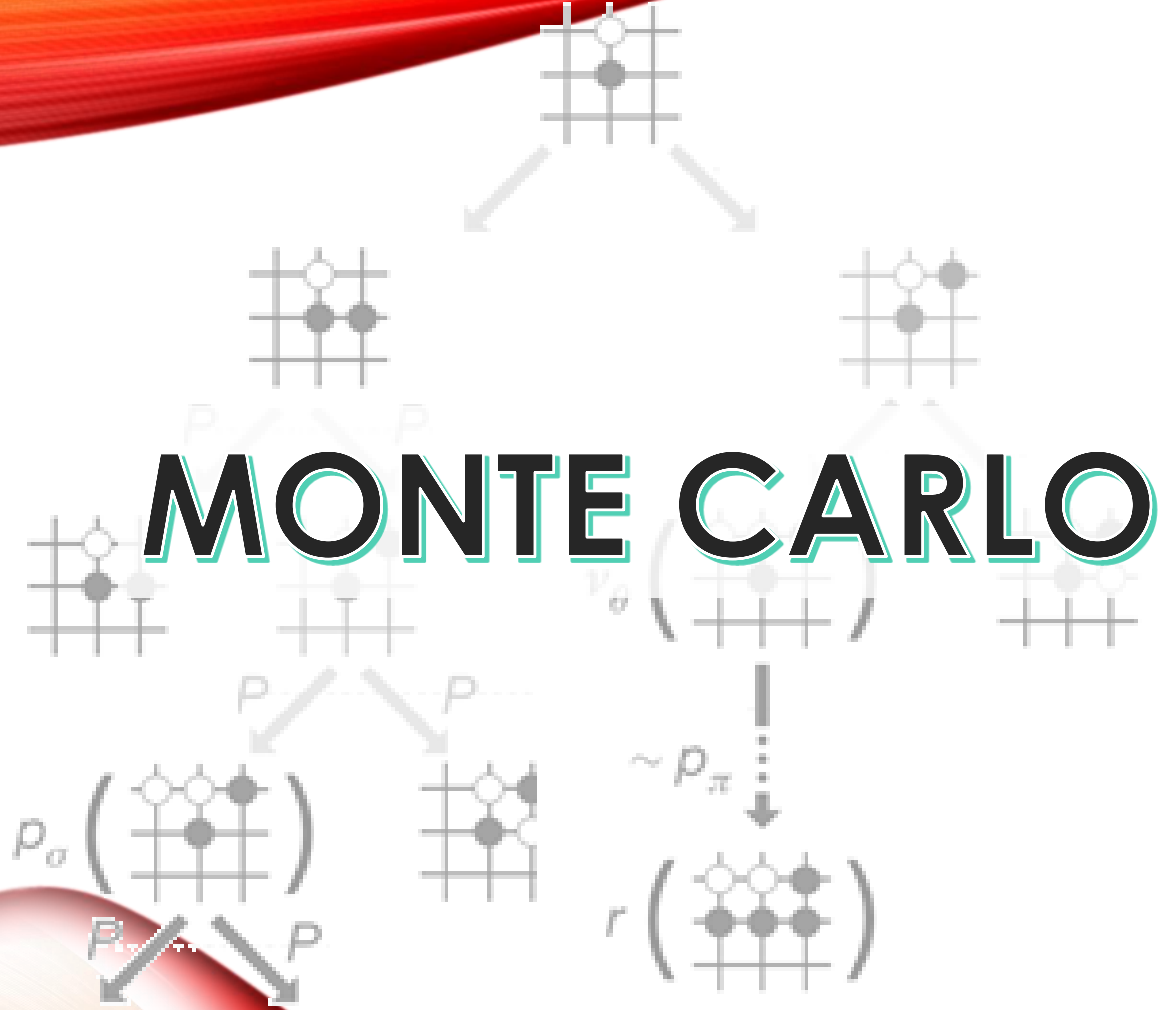
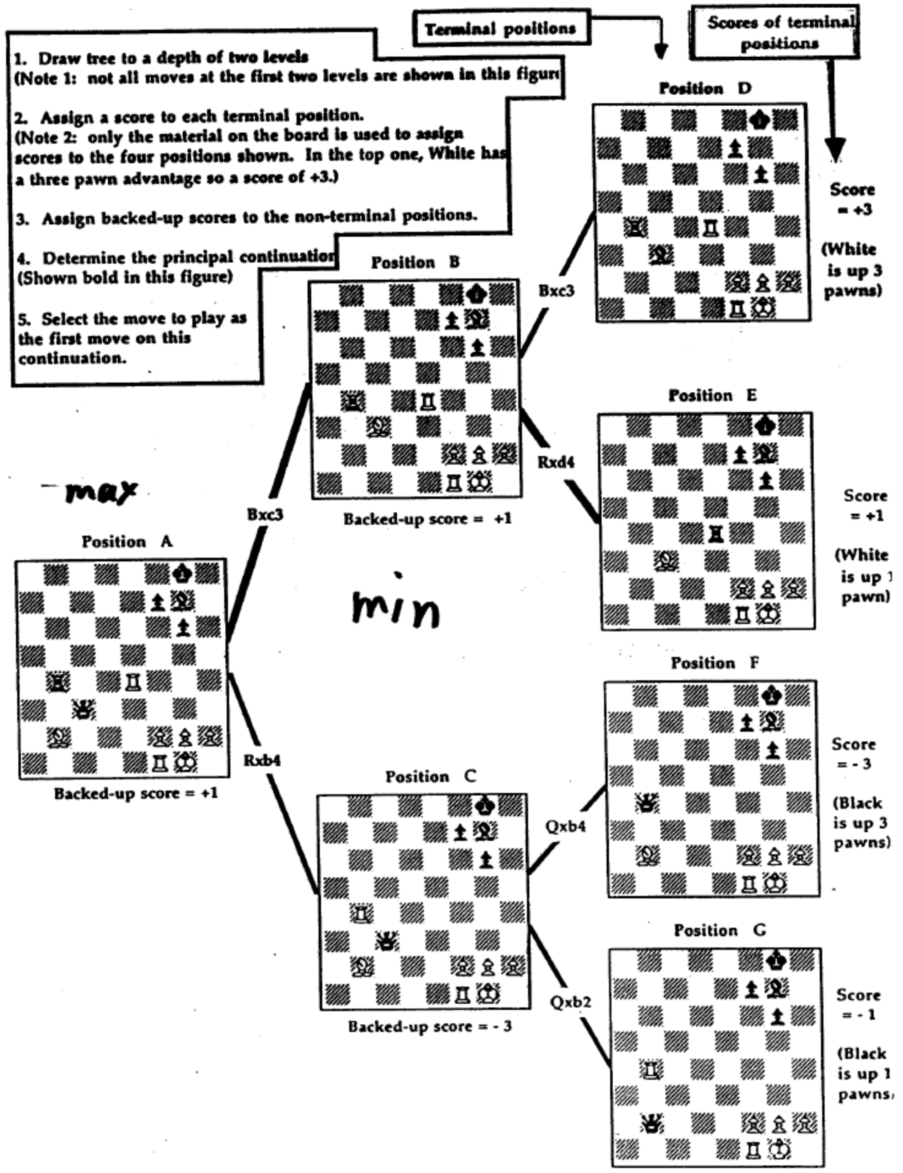
# MONTE CARLO Tree Search



الدكتور مصطفى شبل

# BEFORE DEEP BLUE

- Claude Shannon, Alan Turing: Minimax search with scoring function (1950)
- Only show a few branches here

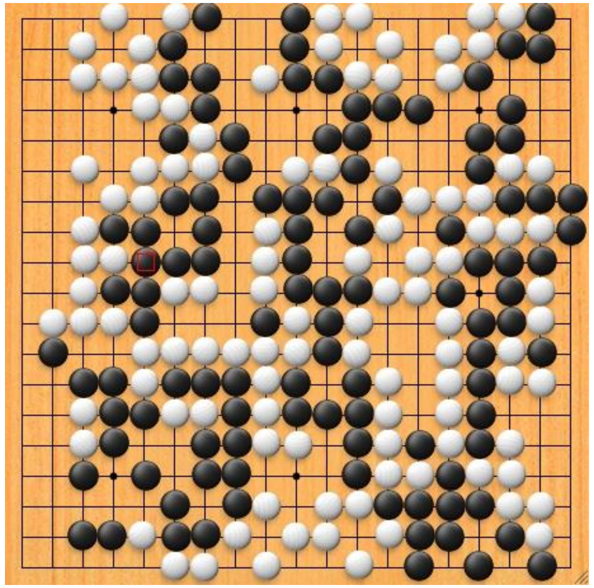


# HOW DEEP BLUE WORKS

- ~200 million moves / second
- 1 sec corresponds to 380 years of human thinking time
- Specialized hardware searches last 5 ply
- Hardware requirement
  - 32-node RS6000 SP multicomputer
  - Each node had
    - 1 IBM Power2 Super Chip (P2SC)
  - 16 chess chips
    - Move generation (often takes 40-50% of time)
    - Evaluation
    - Some endgame heuristics & small endgame databases
  - 32GB opening & endgame database

# Why Alpha-beta not Sufficient

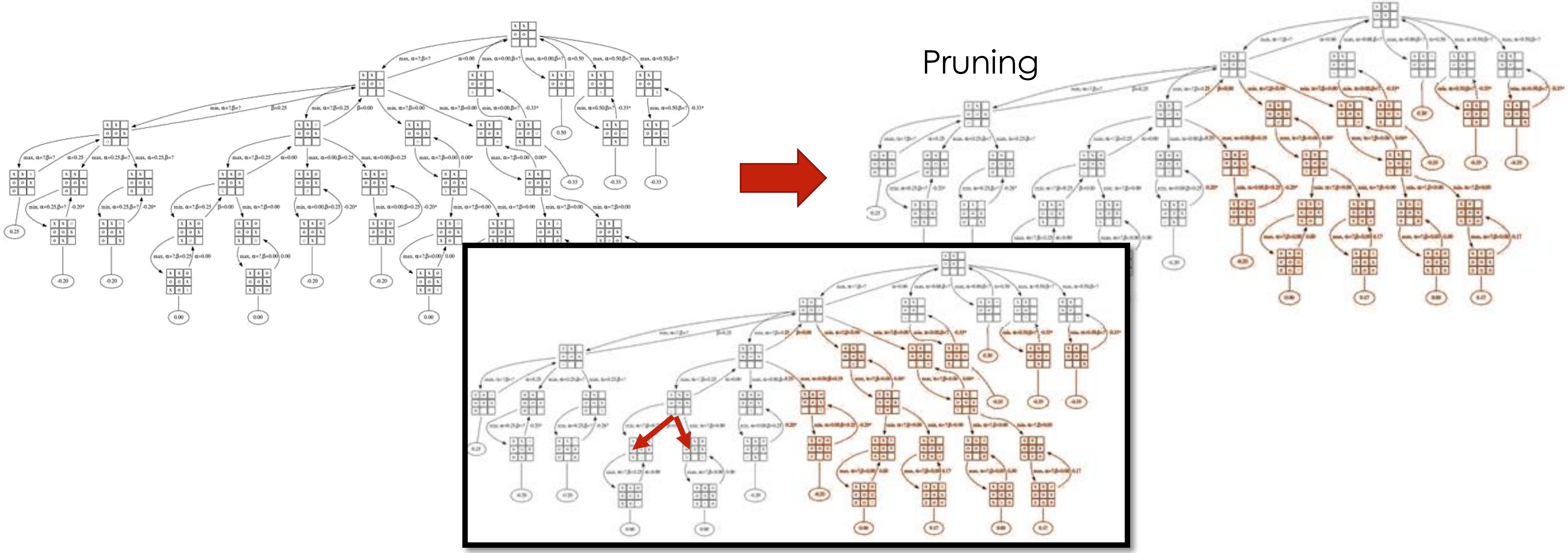
- *1997:* Super human Chess w/ Alpha-Beta + Fast Computer
- *2005*: Computer Go is impossible!



- Branching Factor
  - Chess ≈ 35
  - Go ≈ 250
- Required search depth
  - Chess ≈ 14
  - Go ≈ much larger
- Leaf Evaluation Function
  - Chess – good hand-coded function
  - Go – no good hand-coded function

# INTRODUCTION

- **Monte Carlo methods** are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.

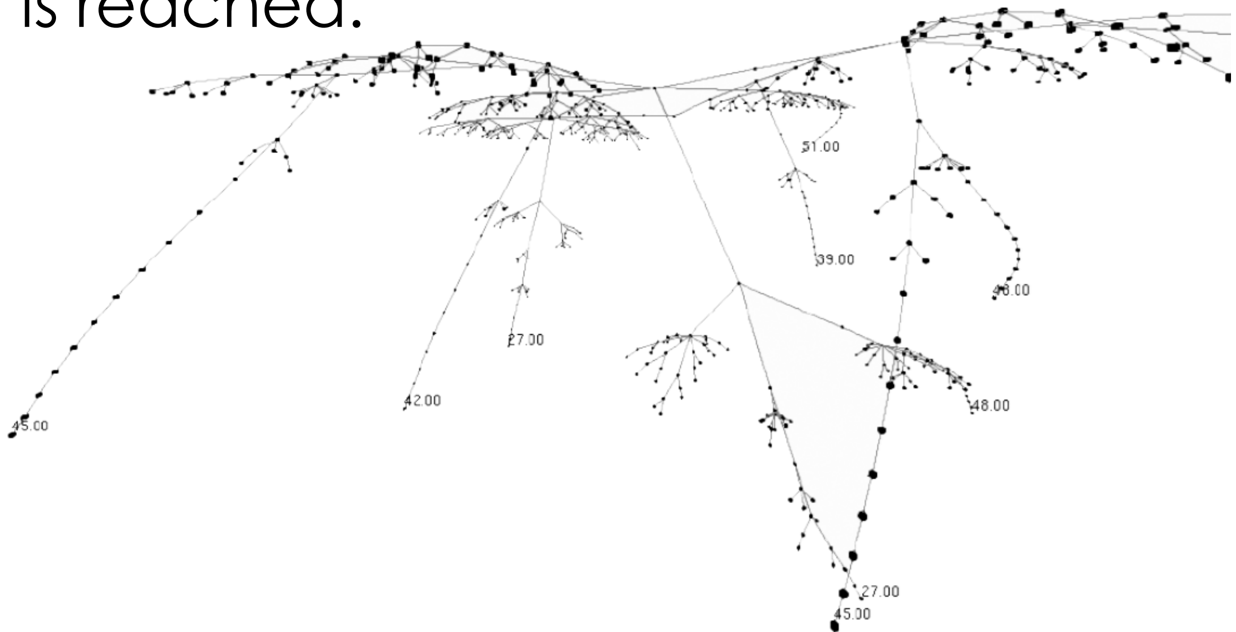


# MONTE CARLO Tree Search (MCTS)

- Builds and searches an asymmetric game tree to make each move
- Phases are: – Tree search:
    - ❑select node to expand using tree policy.

      It balances two factors: exploration a new nodes and exploitation of states that have done well in past
    - ❑Perform random roll-out (simulation )to end of game when true value is known.

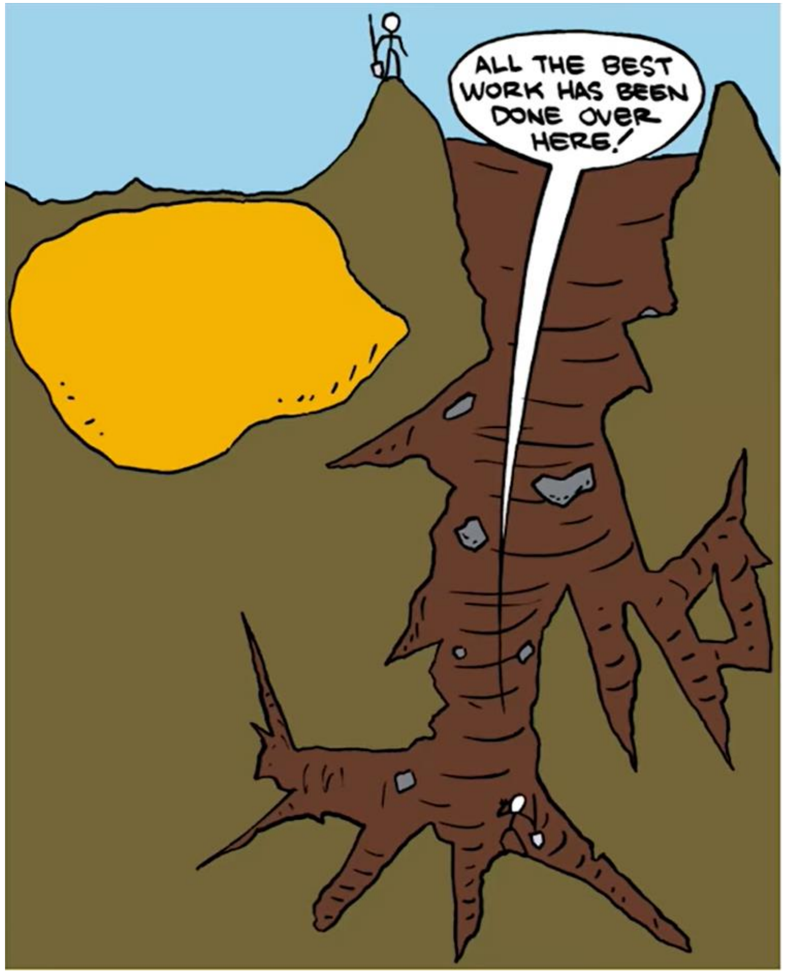
      Repeating until a terminal position is reached.
    - ❑Back the value up the tree.



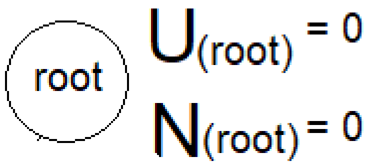
# UPPER CONFIDENCE BOUND FORMULA

$$UCB1(n) = \underbrace{\frac{U(n)}{N(n)}}_{\text{exploitation}} + C \times \underbrace{\sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}}_{\text{exploration}}$$

- U(n) is the total utility of all playouts that went through node n,
- N(n) is the number of playouts through node n,
- PARENT(n) is the parent node of n in the tree.

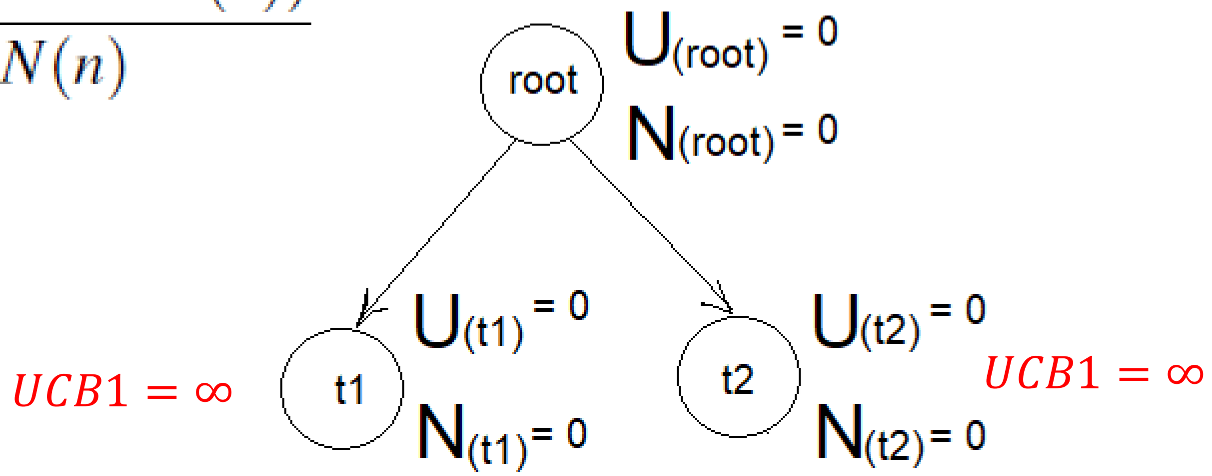


# EXAMPLE



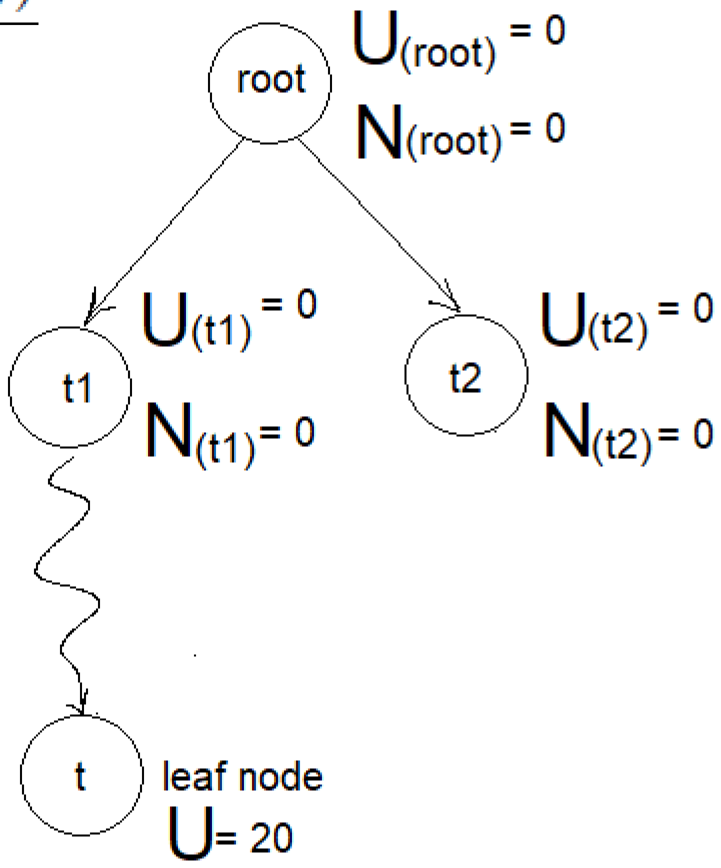
# EXAMPLE

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}$$



# EXAMPLE

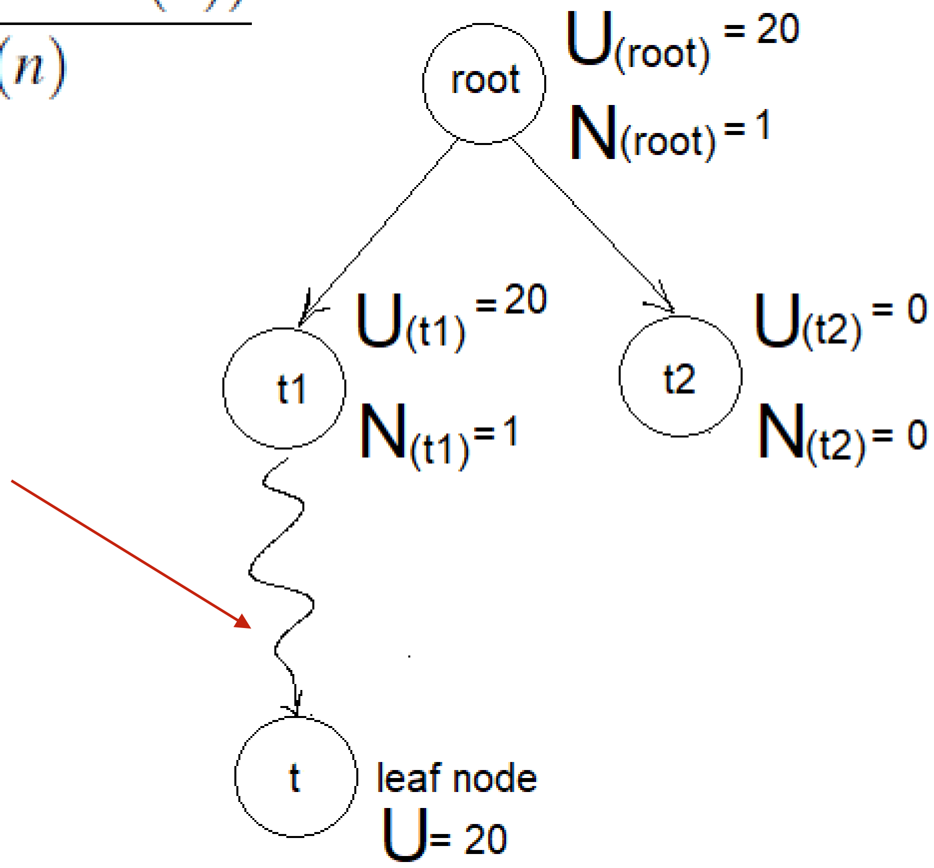
$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}$$



# EXAMPLE

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$

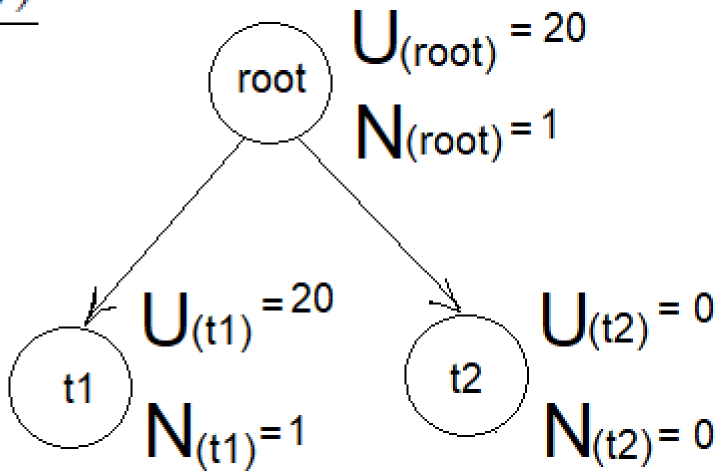
Not recorded



# EXAMPLE

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$

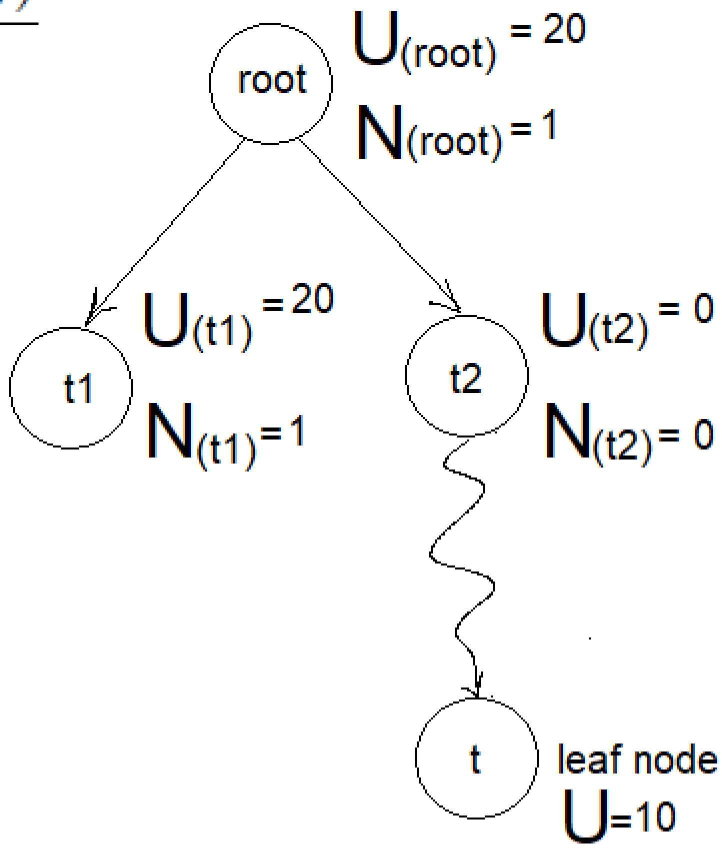
$$UCB1 = \frac{20}{1} + 2 \times \sqrt{\frac{\log(1)}{1}} = 20$$




$$UCB1 = \infty$$

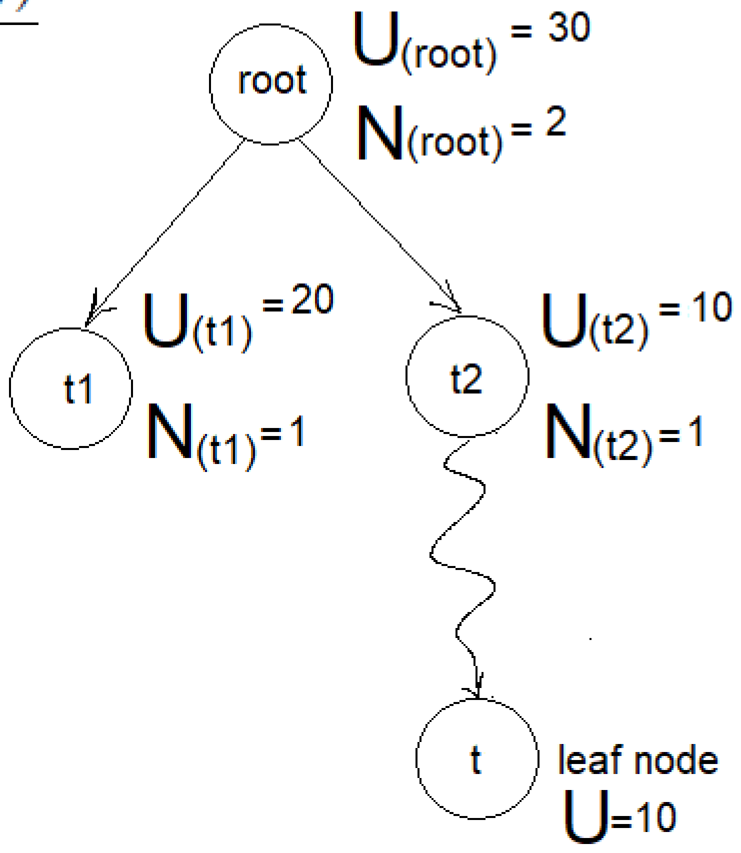
# EXAMPLE

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}$$



# EXAMPLE

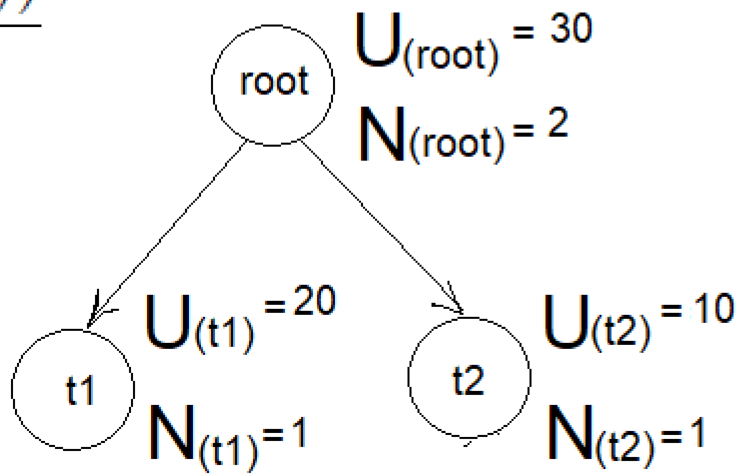
$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}$$



# EXAMPLE

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$

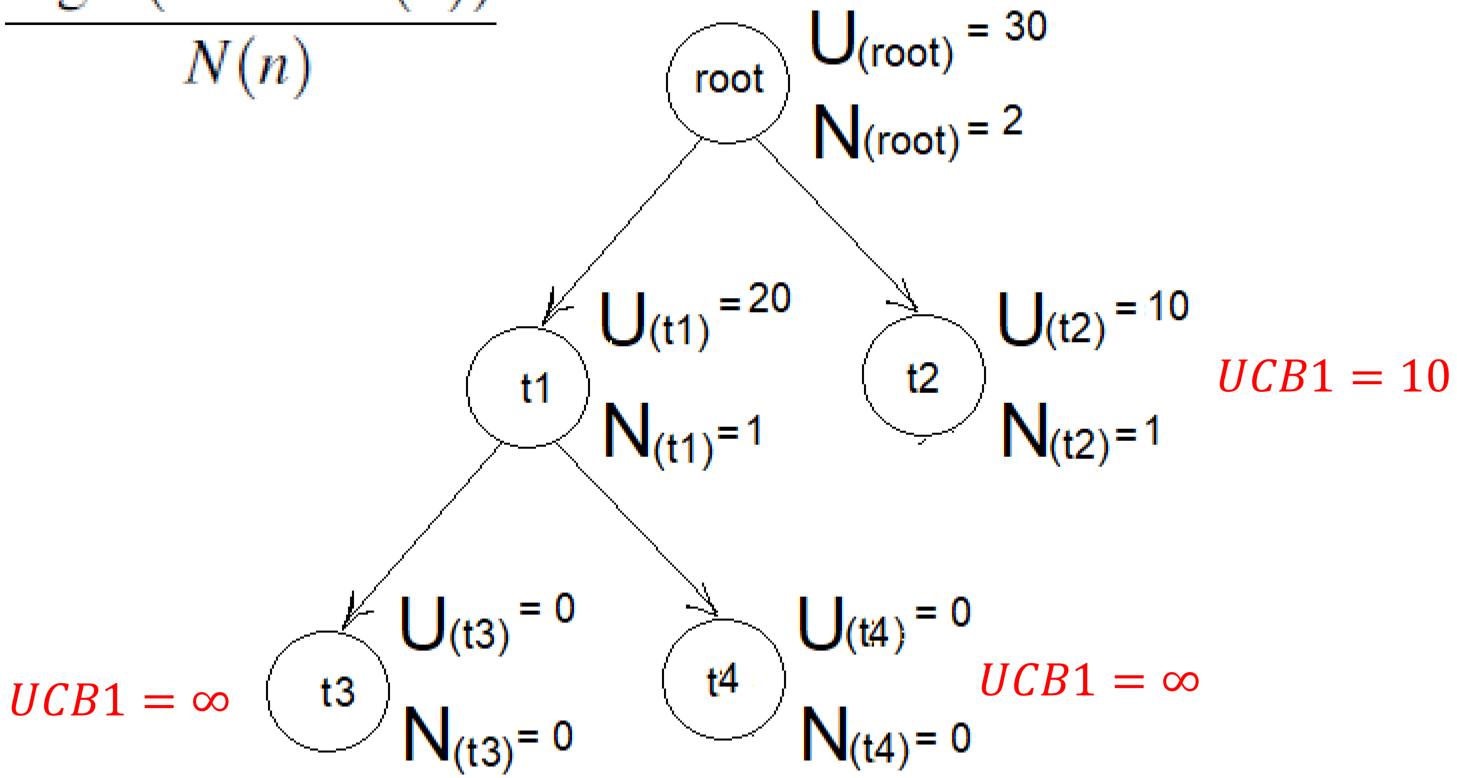
$$UCB1 = \frac{20}{1} + 2 \times \sqrt{\frac{\log(2)}{1}} = 20$$



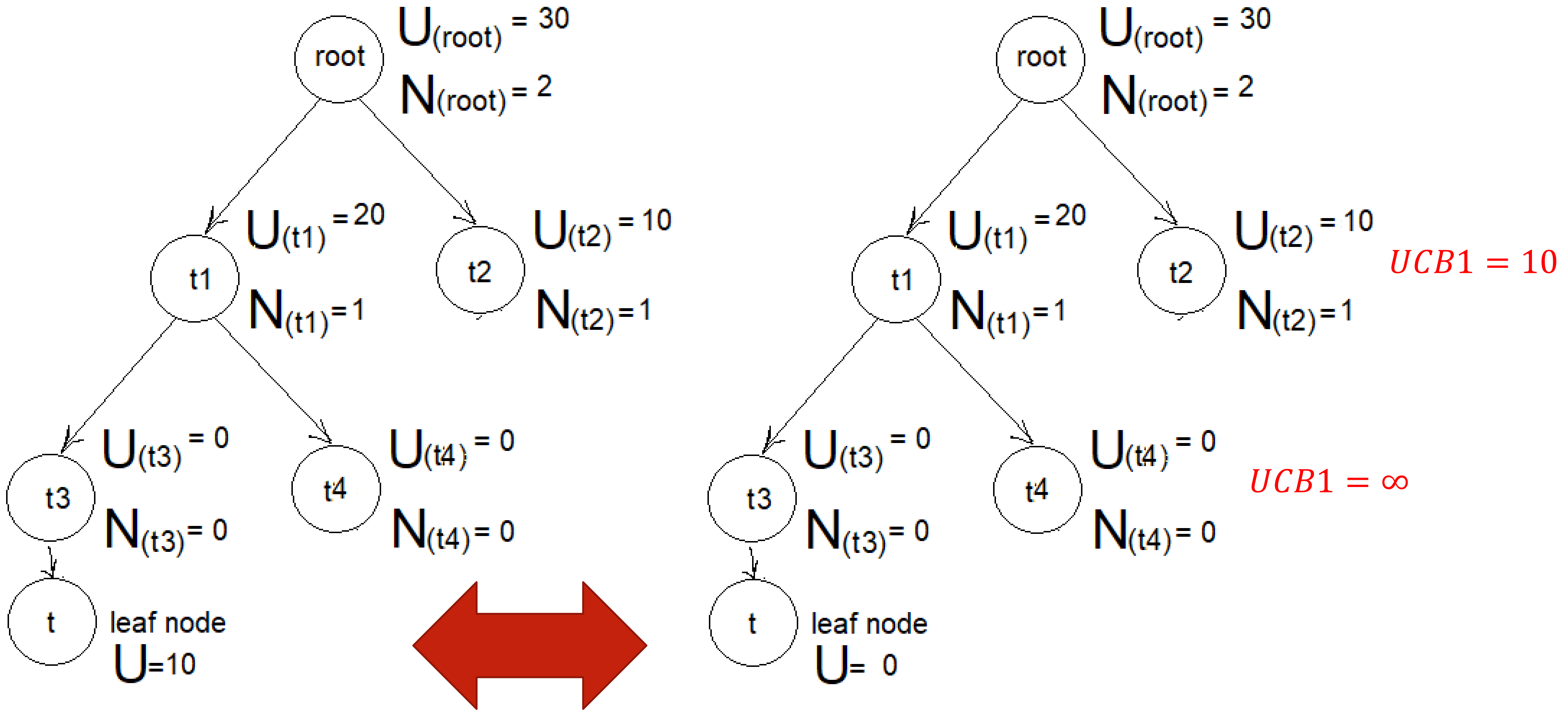

$$UCB1 = \frac{10}{1} + 2 \times \sqrt{\frac{\log(2)}{1}} = 10$$

# EXAMPLE

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}$$

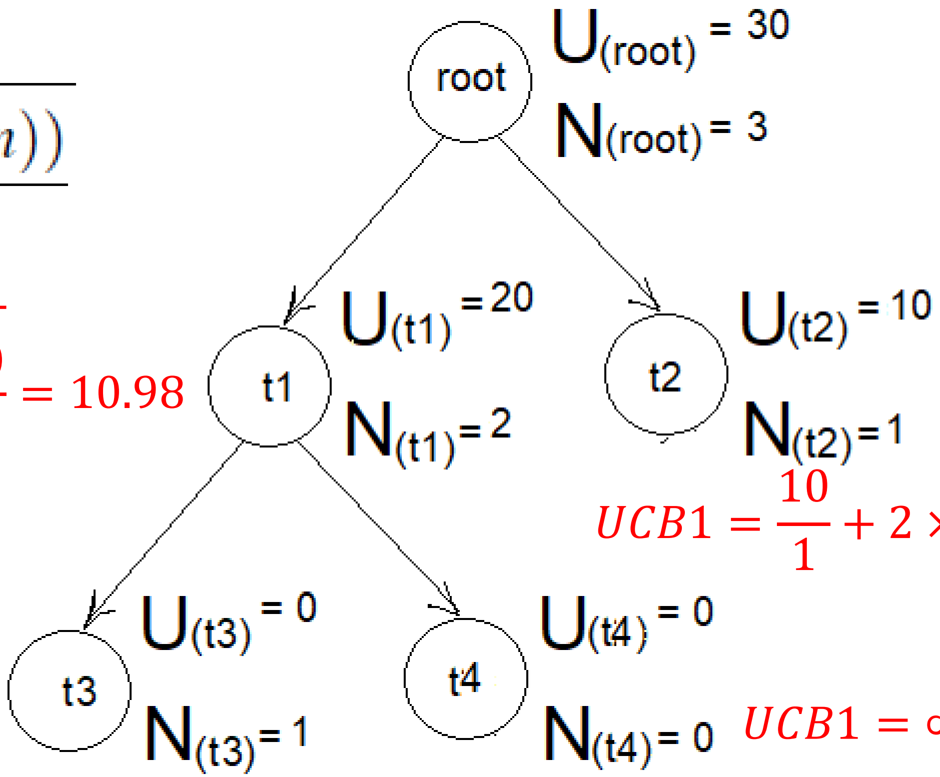


# EXAMPLE


root
U(root) = 30
N(root) = 2
t1
U(t1) = 20
N(t1) = 1
t2
U(t2) = 10
N(t2) = 1
t3
U(t3) = 0
N(t3) = 0
t4
U(t4) = 0
N(t4) = 0
t
leaf node
U=10
root
U(root) = 30
N(root) = 2
t1
U(t1) = 20
N(t1) = 1
t2
U(t2) = 10
N(t2) = 1
UCB1 = 10
t3
U(t3) = 0
N(t3) = 0
t4
U(t4) = 0
N(t4) = 0
UCB1 = ∞
t
leaf node
U= 0

# EXAMPLE

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$




$$UCB1 = \frac{20}{2} + 2 \times \sqrt{\frac{\log(3)}{2}} = 10.98$$

$$UCB1 = \frac{10}{1} + 2 \times \sqrt{\frac{\log(3)}{1}} = 11.38$$

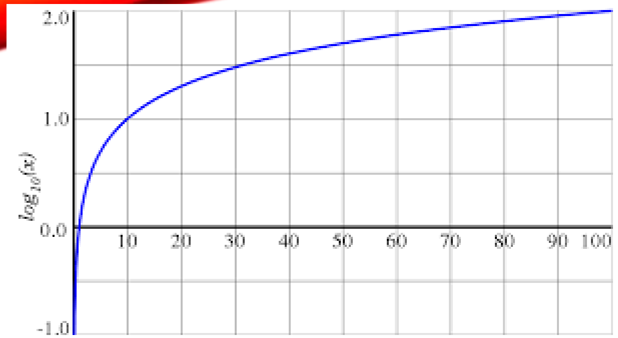
$$UCB1 = \frac{0}{1} + 2 \times \sqrt{\frac{\log(2)}{1}} = 0$$

$$UCB1 = \infty$$

exploration

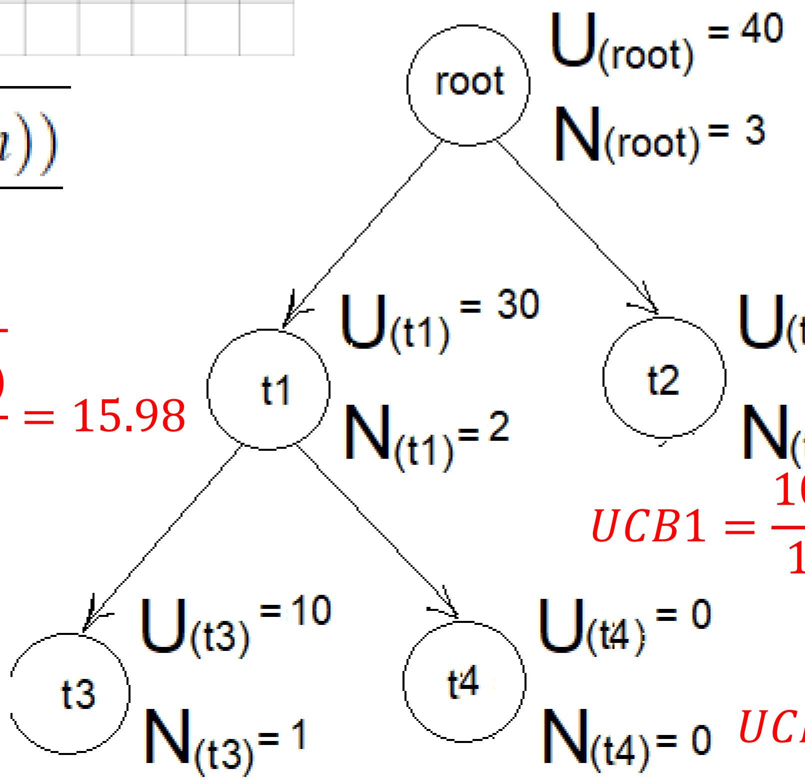
Recalculate if U(t3)=10

# EXAMPLE




$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}$$

exploitation




$$UCB1 = \frac{30}{2} + 2 \times \sqrt{\frac{\log(3)}{2}} = 15.98$$

$$UCB1 = \frac{10}{1} + 2 \times \sqrt{\frac{\log(3)}{1}} = 11.38$$

$$UCB1 = \frac{10}{1} + 2 \times \sqrt{\frac{\log(2)}{1}} = 10$$

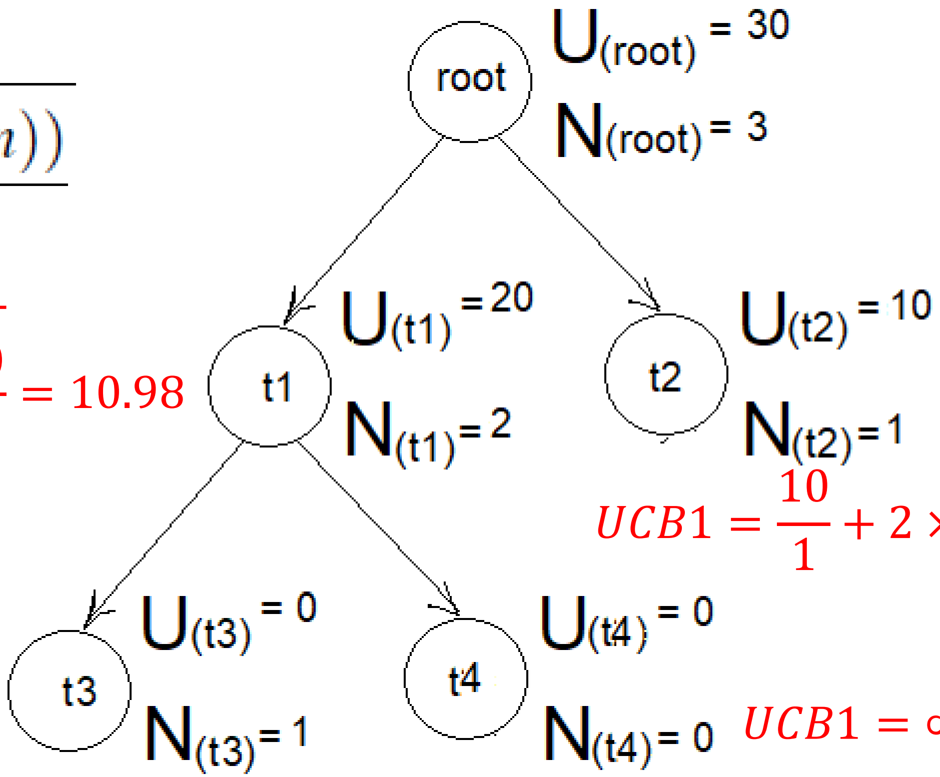
$$UCB1 = \infty$$

Recalculate if U(t3)=10

exploitation

# Constant Selection

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$




$$UCB1 = \frac{20}{2} + 2 \times \sqrt{\frac{\log(3)}{2}} = 10.98$$

$$UCB1 = \frac{10}{1} + 2 \times \sqrt{\frac{\log(3)}{1}} = 11.38$$

$$UCB1 = \frac{0}{1} + 2 \times \sqrt{\frac{\log(2)}{1}} = 0$$

$$UCB1 = \infty$$

exploration

Recalculate if U(t3)=10

# The MONTE CARLO Tree Search Algorithm

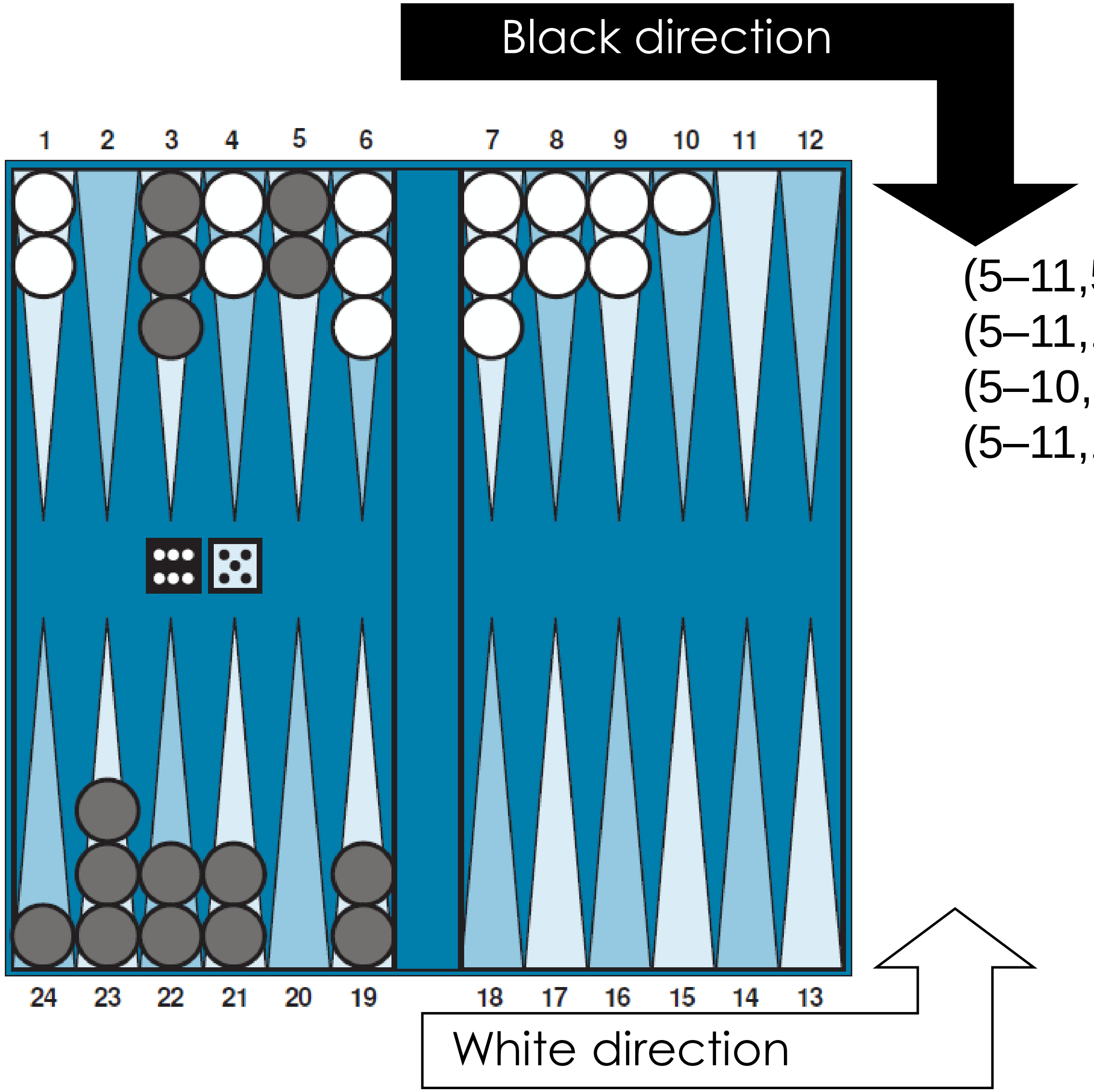
```
function MONTE-CARLO-TREE-SEARCH(state) returns an action
  tree ← NODE(state)
  while IS-TIME-REMAINING() do
    leaf ← SELECT(tree)
    child ← EXPAND(leaf)
    result ← SIMULATE(child)
    BACK-PROPAGATE(result, child)
  return the move in ACTIONS(state) whose node has highest number of playouts
```

# Early Playout Termination

- It is also possible to combine aspects of alpha–beta and Monte Carlo search.

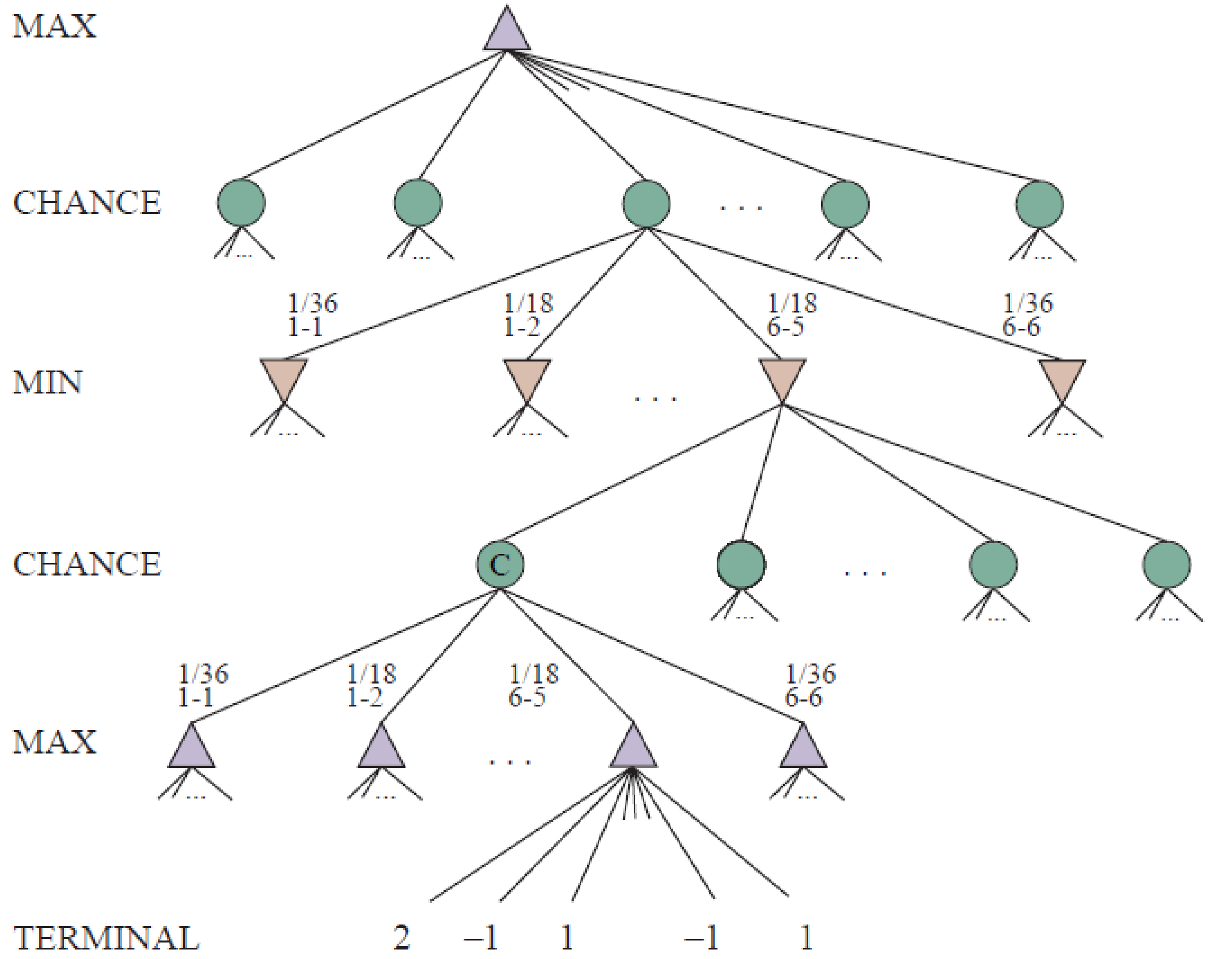
# STOCHASTIC GAMES

# BACKGAMMON




(5–11,5–10),
(5–11,19–24),
(5–10,10–16),
(5–11,11–16)

# Chance Node

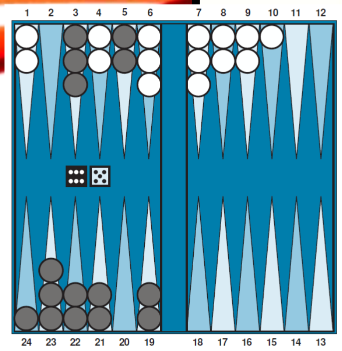


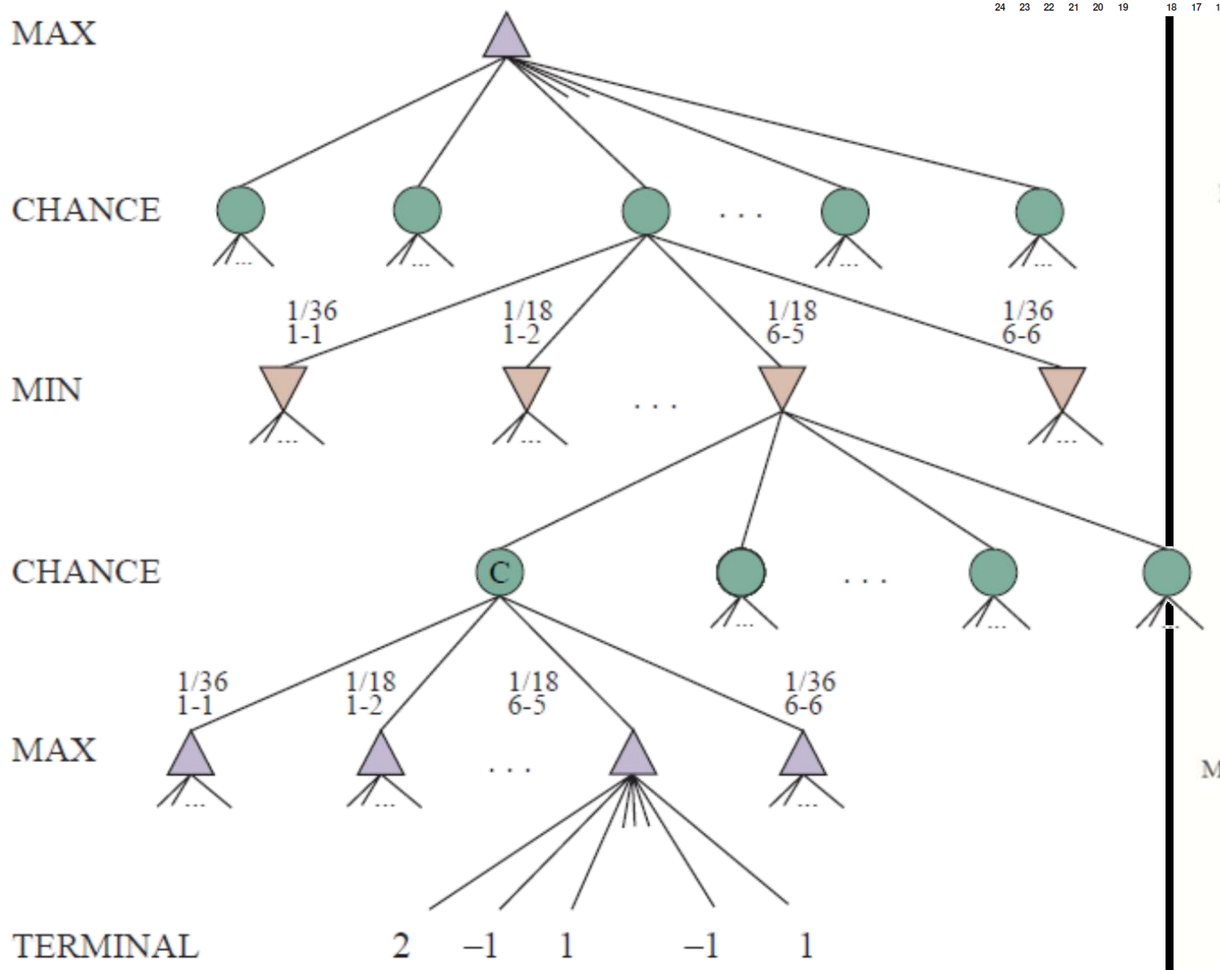


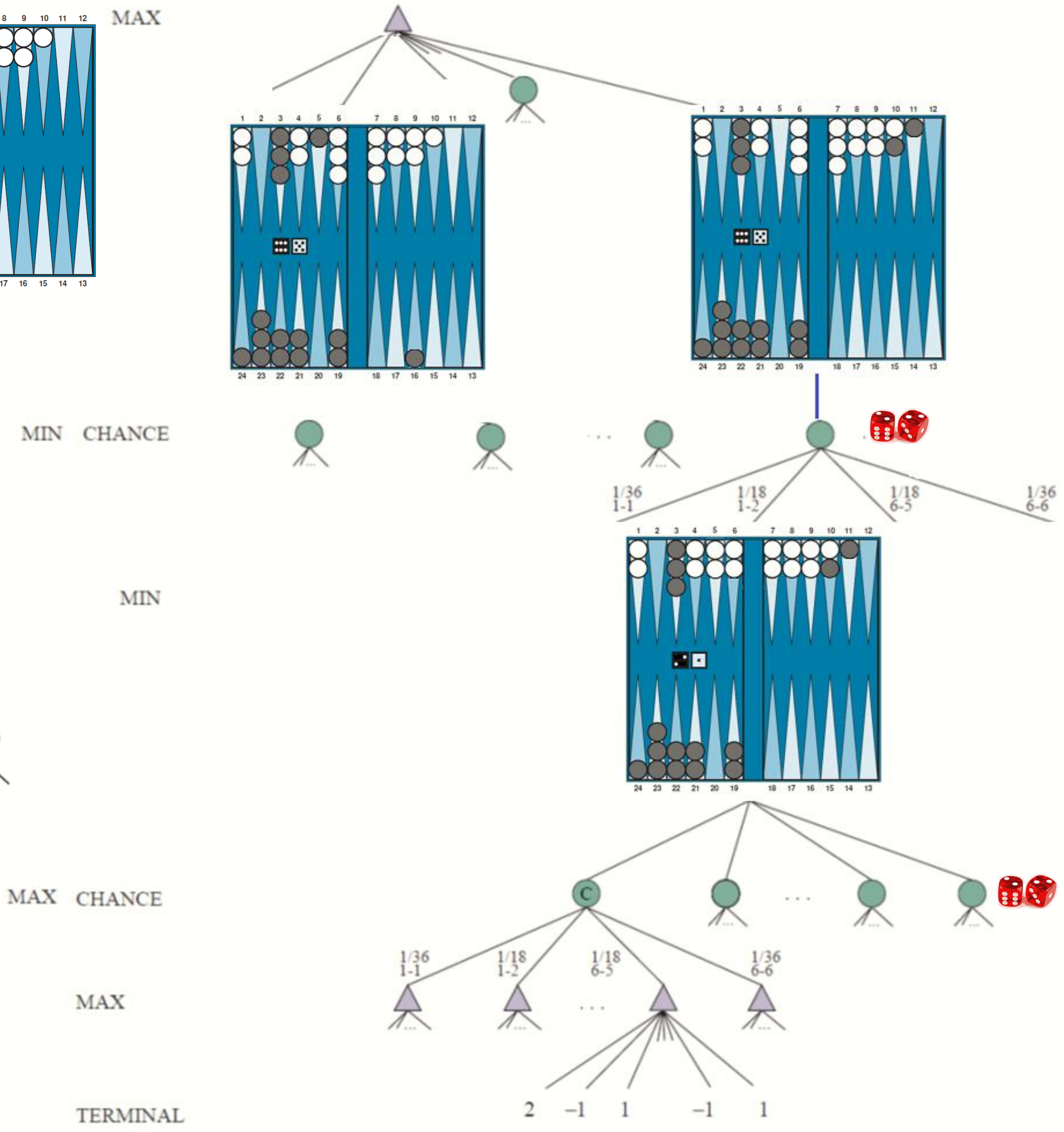
$= \frac{1}{18}$

$= \frac{1}{18}$

$= \frac{1}{36}$


1 2 3 4 5 6 7 8 9 10 11 12
24 23 22 21 20 19 18 17 16 15 14 13


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

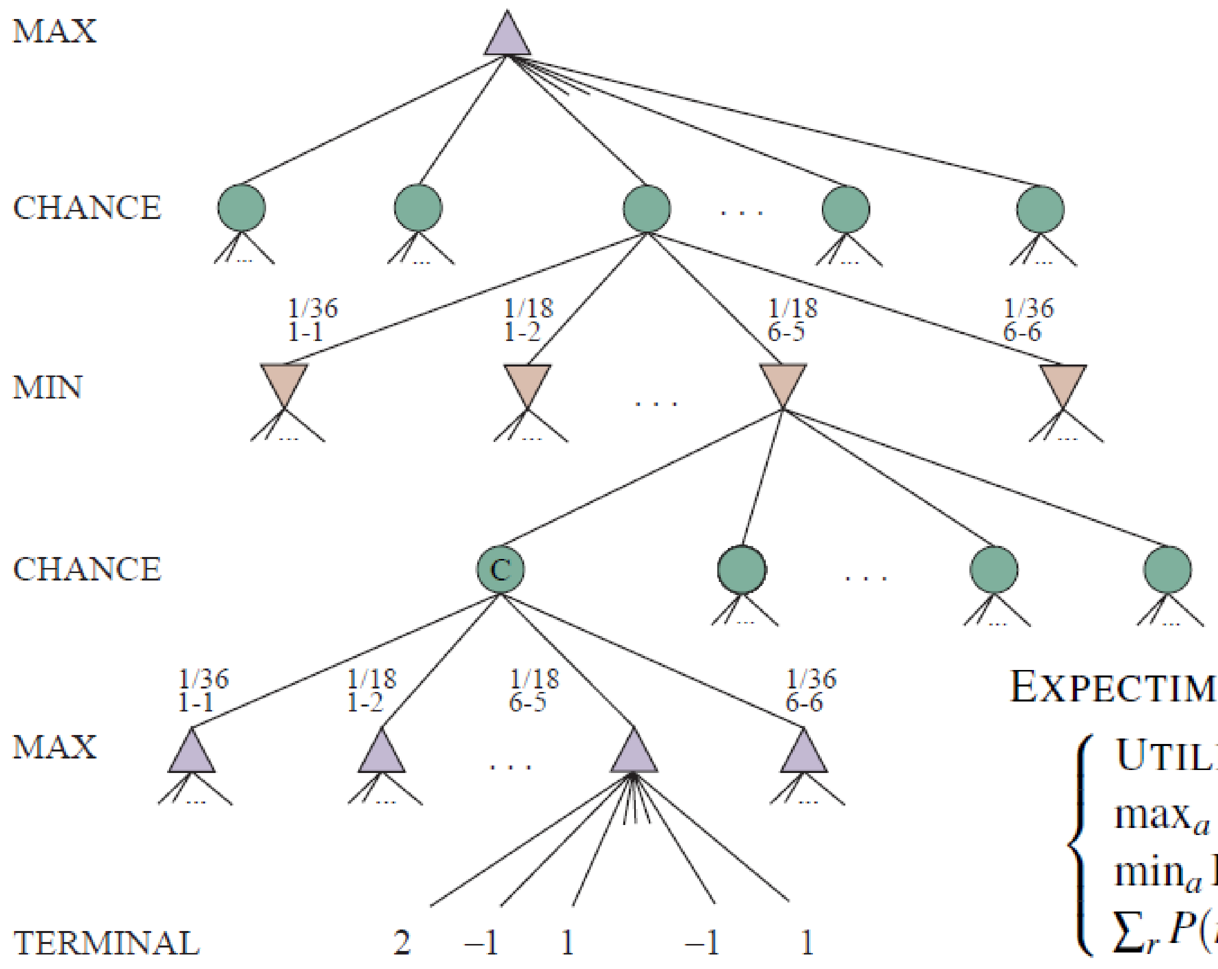

MAX
MIN CHANCE
1/36 1-1
1/18 1-2
1/18 6-5
1/36 6-6
MIN
MAX CHANCE
C
1/36 1-1
1/18 1-2
1/18 6-5
1/36 6-6
MAX
TERMINAL
2 −1 1 −1 1

# Expected Value (Chance Node)

- The sum of the probability of each possible outcome multiplied by its value:

$$E(X) = \sum_i p_i x_i$$

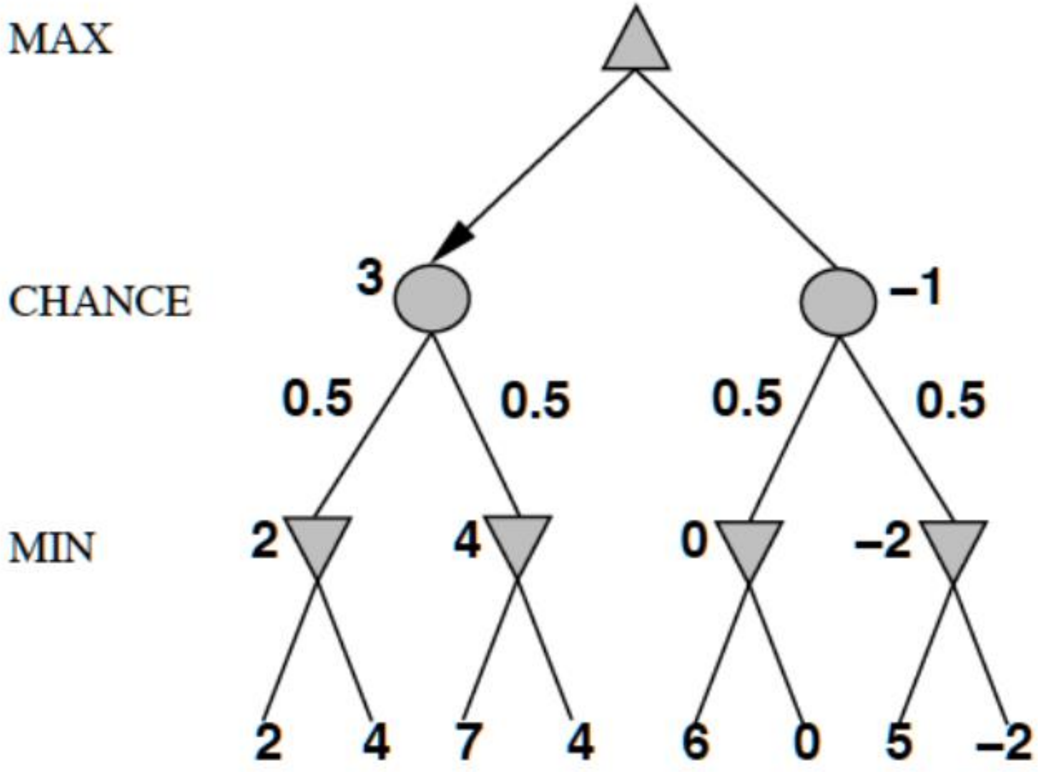
# Expectiminimax Value



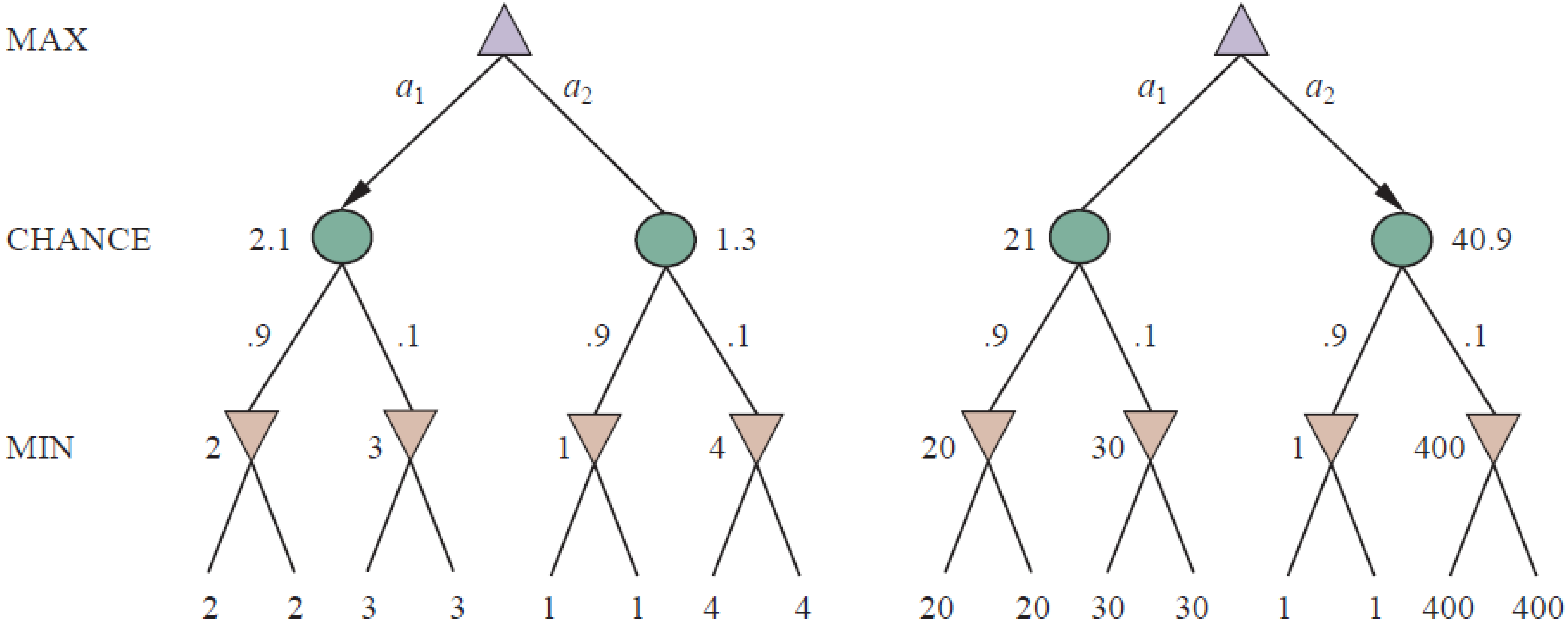

$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if TO-MOVE}(s) = \text{MIN} \\ \sum_r P(r)\,\text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & \text{if TO-MOVE}(s) = \text{CHANCE} \end{cases}$$

# Expectiminimax Value

In nondeterministic games, chance introduced by dice, card shuffling, coin-flipping ..etc



# Order-preserving Transformation



# THANKS