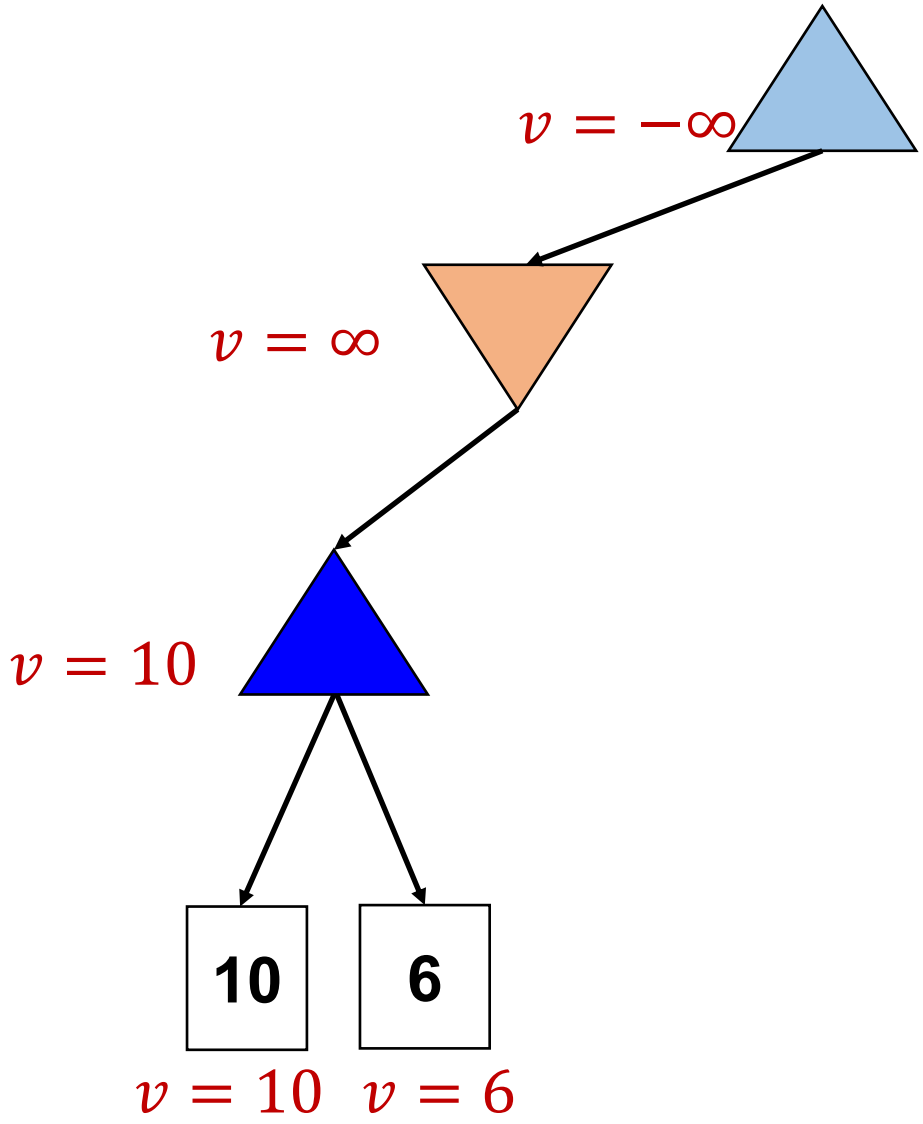
极小极大搜索：算法测试



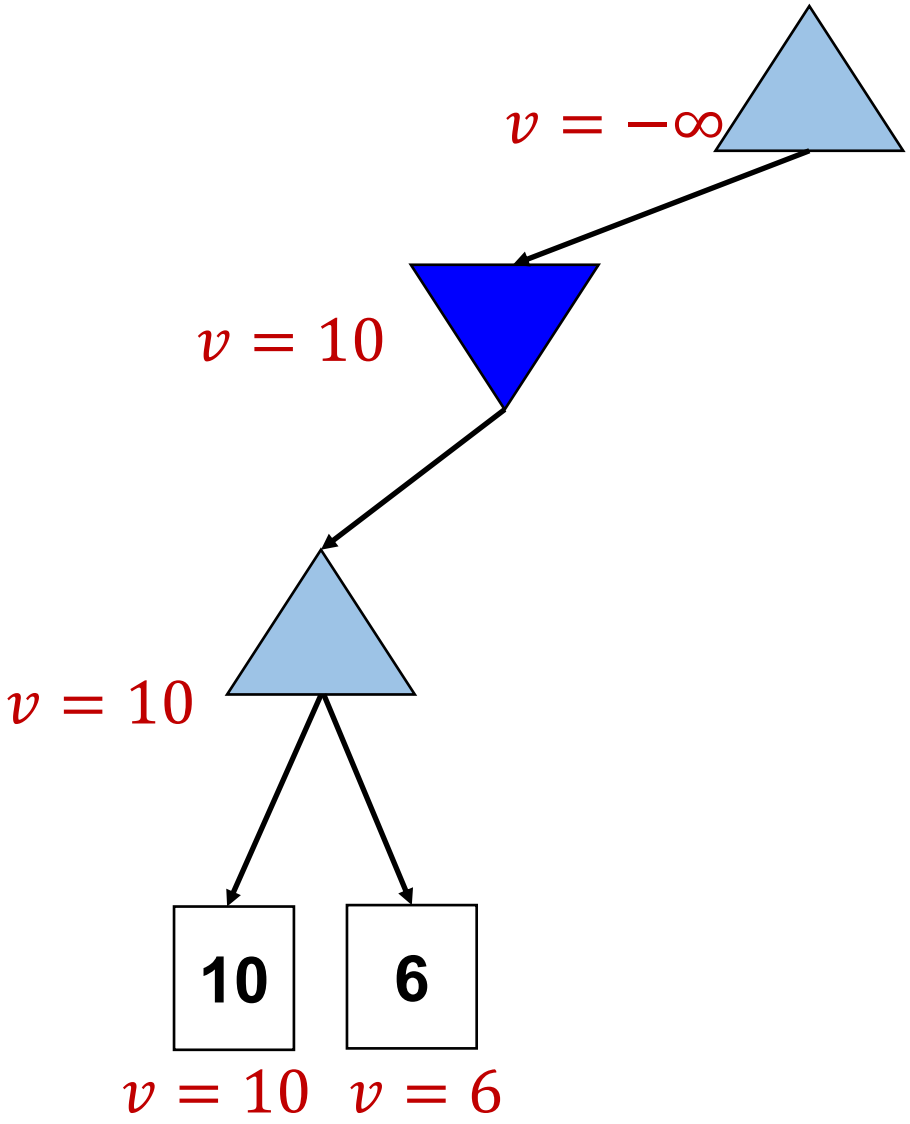
极小极大搜索：算法测试



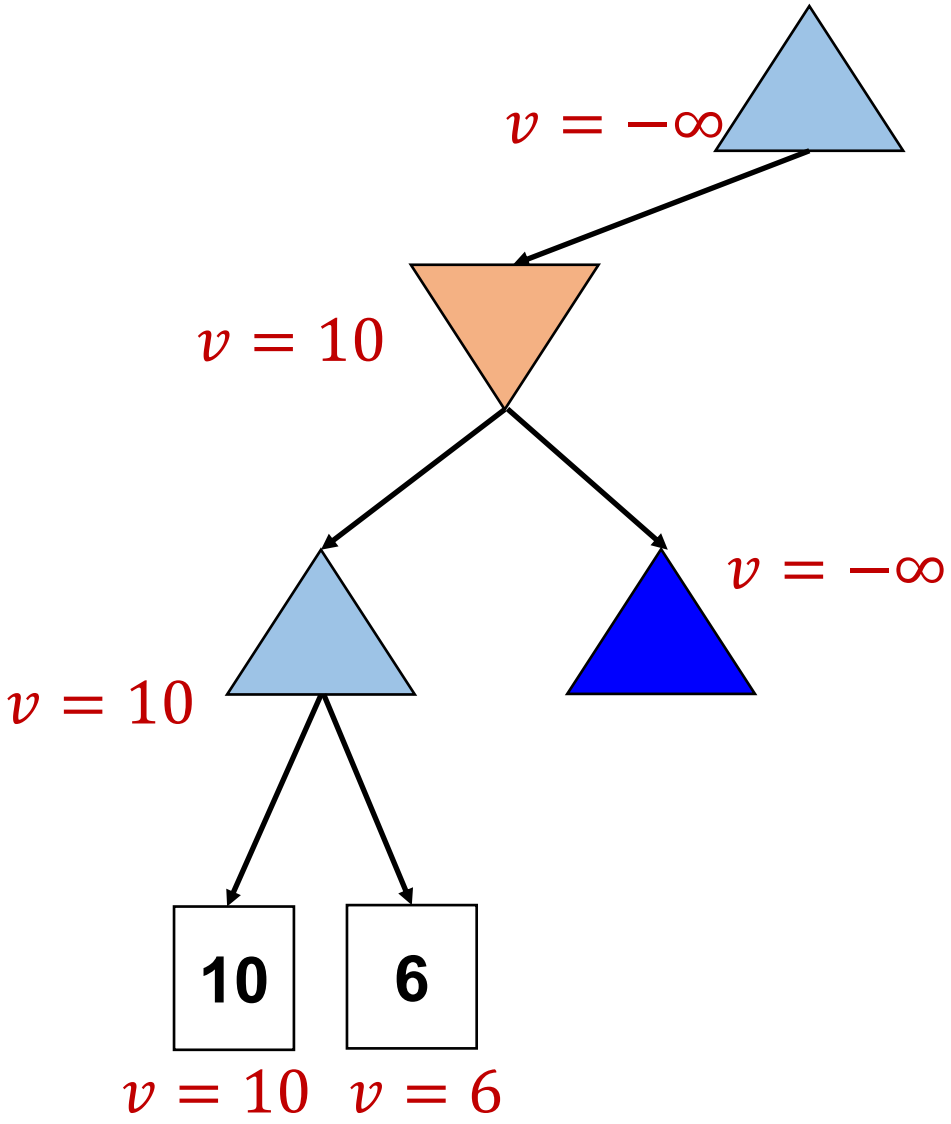
极小极大搜索：算法测试



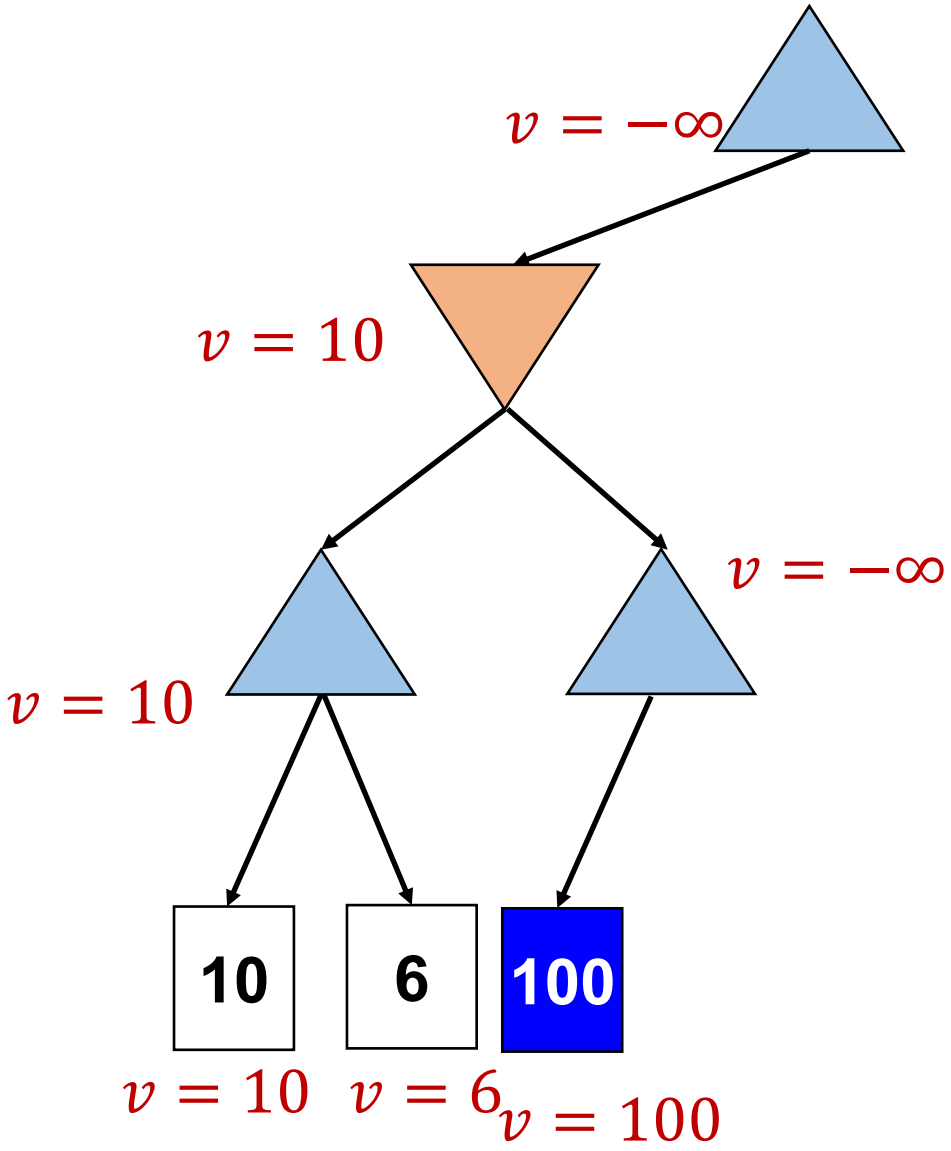
极小极大搜索：算法测试



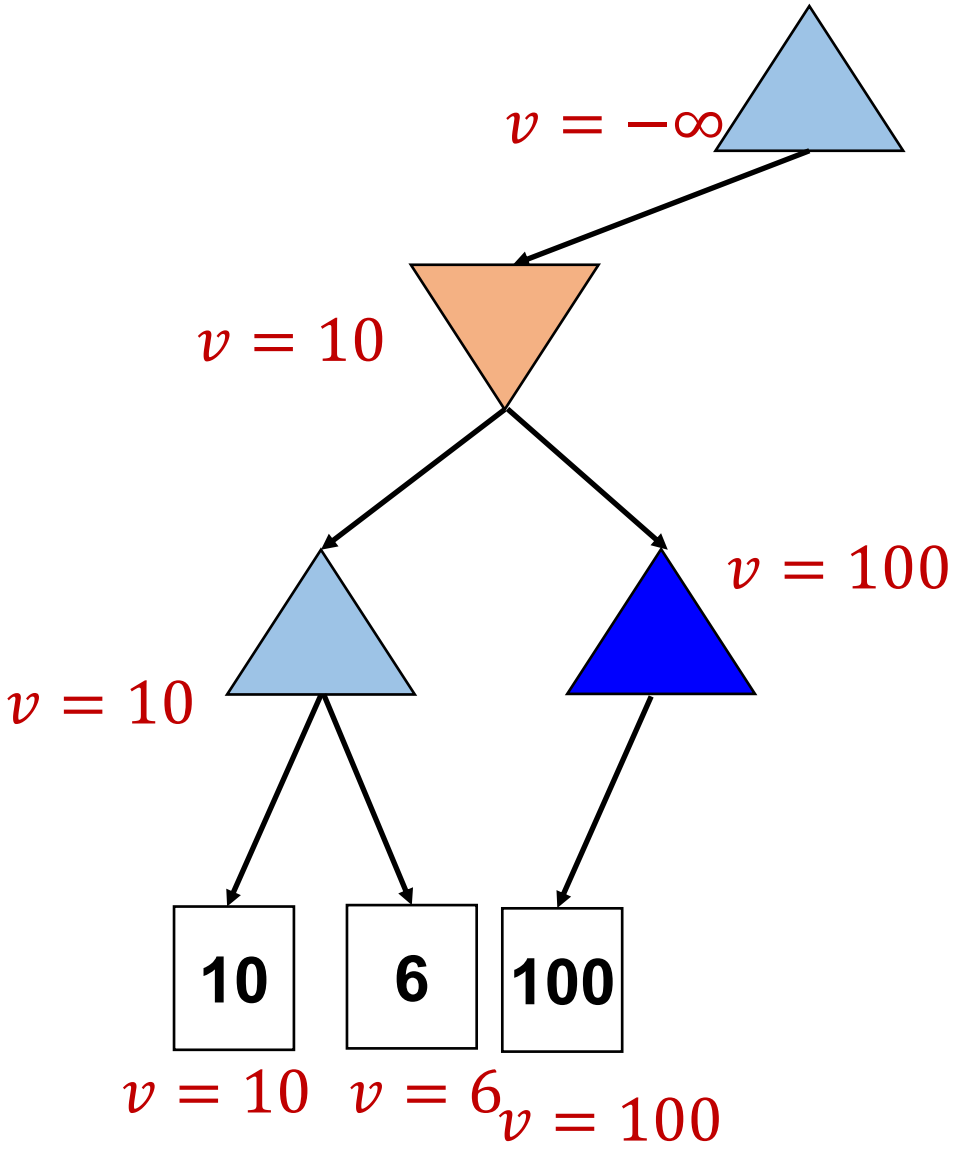
极小极大搜索：算法测试



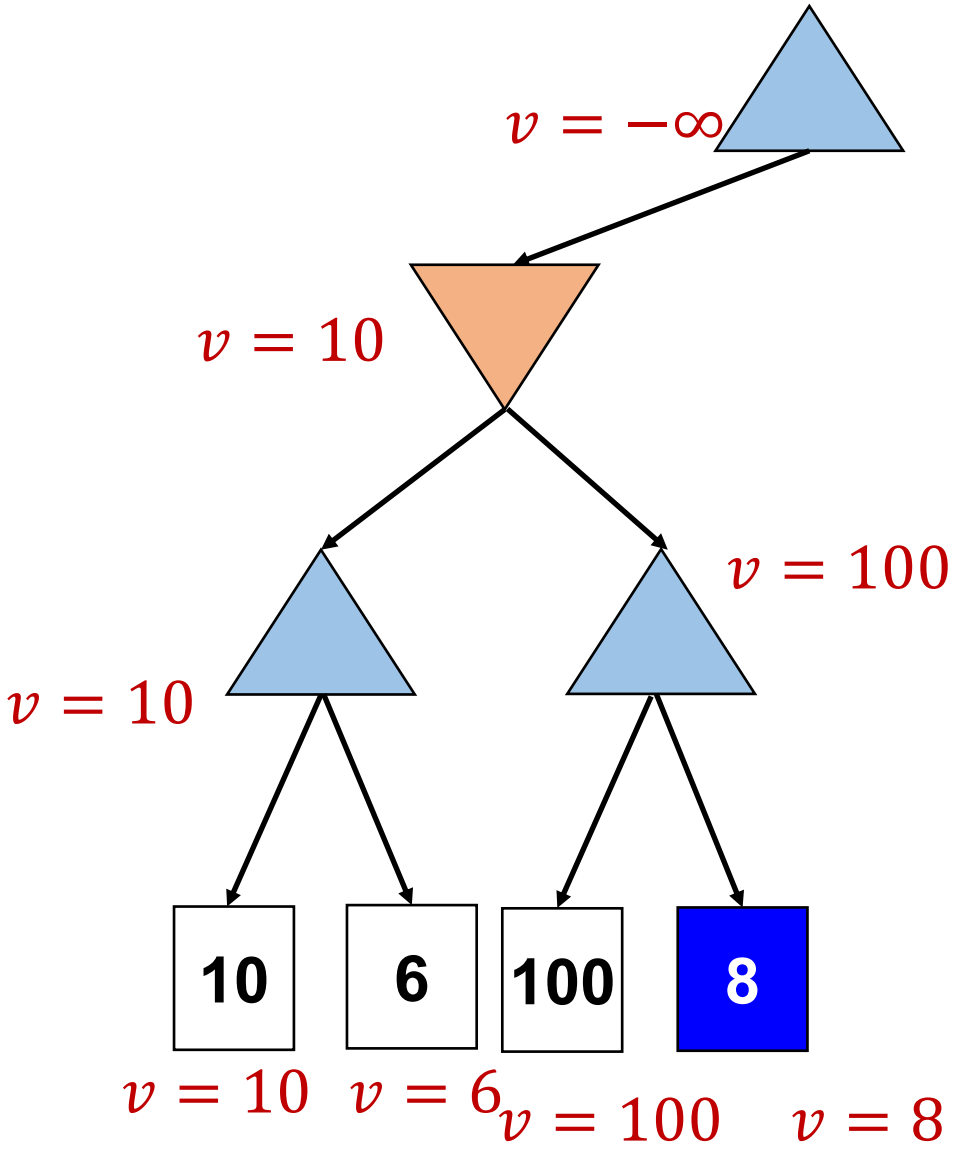
极小极大搜索：算法测试



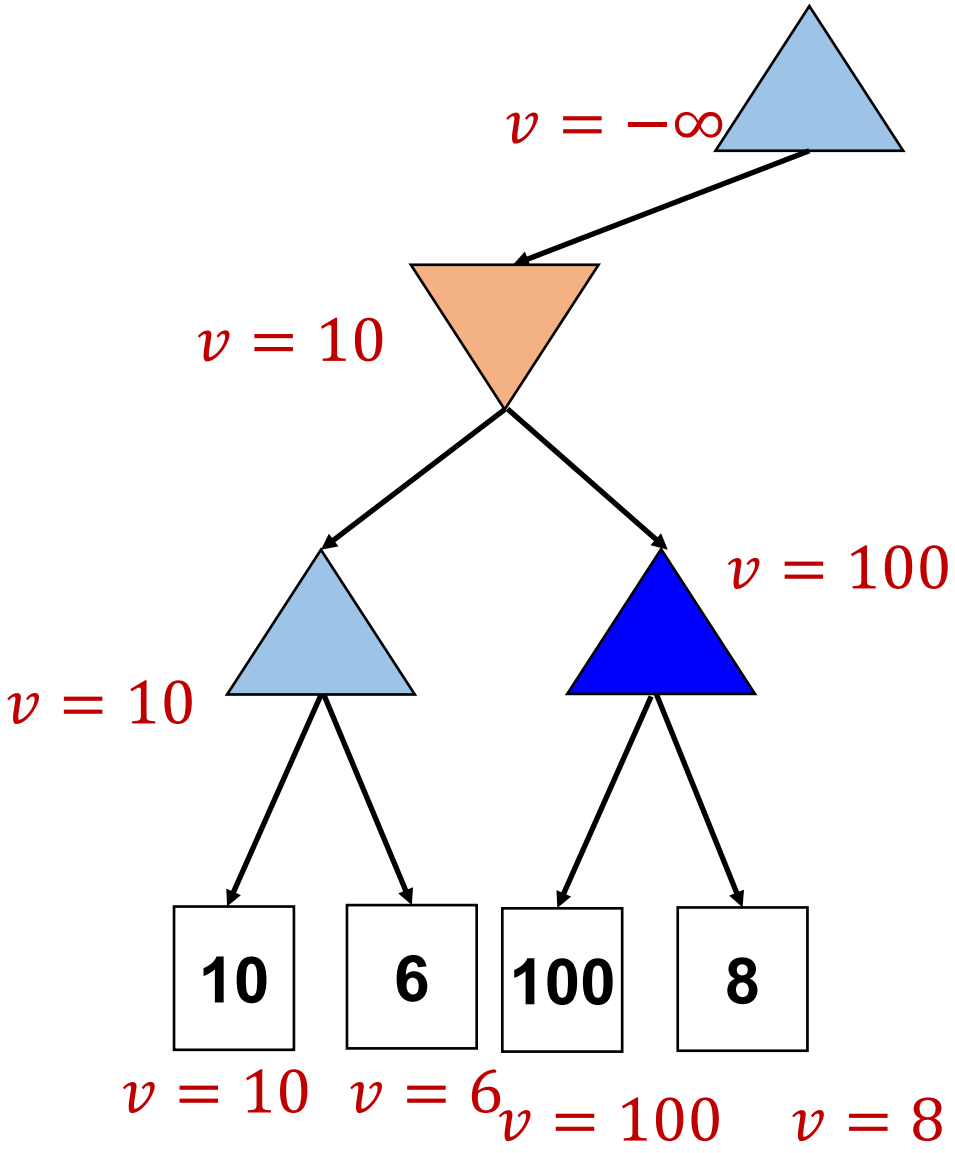
极小极大搜索：算法测试



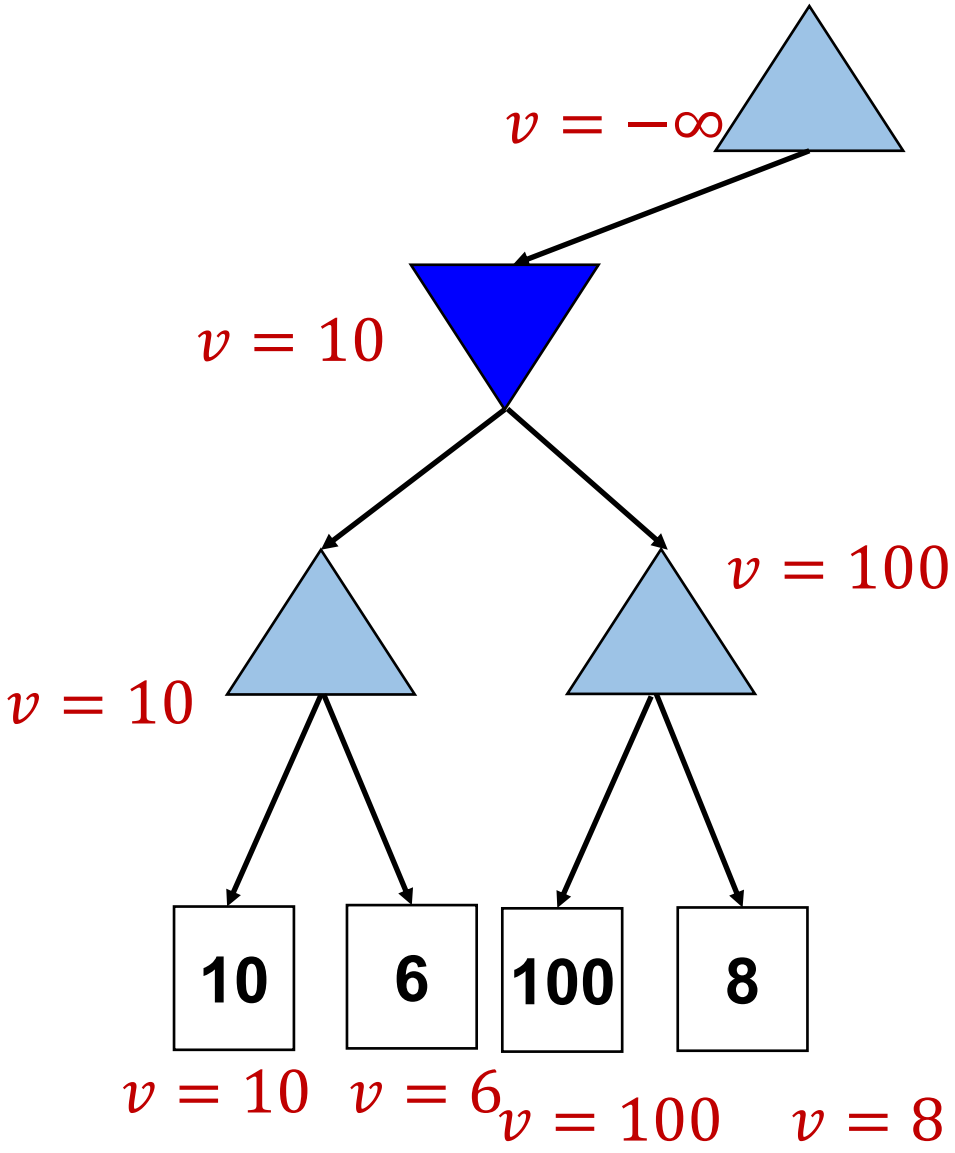
极小极大搜索：算法测试



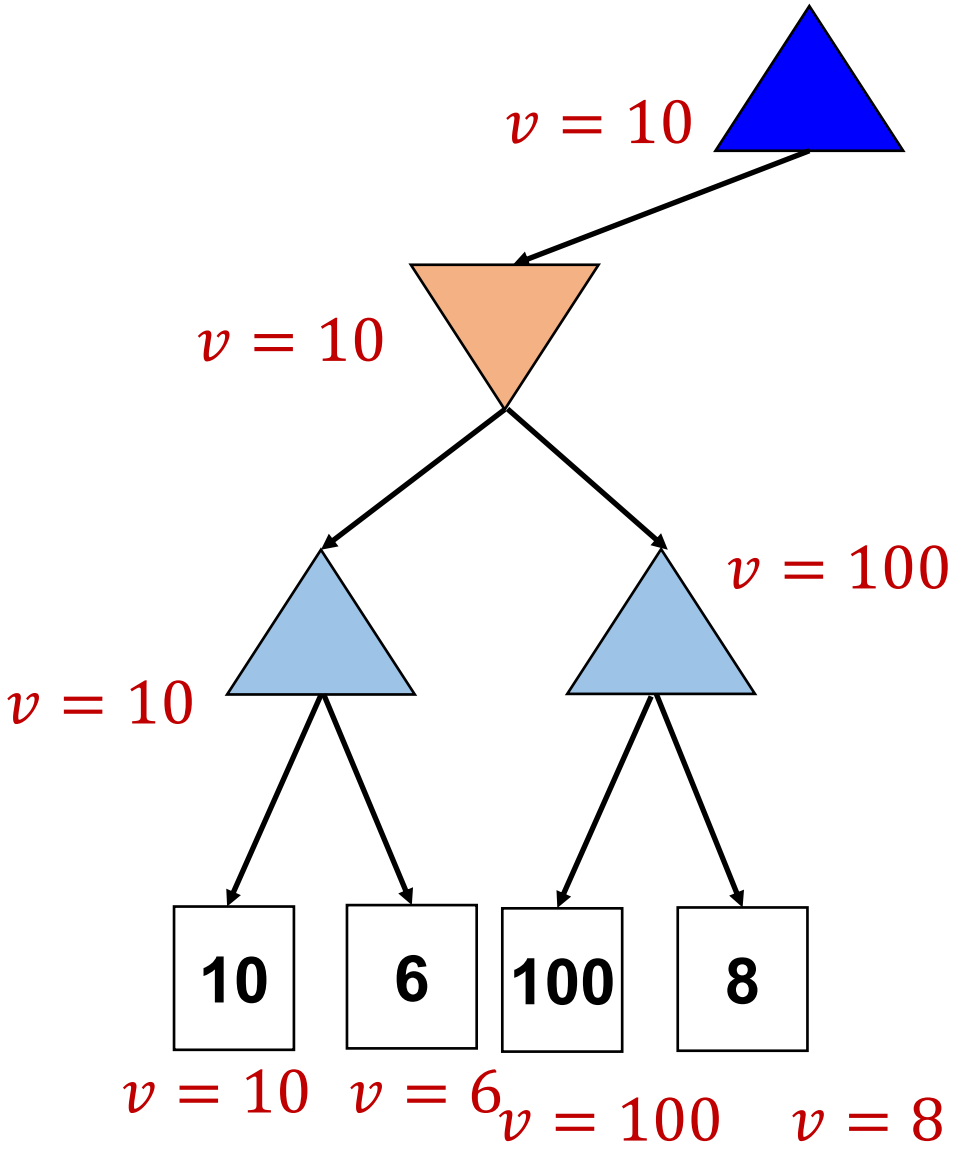
极小极大搜索：算法测试



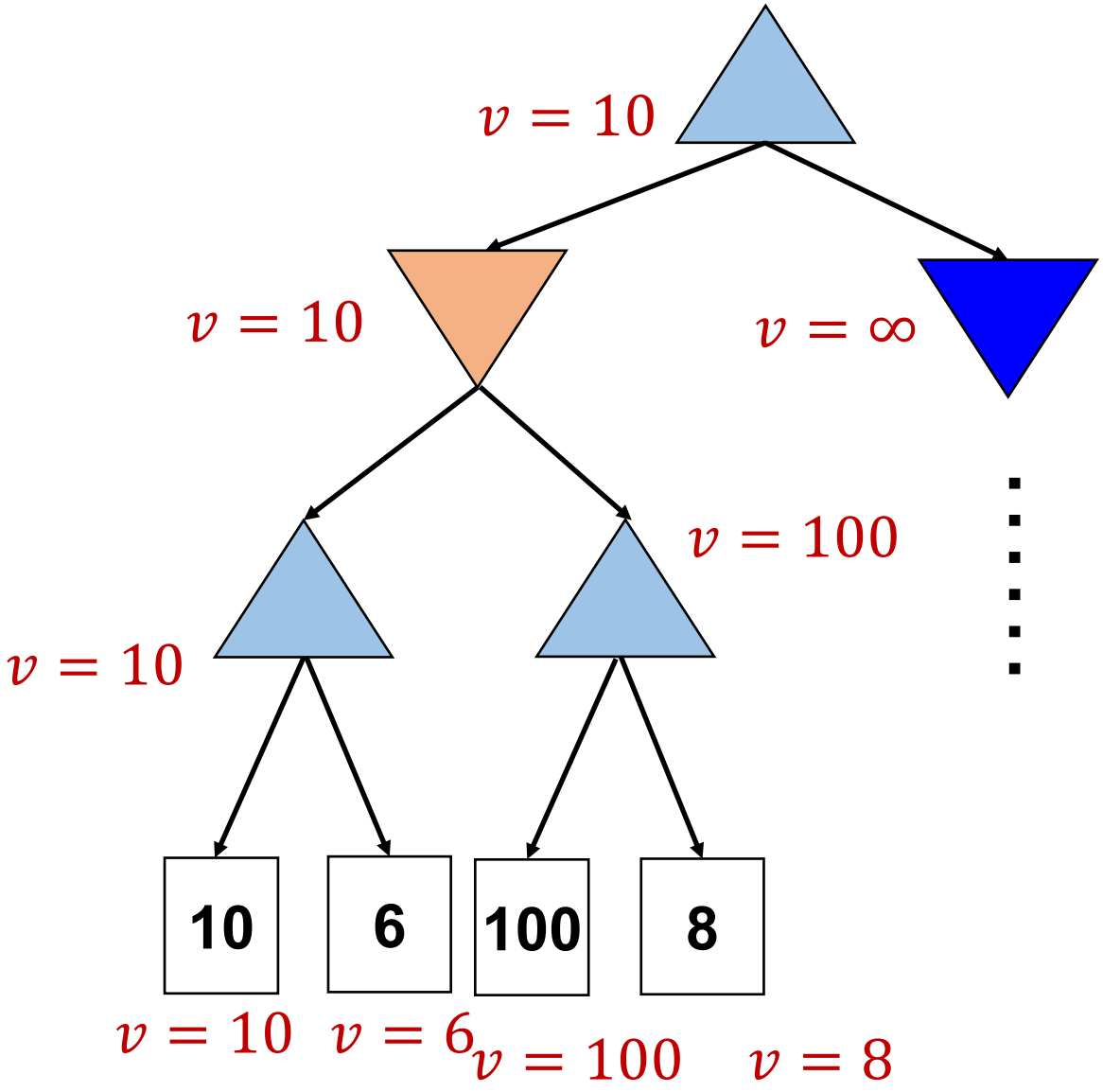
极小极大搜索：算法测试



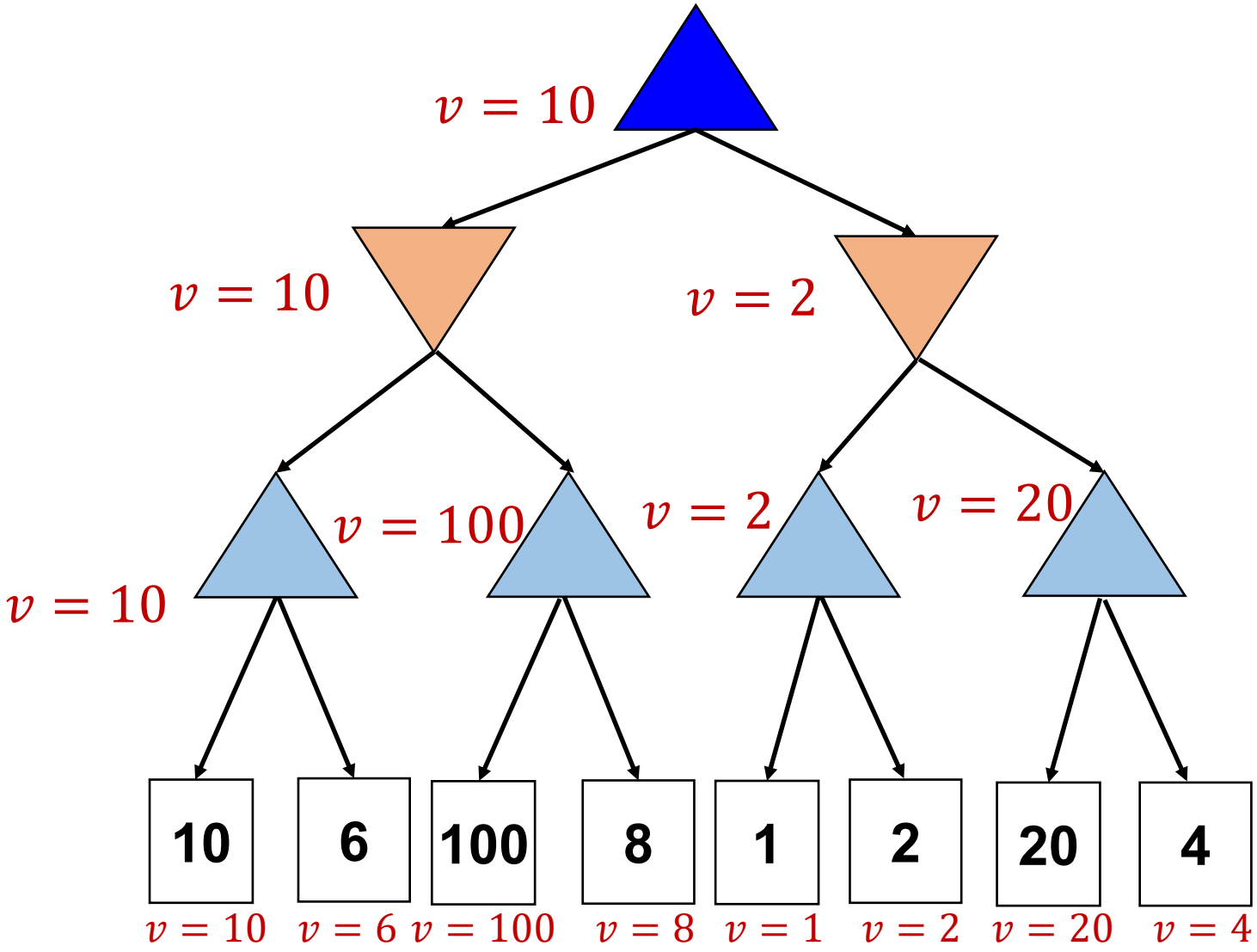
极小极大搜索：算法测试



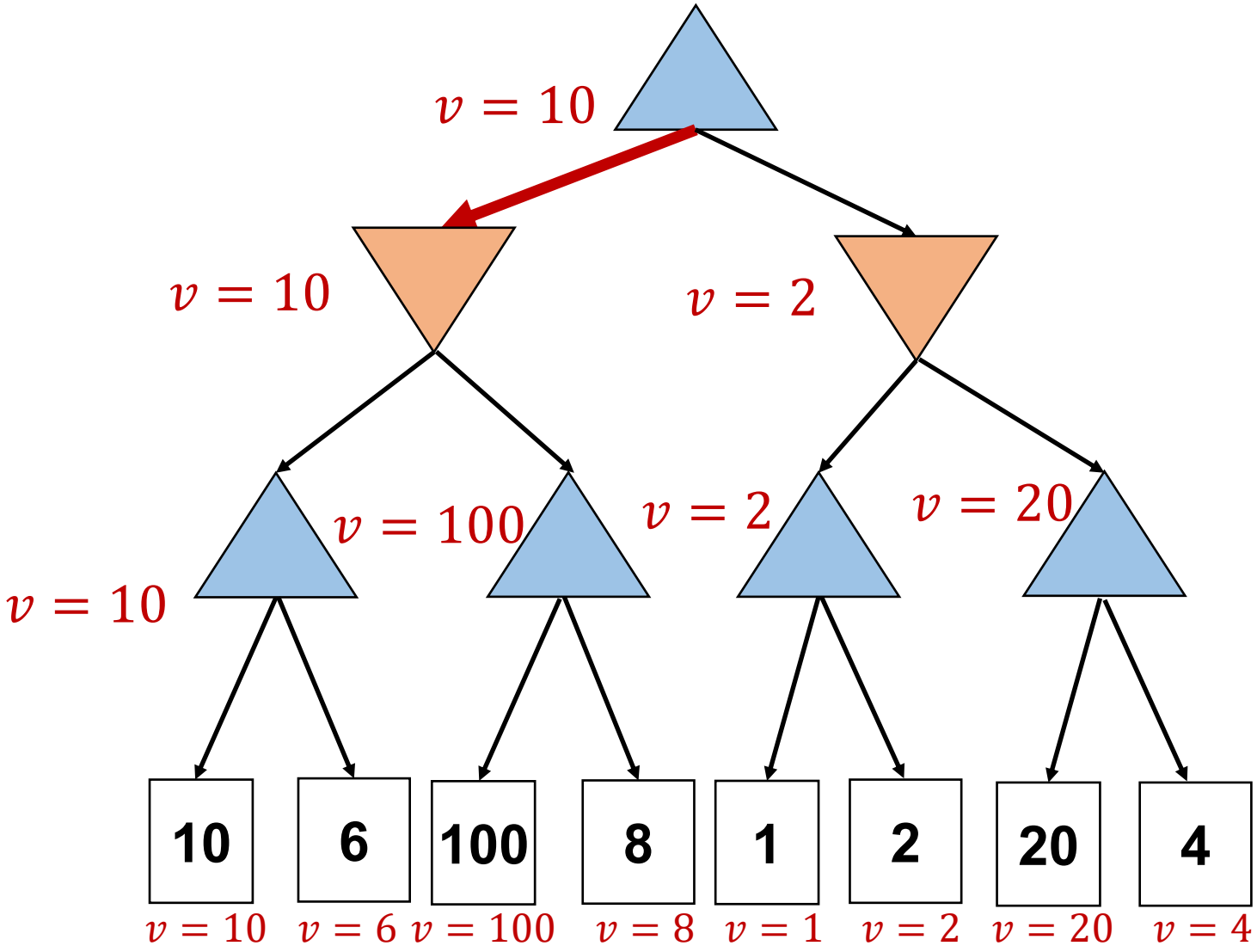
极小极大搜索：算法测试



极小极大搜索：算法测试

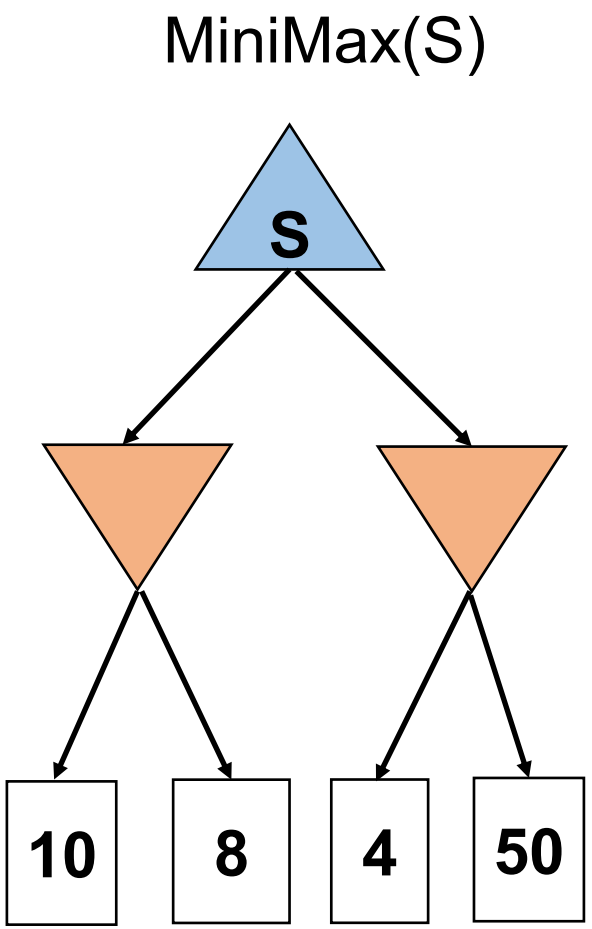

$v = 10$
$v = 10$
$v = 2$
$v = 10$
$v = 100$
$v = 2$
$v = 20$
10
6
100
8
1
2
20
4
$v = 10$
$v = 6$
$v = 100$
$v = 8$
$v = 1$
$v = 2$
$v = 20$
$v = 4$

极小极大搜索：算法测试


$v = 10$
$v = 10$
$v = 2$
$v = 10$
$v = 100$
$v = 2$
$v = 20$
10
6
100
8
1
2
20
4
$v = 10$
$v = 6$
$v = 100$
$v = 8$
$v = 1$
$v = 2$
$v = 20$
$v = 4$

极小极大搜索：练习

```
function MiniMax(state)
If Terminal-Test(state) return Utility(state)
If player(state) = MAX {
        v = -∞
        for each child {
            v = max(v, MiniMax(child))
        }
 else {
        v = ∞
        for each child {
            v = min(v, MiniMax(child))
         }
return v//返回值
```



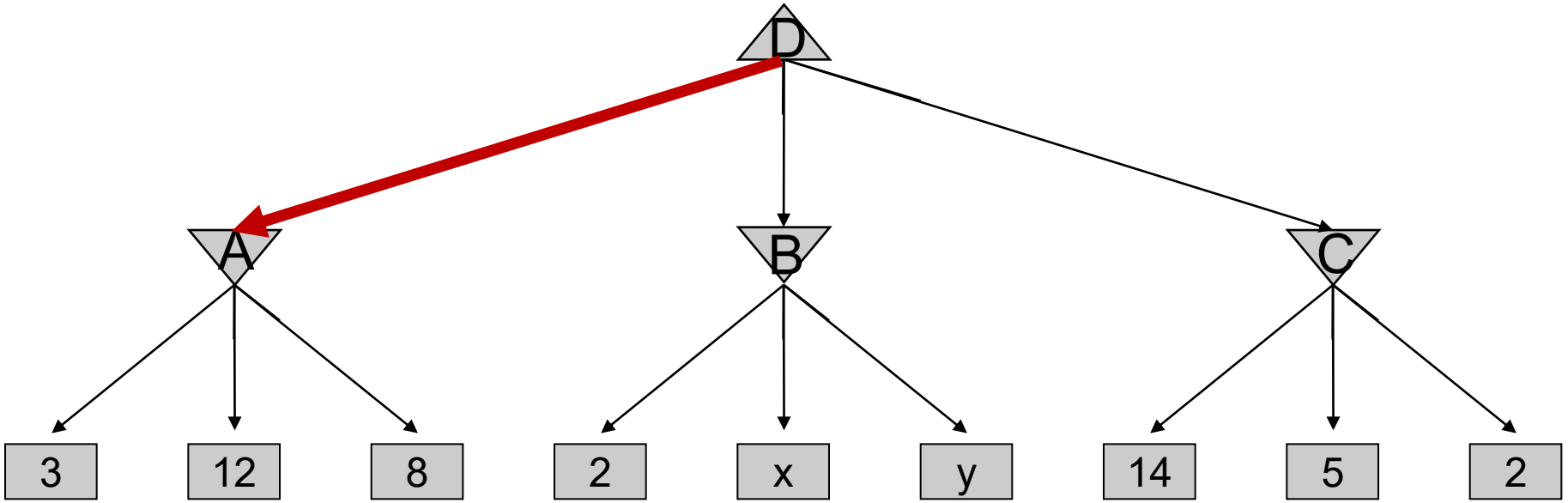
本节课安排

- 博弈搜索问题
  Game Search Problem
  Adversarial Search Problem
- 极小极大搜索
  Minimax Search
- α-β 剪枝
- 练习

极小极大搜索的效率

- **与深度优先遍历相同**:
  - m层（到达终局的步数），b分支(行动个数)
  - 时间复杂度: $\mathcal{O}(b^m)$
  - 空间复杂度: $\mathcal{O}(bm)$
  - 结论：很多情况下无法运行（比如：围棋）
- **在初始状态做出最优行为决策是否有必要遍历整棵树?**

是否有必要遍历整棵树?



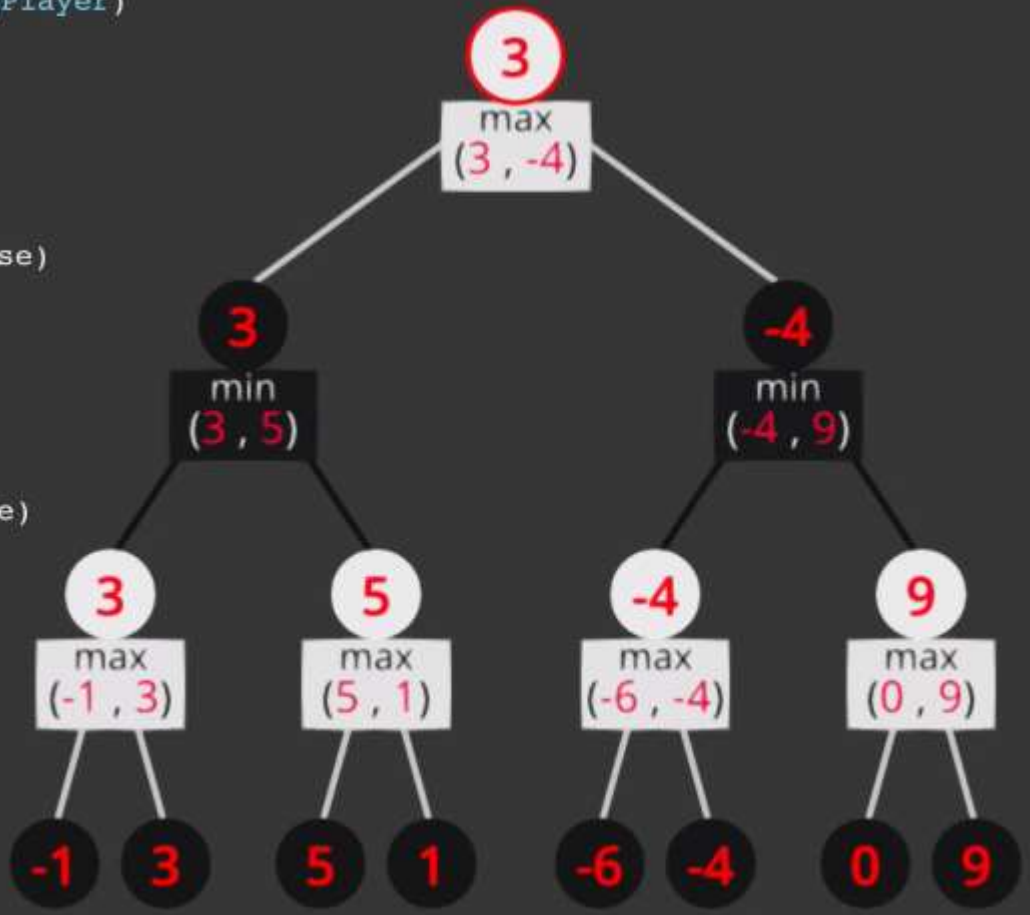
$$\begin{aligned}&\text{MiniMax(D)}\\&= \max(\min(3,12,8)\,,\min(2,x,y)\,,\min(14,5,2))\\&= \max(3,\min(2,x,y),2) = 3\end{aligned}$$

```
function minimax(position, depth, maximizingPlayer)
    if depth == 0 or game over in position
        return static evaluation of position

    if maximizingPlayer
        maxEval = -infinity
        for each child of position
            eval = minimax(child, depth - 1, false)
            maxEval = max(maxEval, eval)
        return maxEval

    else
        minEval = +infinity
        for each child of position
            eval = minimax(child, depth - 1, true)
            minEval = min(minEval, eval)
        return minEval
```

```
// initial call
minimax(currentPosition, 3, true)
```



α-β 决策

- α-β决策：
  - 类似MiniMax决策
  - α：MiniMax的下界，初始值= $-\infty$
  - β：MiniMax的上界，初始值= ∞
  - 减少博弈树的分枝，不影响决策的最优性，即：等同于MiniMax决策

function AlphaBeta-Decision(state)

If player(state) = MAX

return $\arg\max_a$ AlphaBeta(Result($state, a$), $-\infty, \infty$)

If player(state) = MIN

return $\arg\min_a$ AlphaBeta(Result($state, a$), $-\infty, \infty$)

如何实现AlphaBeta(state, α, β)函数？

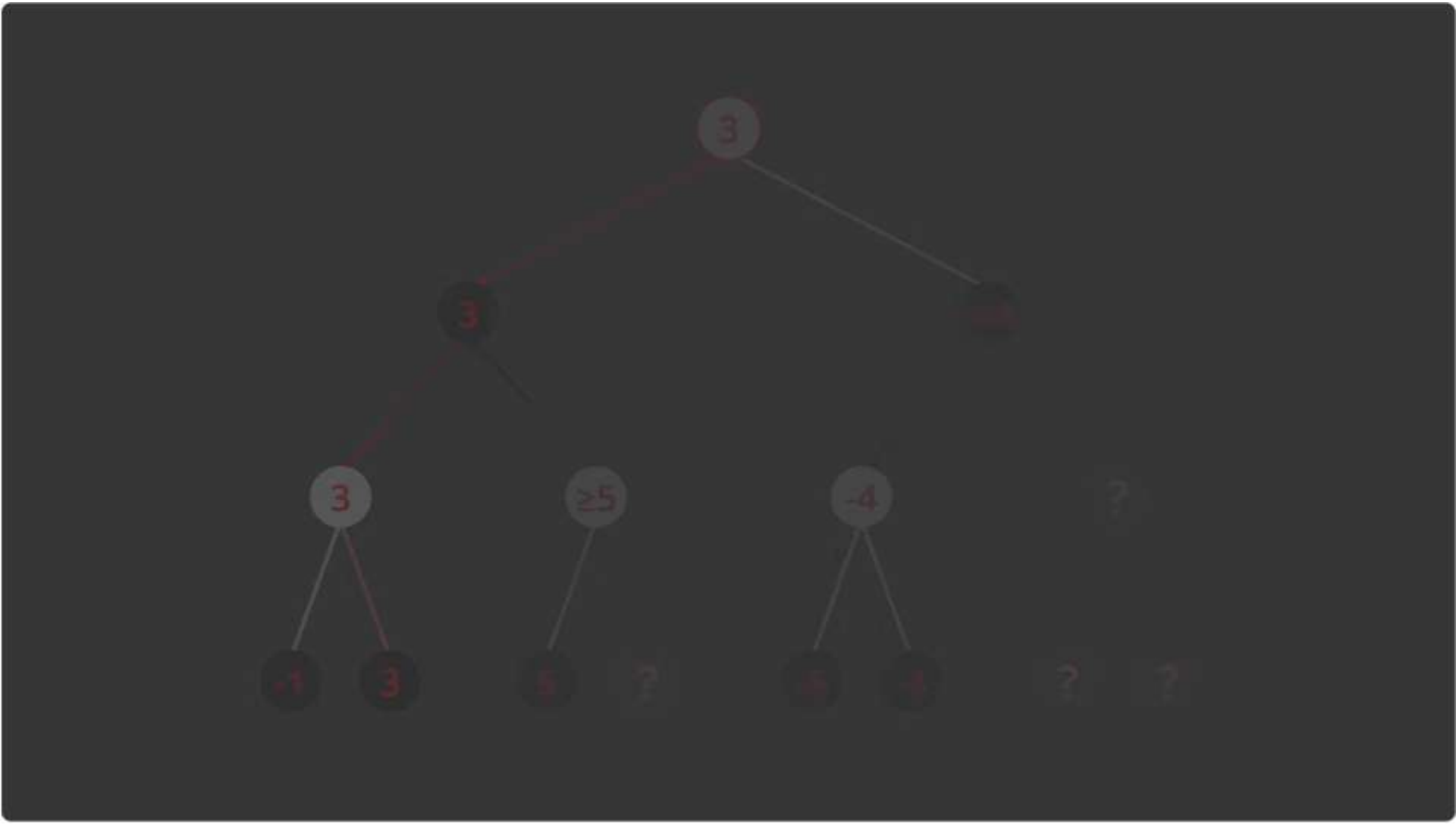
α-β 剪枝

```
function AlphaBeta(state, α, β)
If Terminal-Test(state) return Utility(state)
If player(state) = MAX {
        v = -∞
        for each child {
           v = max(v, AlphaBeta(child, α, β))
           α = max(α, v)
           if β ≤ α then break // 裁剪 } }
 else {
        v = ∞
        for each child {
           v = min(v, AlphaBeta(child, α, β))
           β = min(β, v)
           if β ≤ α then break // 裁剪 } }
return v//返回值
```

- α: MiniMax值的下界，不断增加
- β: MiniMax值的上届，不断减小
- 裁剪条件：

$$\beta \leq \alpha$$

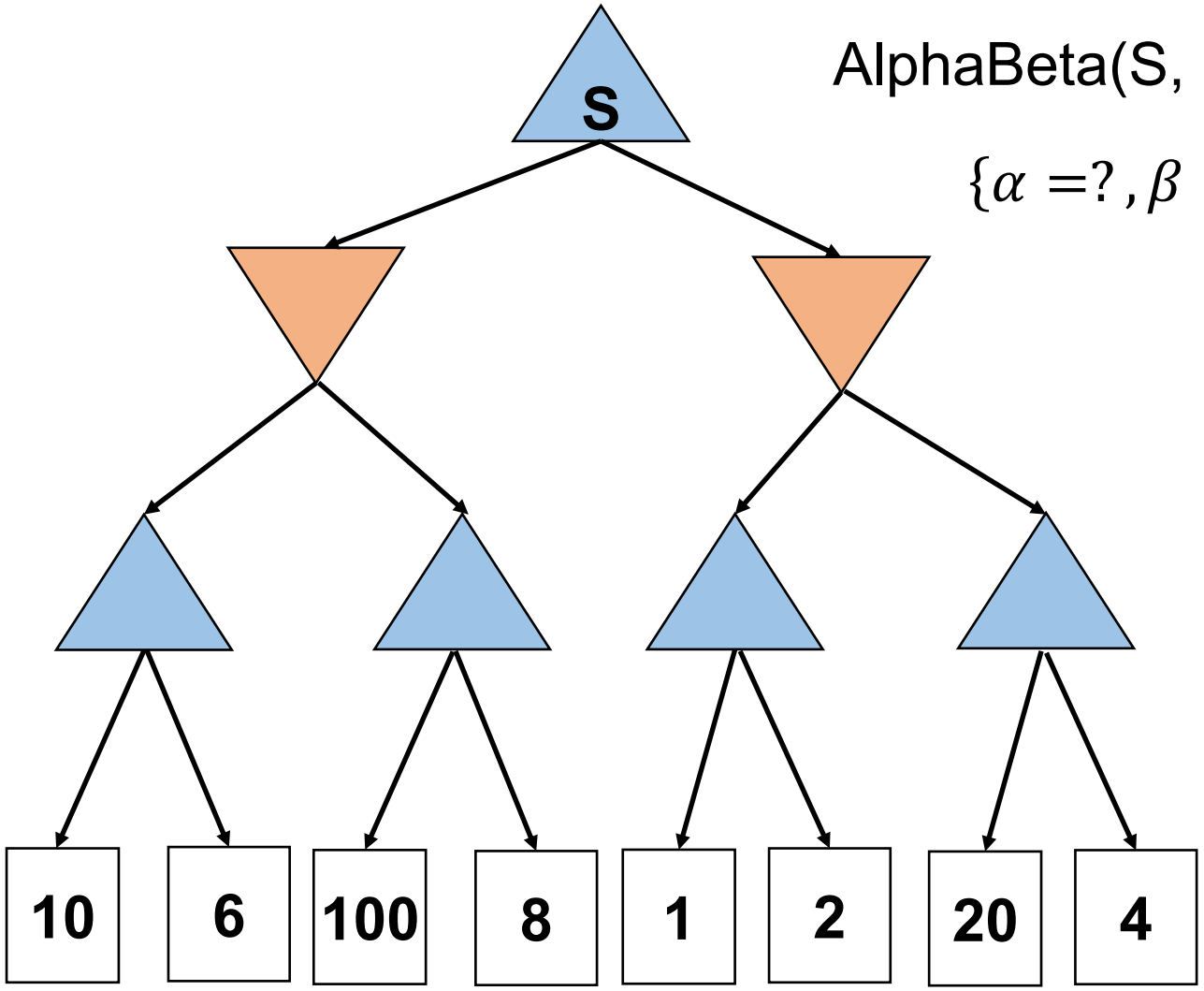
- **关注每个状态：**
  - **向子节点传递α与β**
  - **接受子节点的返回值v，更新α（MAX）或β（MIN）**
  - **返回值v**


3
3
3
≥5
-4
?
-1
3
5
?
-5
-1
?
?

α-β 剪枝：算法测试

AlphaBeta(S, $-\infty, \infty$)

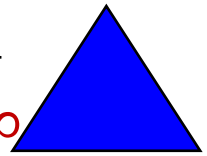
$\{\alpha = ?, \beta = ?\}, v = ?$



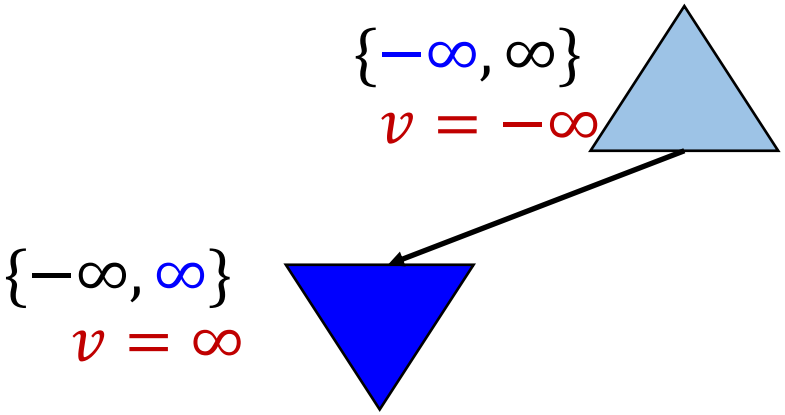
α-β 剪枝：算法测试

$\{-\infty, \infty\}$

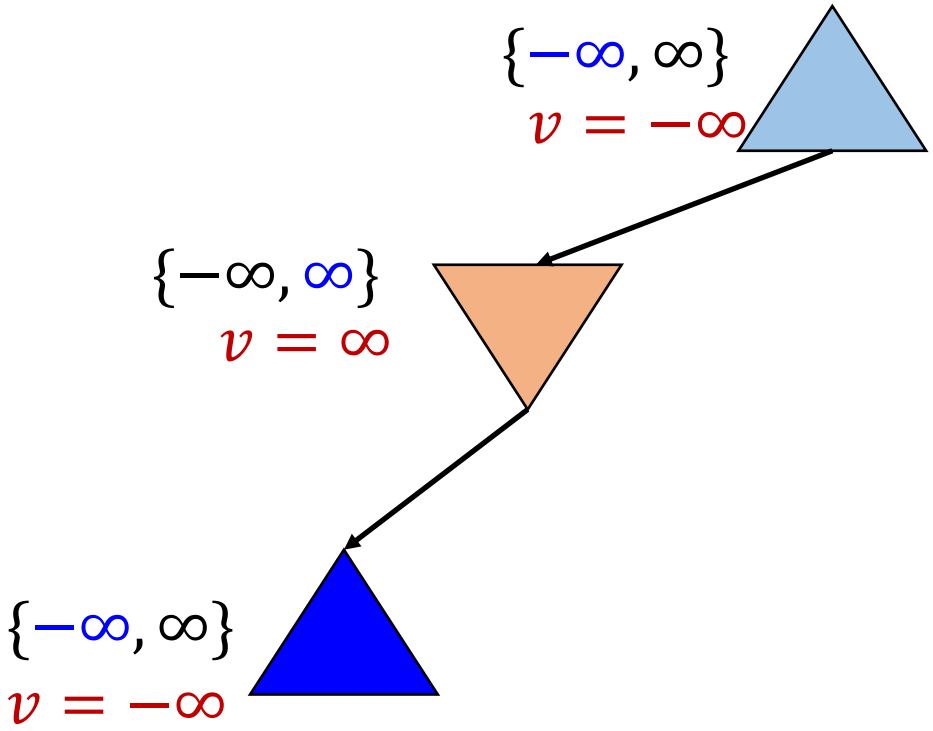
$v = -\infty$



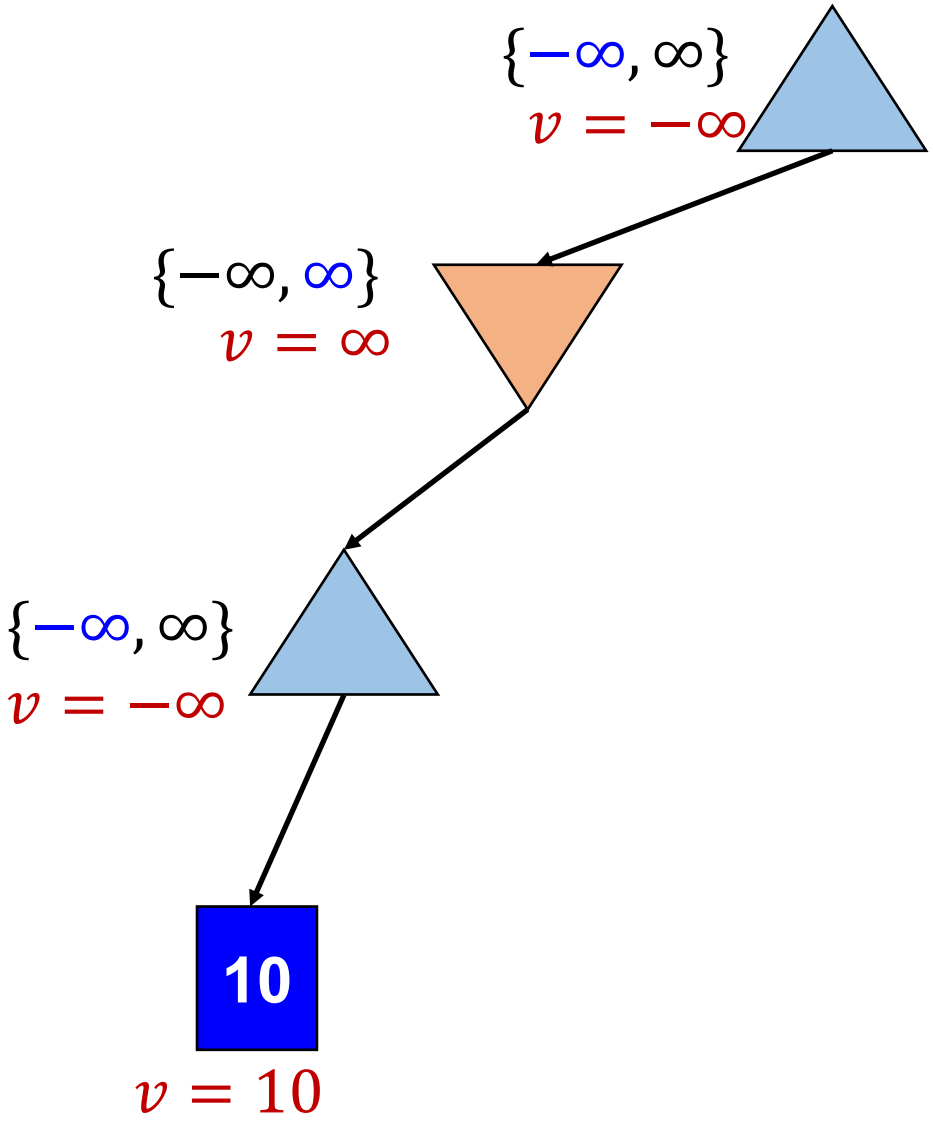
α-β 剪枝：算法测试



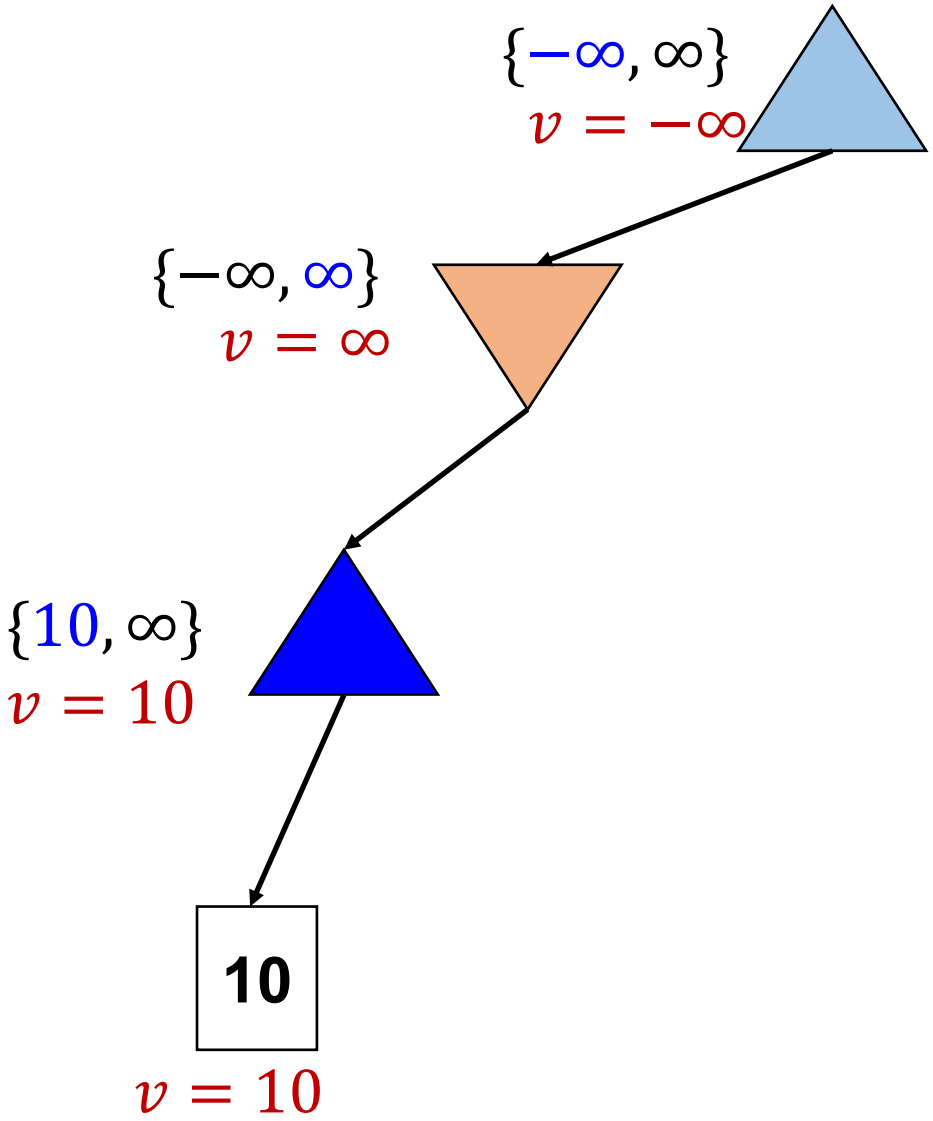
α-β 剪枝：算法测试



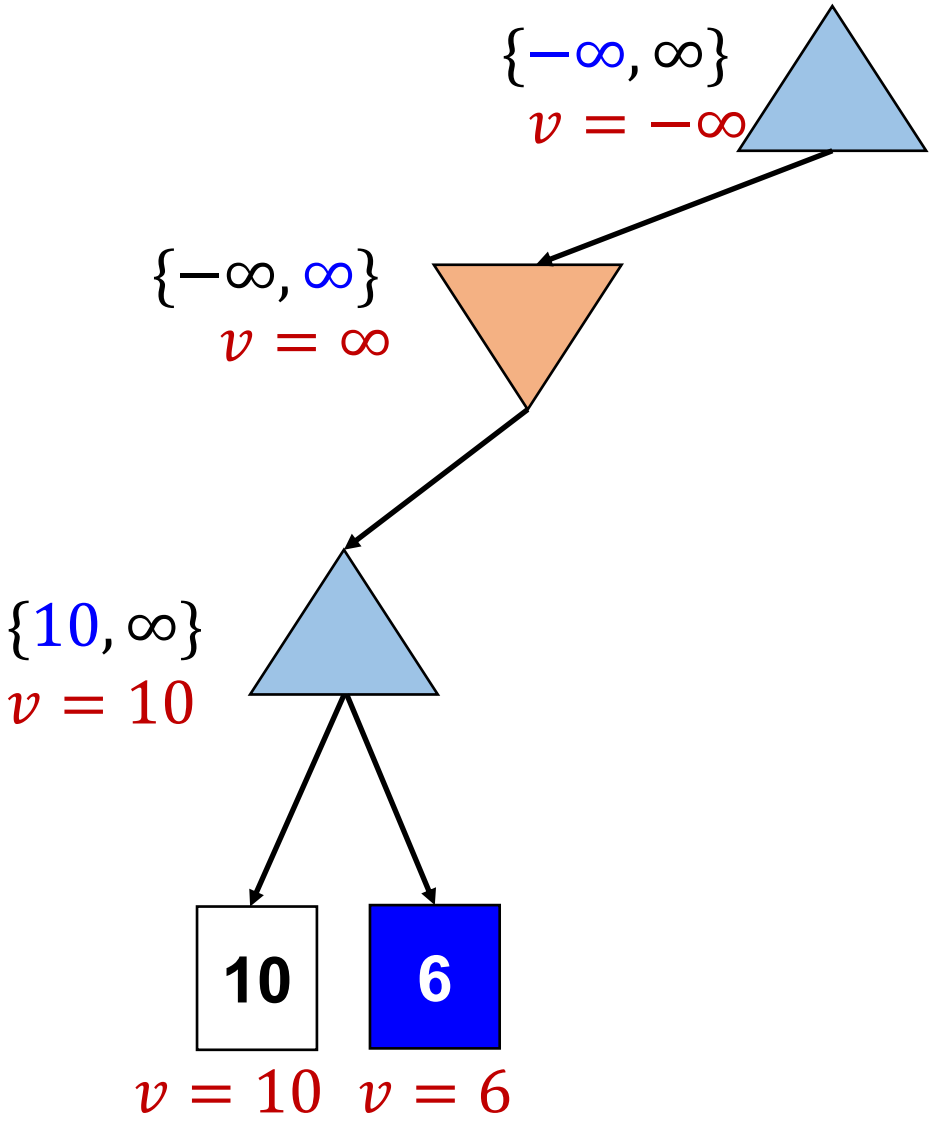
α-β 剪枝：算法测试



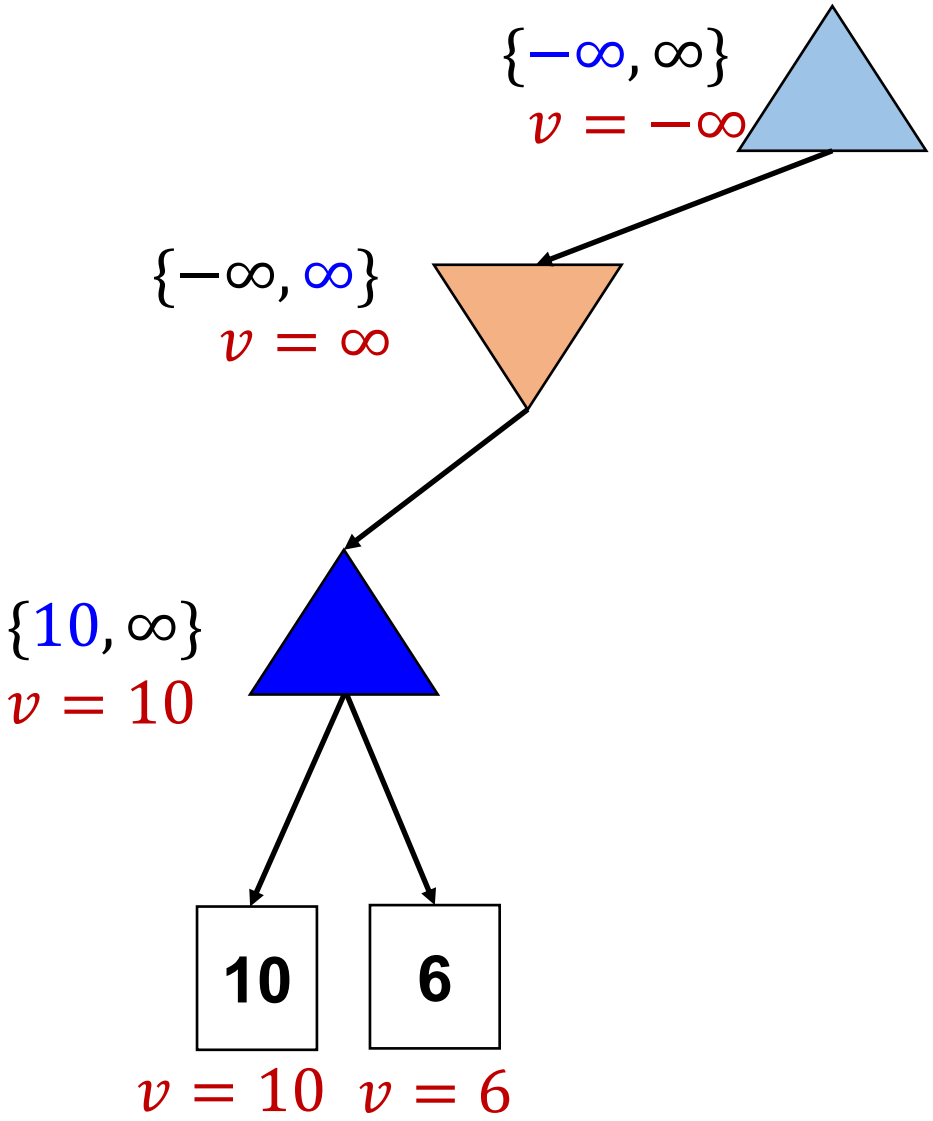
α-β 剪枝：算法测试



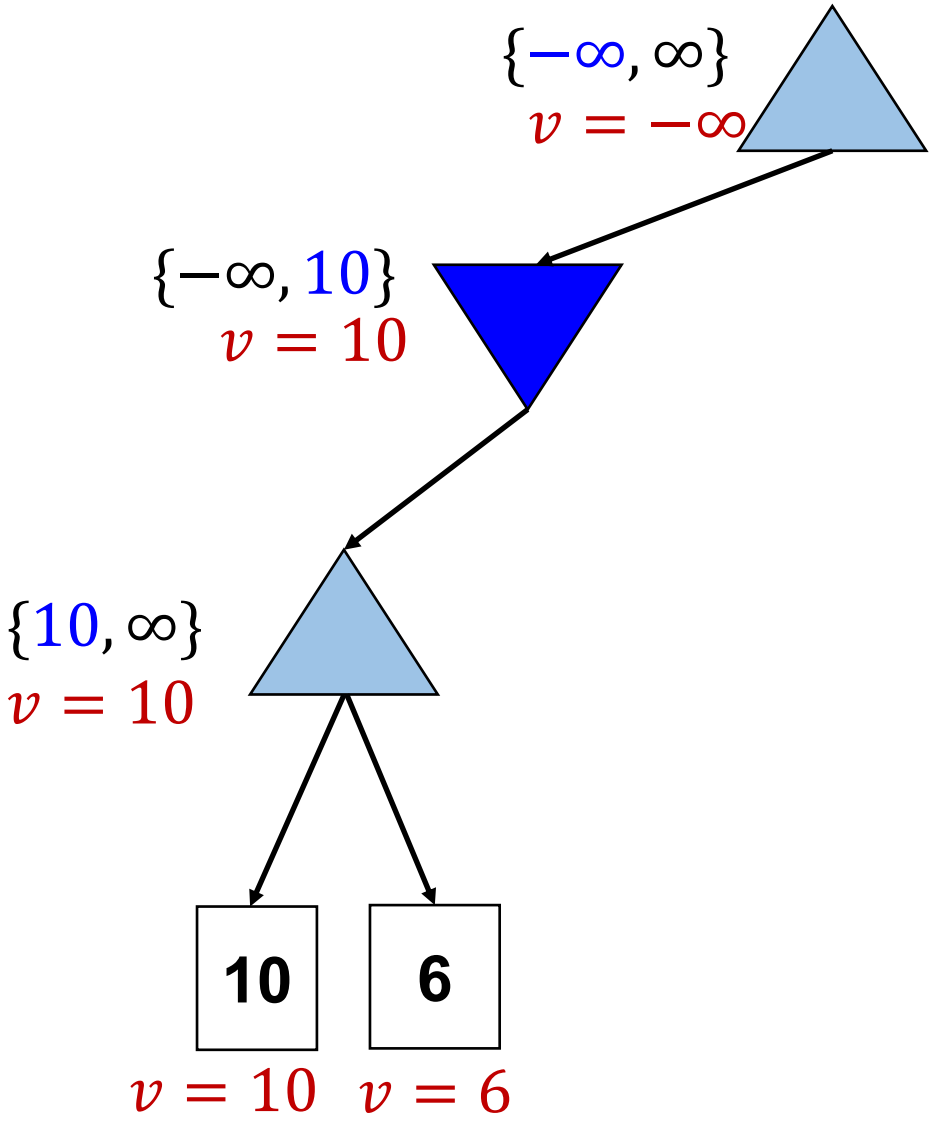
α-β 剪枝：算法测试



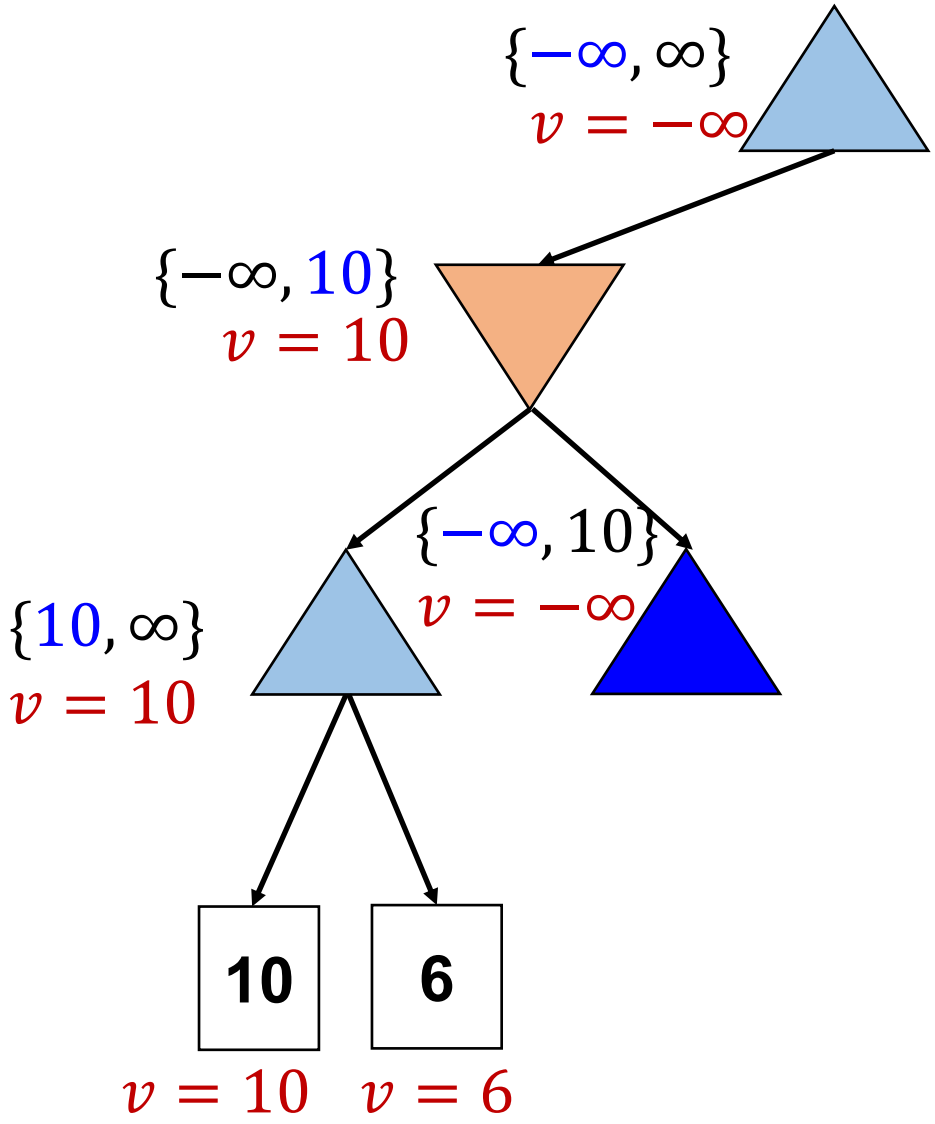
α-β 剪枝：算法测试



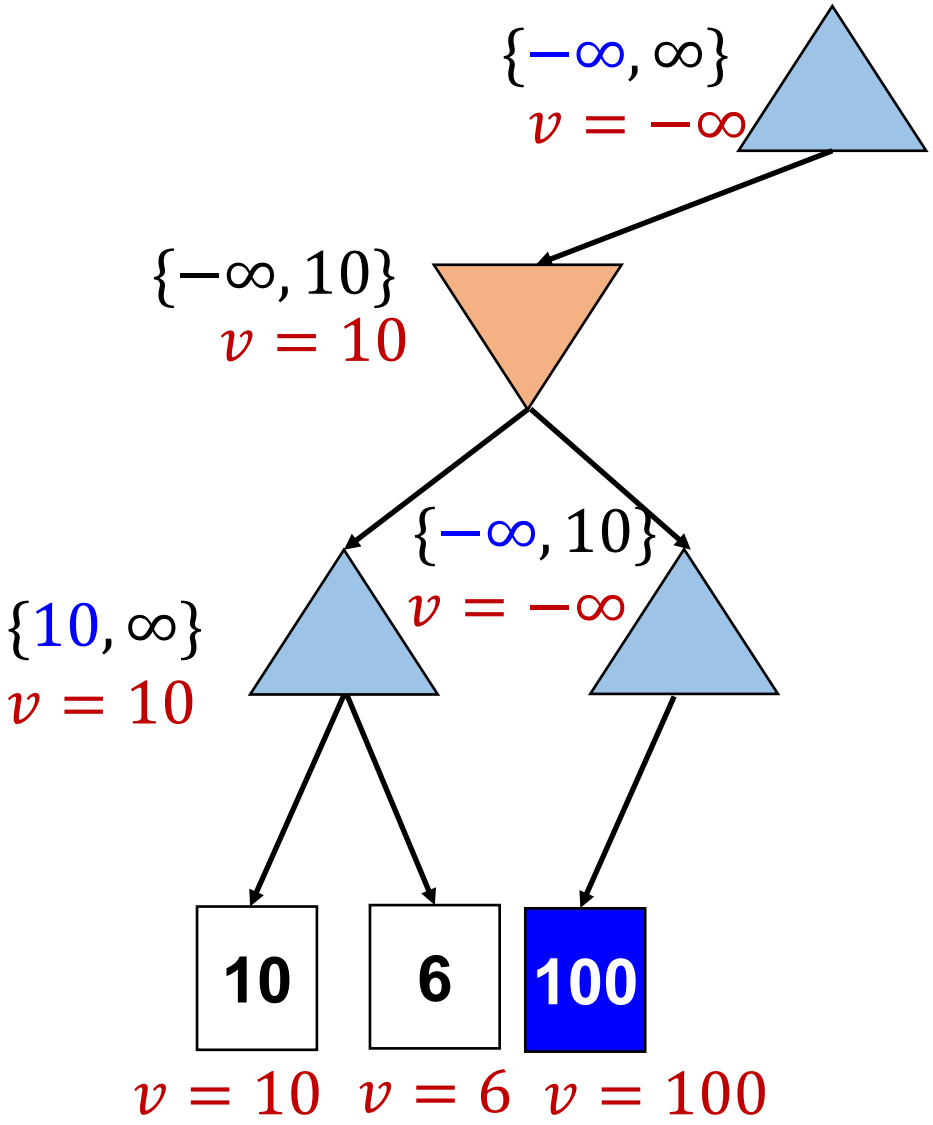
α-β 剪枝：算法测试



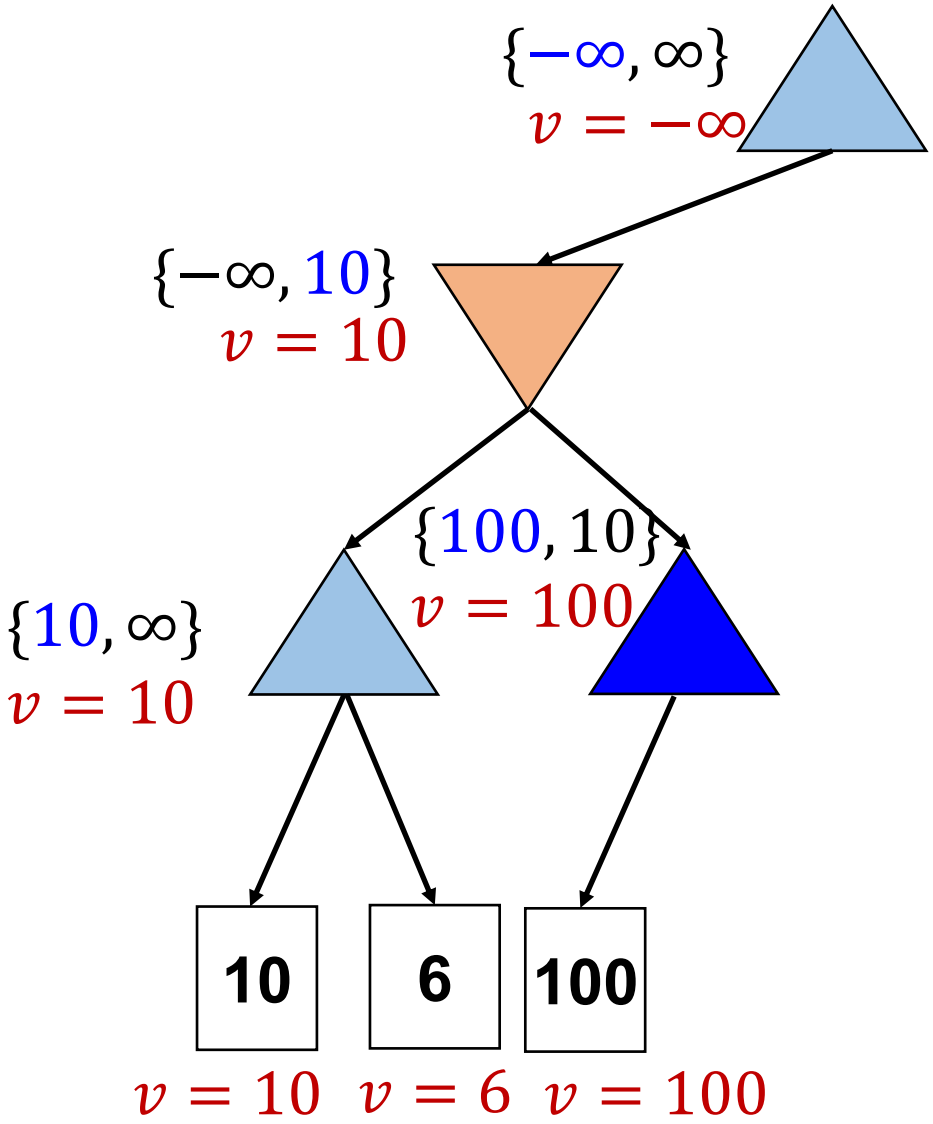
α-β 剪枝：算法测试



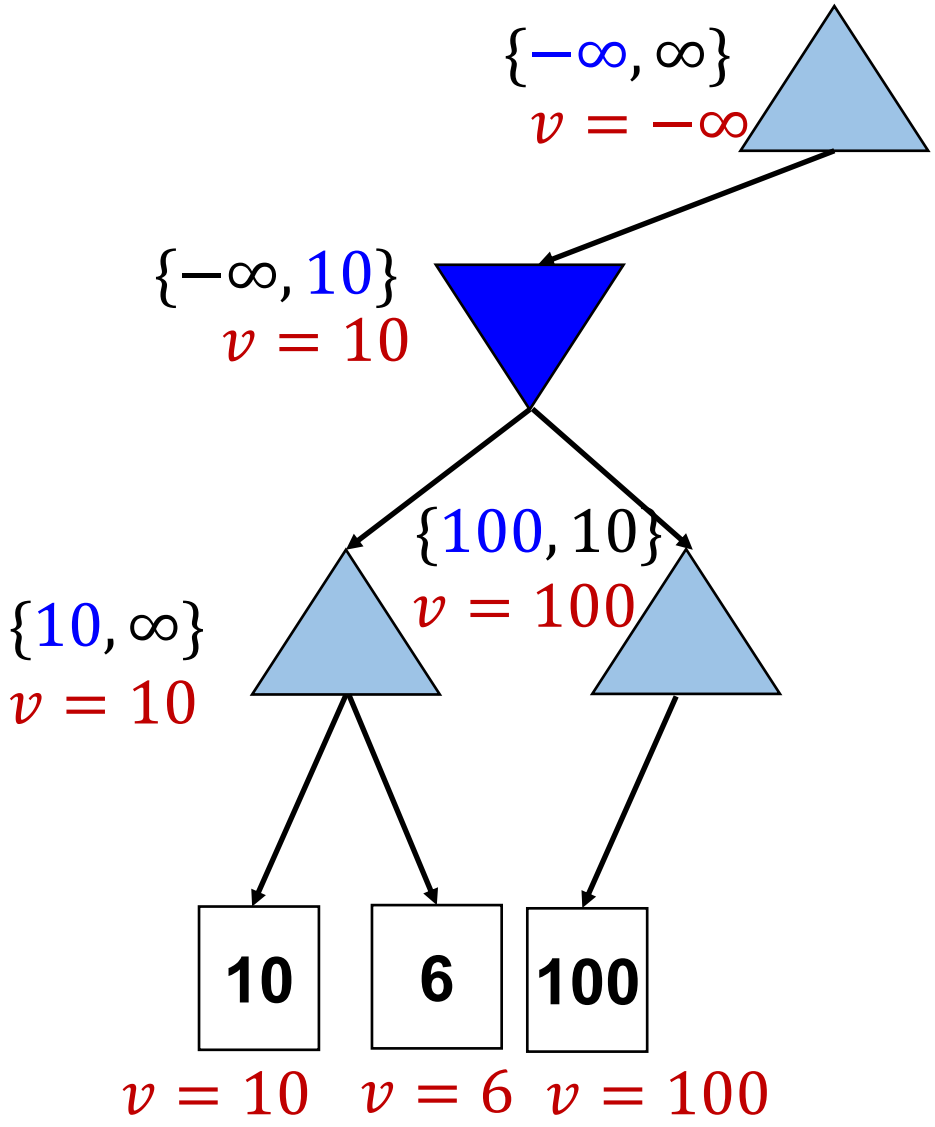
α-β 剪枝：算法测试



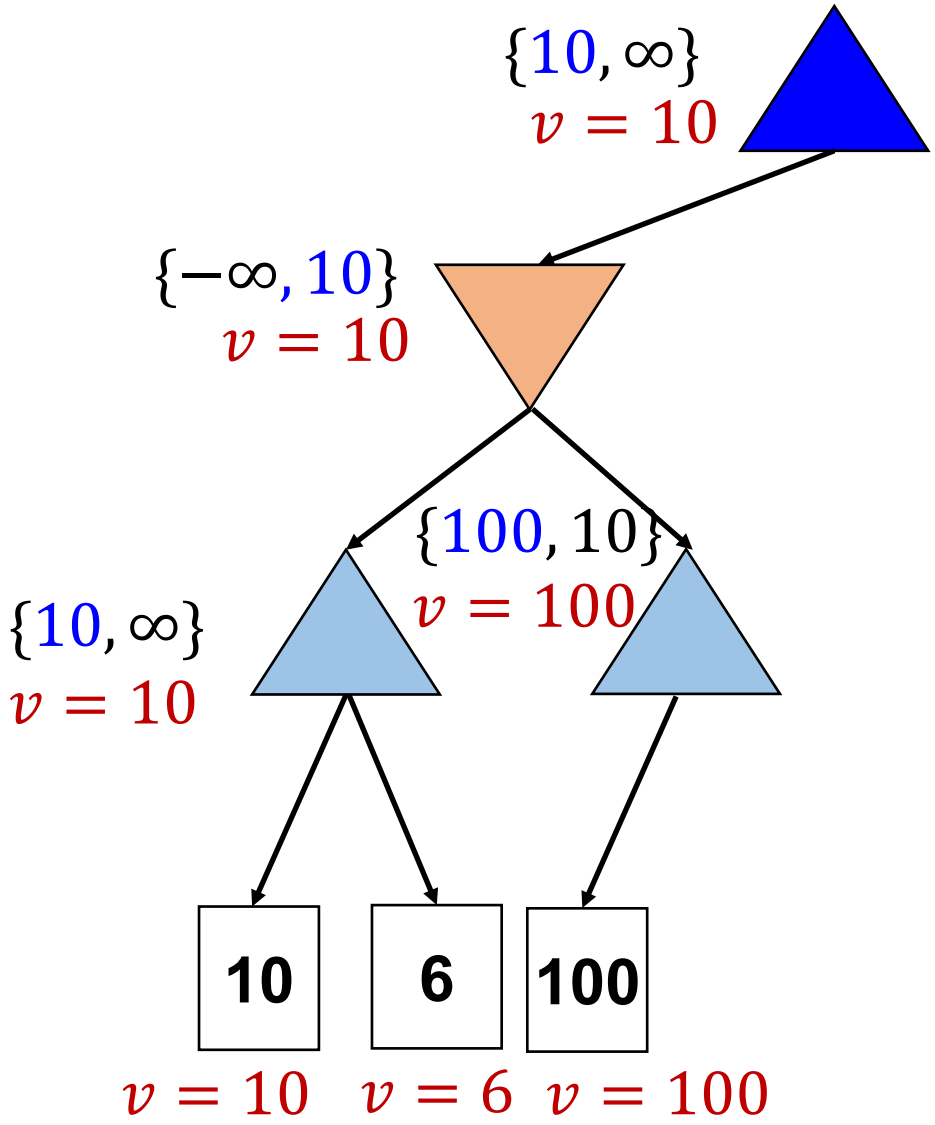
α-β 剪枝：算法测试



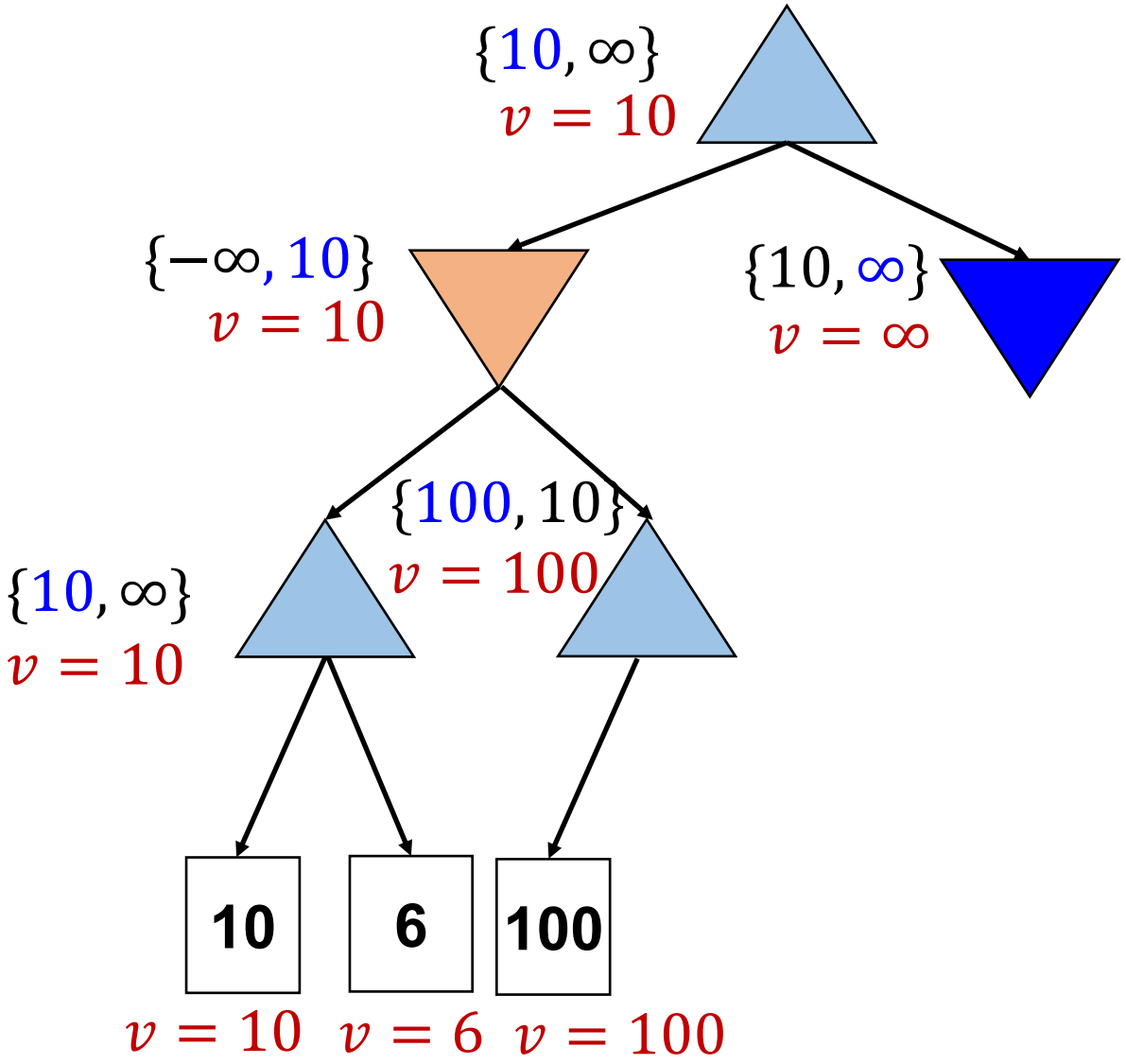
α-β 剪枝：算法测试



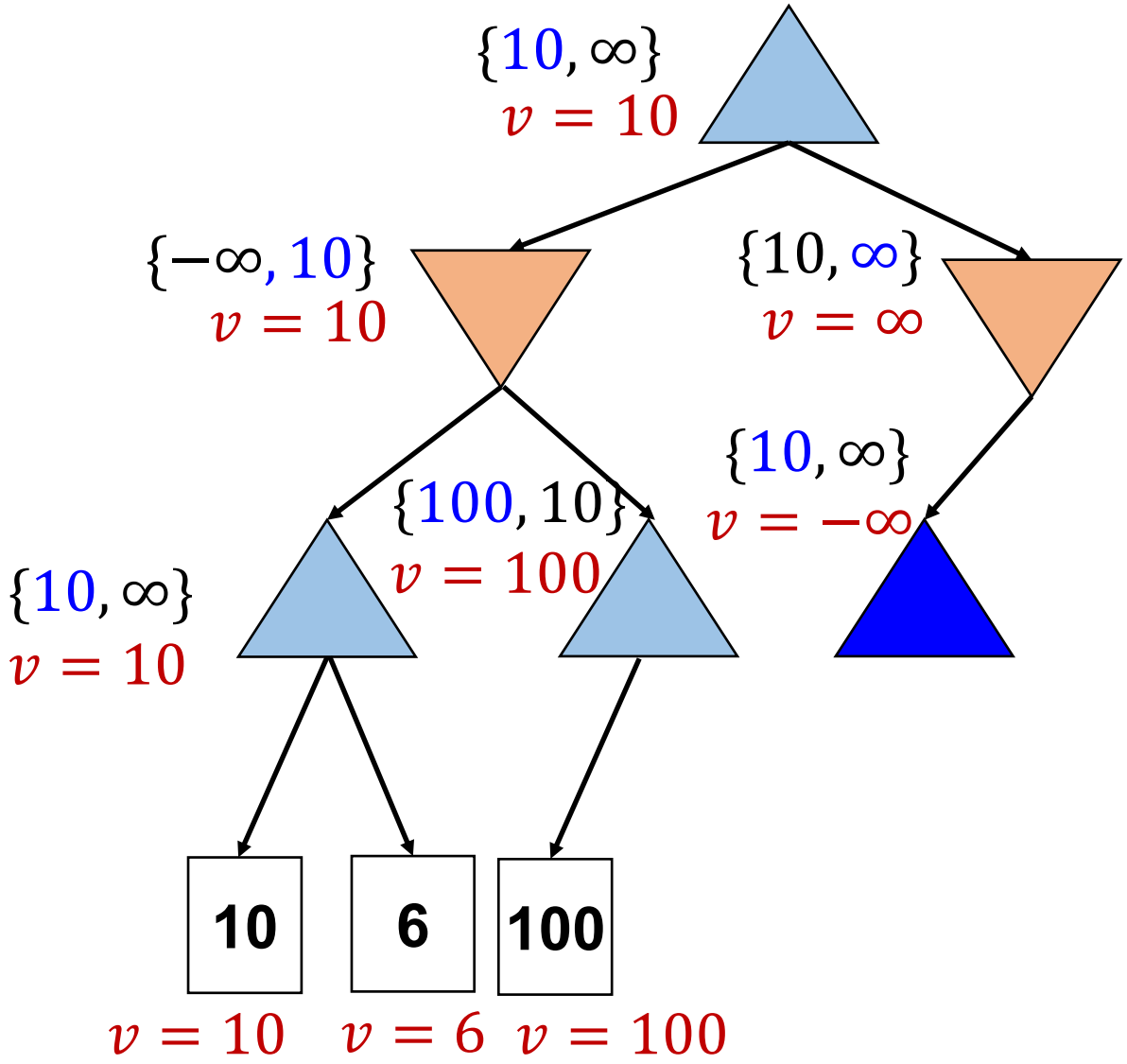
α-β 剪枝：算法测试



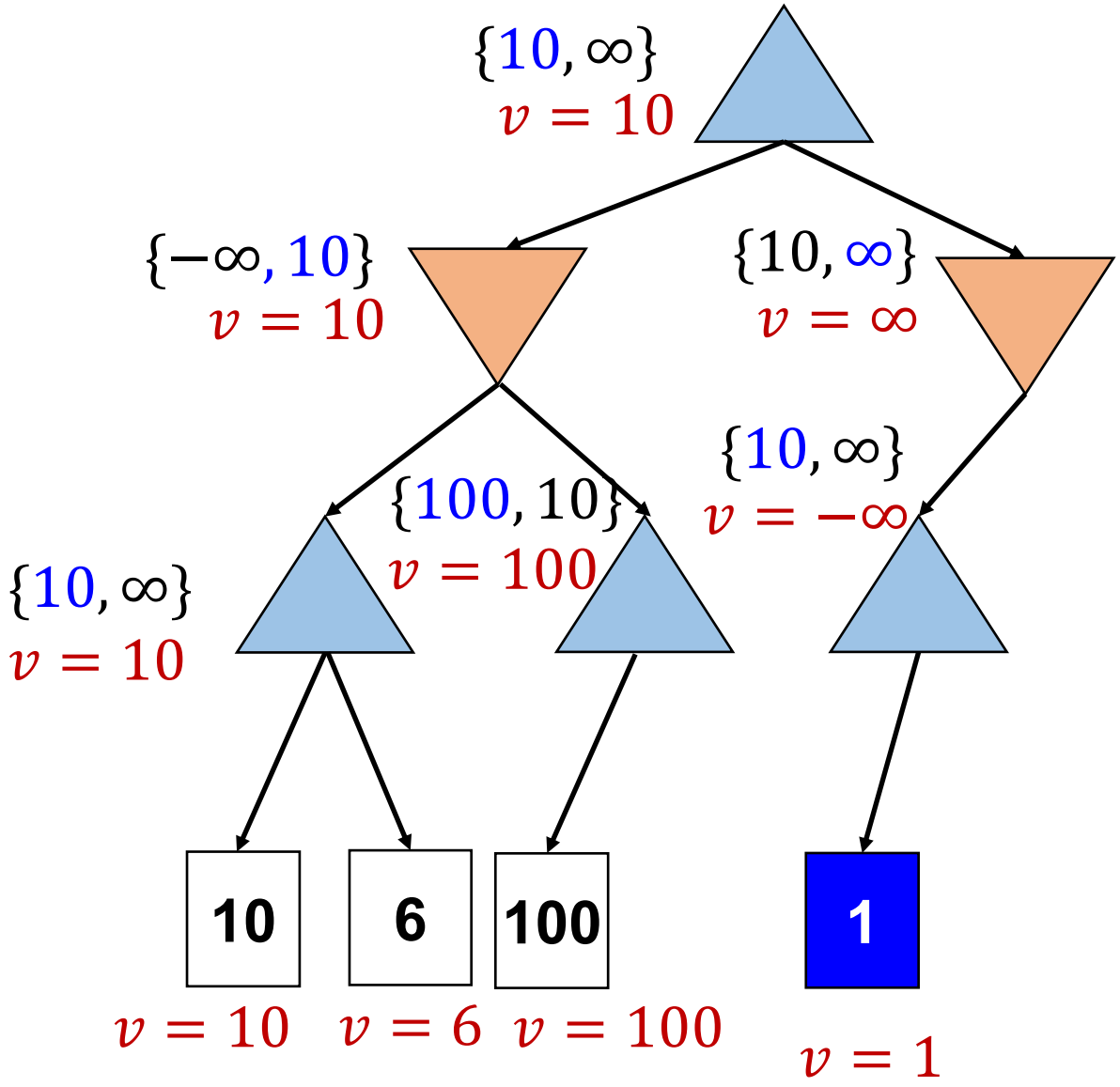
α-β 剪枝：算法测试



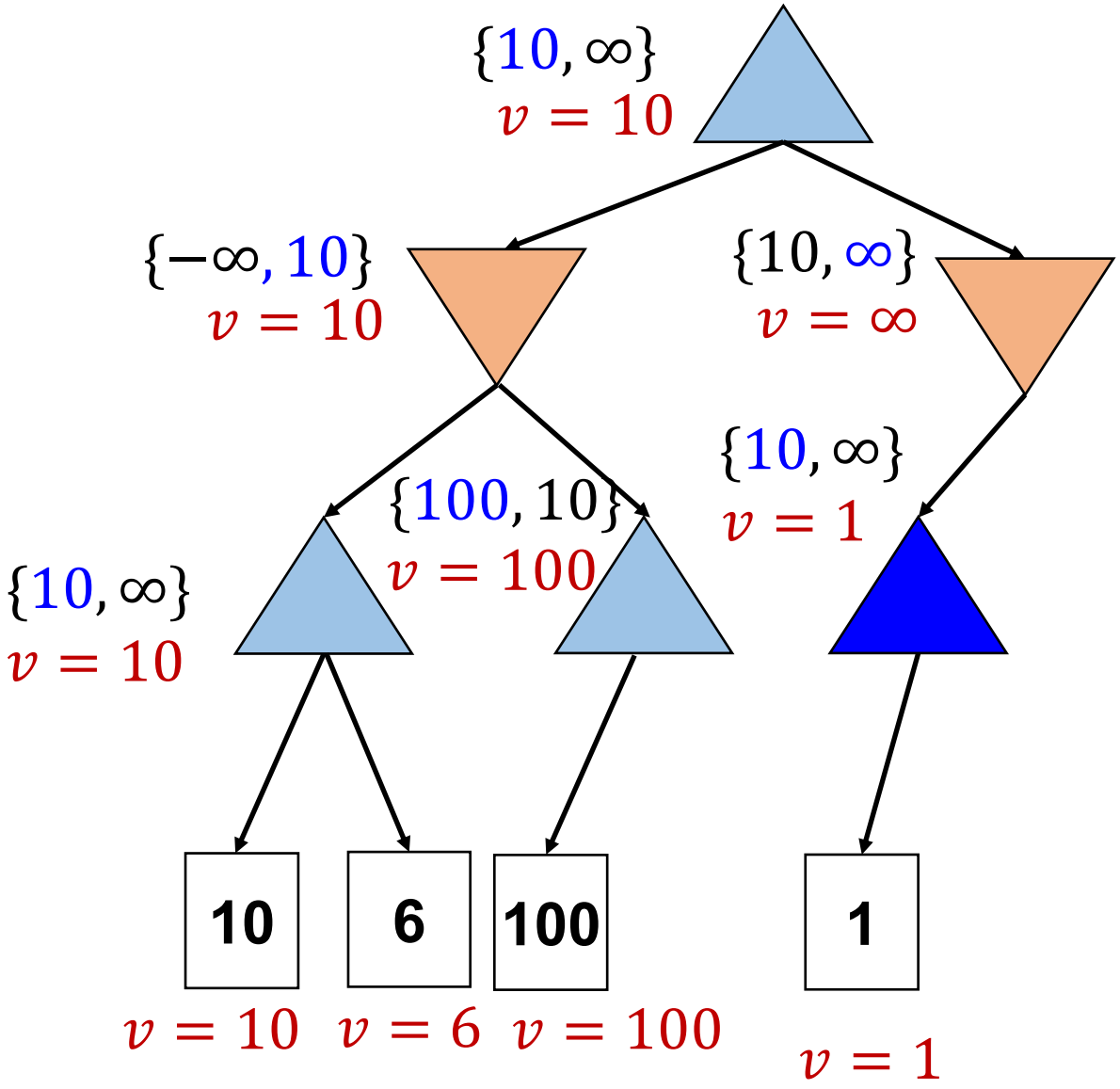
α-β 剪枝：算法测试



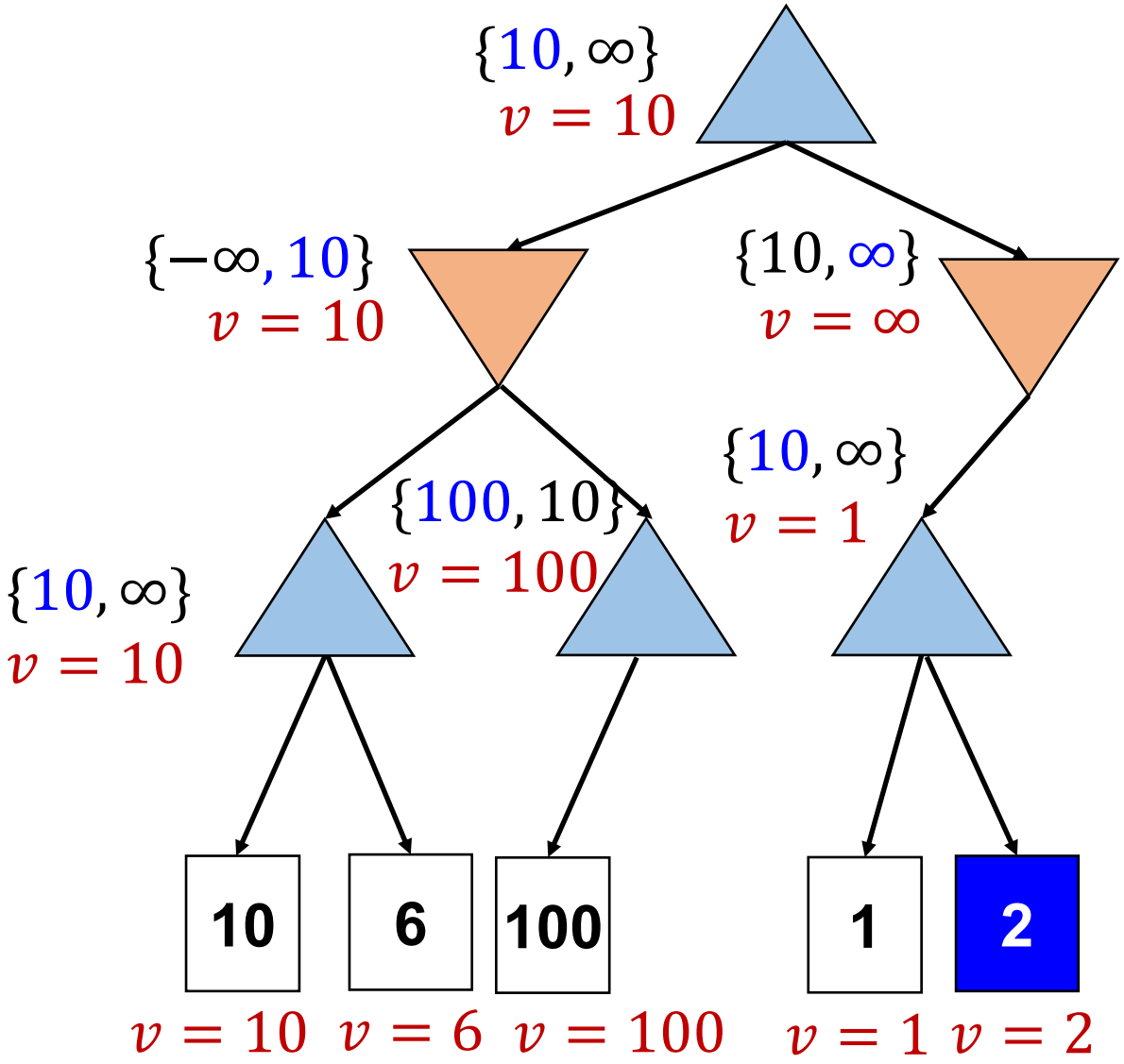
α-β 剪枝：算法测试



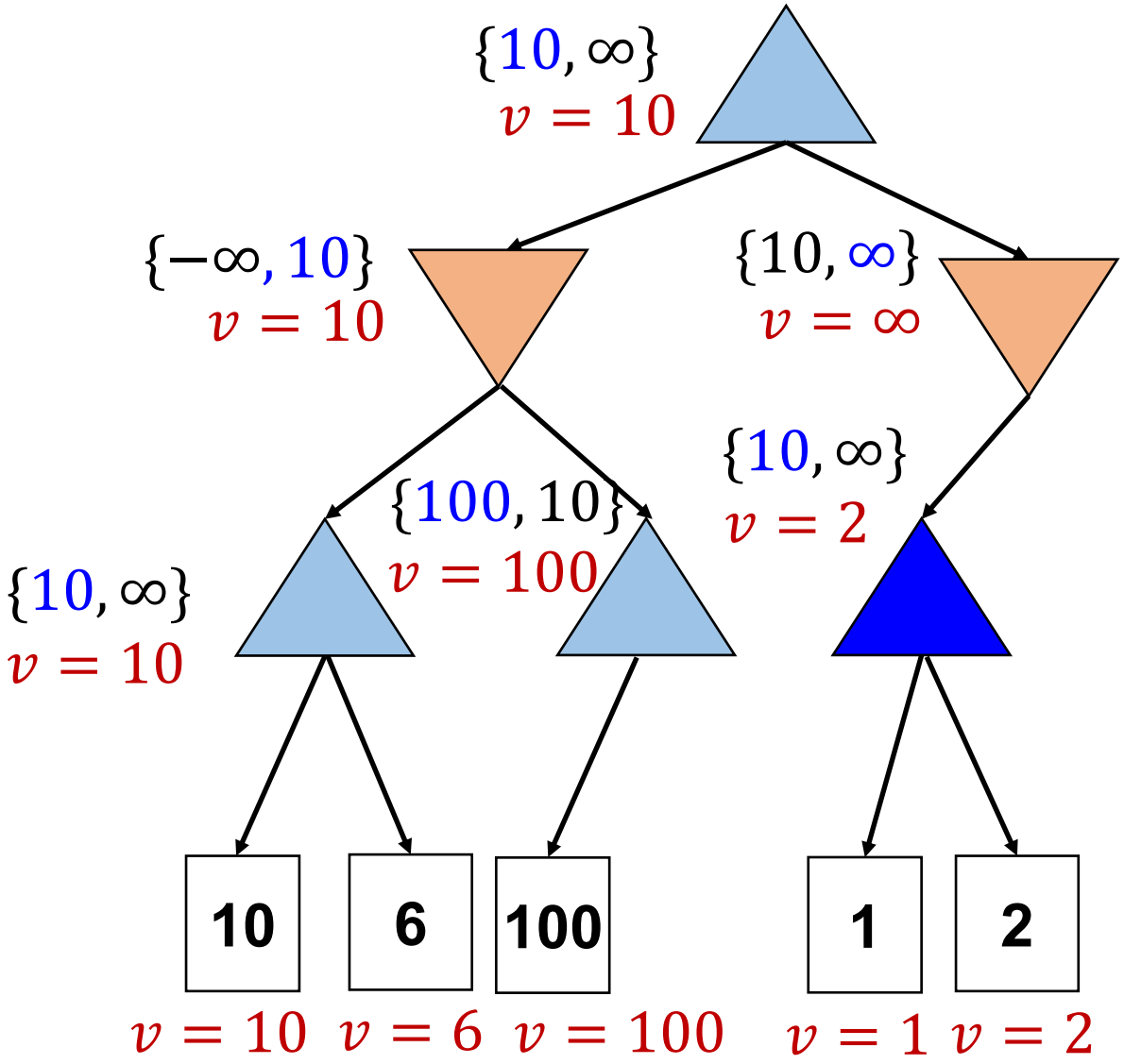
α-β 剪枝：算法测试



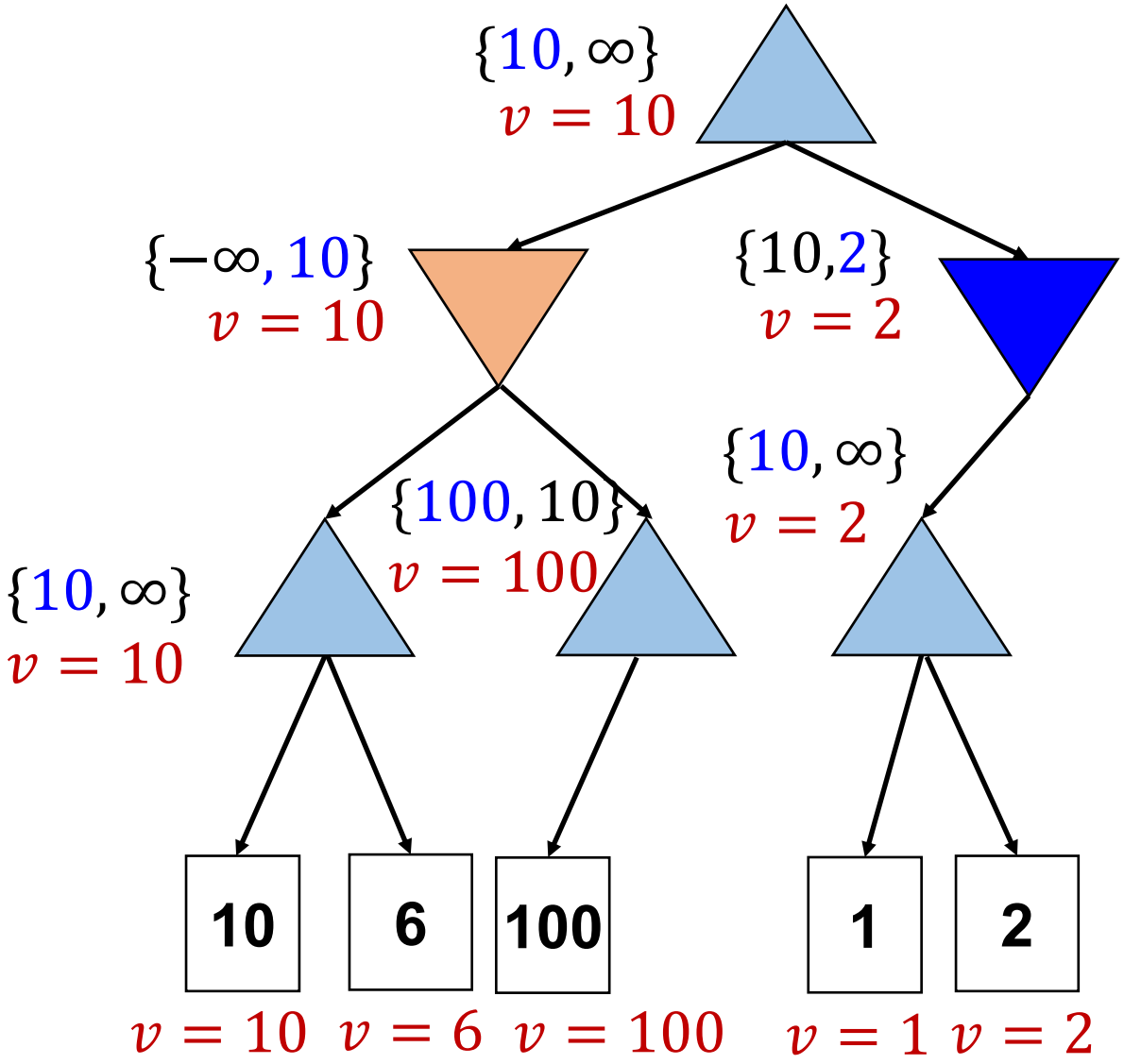
α-β 剪枝：算法测试


{10, ∞}
v = 10
{−∞, 10}
v = 10
{10, ∞}
v = ∞
{100, 10}
v = 100
{10, ∞}
v = 1
{10, ∞}
v = 10
10
6
100
1
2
v = 10
v = 6
v = 100
v = 1
v = 2

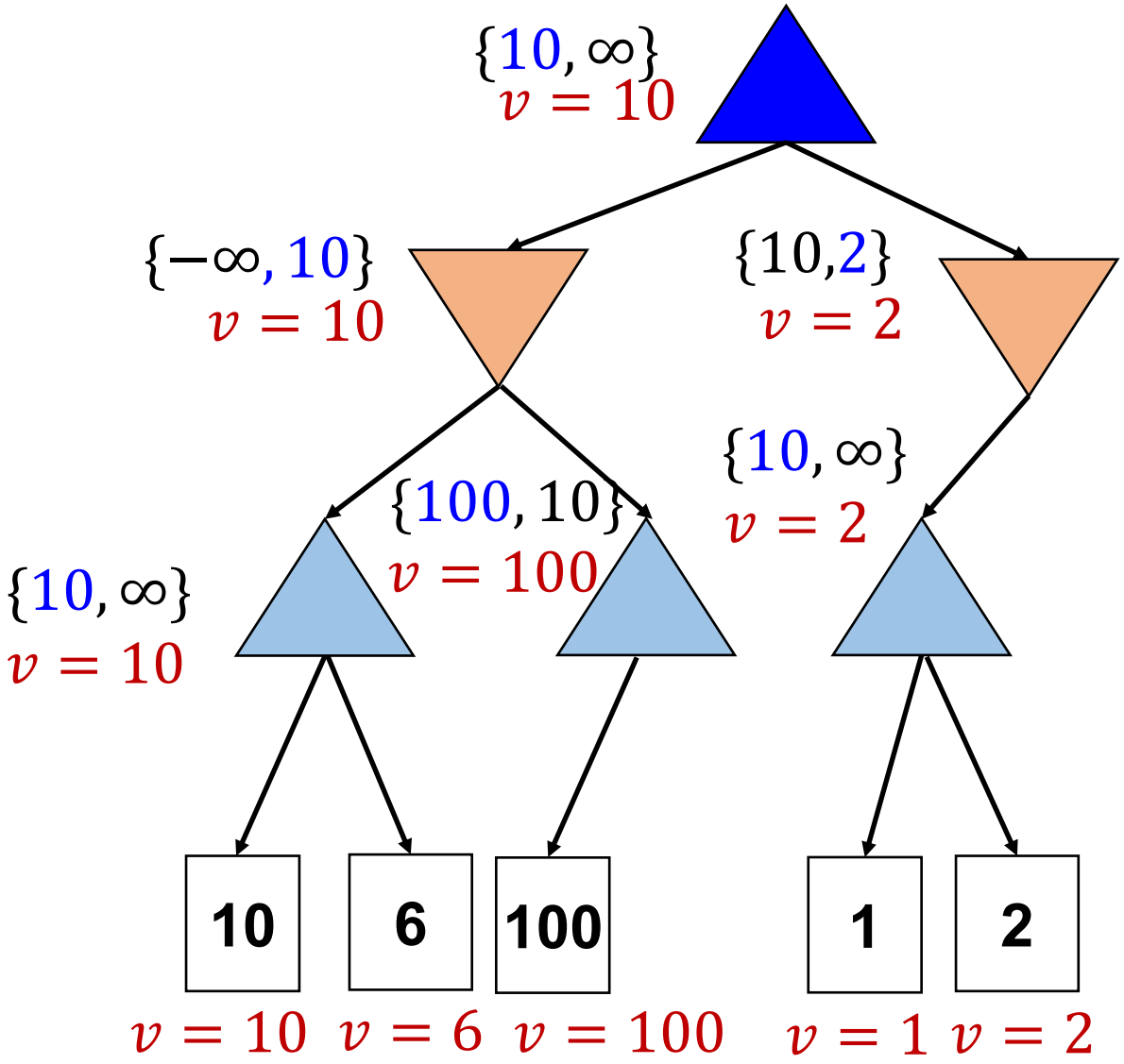
α-β 剪枝：算法测试


{10, ∞}
v = 10
{−∞, 10}
v = 10
{10, ∞}
v = ∞
{10, ∞}
v = 2
{100, 10}
v = 100
{10, ∞}
v = 10
10
6
100
1
2
v = 10
v = 6
v = 100
v = 1
v = 2

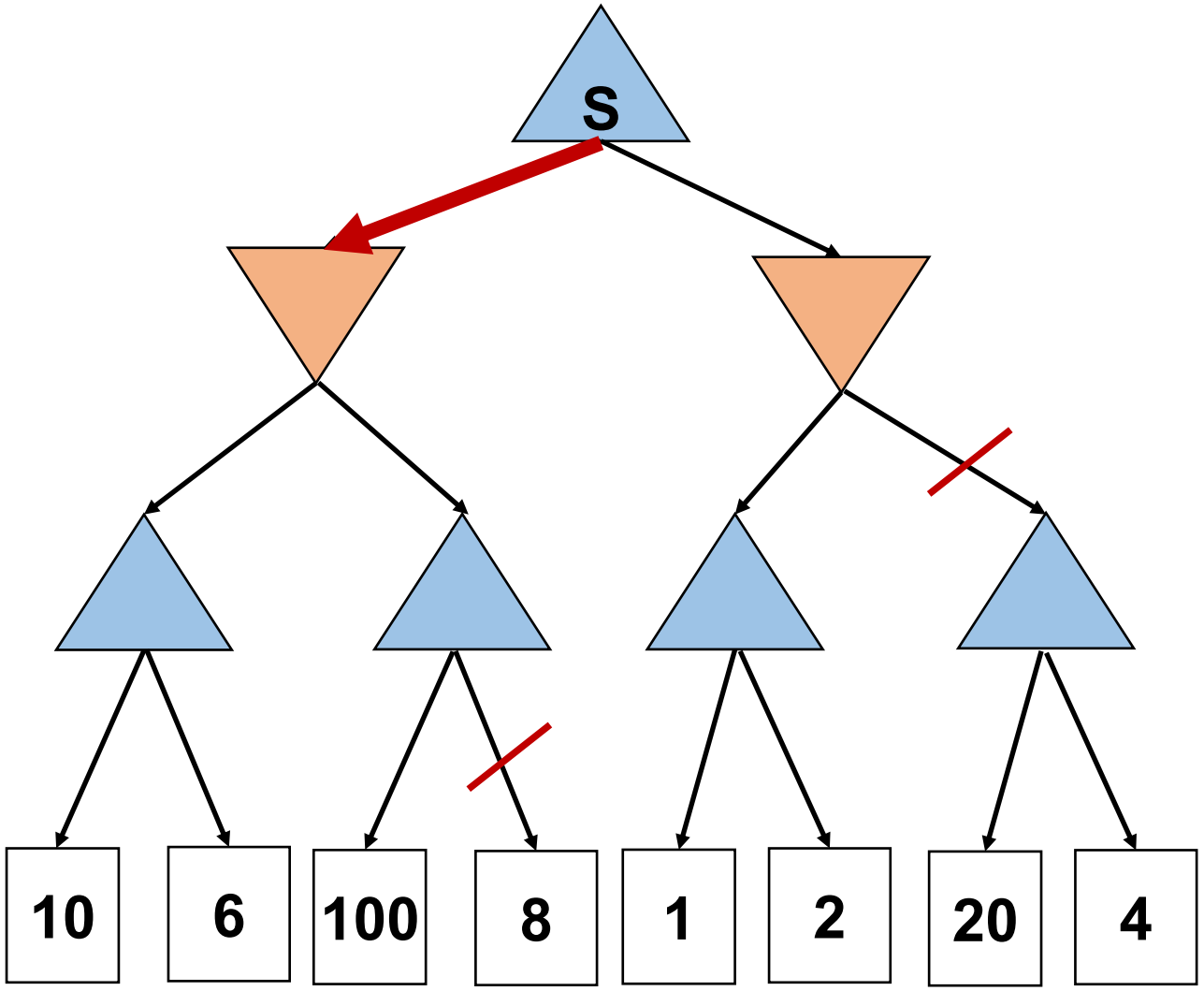
α-β 剪枝：算法测试


{10, ∞}
v = 10
{−∞, 10}
v = 10
{10,2}
v = 2
{10, ∞}
v = 10
{100, 10}
v = 100
{10, ∞}
v = 2
10
6
100
1
2
v = 10
v = 6
v = 100
v = 1
v = 2

α-β 剪枝：算法测试


{10, ∞}
v = 10
{−∞, 10}
v = 10
{10,2}
v = 2
{10, ∞}
v = 10
{100, 10}
v = 100
{10, ∞}
v = 2
10
6
100
1
2
v = 10
v = 6
v = 100
v = 1
v = 2

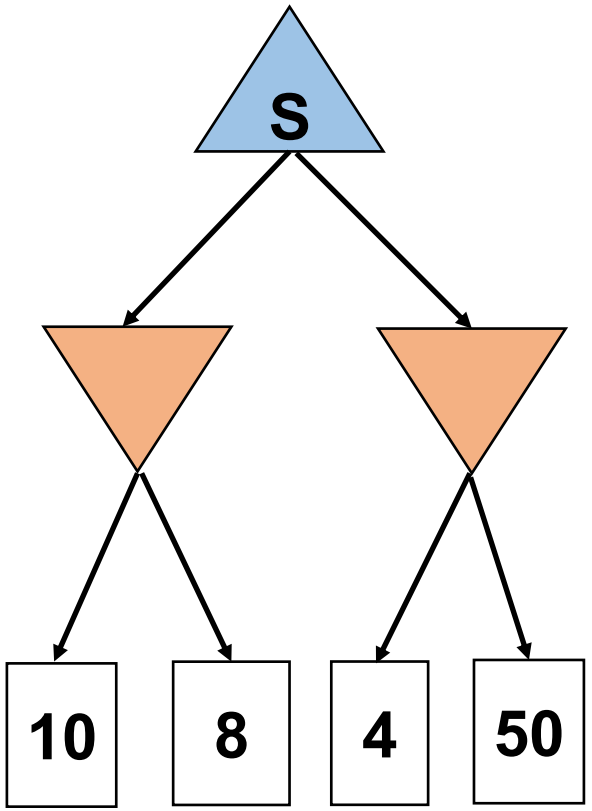
α-β 剪枝结果



α-β 剪枝：练习

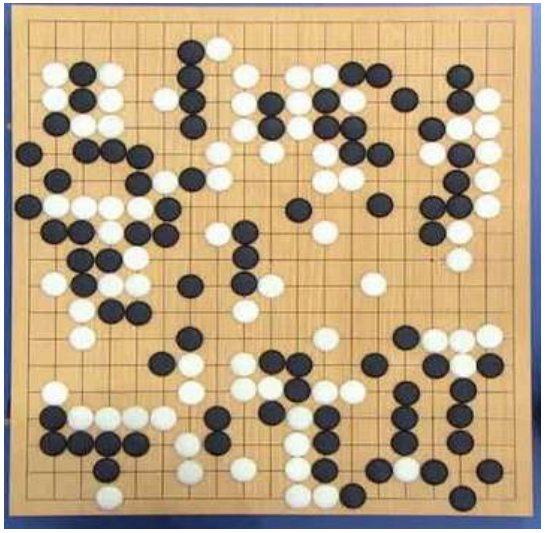
```
function AlphaBeta(state, α, β)
If Terminal-Test(state) return Utility(state)
If player(state) = MAX {
        v = -∞
        for each child {
            v = max(v, AlphaBeta(child, α, β))
            α = max(α, v)
            if β ≤ α then break // 裁剪 } }
else {
        v = ∞
        for each child {
            v = min(v, AlphaBeta(child, α, β))
            β = min(β, v)
            if β ≤ α then break // 裁剪 } }
return v //返回值
```

AlphaBeta(S, −∞, ∞)



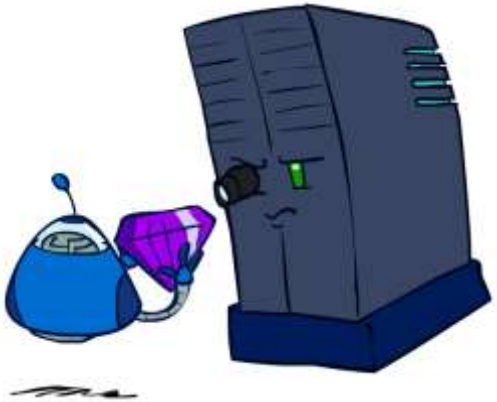
α-β 剪枝的特点

- 裁剪不影响根节点的MiniMax值，不影响决策的最优性
- 中间节点的MiniMax值可能不精确
- 剪枝效率和节点的访问顺序有关
  - 完美顺序下，时间复杂度 $\mathcal{O}(b^{m/2})$ — $\sqrt{b}$分支
  - 随机顺序下，时间复杂度 $\mathcal{O}(b^{3m/4})$ — $b^{3/4}$分支
- 对于困难问题，计算复杂度仍然过高

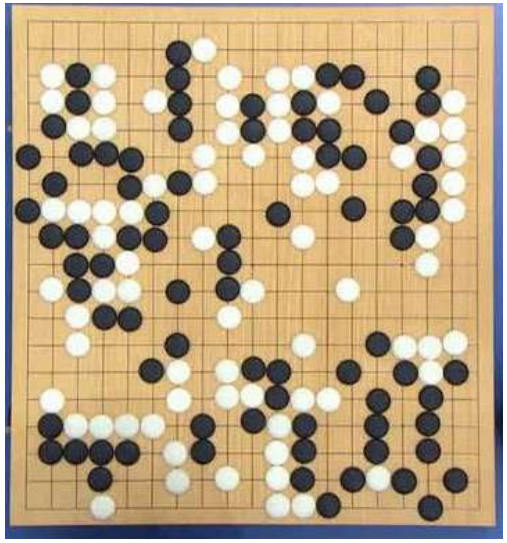


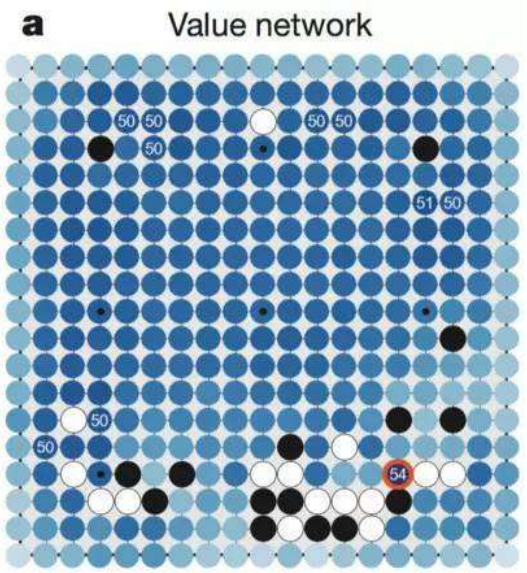
状态评估

- 设计评估函数（evaluation function），估计非终端节点的得分（utilities）



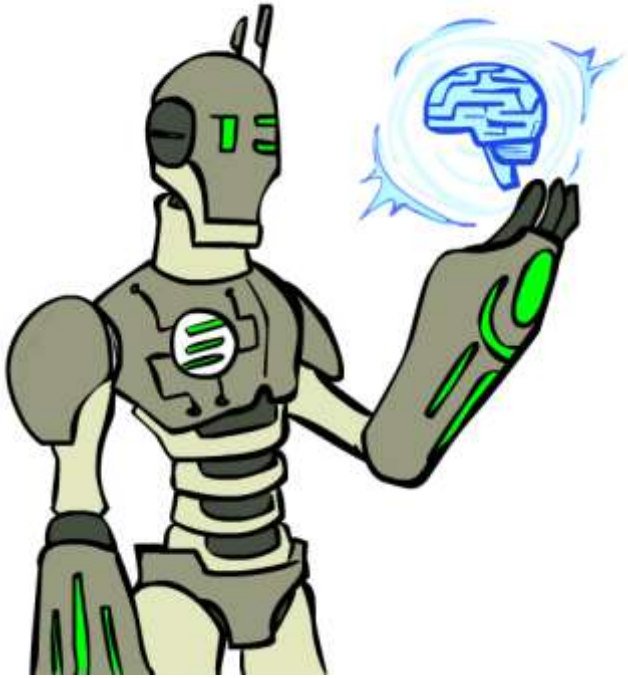
- 理想：得到实际的MiniMax值
- 现实：估计每个状态的获胜概率
  - **Alpha Go：机器学习**





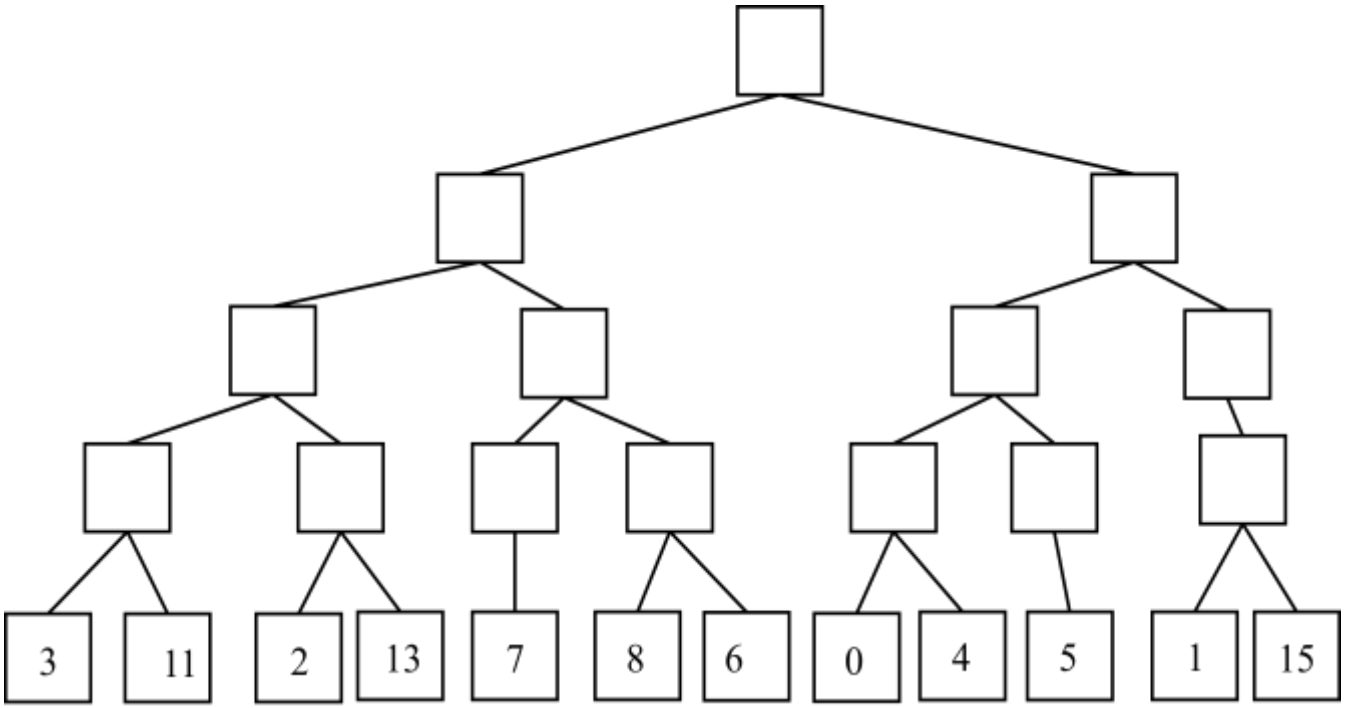
本节课安排

- 博弈搜索问题
  Game Search Problem
  Adversarial Search Problem
- 极小极大搜索
  Minimax Search
- α-β 剪枝
- 练习



练习

1. 考虑下图所示的零和博弈树，假设博弈树都是从MAX层出发，MAX-MIN层交替进行。如果采用$\alpha-\beta$剪枝操作，请在需要剪除的分支上打╳（本小题共6分）。



练习：答案

1. 考虑下图所示的零和博弈树，假设博弈树都是从MAX层出发，MAX-MIN层交替进行。如果采用$\alpha-\beta$剪枝操作，请在需要剪除的分支上打╳（本小题共6分）。