Week 4: Games

Dr. Miqing Li

Week3 exercise

- Consider the following 4x4 Sudoku problem, where each column, each row, and each of the four regions contain all of the digits from 1 to 4. Use backtracking search with forward checking and ordering to solve this problem. Give the order of all states to be visited. Let us assume that tie of cells is broken first from top to bottom and then from left to right, and tie of numbers is broken numerically.

3	4		
	2		
		4	

  - Backtracking:
    - Check constraints as you go
    - Consider one variable at a layer
  - Forward Checking:
    - When assigning a variable, cross off anything that is now violated on all of its neighbours' domains.
  - Ordering:
    - Choose the variable with the fewest legal values left in its domain.

Question 1

- Consider the following 4x4 Sudoku problem, where each column, each row, and each of the four regions contain all of the digits from 1 to 4. Use backtracking search with forward checking and ordering to solve this problem. Give the order of all states to be visited. Let us assume that tie of cells is broken first from top to bottom and then from left to right, and tie of numbers is broken numerically.

3	4	{1,2}	{1,2}
{1}	2	{1,3}	{1,3,4}
{1,2,4}	{1,3}	{1,2,3}	{1,2,3}
{1,2}	{1,3}	4	{1,2,3}

3	4	{1,2}	{1,2}
1	2	{3}	{3,4}
{2,4}	{1,3}	{1,2,3}	{1,2,3}
{2}	{1,3}	4	{1,2,3}

3	4	{1,2}	{1,2}
1	2	3	{4}
{2,4}	{1,3}	{1,2}	{1,2,3}
{2}	{1,3}	4	{1,2,3}

3	4	{1,2}	{1,2}
1	2	3	4
{2,4}	{1,3}	{1,2}	{1,2,3}
{2}	{1,3}	4	{1,2,3}

3	4	{1,2}	{1,2}
1	2	3	4
{4}	{1,3}	{1,2}	{1,2,3}
2	{1,3}	4	{1,3}

3	4	{1,2}	{1,2}
1	2	3	4
4	{1,3}	{1,2}	{1,2,3}
2	{1,3}	4	{1,3}

Question 1

- Consider the following 4x4 Sudoku problem, where each column, each row, and each of the four regions contain all of the digits from 1 to 4. Use backtracking search with forward checking and ordering to solve this problem. Give the order of all states to be visited. Let us assume that tie of cells is broken first from top to bottom and then from left to right, and tie of numbers is broken numerically.

3	4	1	{2}
1	2	3	4
4	{1,3}	{2}	{1,2,3}
2	{1,3}	4	{1,3}

3	4	1	2
1	2	3	4
4	{1,3}	{2}	{1,3}
2	{1,3}	4	{1,3}

3	4	1	2
1	2	3	4
4	{1,3}	2	{1,3}
2	{1,3}	4	{1,3}

3	4	1	2
1	2	3	4
4	1	2	{3}
2	{3}	4	{1,3}

3	4	1	2
1	2	3	4
4	1	2	3
2	{3}	4	{1}

3	4	1	2
1	2	3	4
4	1	2	3
2	3	4	{1}

Question 1

- Consider the following 4x4 Sudoku problem, where each column, each row, and each of the four regions contain all of the digits from 1 to 4. Use backtracking search with forward checking and ordering to solve this problem. Give the order of all states to be visited. Let us assume that tie of cells is broken first from top to bottom and then from left to right, and tie of numbers is broken numerically.

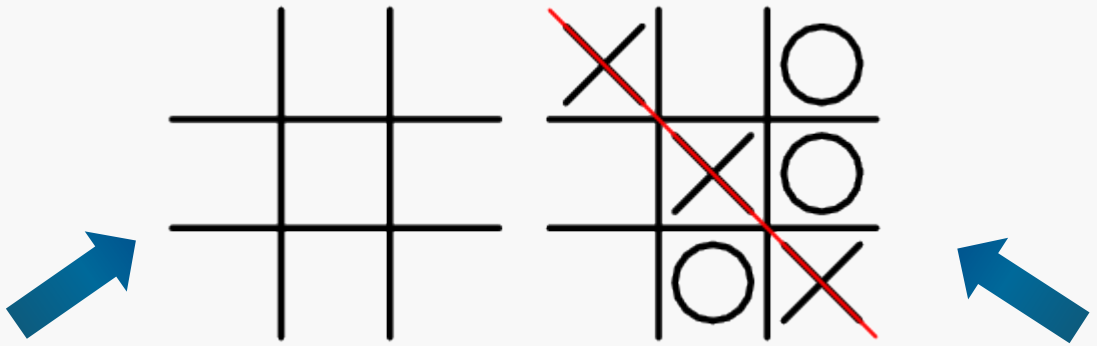
3	4	1	2
1	2	3	4
4	1	2	3
2	3	4	1

The order of states to be visited:

$Cell_{21}=1 \Rightarrow Cell_{23}=3 \Rightarrow Cell_{24}=4 \Rightarrow Cell_{41}=2 \Rightarrow Cell_{31}=4 \Rightarrow Cell_{13}=1 \Rightarrow Cell_{14}=2 \Rightarrow Cell_{33}=2$
$\Rightarrow Cell_{32}=1 \Rightarrow Cell_{34}=3 \Rightarrow Cell_{42}=3 \Rightarrow Cell_{44}=1$

Game is a search problem

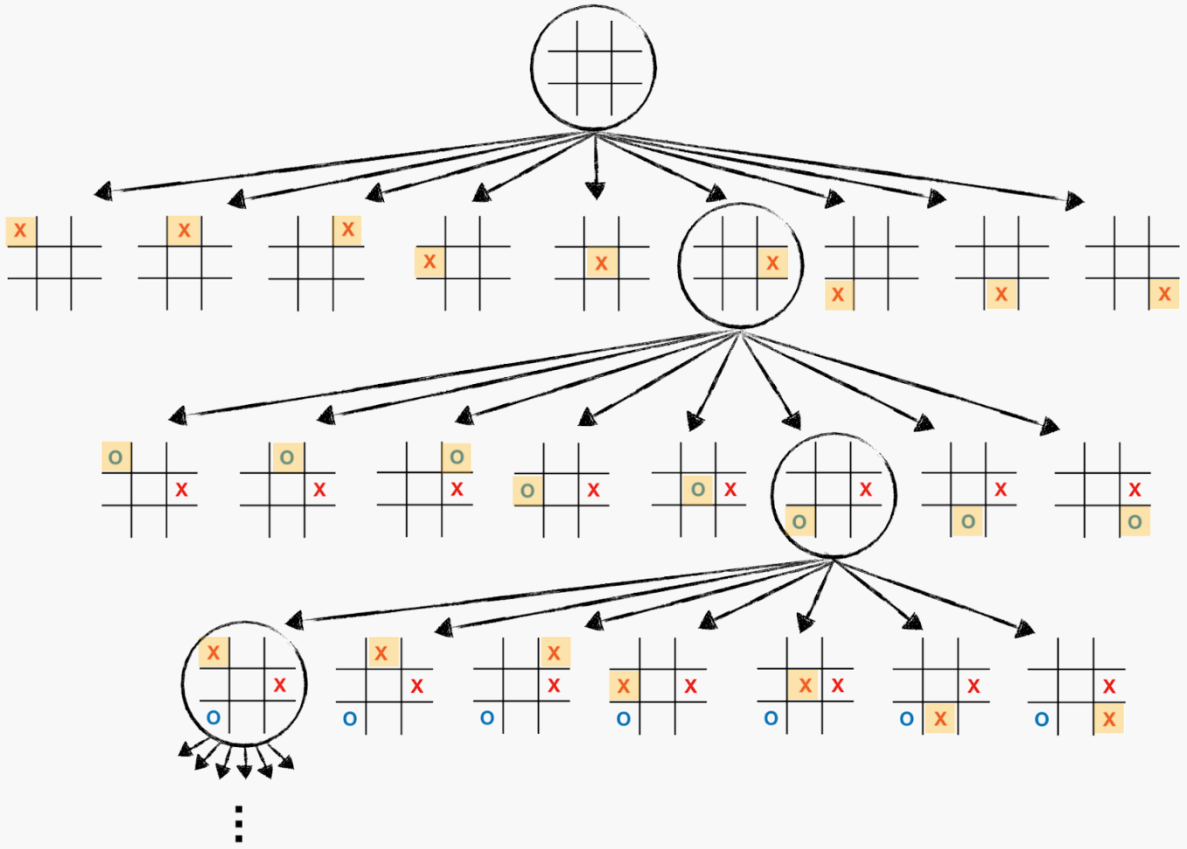
- Example: tic-tac-toe (also called noughts and crosses) is a game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a diagonal, horizontal, or vertical row is the winner.



Start state

Goal state

Game Tree



Game is typically an **Adversarial** Search problem where your opponent has something to say about your strategy.

Types of Games

- Many different types of games
  - Is there some randomness element in the game
    - Deterministic, e.g. Tic-Tac-Toe, Chess, Chinese Chess, Go
    - Stochastic, e.g. Poker, Mahjong
  - How many players
    - One, e.g. Solitaire, various puzzle games
    - Two, e.g. Tic-Tac-Toe, Chess, Chinese Chess, Go
    - More than two, e.g. many Poker games, Mahjong
  - Are you playing against each other (strictly competitive)
    - Zero-sum, e.g. Tic-Tac-Toe, Chess, Go, Poker
    - Non-zero-sum, e.g. Prisoner's Dilemma
  - Can you see the state
    - Perfect information, e.g. Tic-Tac-Toe, Chess, Chinese Chess, Go
    - Imperfect information, e.g. Poker, Mahjong

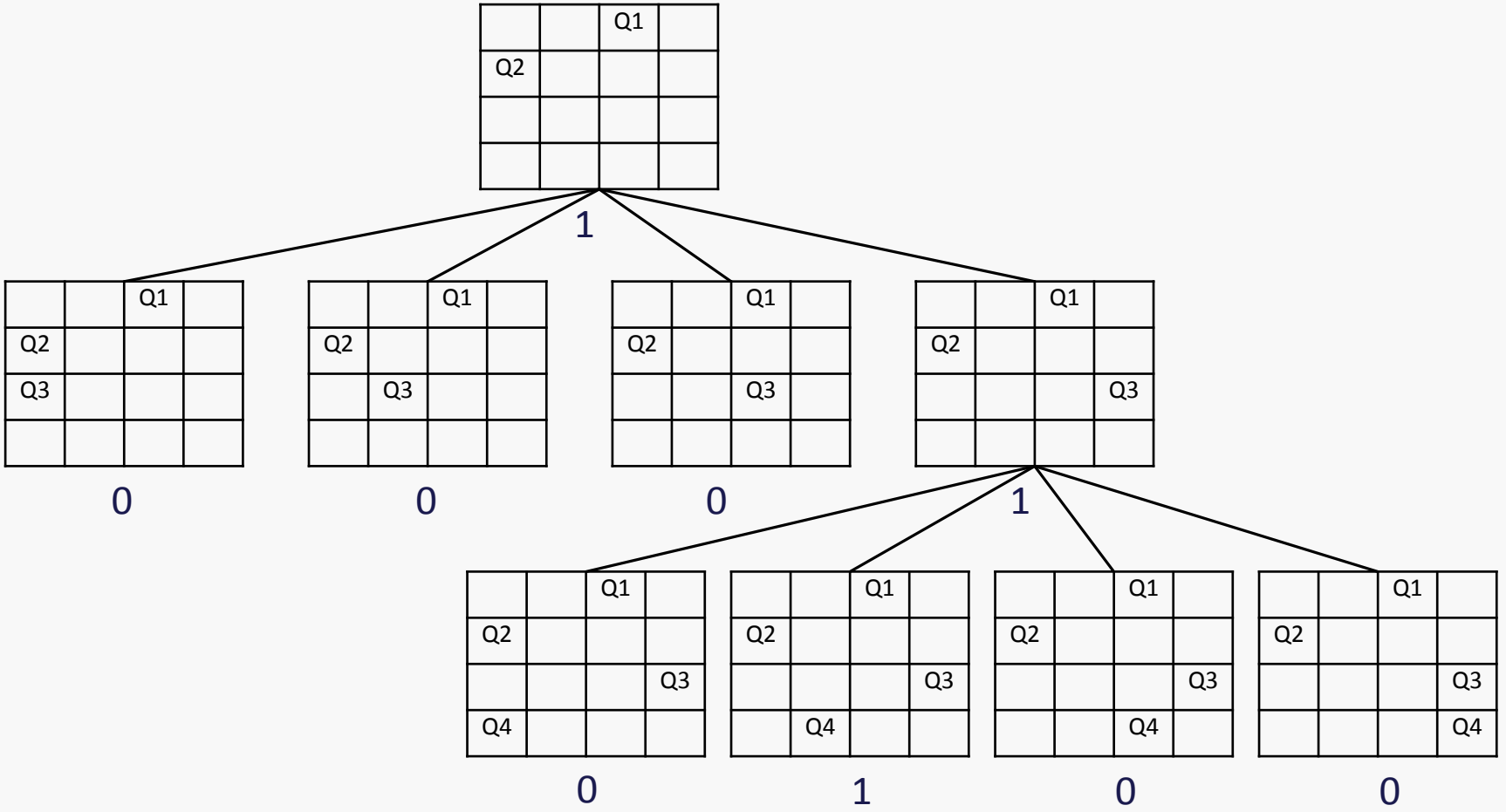
Formalisation

- States : S (Start State s_0)
- Actions: A (may depend on player/state)
- Transition function: $S \times A \rightarrow S$
- Terminal test: $S \rightarrow (ture, false)$
- Players: $P = (1, \dots, N)$ (usually take turns)
- Utilities: $S_{terminal} \times P \rightarrow R$ (values on outcomes)
  - A utility function (also called an objective function or payoff function) defines the final numeric value for a game that ends in terminal state s for a player p.

We want algorithms to find a strategy (**Policy**) which recommends a move for each state, i.e. $S \rightarrow A$

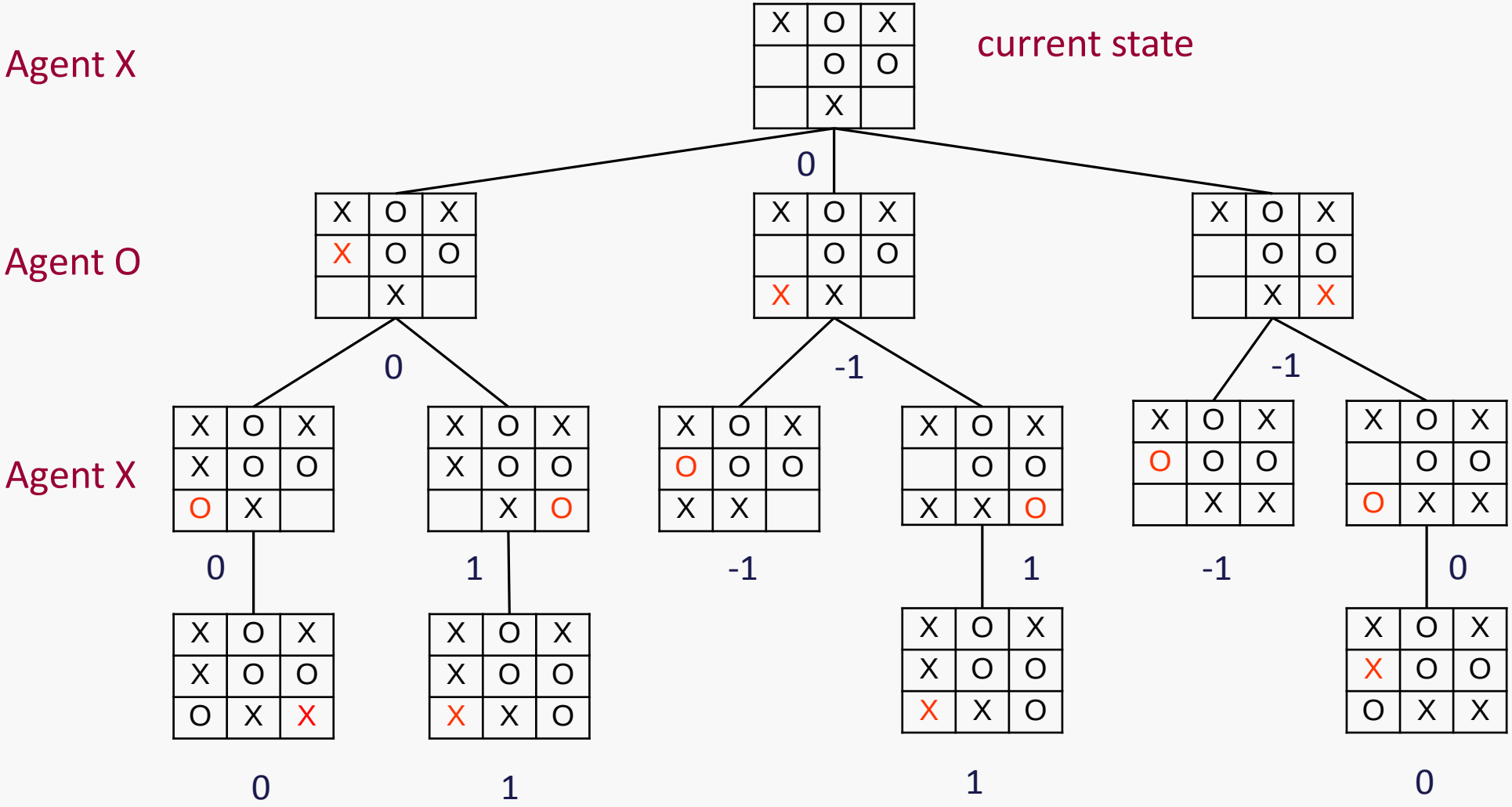
Value of a State

- Value of a state: the best achievable outcome (utility) from that state.
- Example: 4-Queens puzzle, let us say the utility of a valid solution is 1 and an invalid solution is 0.



State value in adversarial games

- Example: tic-tac-toe, utility for X: win (1), draw (0), loss (-1).



Minimax

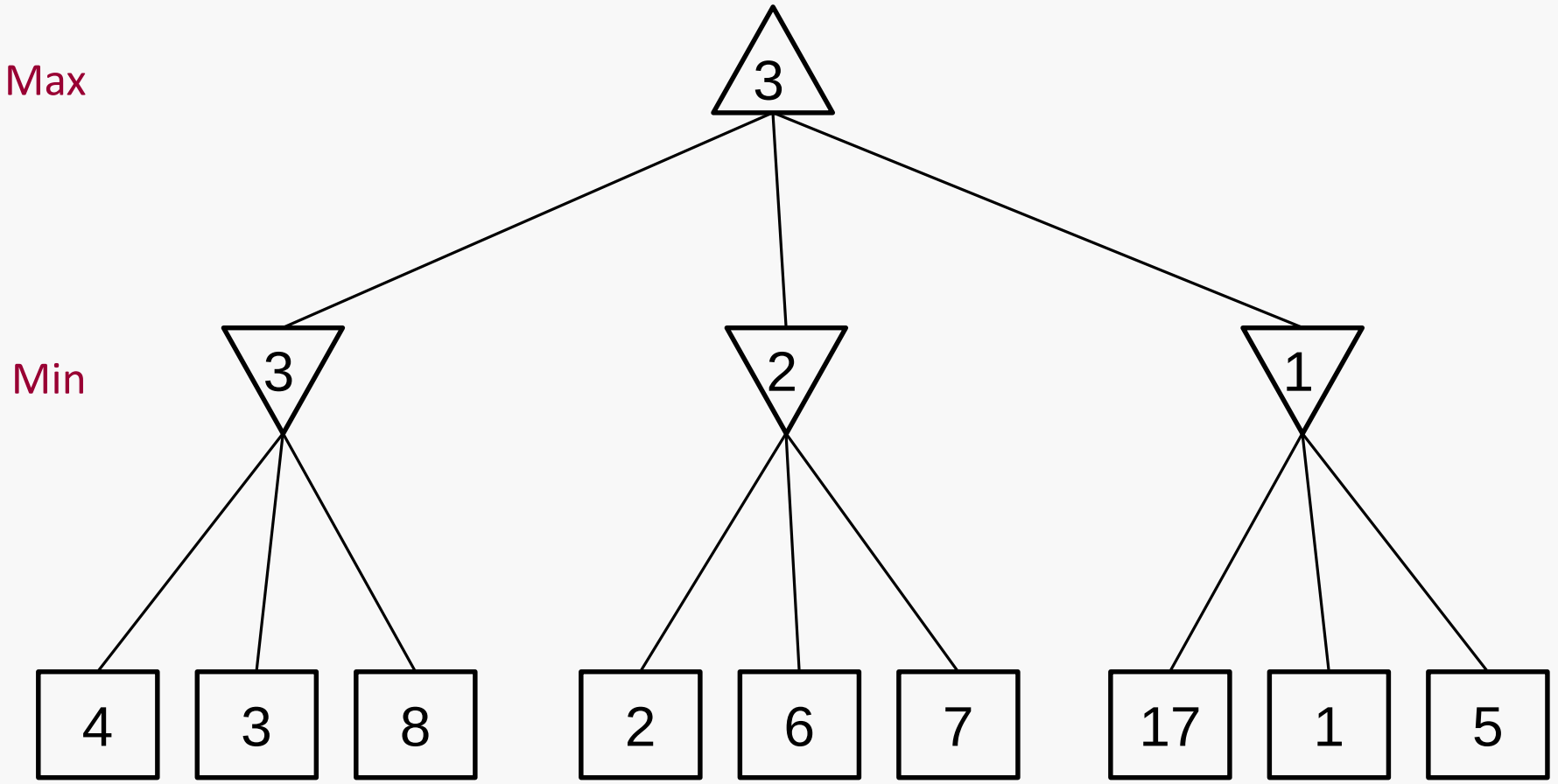
- Minimax value of a node is the utility of the terminal state to which both players play optimally from that node.

- So the process of a two-player game is that one player (called player Max) is to maximise its utility whereas its opponent (called player Min) is to minimise the utility of Max.

- For a state s, its minimax value $minimax(s)$ is

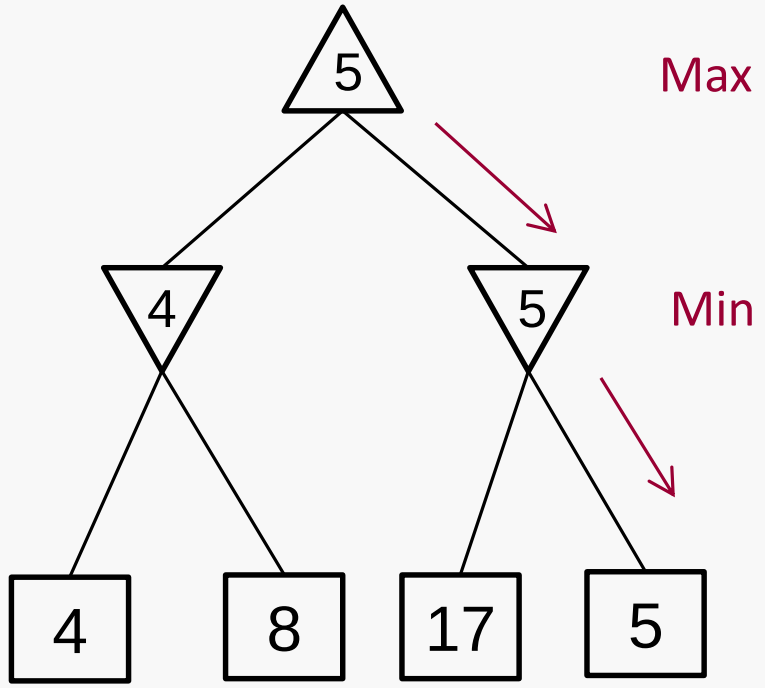
$$\begin{cases} utility(s) & \text{if } s \text{ is a terminal sate} \\ max_{s' \in \text{successor}(s)}\big(minimax(s')\big) & \text{if player is } Max \\ min_{s' \in \text{successor}(s)}\big(minimax(s')\big) & \text{if player is } Min \end{cases}$$

Minimax: example



Adversarial Search (Minimax)

- Deterministic, zero-sum games
  - Tic-tac-toe, chess, go
  - One player maximises result and the other minimises result
- Minimax search:
  - A state-space search tree
  - Players alternate turns
  - Compute each node's minimax value, i.e. the best achievable utility against an optimal adversary



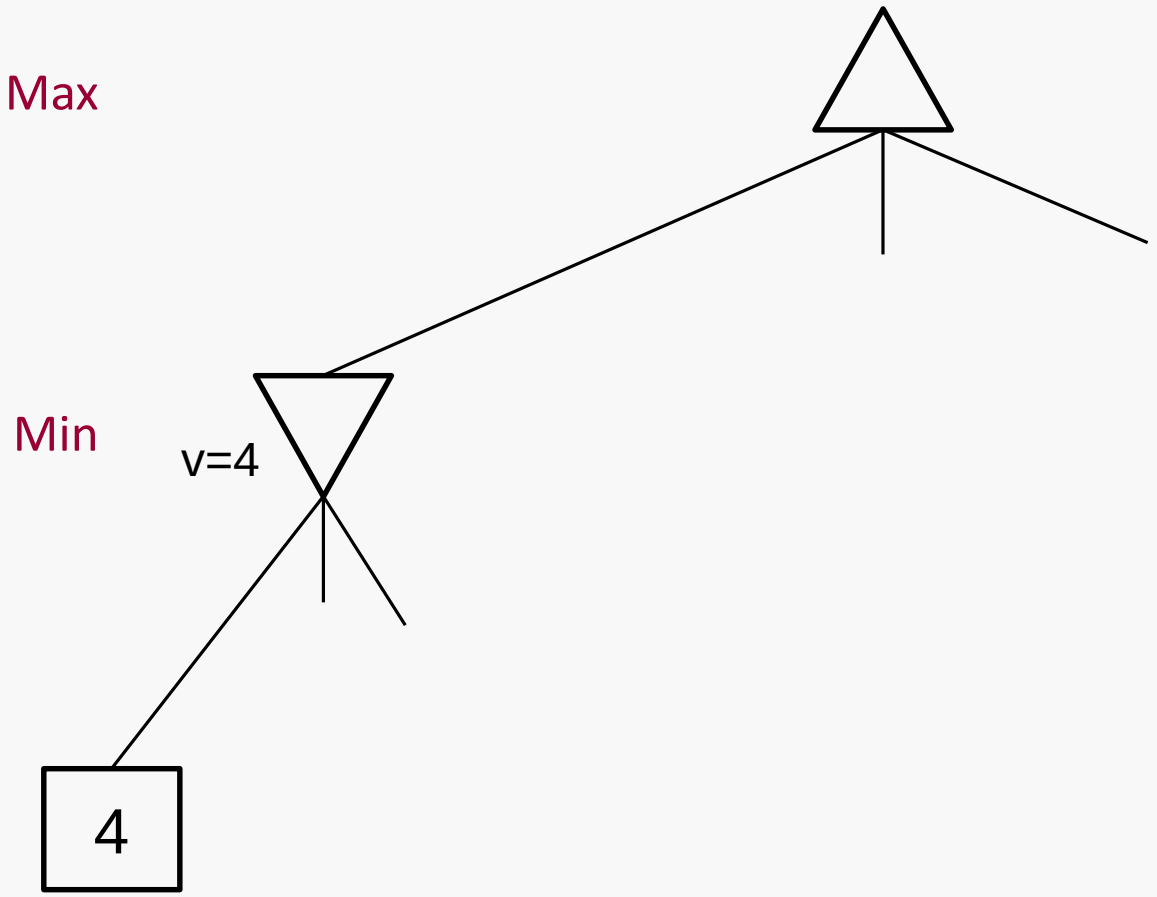
Implementing Minimax

```
function minimax_value (state) return its minimax value
    if state is a terminal state
        return its utility
    if state is for agent Max to take an action
        return max_value(state)
    if state is for agent Min to take an action
        return min_value(state)
```

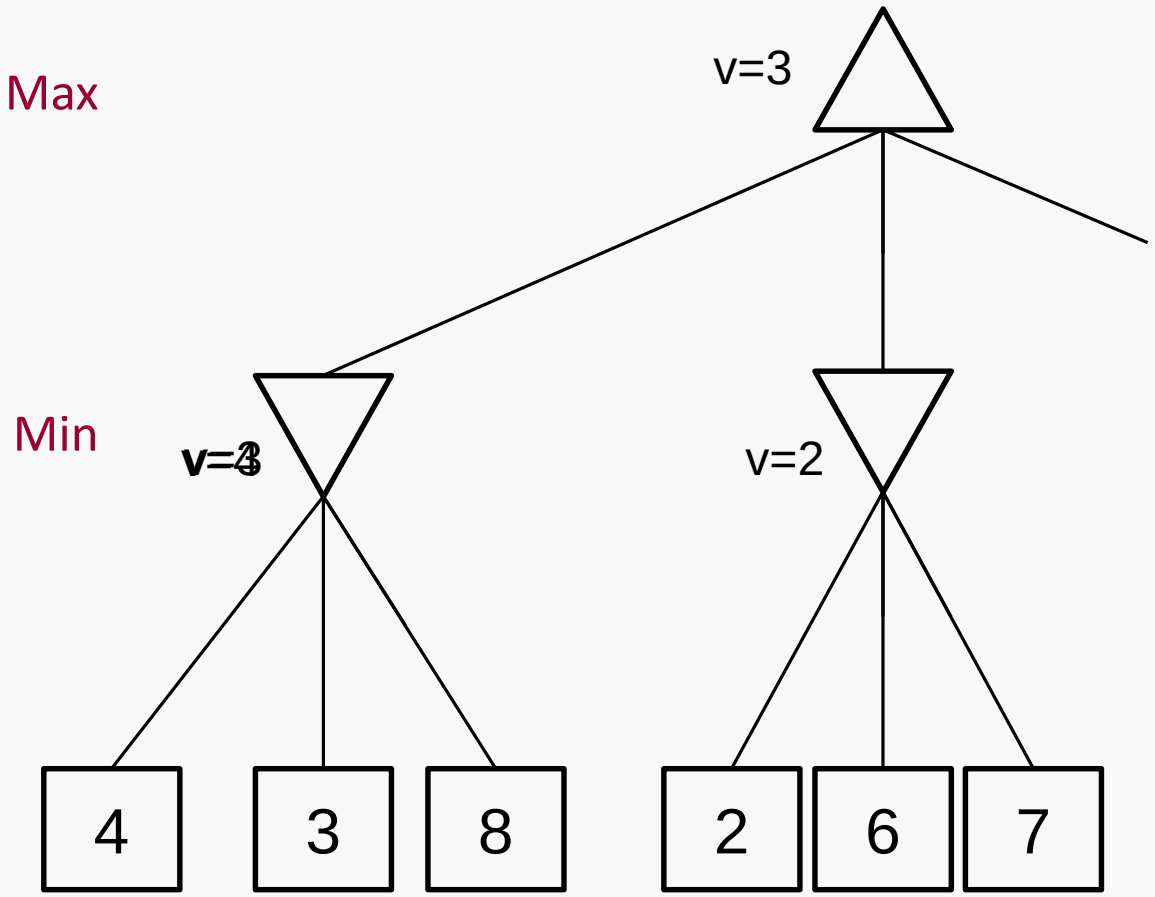
```
function max_value (state) return its minimax value v
    initialise v = −∞
    for each successor of state
        v = max(v, minimax_value(successor))
    return v
```

```
function min_value (state) return its minimax value v
    initialise v = +∞
    for each successor of state
        v = min(v, minimax_value(successor))
    return v
```

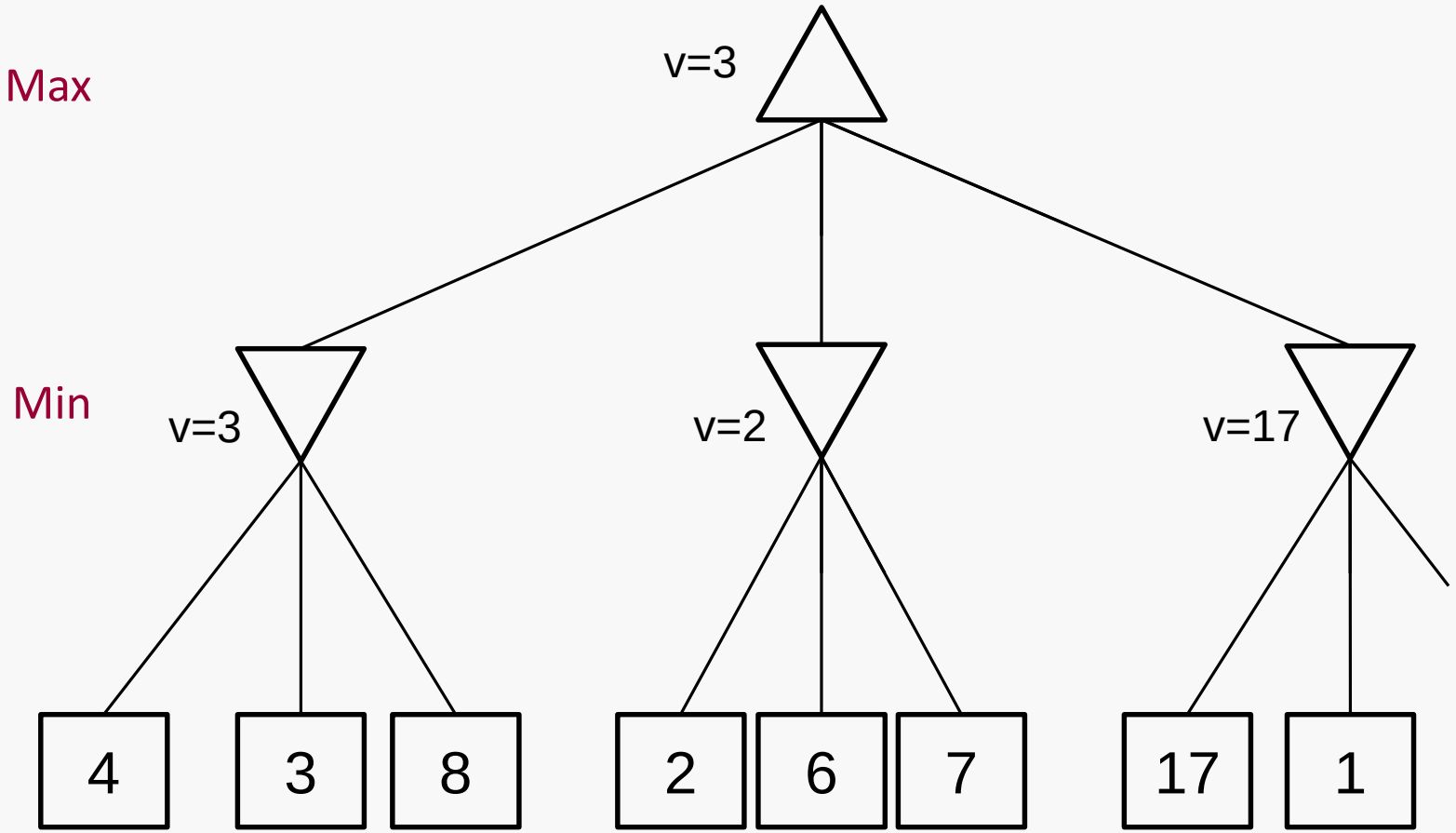
Minimax example



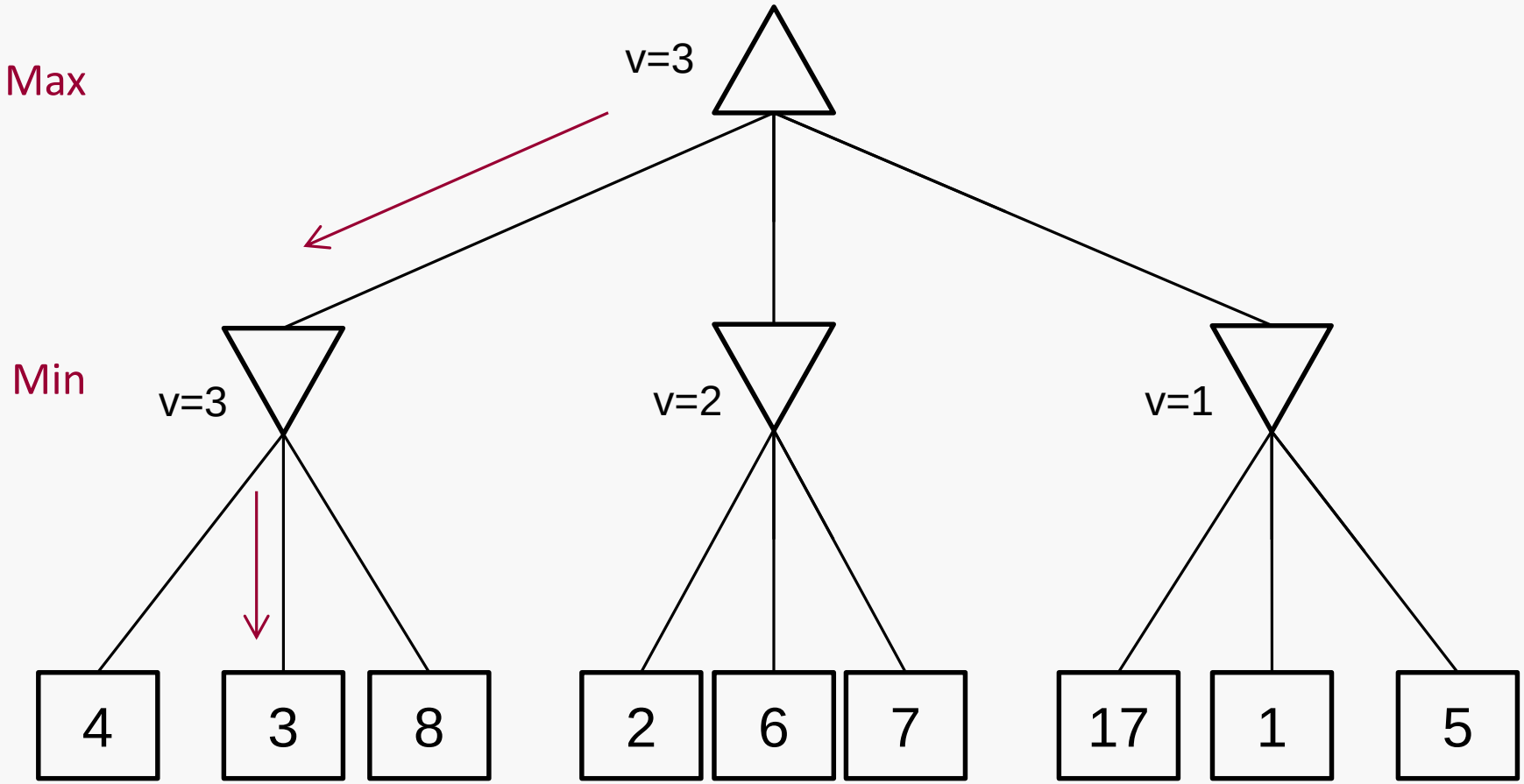
Minimax example



Minimax example



Minimax example



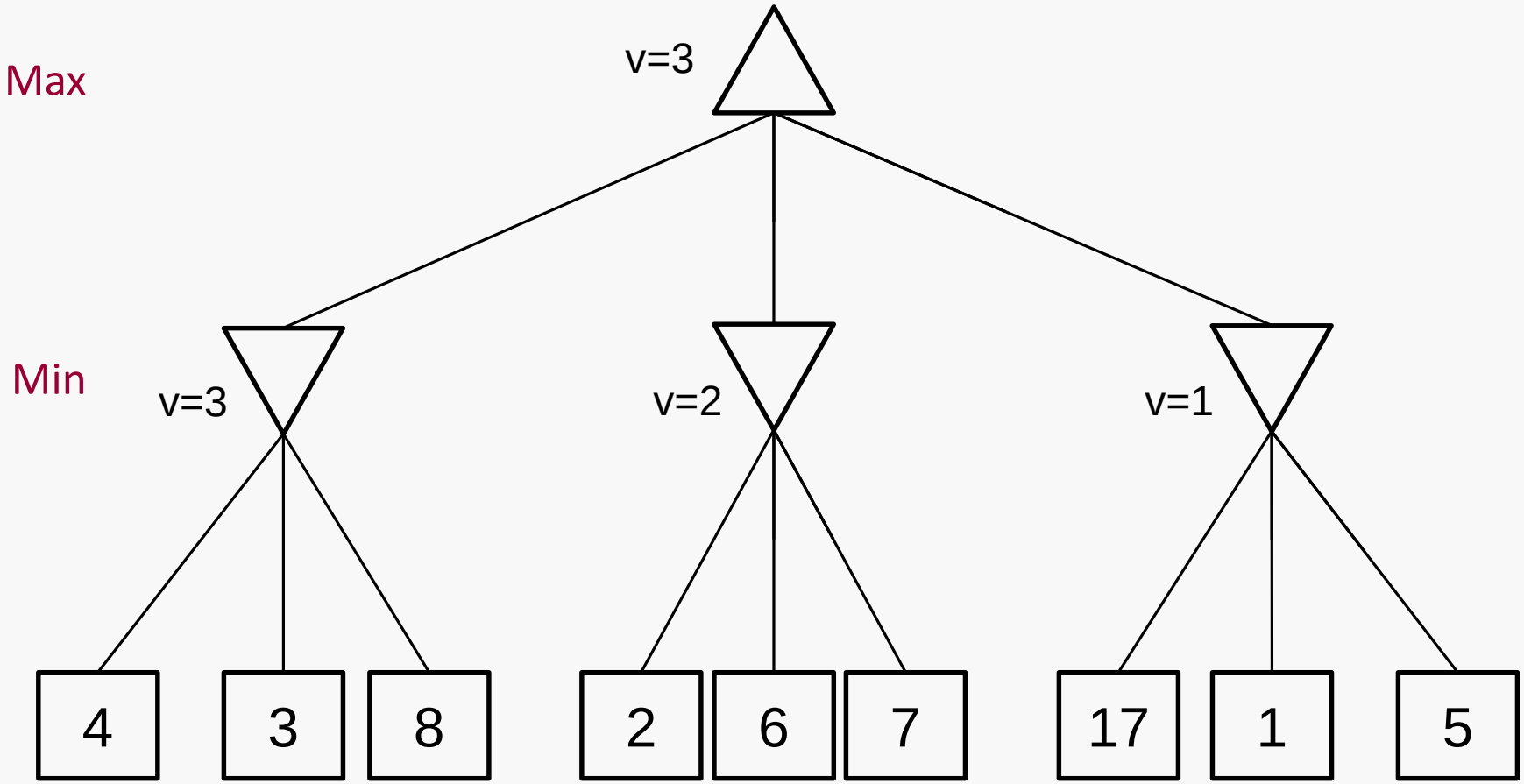
Computational Complexity of Minimax

- How efficient is minimax
  - DFS (exhaustive)
  - Time: $O(b^m)$, b is branching factor, m is maximum depth of the tree.
  - Space: $O(bm)$
- Examples
  - Chess: $b \approx 35,\ m \approx 100$
  - Go: $b \approx 250,\ m \approx 150$

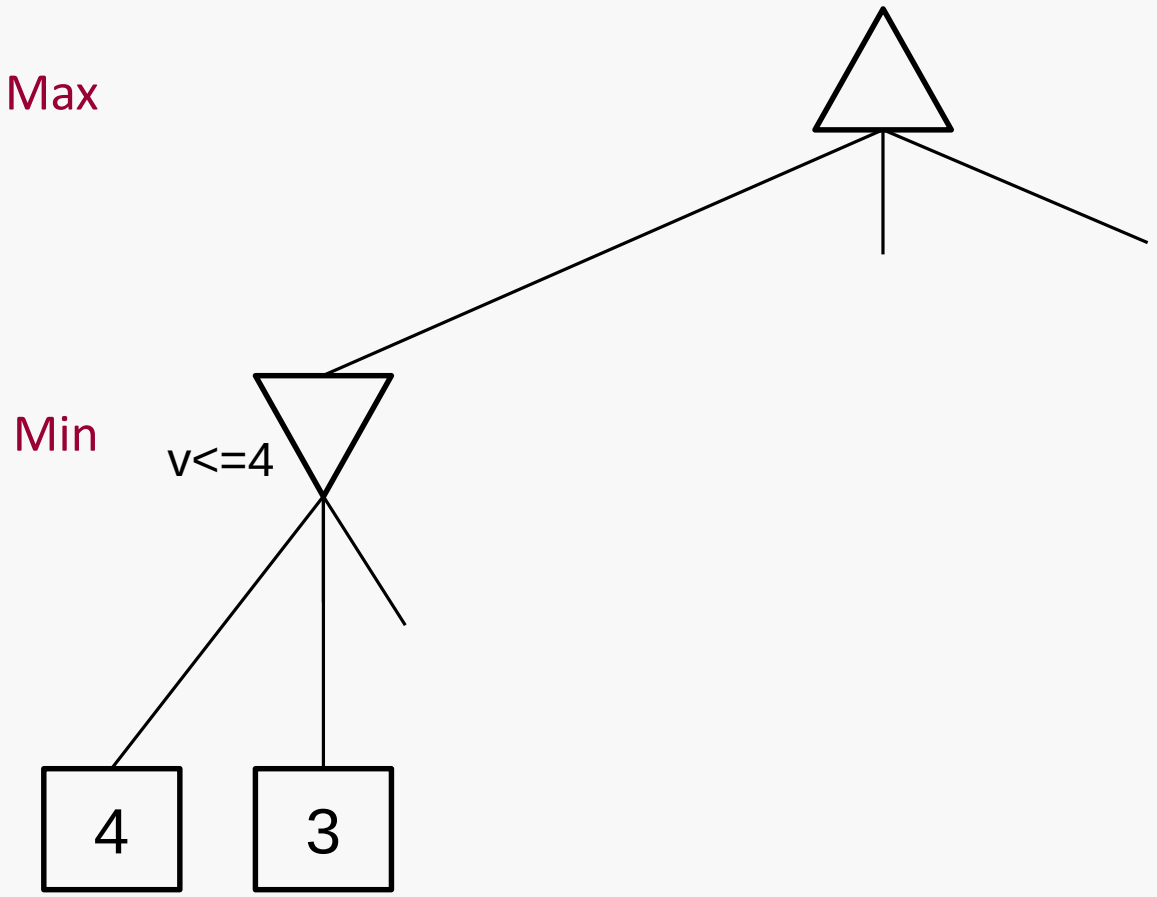
Game	Game-tree complexity (appr)
Tic-Tac-Toe	10^5
Connect Four	10^{21}
Checkers	10^{50}
Nine men's morris	10^{50}
Draughts	10^{54}
Chess	10^{123}
Backgammon	10^{144}
Chinese Chess	10^{150}
Shoji	10^{226}
Go	10^{360}

Solving them is completely infeasible in most cases, but do we need to explore the entire tree to find the minimax value?

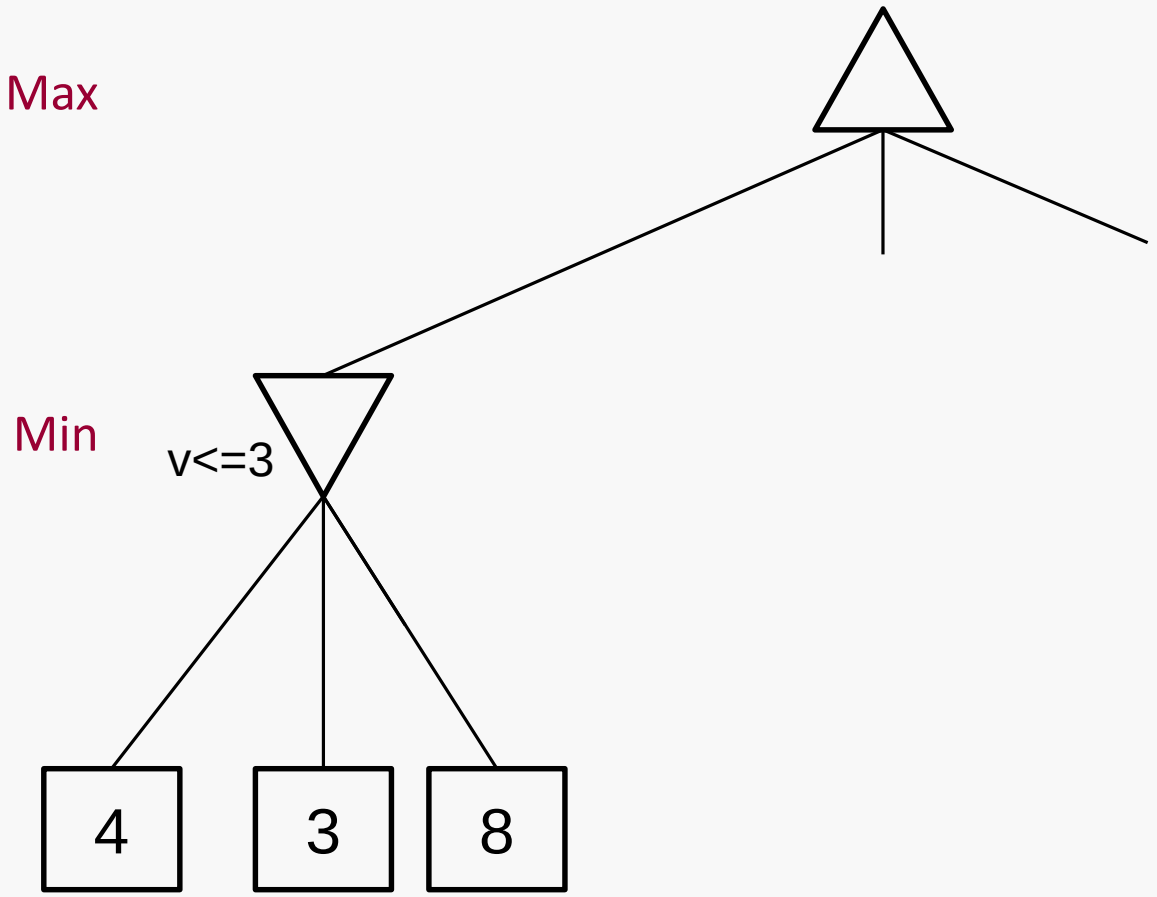
Game Tree Pruning



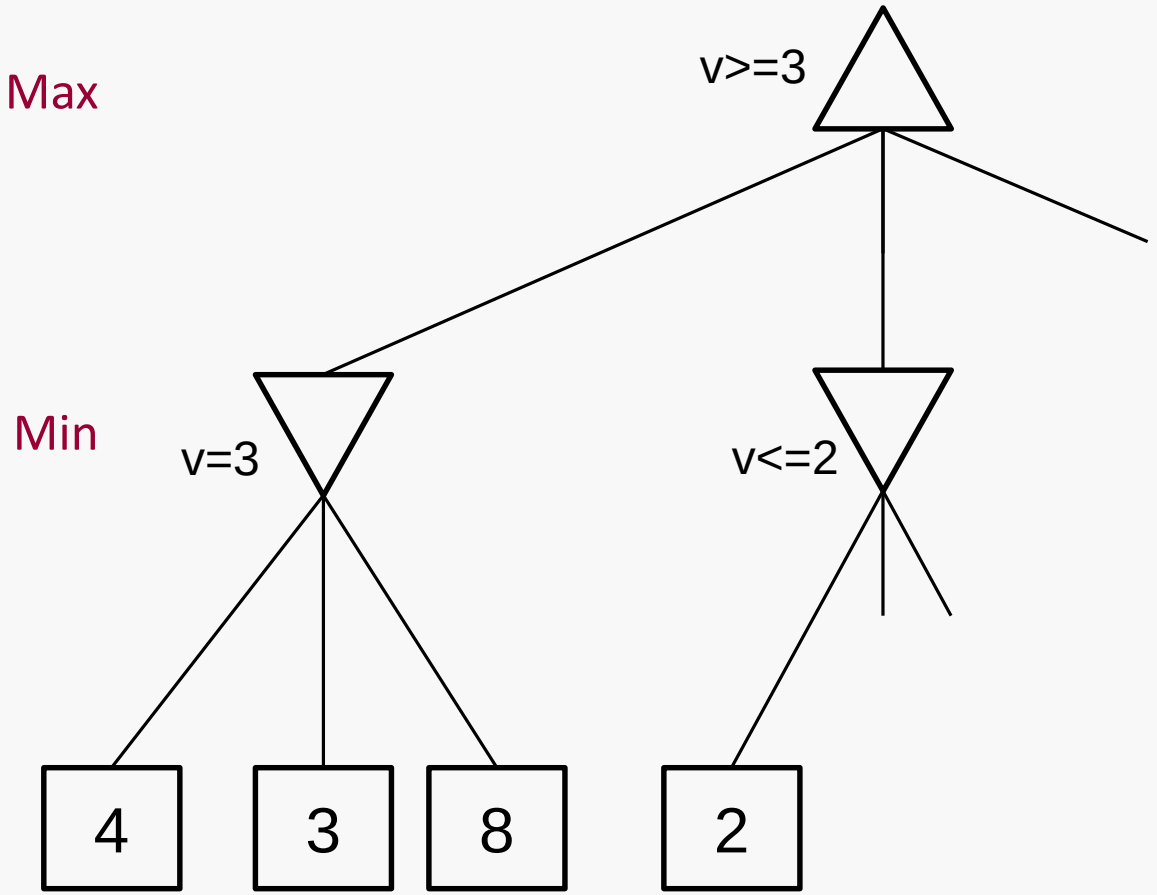
Game Tree Pruning



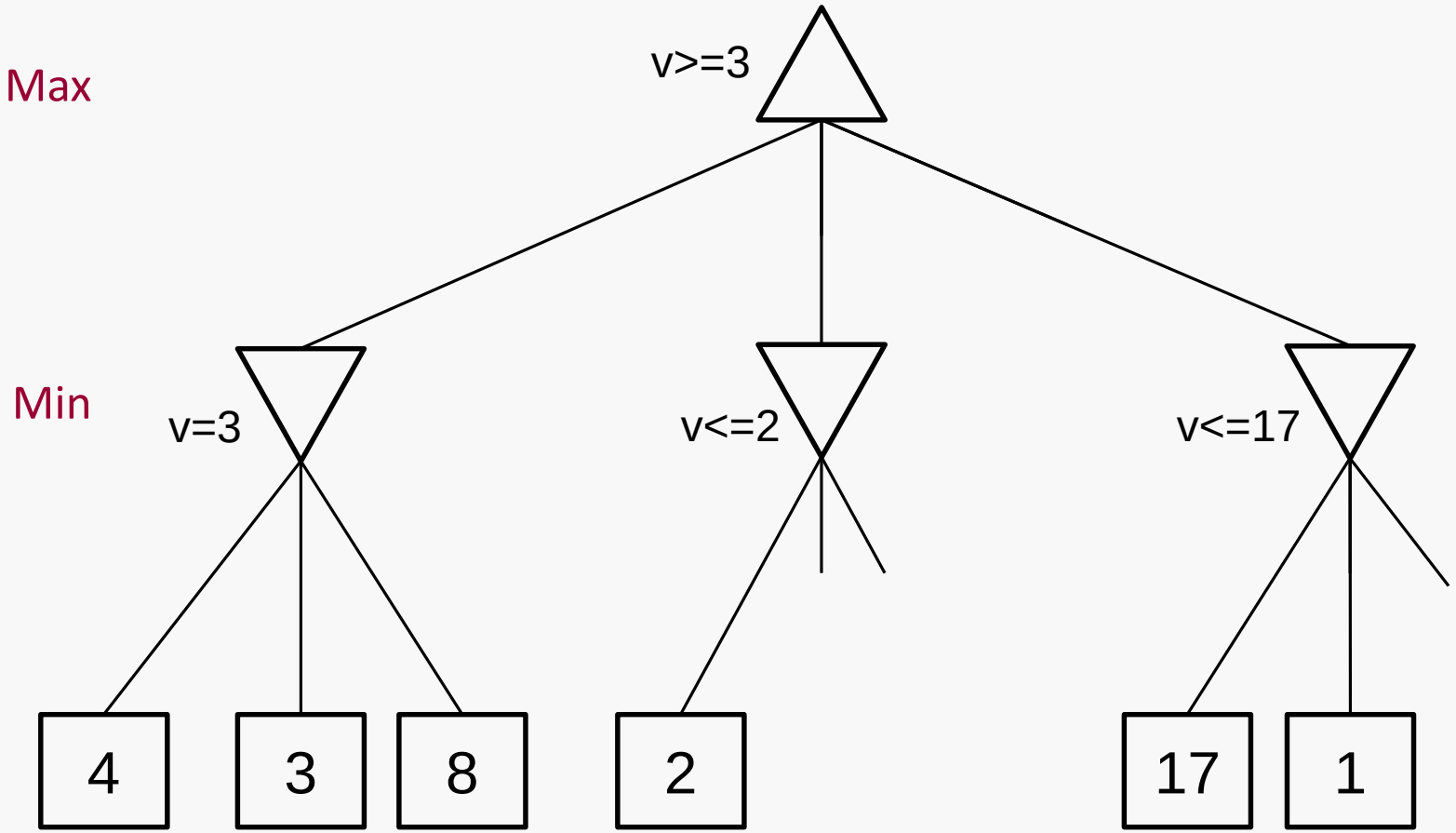
Game Tree Pruning



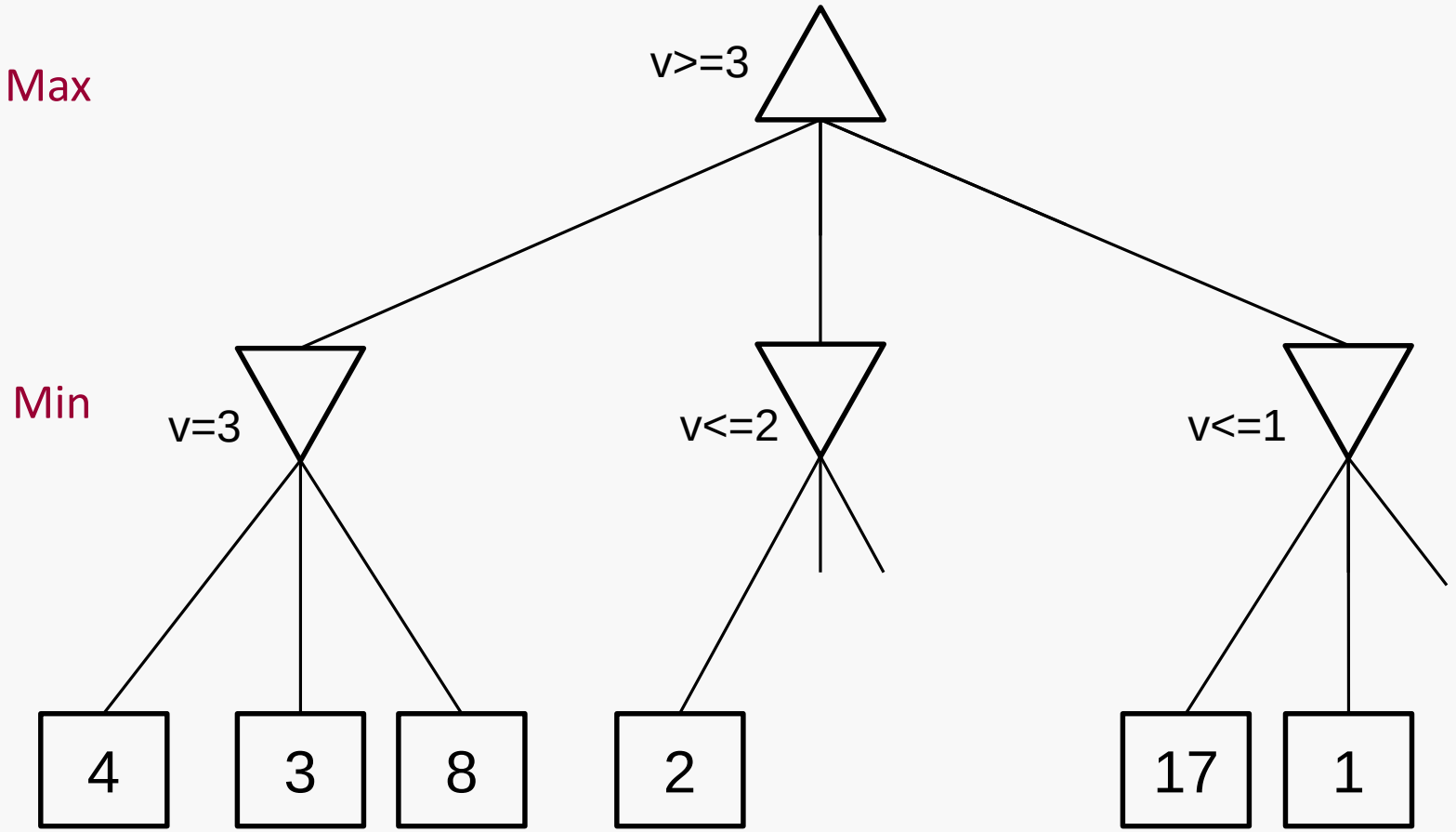
Game Tree Pruning



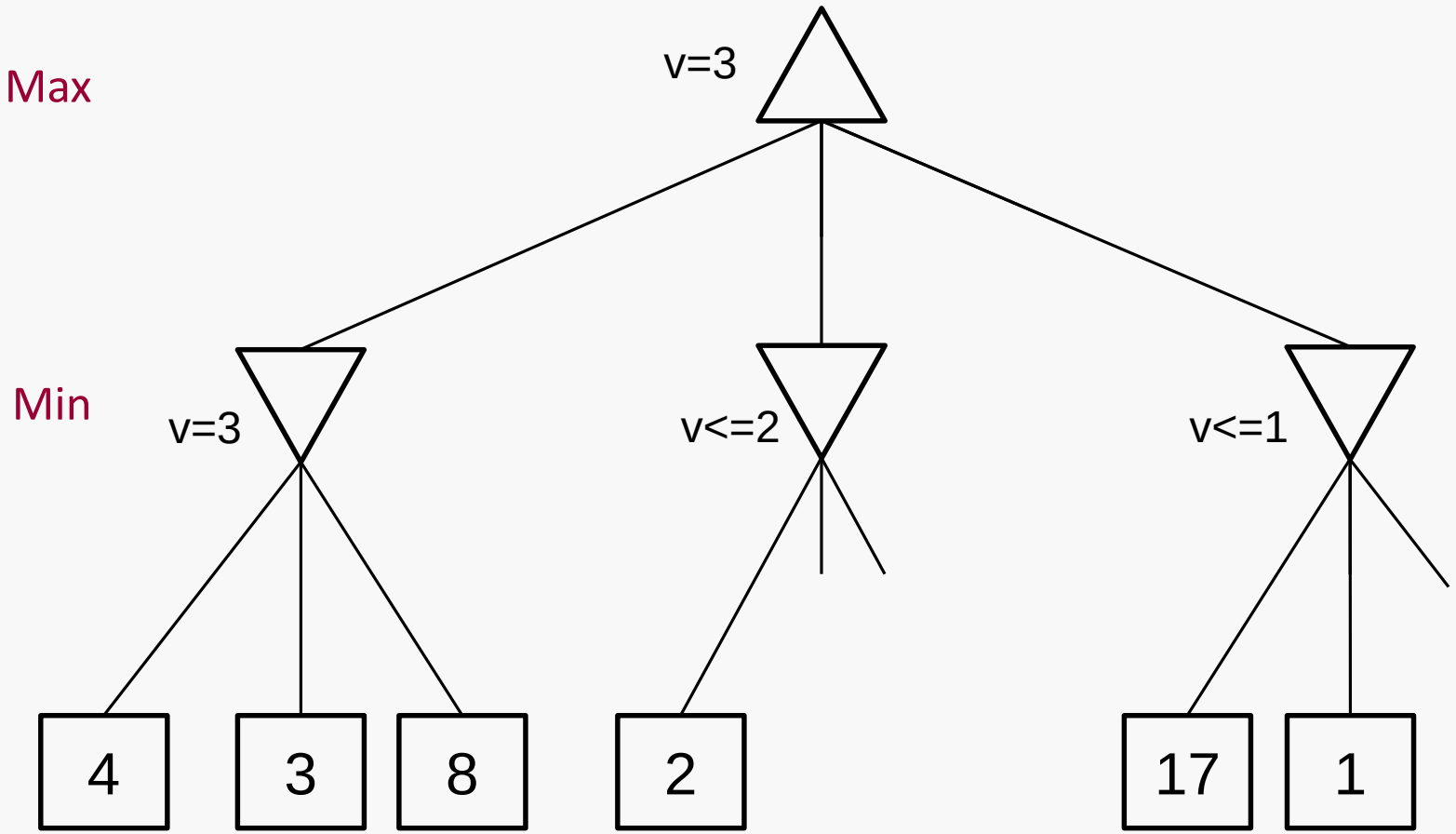
Game Tree Pruning



Game Tree Pruning

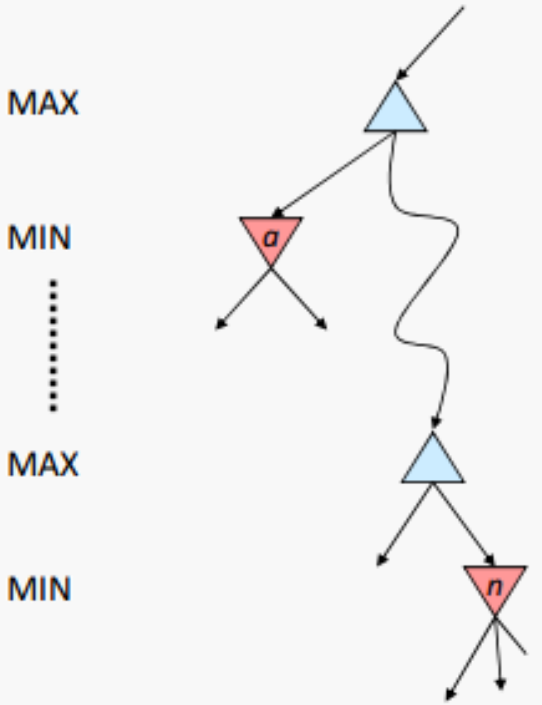


Game Tree Pruning



Alpha-Beta Pruning

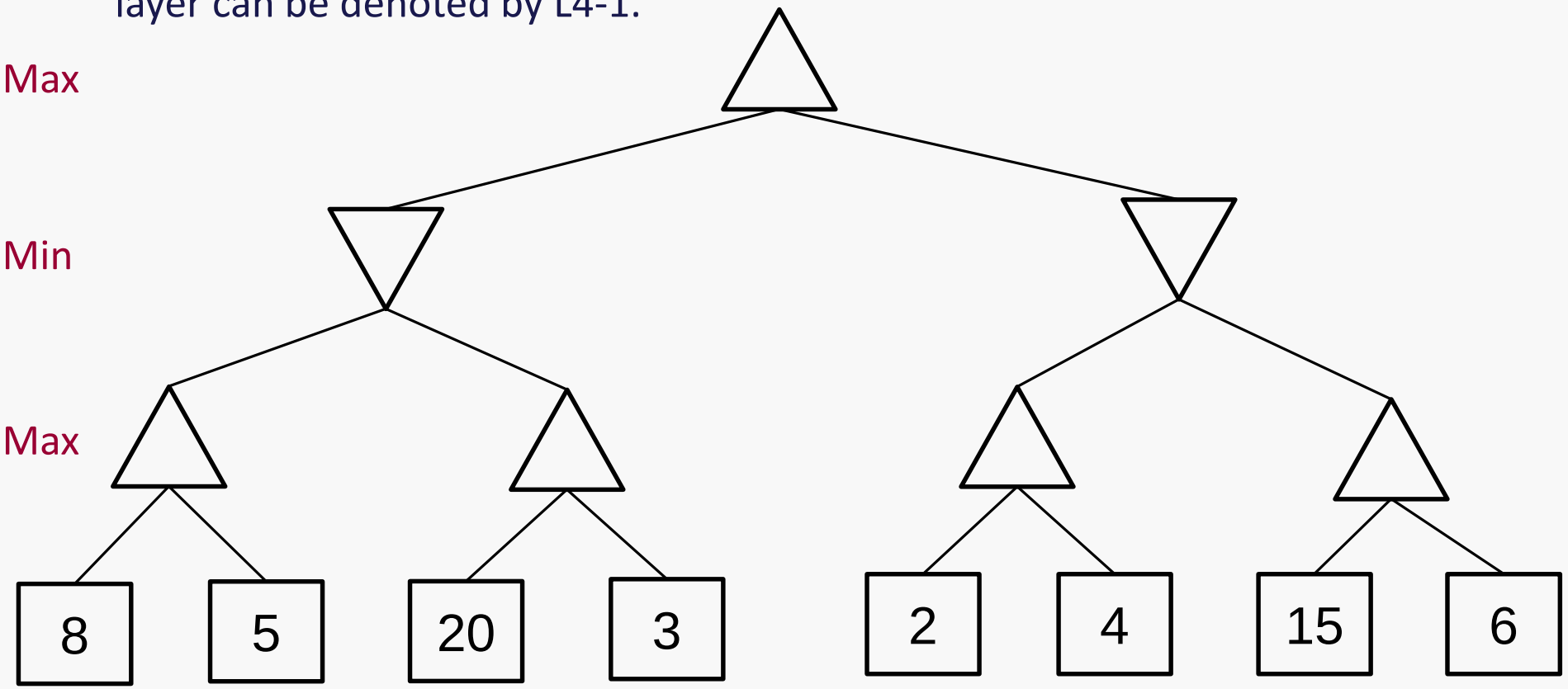
- General configuration (for agent Max)
  - Let a be the value that Max can currently get at least.
  - We are now computing the *min_value* at some node n
  - When we explore n's children, if we find that the value of n will never be better than a (for agent Max), then we can stop considering n's other children.



- Properties of alpha-beta pruning
  - The pruning has no effect on minimax value for the root
  - Good child ordering improves effectiveness of pruning
  - Complexity of perfect ordering: $O\left(b^{m/2}\right)$
  - Full search of many games (e.g. Chess) is still hopeless

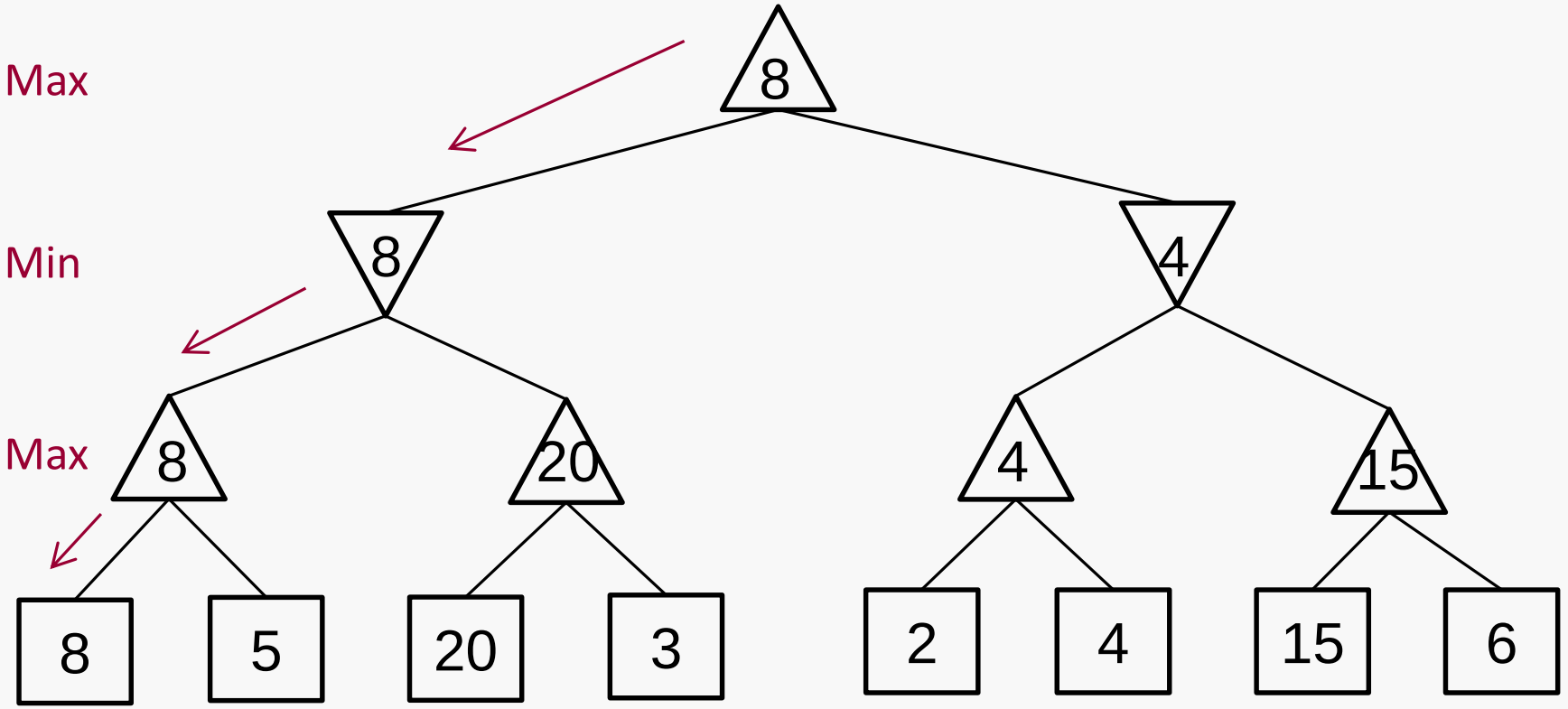
Alpha-Beta quiz

- Question 1: Give the minimax value at each node for the game tree below.
  Question 2: Find the nodes of the following tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 8) at the bottom layer can be denoted by L4-1.



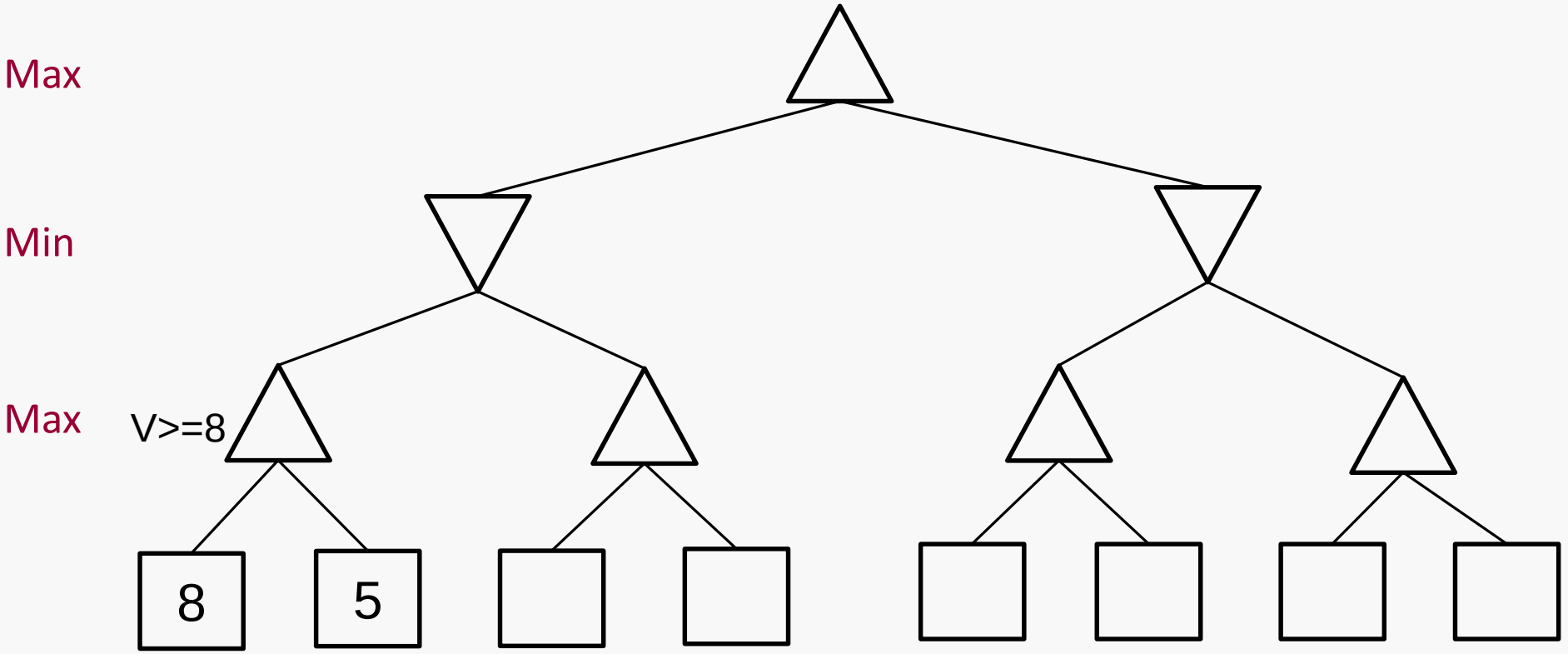
Question 1

- Give the minimax value at each node for the game tree below.



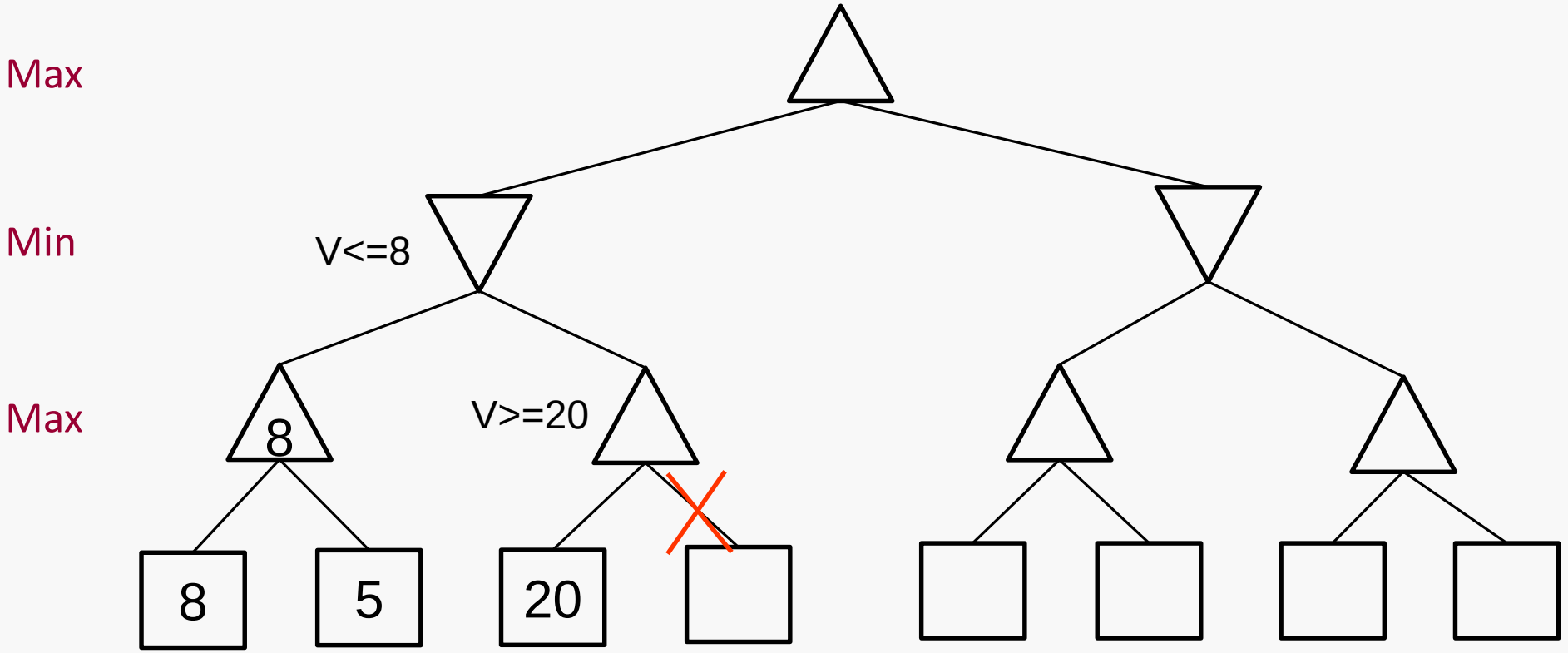
Question 2

- Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 8) at the bottom layer can be denoted by L4-1.



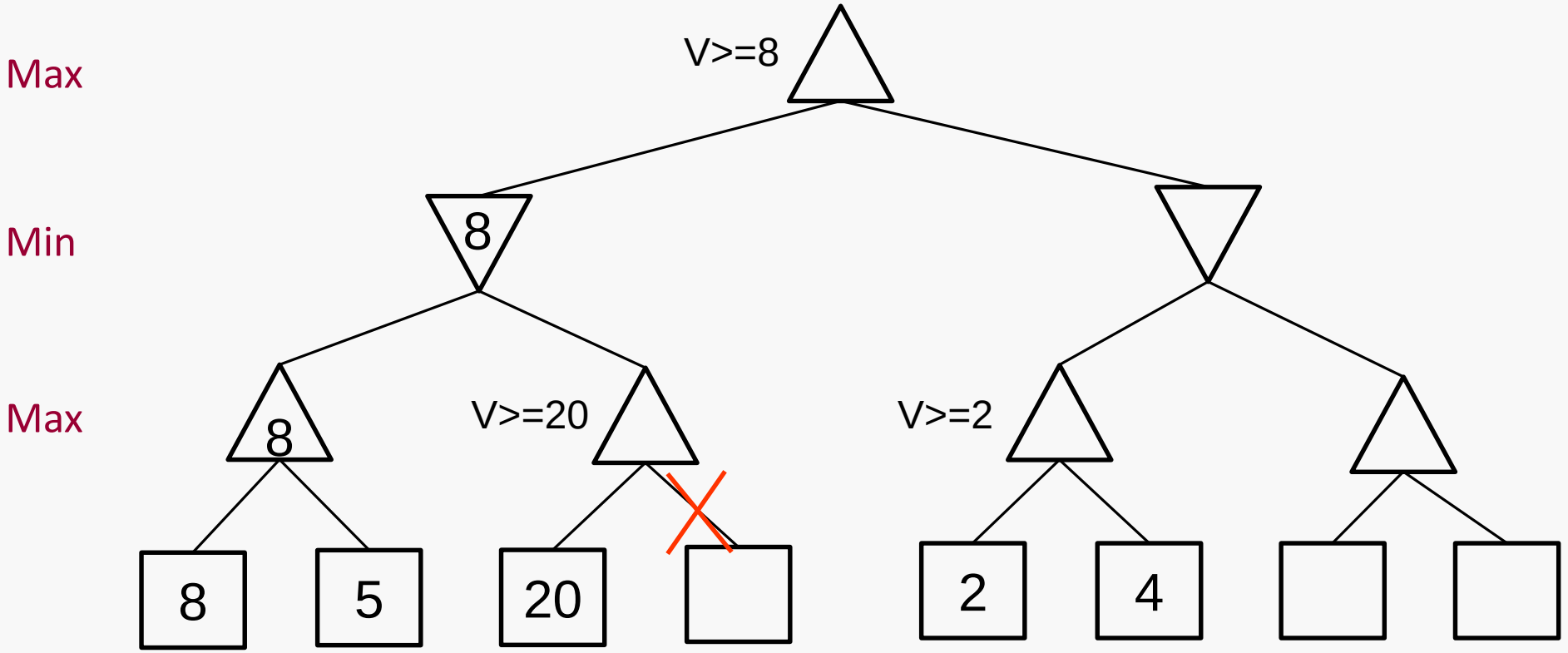
Question 2

- Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 8) at the bottom layer can be denoted by L4-1.



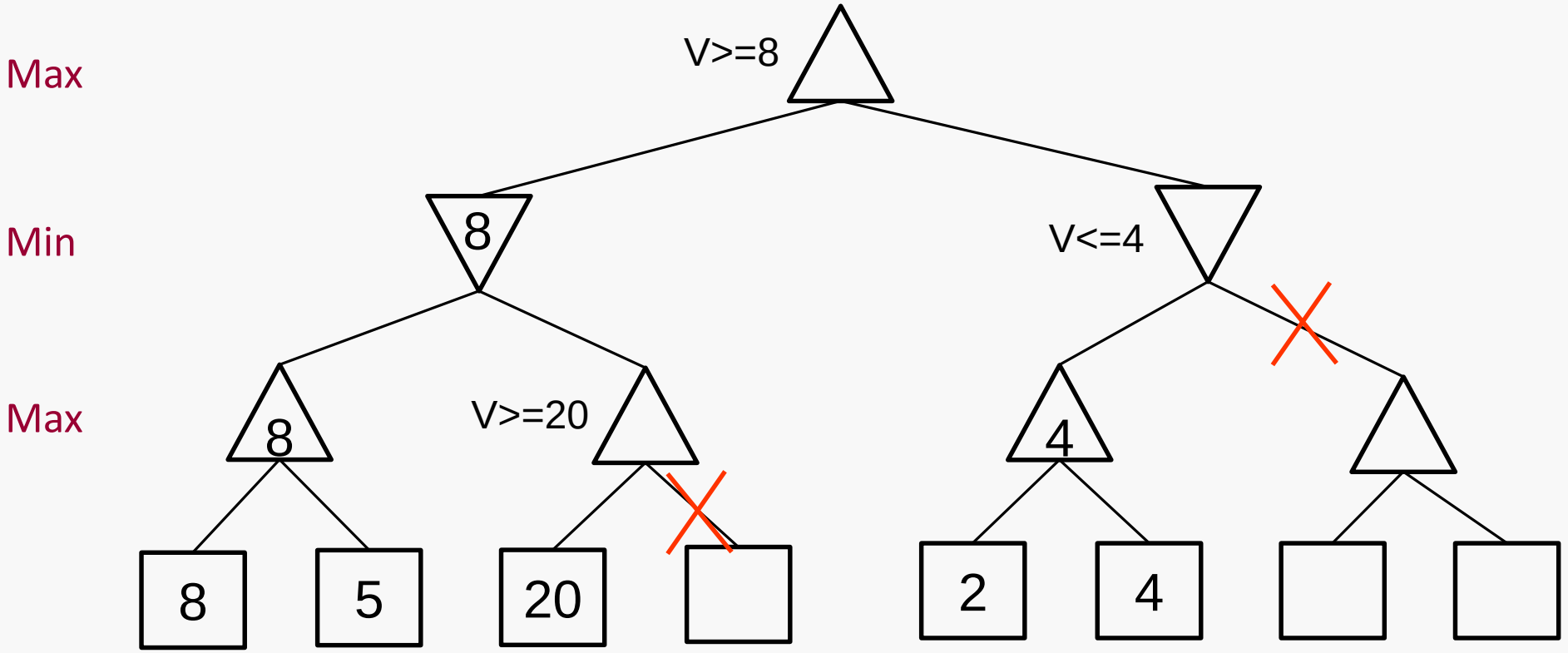
Question 2

- Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 8) at the bottom layer can be denoted by L4-1.



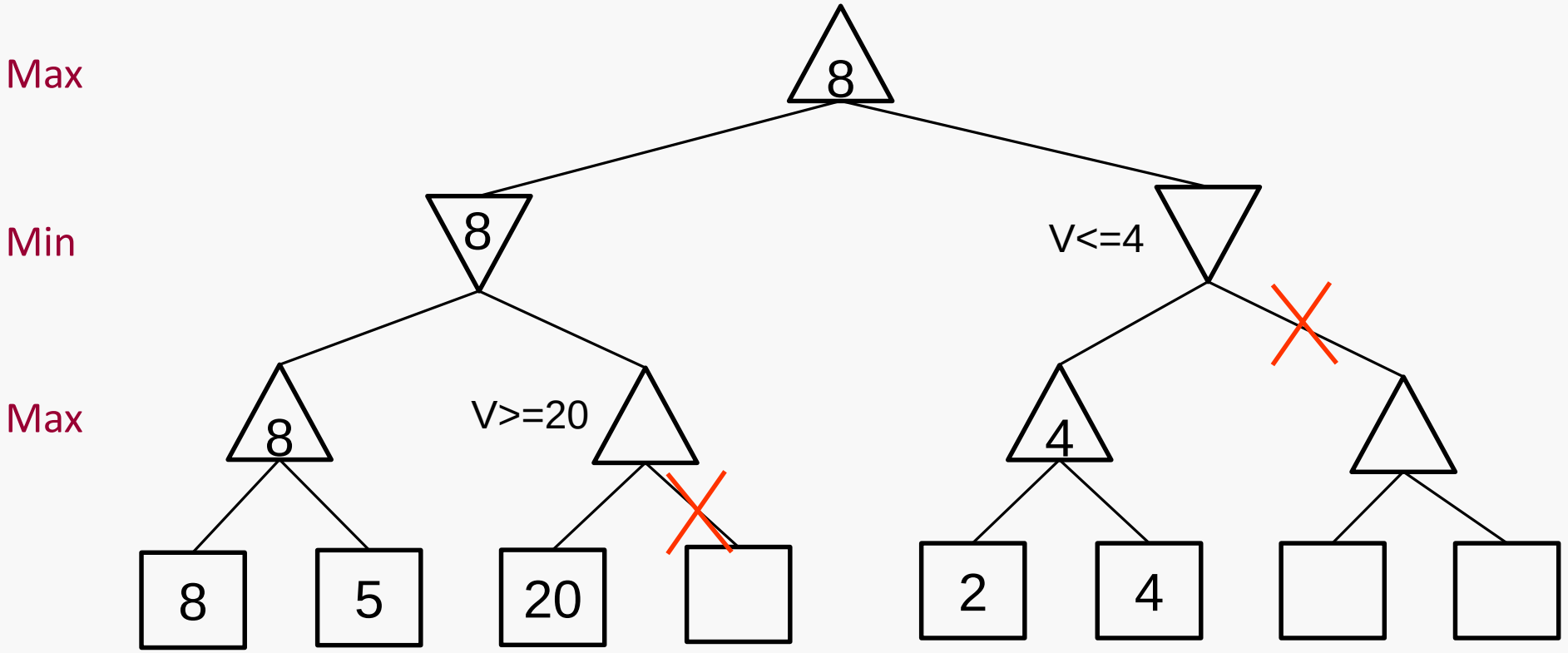
Question 2

- Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 8) at the bottom layer can be denoted by L4-1.



Question 2

- Find the nodes of the above tree pruned by alpha-beta pruning algorithm. Assuming child nodes are visited from left to right. There are four layers and you can use Lm-n to denote the nth node from left to right in the layer m, e.g., the first node (with value 8) at the bottom layer can be denoted by L4-1.



The pruned nodes: L4-4, L3-4, L4-7, L4-8