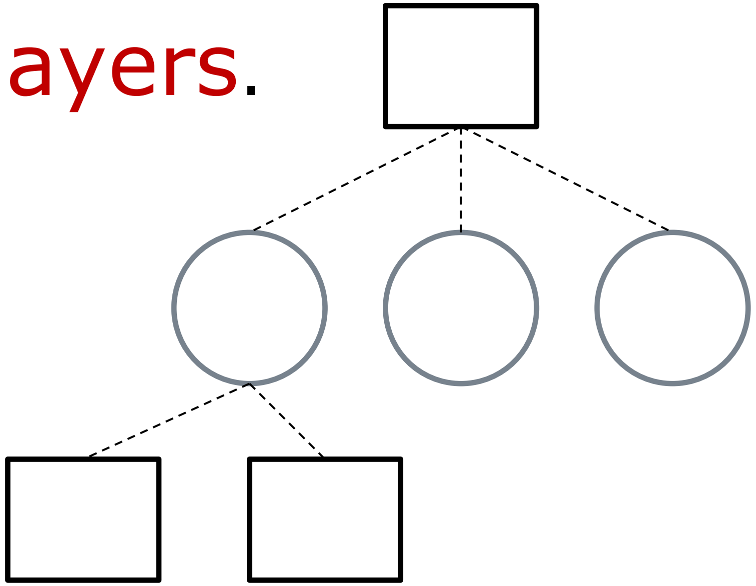
ME 620: Fundamentals of Artificial Intelligence

Lecture 12: Minimax and Alpha-Beta Pruning

Shyamanta M Hazarika
Biomimetic Robotics and Artificial Intelligence Lab
Mechanical Engineering and M F School of Data Sc. & AI
IIT Guwahati

Game Trees

- A game is represented by a game tree.
  - Game tree is a layered tree in which at each alternating level, one or the other player makes the choice.
  - Layers - MAX layers and the MIN layers.



A game starts at the root with MAX playing first and ends at the leaf node.

Leaves of a game tree are labelled with outcome of the game and the game ends there.

Minimax Procedure

- For complex games such as chess or checkers, search to termination is out of question.

  Complete game tree for chess has approximately 10^{40} nodes.

  Even for a game as simple as Tic-tac-toe there are over 3,50,000 nodes in the complete game tree.

- Good First Move?
  - A good first move can be extracted by a procedure called the Minimax.
  - This estimate can be made by applying a static evaluation function to the leaf node.
  - Back-up values level by level.
    - MAX parent of MIN nodes is assigned backed-up value equal to maximum of the evaluations of the nodes.
    - MIN parent of MAX nodes is assigned the minimum.

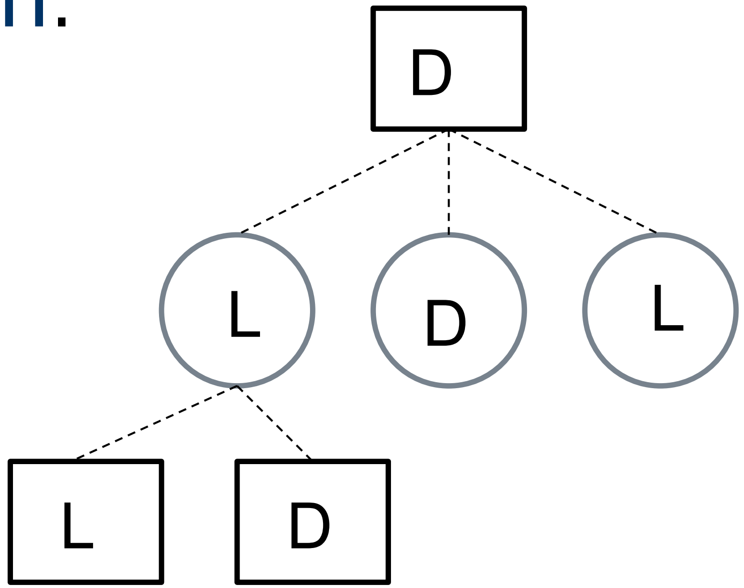
Minimax Rule

- The minimax rule backs up values from the children of a node.
  - For a MAX node, it backs up the maximum of the values of the children.
  - For a MIN node, the minimum.

L – Loss; D – Draw and W- Win

Is from the perspective of MAX; The leaves can be labelled equivalently with numbers

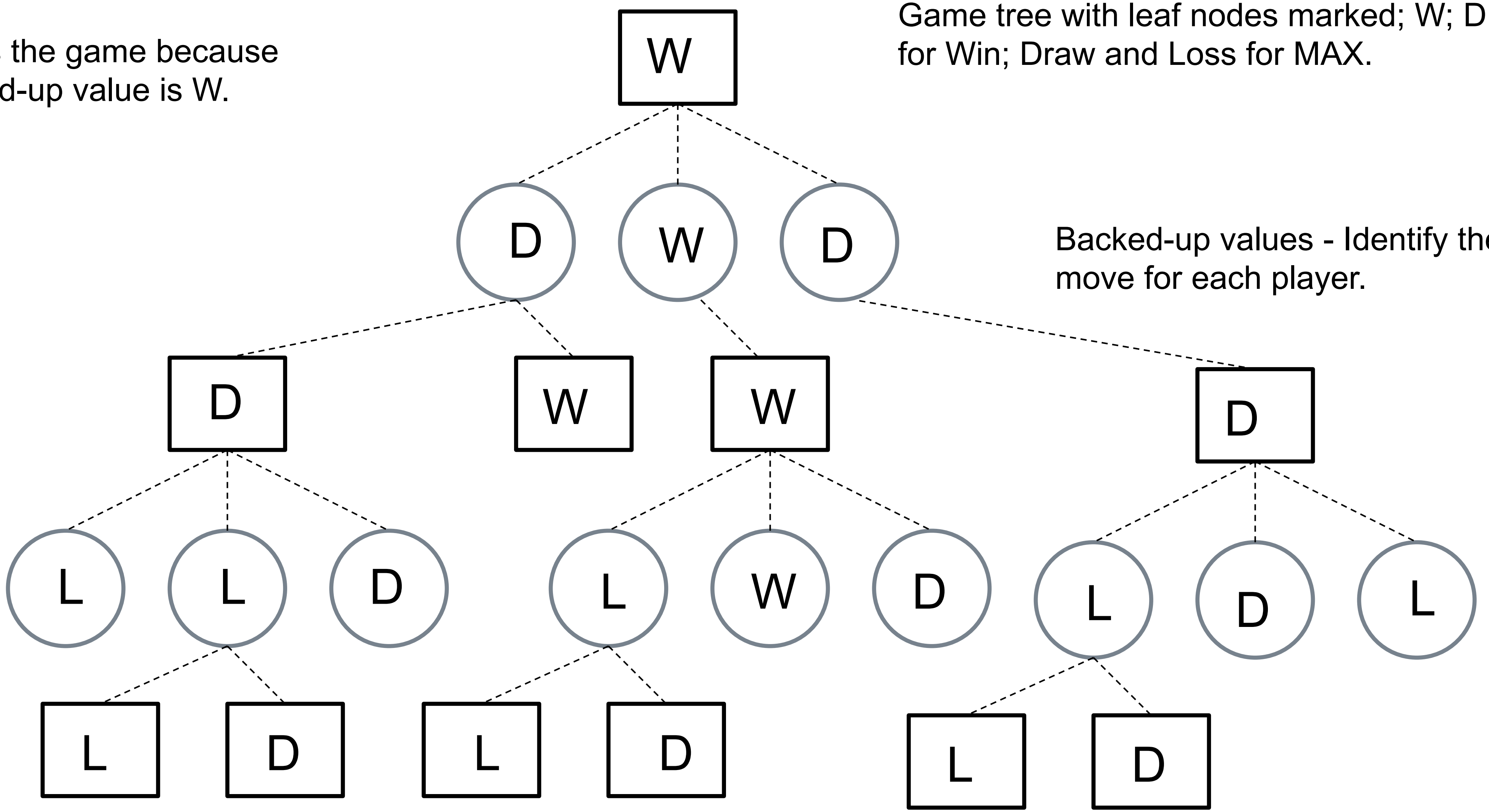
-1 – Loss; 0 – Draw and 1 – Win



Game Trees

MAX wins the game because the backed-up value is W.

Game tree with leaf nodes marked; W; D; L for Win; Draw and Loss for MAX.



Backed-up values - Identify the best move for each player.

Use of an Evaluation Function

Cannot inspect the complete game tree and compute the minimax value!

- Resort to other means to select the BEST move to make.
- Instead of BEST; select moves that appears to be BEST.

Tic-Tac-Toe

Evaluation Function

e(p) = number of directions open for Max –
number of directions open for Min

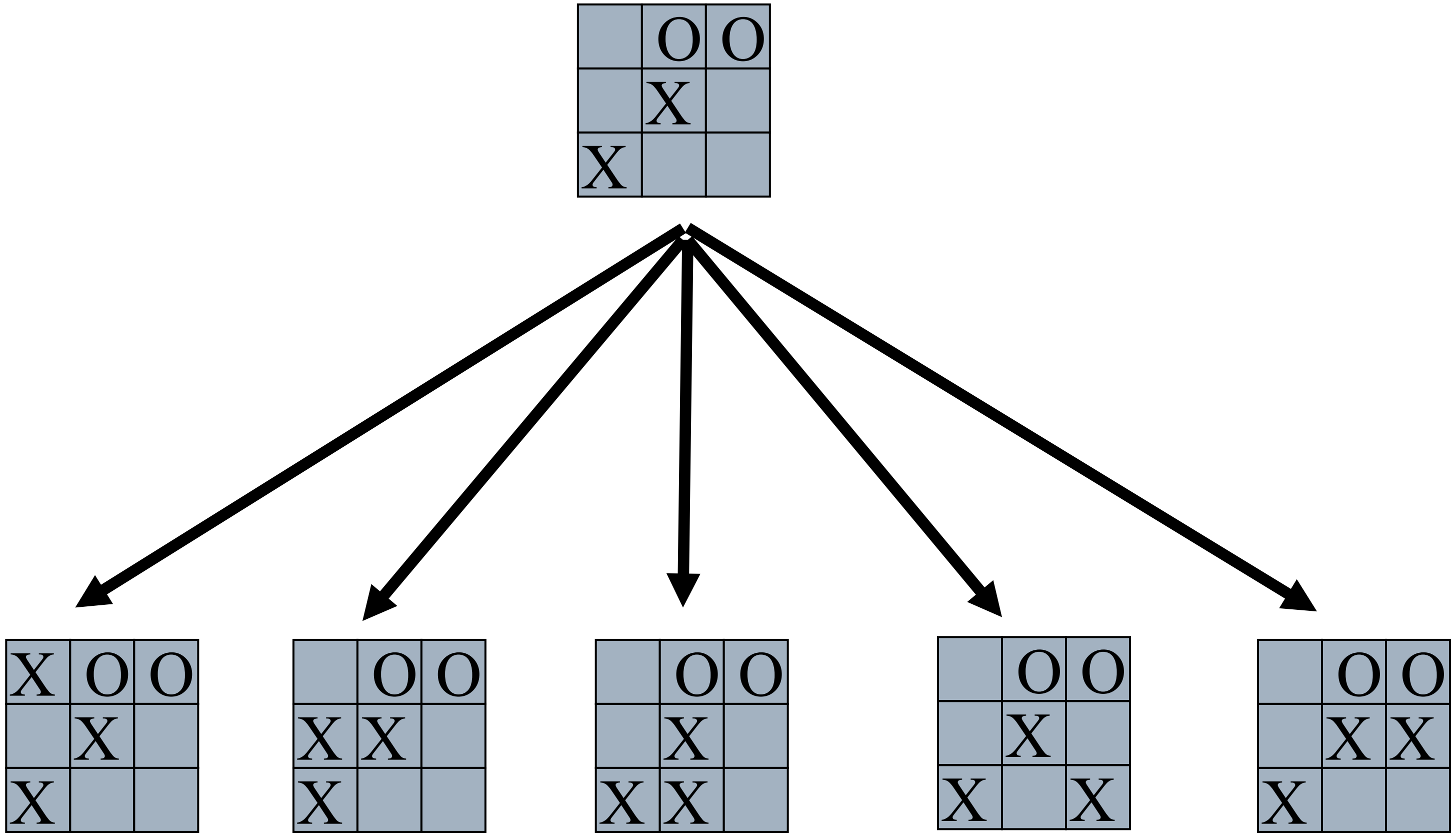
e(p) = + inf if win for Max

e(p) = - inf if win for Min

e(p) = 6 – 4 = 2

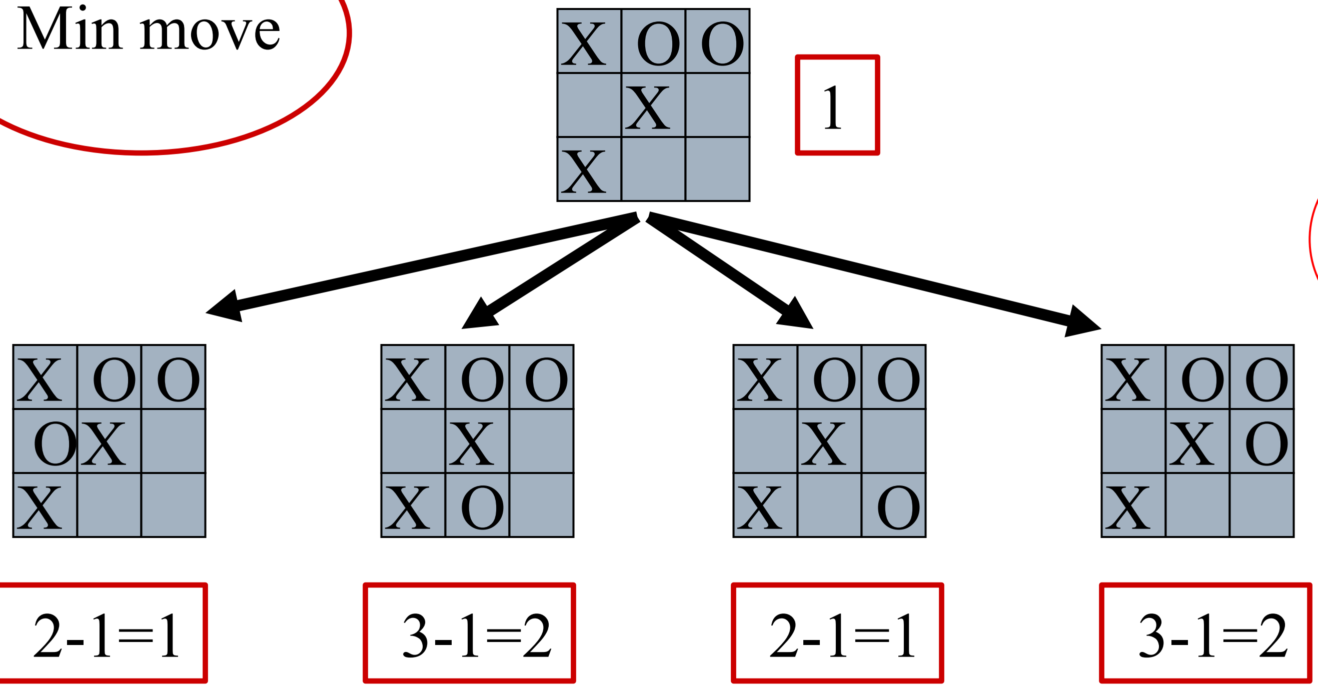
	O	
	X	

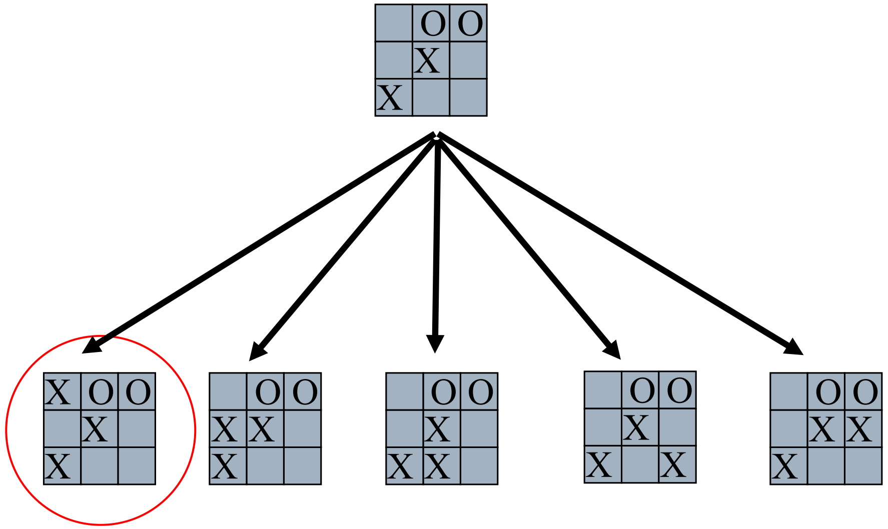
Tic-Tac-Toe



Tic-Tac-Toe

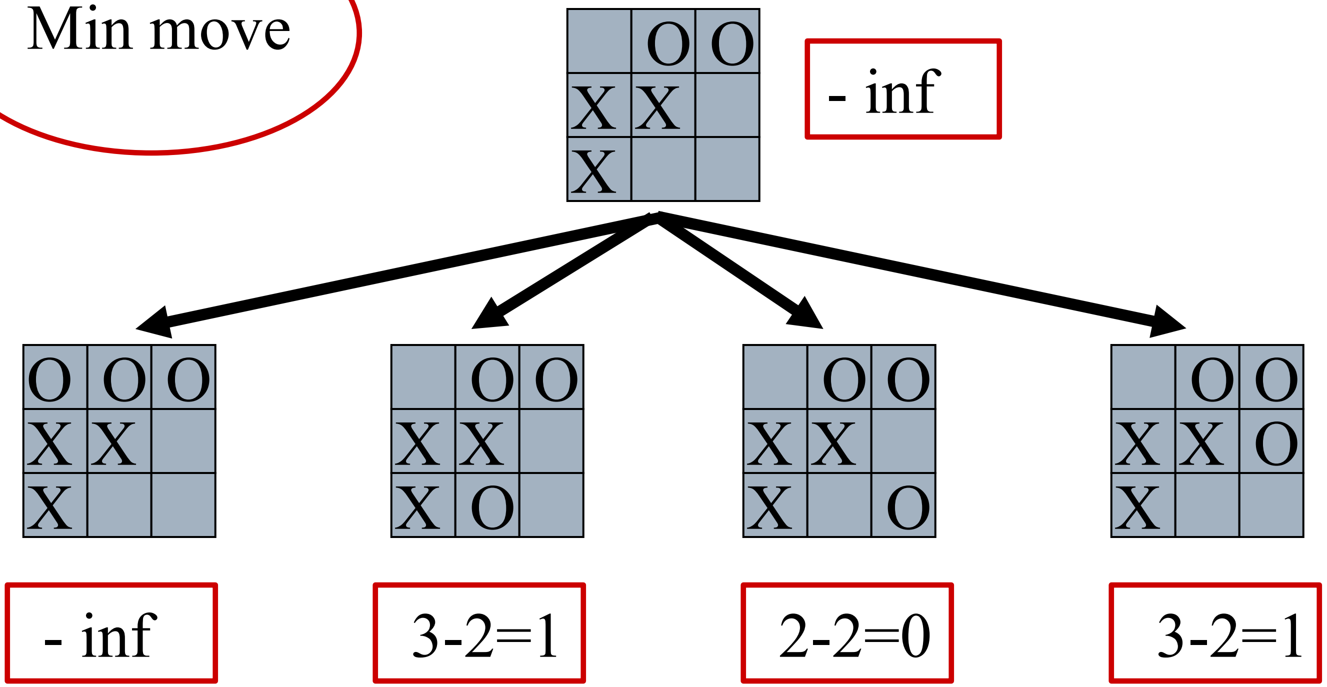
Min move

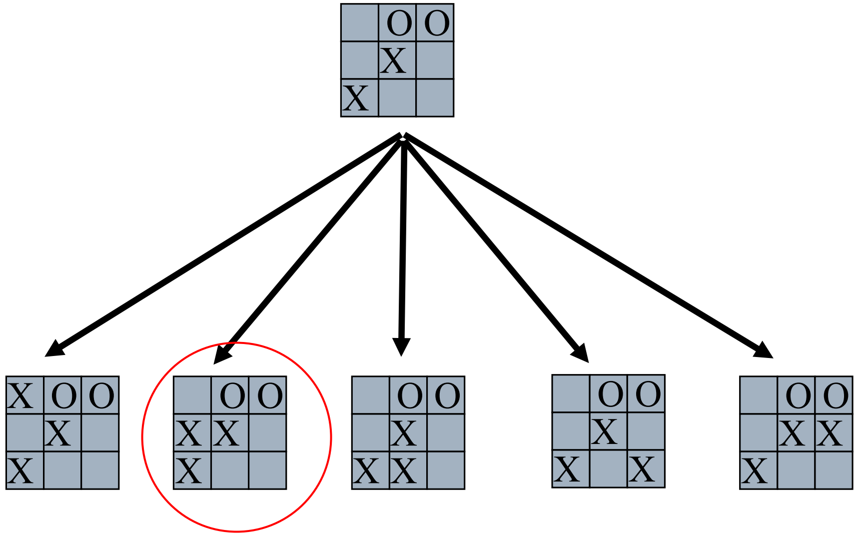




Tic-Tac-Toe

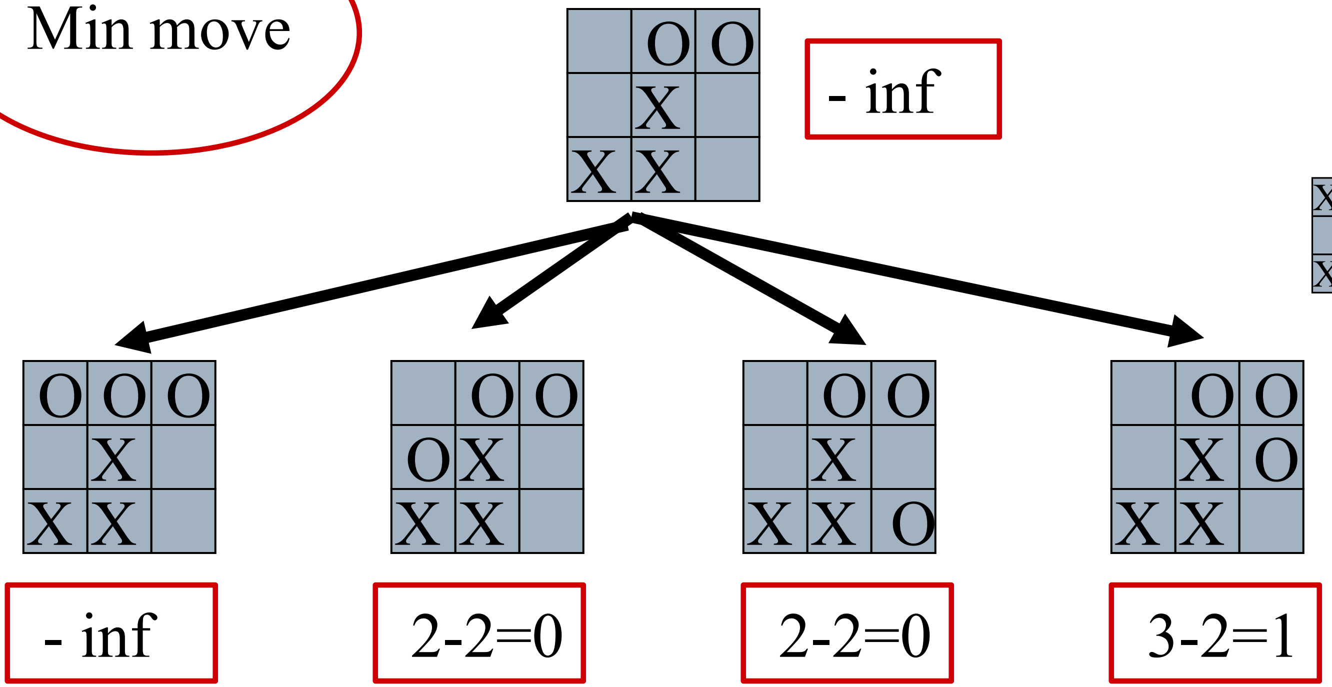
Min move

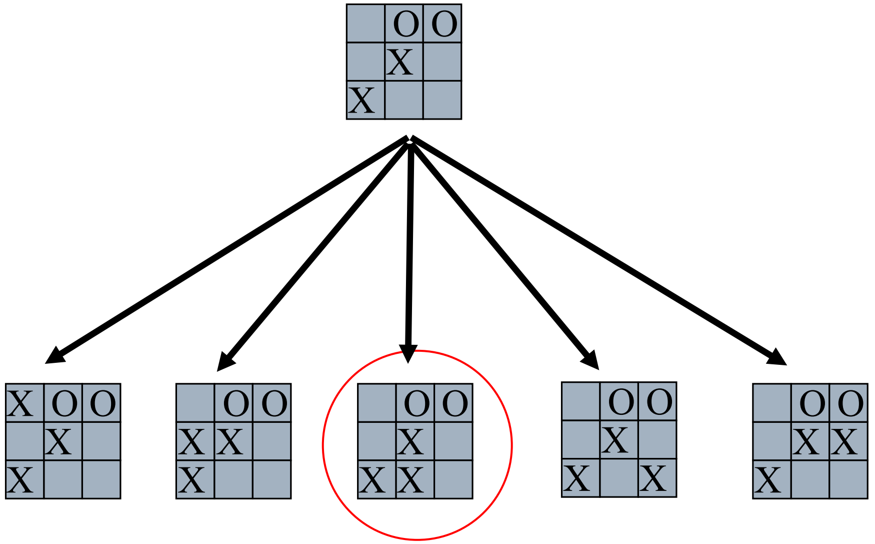




Tic-Tac-Toe

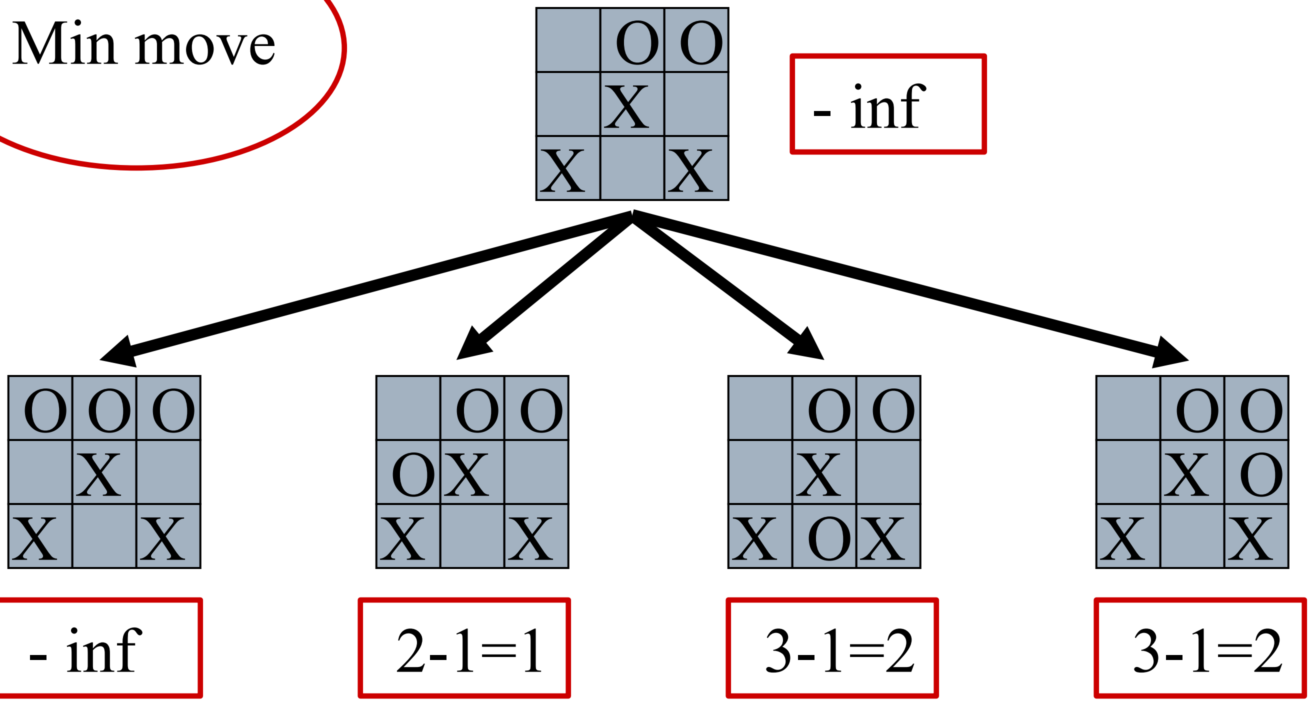
Min move

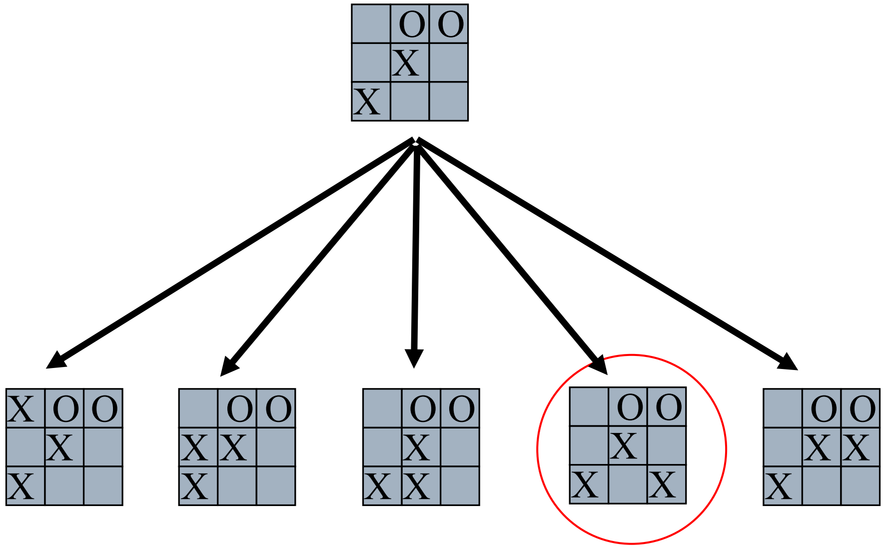




Tic-Tac-Toe

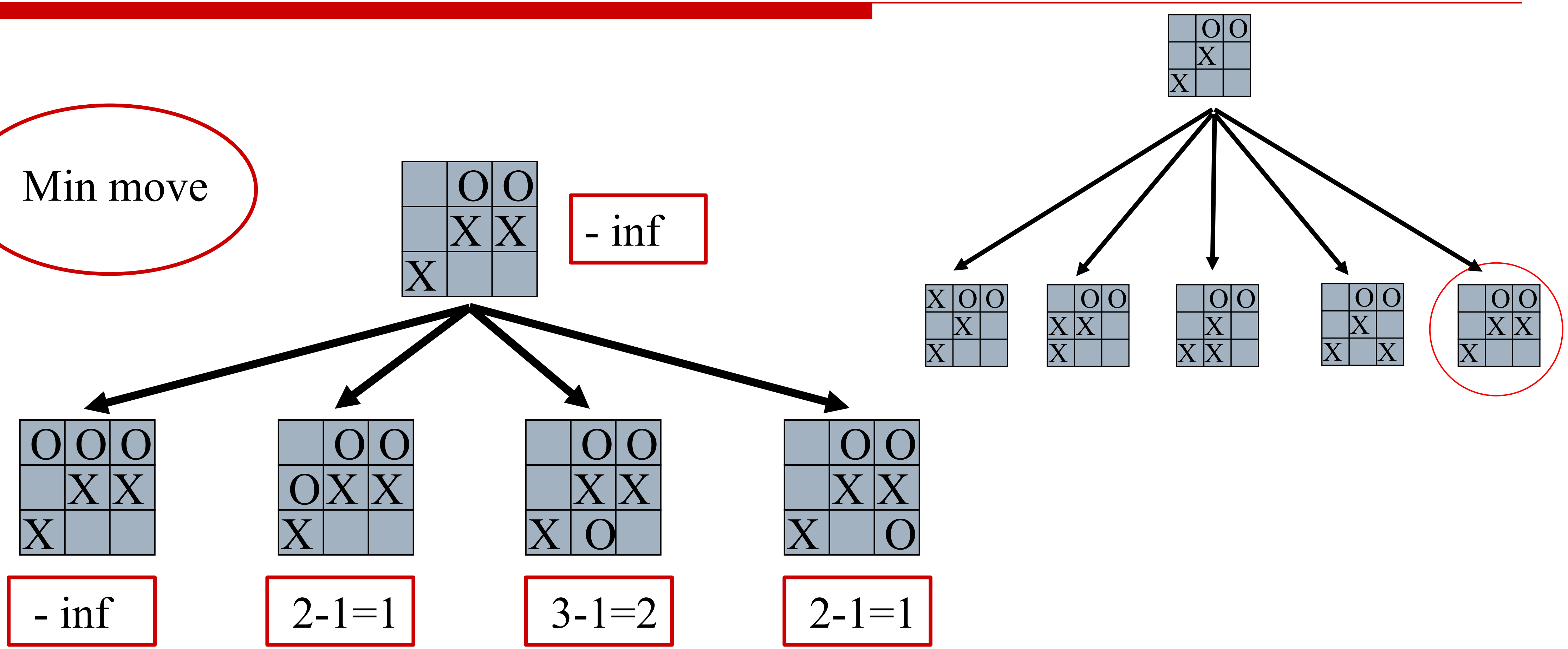
Min move



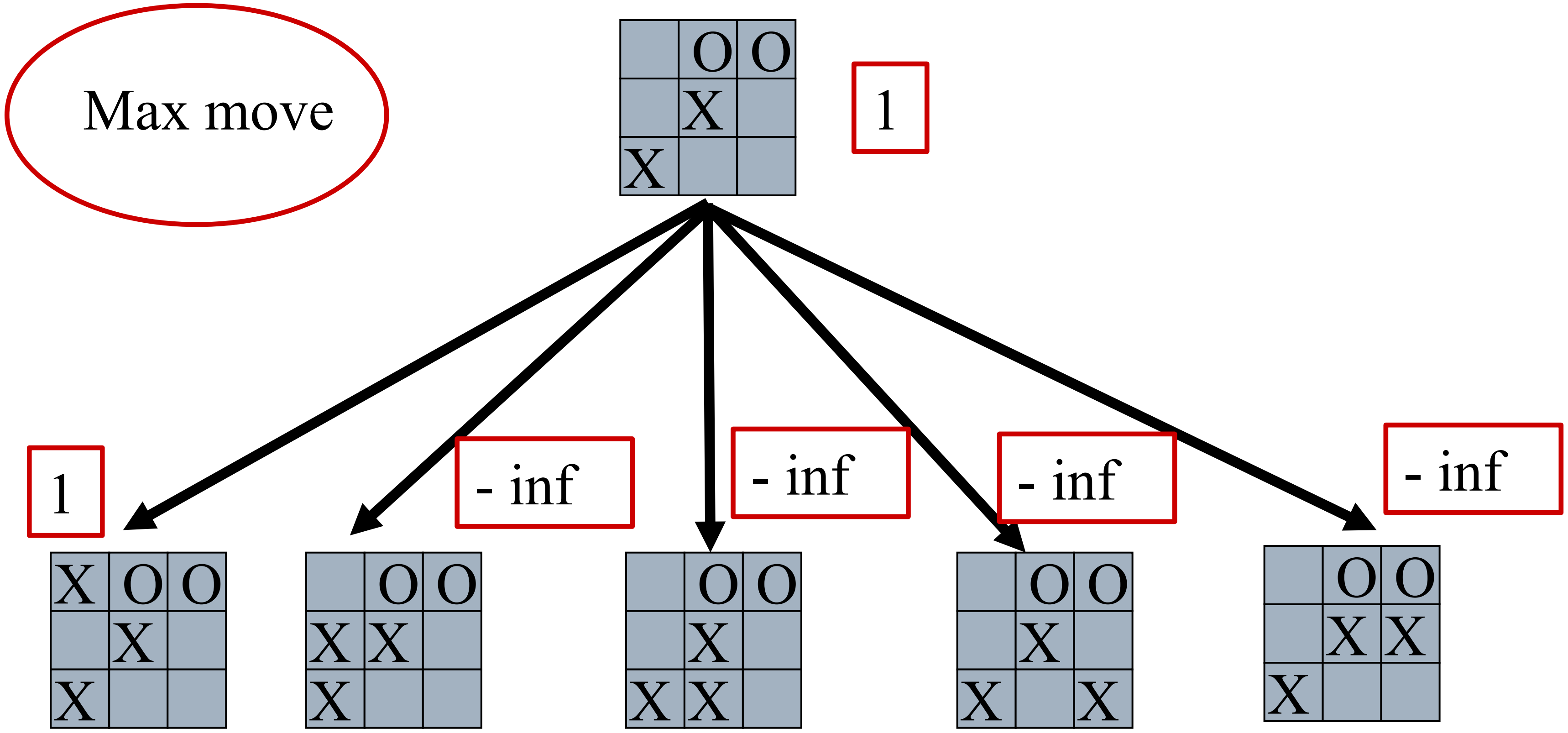


Tic-Tac-Toe

Min move



Tic-Tac-Toe


Max move
O O
X
X
1
1
X O O
X
X
- inf
O O
X X
X
- inf
O O
X
X X
- inf
O O
X
X X
- inf
O O
X X
X

Minimax Procedure

Goal of game tree search

To determine one move for Max player that maximizes the guaranteed payoff for a given game tree for MAX

Regardless of the moves the MIN will take!

The value of each node (Max and MIN) is determined by (back-up from) the values of its children

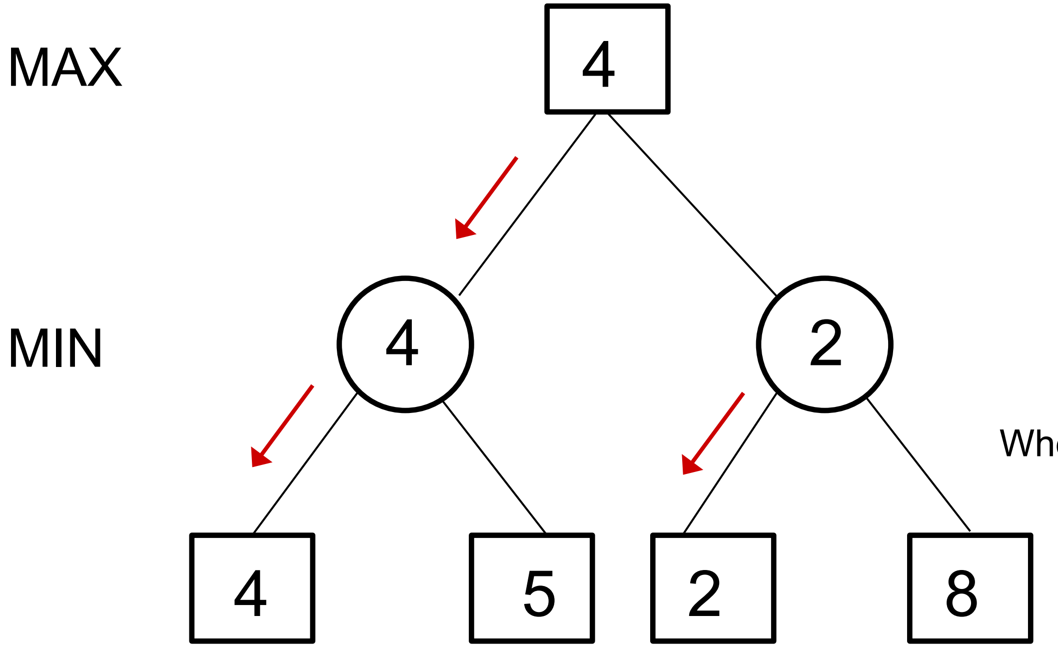
MAX plays the worst case scenario:

Assume MIN to take moves to maximize his own pay-off (i.e., to minimize the pay-off of MAX)

Minimax Procedure

Static evaluation functions measures the worth of a leaf node.

Measurement is based on features thought to influence the worth.
E.g. In checkers – relative piece advantage; control of center etc.



Adversarial game – Competing with each other.

Far shorter than 8, the MAX player wanted.
More than 2 that the MIN player wanted.

Minimax Algorithm

1. Go to the bottom of the tree.
2. Compute static values.
3. Back them up level-by-level.
4. Decide where to go.

Back-up values one level.

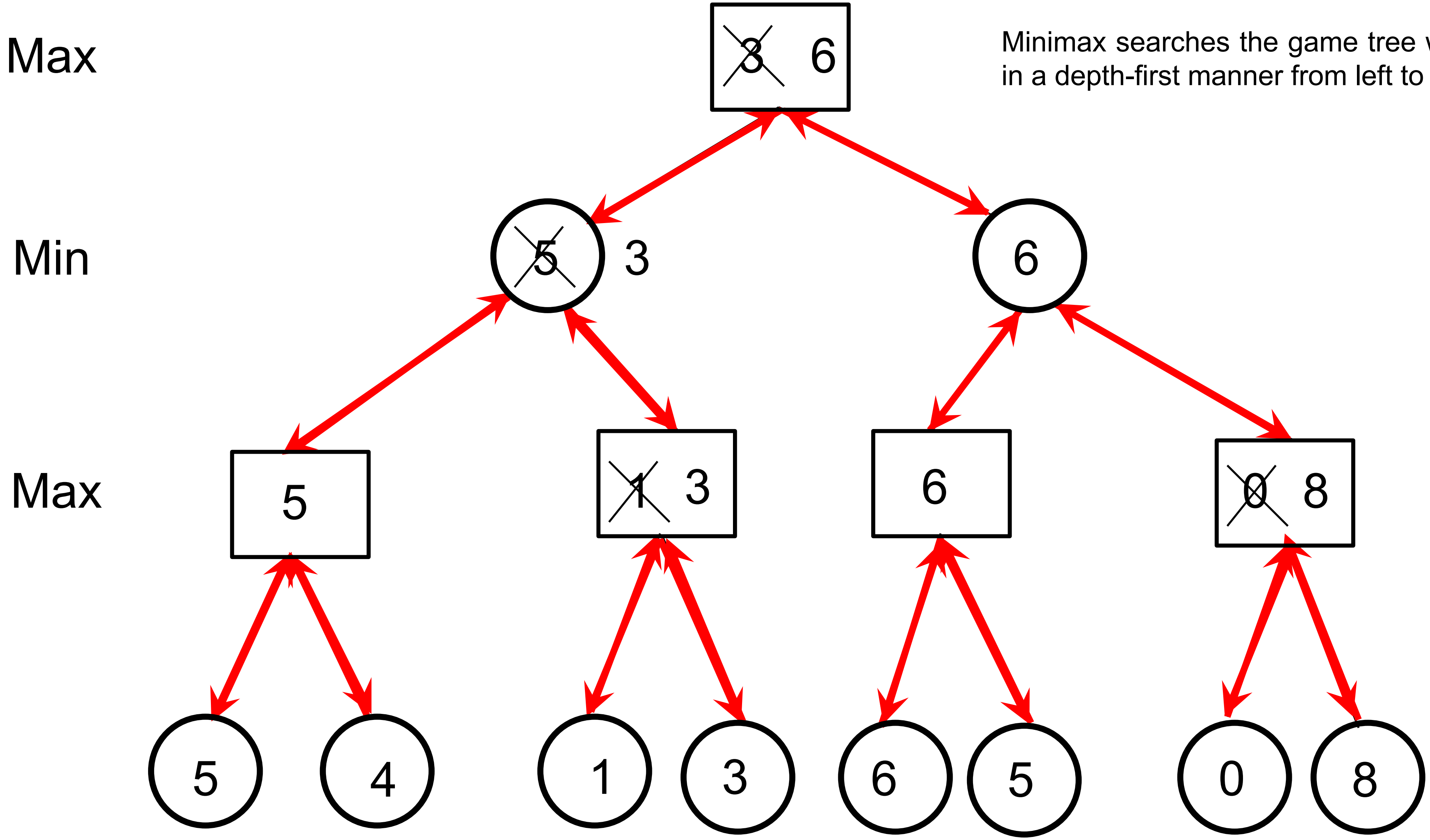
Where is the play going to go?

Value of the board from the perspective of the MAX player.

Do not expect to go where YOU wanted!

Minimax

Minimax searches the game tree with depth k in a depth-first manner from left to right.



Minimax Procedure

- Usually not possible to expand a game to end-game status
- For Lookahead one needs to choose a ply-depth that is achievable with reasonable time and resources.

- For a game tree:

  Can we have ways and means of reducing the search space?

  Each node has b children and a d-ply lookahead is performed.

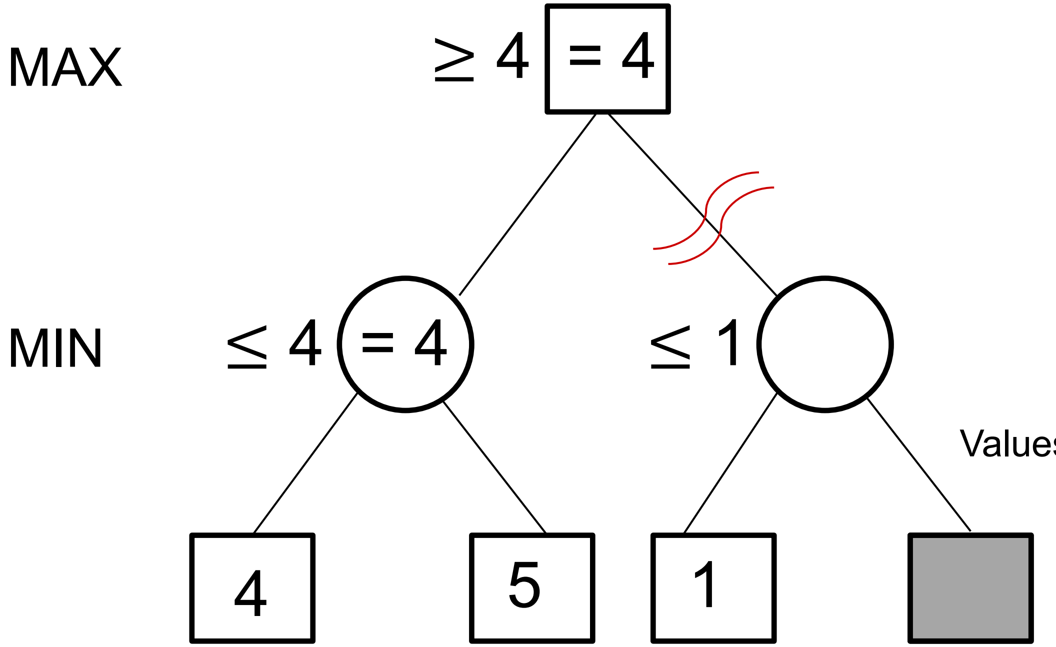
  Examining b^d leaf nodes

Minimax Procedure

- Search procedure for Minimax that we have described separates completely the processes of search-tree generation and position evaluation.
  - Position evaluation begins ONLY AFTER the tree generation is completed.
- Separation of the tree generation and position evaluation results in grossly inefficient strategy!
- Tip-node evaluation and calculation of backed-up values simultaneously with tree generation can be performed.

  Amounting sometimes to many orders of magnitude.
  - Remarkable reductions in the amount of search required to discover equally good moves.

Alpha-beta Pruning



Alpha-beta Algorithm

1. Not a separate algorithm!
2. Layering on top of Minimax.

Essence of Alpha-Beta Pruning; cuts out sections of the search space

Values in the cut-out branch can't affect the value of the root node.

Compute static values one at a time.

We don't need to compute the value at this node.

Alpha-beta Pruning

- Alpha-beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithm.
- It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree.
- It cuts off branches in the game tree which need not be searched because there already exists a better move available.
- It is called Alpha-Beta pruning because it passes two extra parameters in the minimax function
  - alpha and beta.

Alpha-beta Pruning

Traverse the search tree in depth-first order

alpha(n) The best value that MAX currently can guarantee; maximum value found so far.

beta(n) The best value that MIN currently can guarantee; minimum value found so far

The alpha values start at -infinity and only increase, while beta values start at +infinity and only decrease.

Alpha-beta Pruning

- **Beta cutoff**

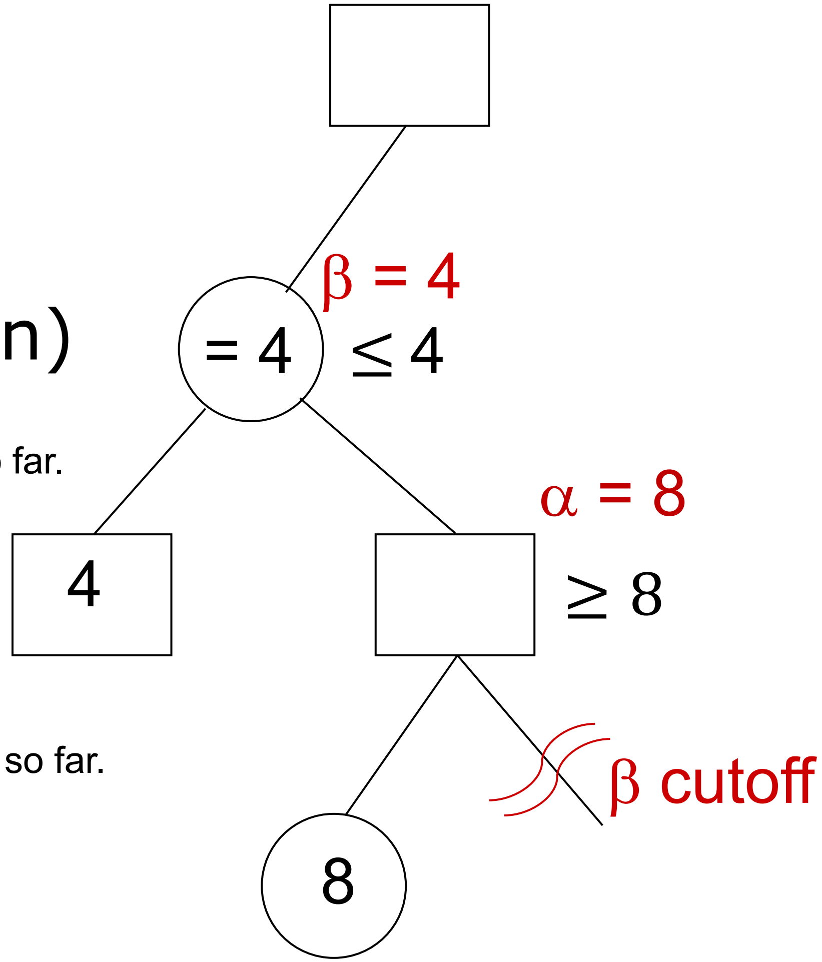
Cutoff- Do not generate or examine any more of n's children.

Given a MAX node n

Cutoff search below n

if alpha(n) $\geq$ beta(i)

(for some MIN node ancestor i of n)



beta - best value that MIN currently can guarantee; minimum value found so far.

alpha - best value that MAX currently can guarantee; maximum value found so far.

Alpha-beta Pruning

- **Alpha cutoff**

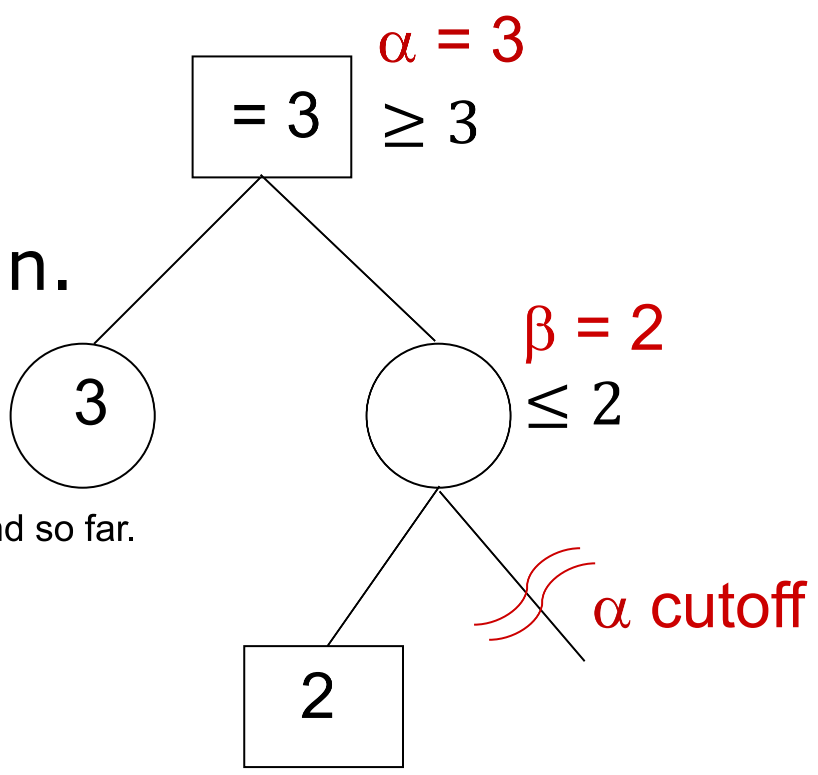
Cutoff- Do not generate or examine any more of n's children.

Given a MIN node n

Cutoff search below n

if beta(n) $\leq$ alpha(i)

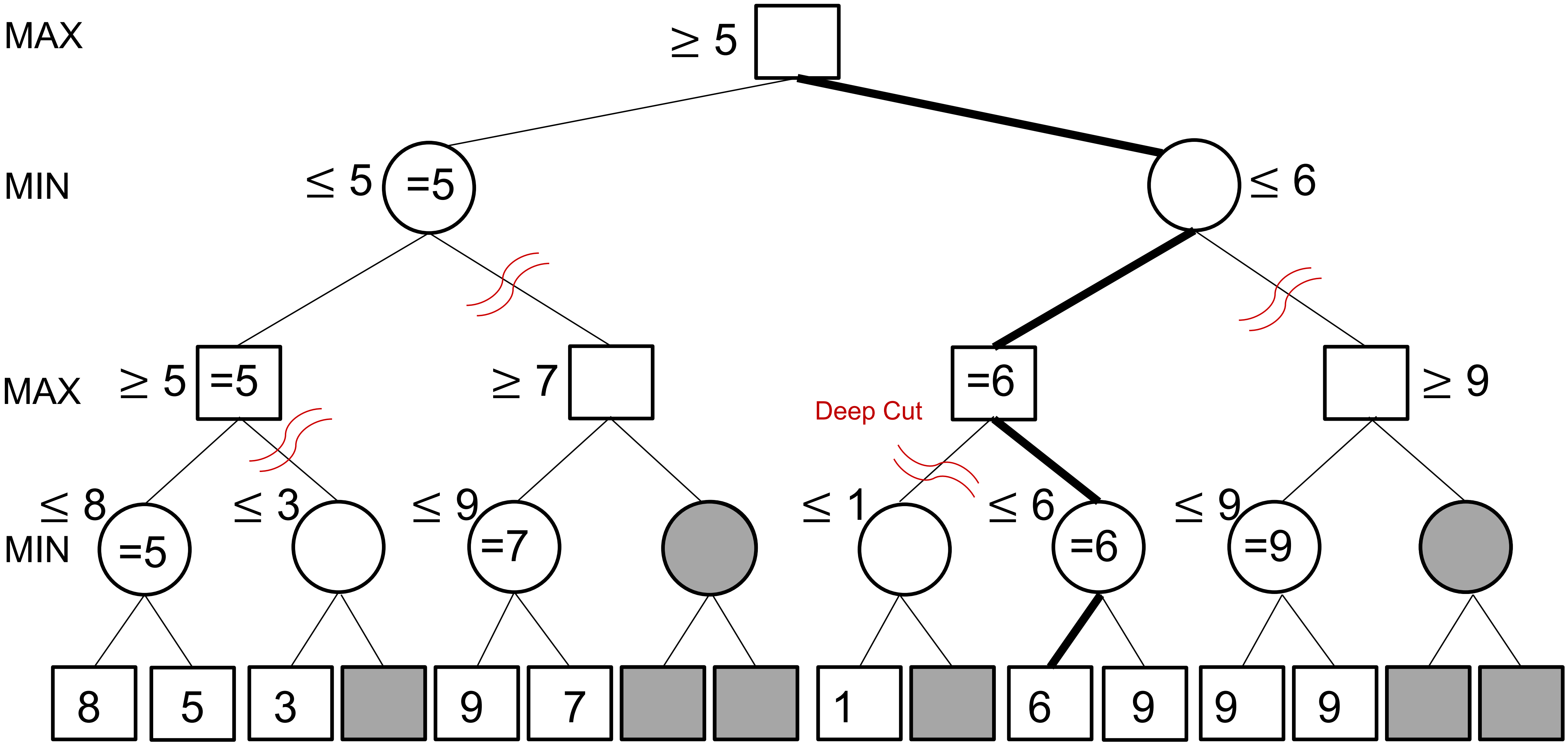
for some MAX node ancestor i of n.



beta - best value that MIN currently can guarantee; minimum value found so far.

alpha - best value that MAX currently can guarantee; maximum value found so far.

Alpha-beta Pruning



Effectiveness of Alpha-beta Pruning

- Alpha-beta is guaranteed to compute the same value for the root node as computed by minimax, with less or equal computation

Worst case:

- No pruning.
- Examining b^d leaf nodes, where each node has b children and a d-ply search is performed.

Best case:

- Examine only (2b)^(d/2) leaf nodes.
- Result is you can search twice as deep as minimax!

Effectiveness of Alpha-beta Pruning

- Minimax algorithm and its variants - inherently depth-first.
- Iterative deepening is usually used in conjunction with alpha–beta so that a reasonably good move can be returned even if the algorithm is interrupted before it has finished execution.
  - Using iterative deepening can give move-ordering hints at shallower depths; as well as shallow alpha and beta estimates.
    - Both can help produce cutoffs for higher depth searches much earlier than would otherwise be possible.
- There exists algorithms that use the best-first strategy.
  - More time-efficient.
  - But typically at a heavy cost in space-efficiency.