SAPIENZA
UNIVERSITÀ DI ROMA

Artificial Intelligence

2023/2024 Prof: Sara Bernardini

Lab 3:
A*, Local Search and Games

Francesco Argenziano
email: argenziano@diag.uniroma1.it

*The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello, Prof. Daniele Nardi and Dott. Fabio Previtali.

Example: Missionaries and Cannibals

3 missionaries and **3 cannibals** are on one of the sides of a river with a small **boat that can hold 2 passengers at most**. The **number of missionaries must always be more or equal wrt the number of cannibals**, otherwise they are eaten by the cannibals. How can the missionaries cross the river without being eaten?

1. Characterize the **state space**
2. Specify the **operators**
3. Find a minimal sequence of moves to solve the problem
4. Find a good **heuristics** to be used by A*.
5. Draw the **search tree generated by A***. For each node indicate: the number (state), the cost (f, g, and h) and an integer indicating the expansion order.

Example: Missionaries and Cannibals

1) Characterize the **state space**

<Ma, Ca, Ba, Mb, Cb, Bb>

- **Ma:** #of missionaries on side A;
- **Ca:** #of cannibals on side A;
- **Ba:** presence of the boat on side A;
- **Mb:** #of missionaries on side B;
- **Cb:** #of cannibals on side B;
- **Bb:** presence of the boat on side B;

Ma, Mb ∈ N+, with Ma+Mb = 3
Ca, Cb ∈ N+, with Ca+Cb = 3
Ba ∈ {0,1}, Bb ∈ {0,1} and Bb = 1-Ba

Initial State:

<3, 3, 1, 0, 0, 0>

Goal State:

<0,0, _, 3, 3, _>

Example: Missionaries and Cannibals

Operators: <Ma, Ca, Ba, Mb, Cb, Bb> → <Ma*, Ca*, Ba*, Mb*, Cb*, Bb*>

carry(*m*, *c*):

meaning: carry *m* missionaries and *c* cannibals from a to b

preconditions:

- Boat side: Ba = 1, Bb = 0
- Boat capacity: Ma*+Ca* >= Ma+Ca -2
- Negative quantities: Ma*, Ca* >= 0
- Avoid eatings:

  - Ma*>=Ca*, Mb*>=Cb*
  - or Ma* = 0 and any Ca*
  - or Mb* = 0 and any Cb*

new state:

<Ma-m, Ca-c, Ba-1, Mb+m, Cb+c, Bb+1>

carryback(*m*, *c*):

meaning: carryback *m* missionaries and *c* cannibals from b to a

preconditions:

- Boat side: Ba = 0, Bb = 1
- Boat capacity: Mb*+Cb* >= Mb+Cb -2
- Negative quantities: Mb*, Cb* >= 0
- Avoid eatings:

  - Ma*>=Ca*, Mb*>=Cb*
  - or Ma* = 0 and any Ca*
  - or Mb* = 0 and any Cb*

new state:

<Ma+m, Ca+c, Ba+1, Mb-m, Cb-c, Bb-1>

**preconditions* represent the possible configurations of the state-space that allow to execute an action

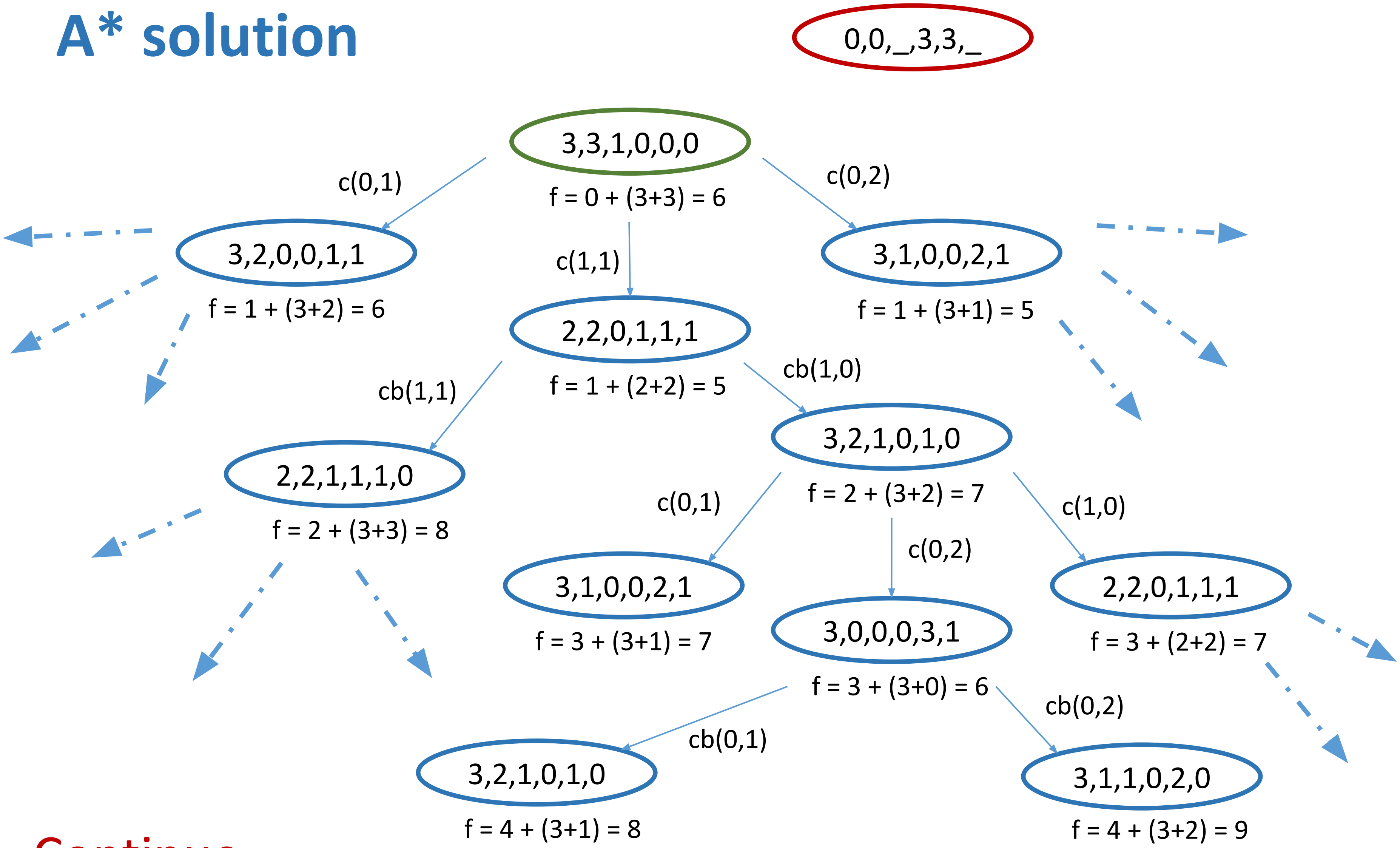
Example: Missionaries and Cannibals

Heuristic

$$h(n) = |Mb-3| + |Cb-3|$$

Example: Missionaries and Cannibals

A* solution



0,0,_,3,3,_

3,3,1,0,0,0

f = 0 + (3+3) = 6

c(0,1)

3,2,0,0,1,1

f = 1 + (3+2) = 6

c(1,1)

2,2,0,1,1,1

f = 1 + (2+2) = 5

c(0,2)

3,1,0,0,2,1

f = 1 + (3+1) = 5

cb(1,1)

2,2,1,1,1,0

f = 2 + (3+3) = 8

cb(1,0)

3,2,1,0,1,0

f = 2 + (3+2) = 7

c(0,1)

3,1,0,0,2,1

f = 3 + (3+1) = 7

c(0,2)

3,0,0,0,3,1

f = 3 + (3+0) = 6

c(1,0)

2,2,0,1,1,1

f = 3 + (2+2) = 7

cb(0,1)

3,2,1,0,1,0

f = 4 + (3+1) = 8

cb(0,2)

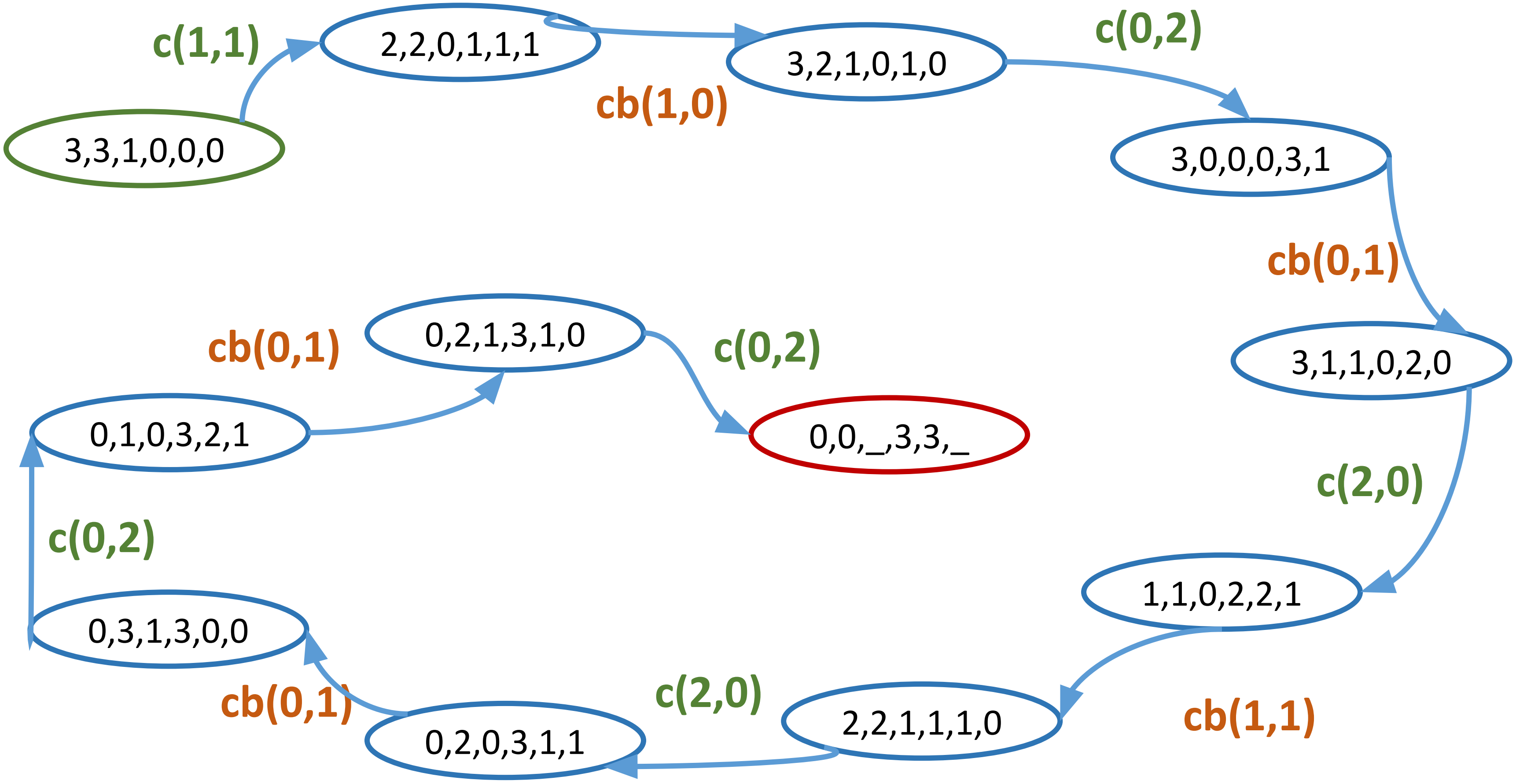
3,1,1,0,2,0

f = 4 + (3+2) = 9

Continue…

Example: Missionaries and Cannibals

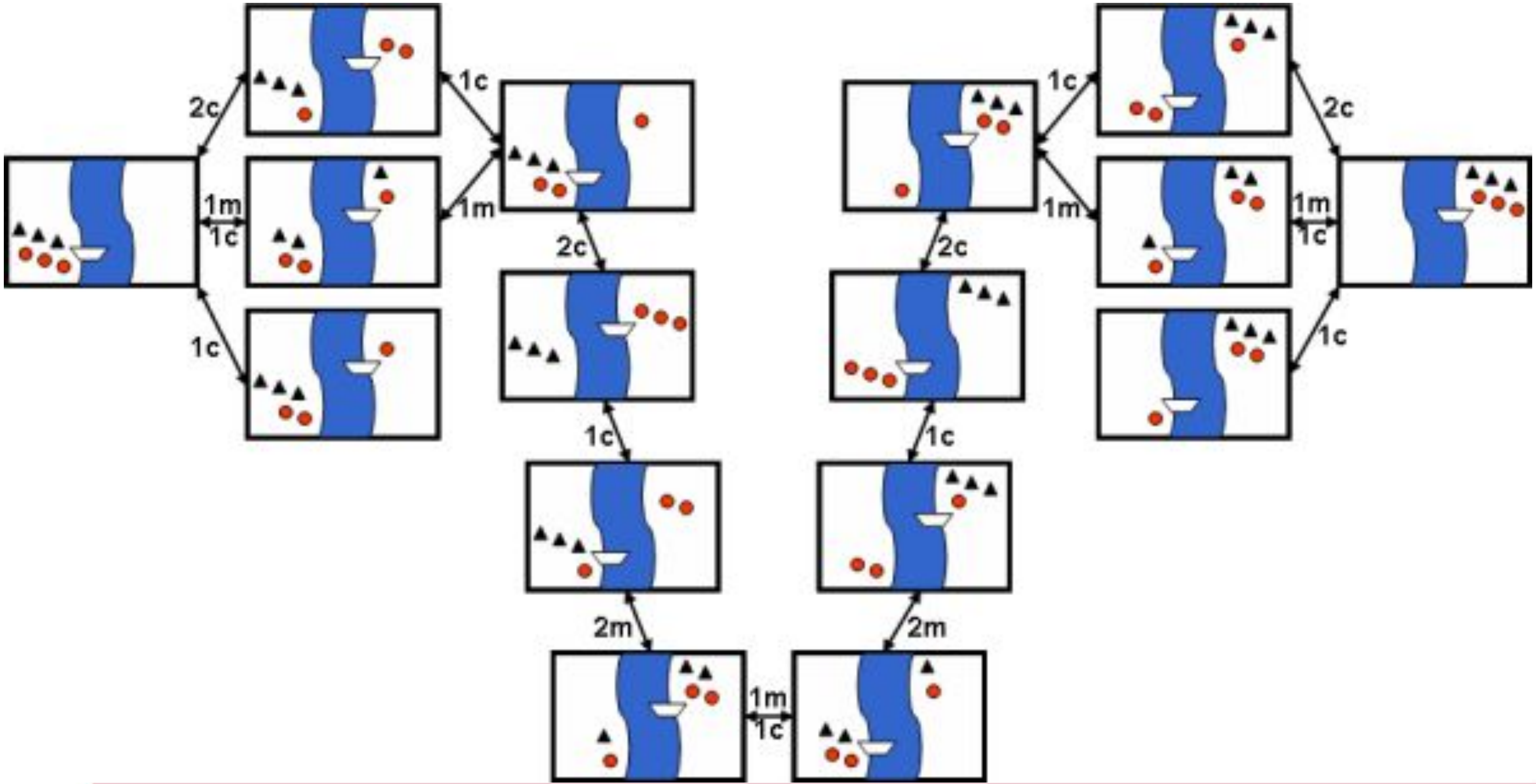
Possible solution



Example: Missionaries and Cannibals

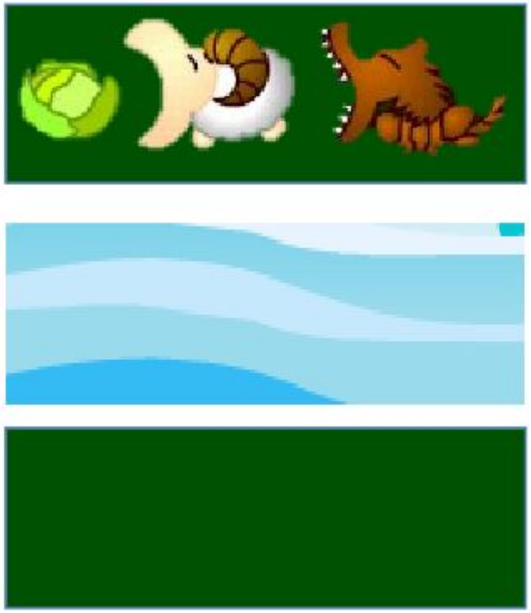
A* solution

0,0,_,3,3,_



Example: Crossing the river

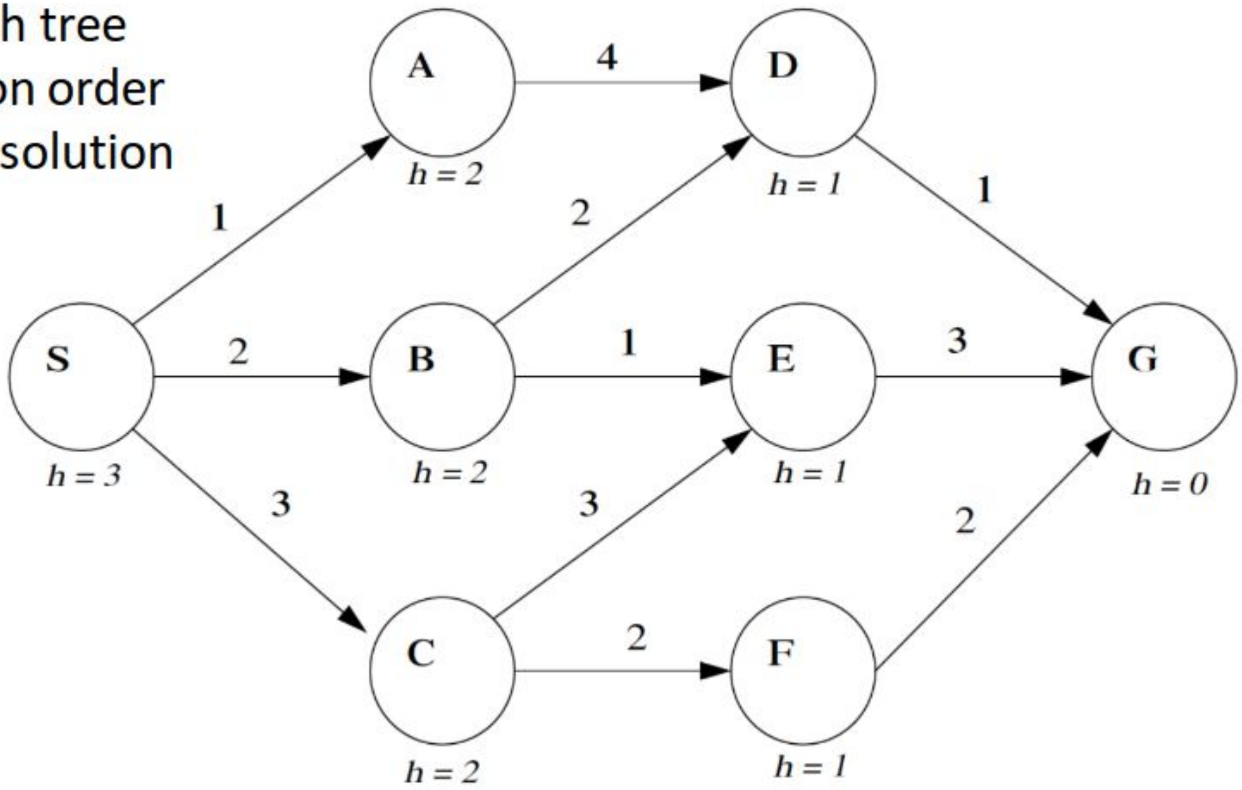
A man has a **wolf**, a **sheep** and a **cabbage**. He is on a river bench with a **boat**, whose maximum **load** for a single trip is the man **plus one of his 3 goods**. The man wants to cross the river with his goods, but he must avoid that - when he is far away - **the wolf eats the sheep** and that, **the sheep eats the cabbage**. How can the man reach is goal?



1. Characterize the **state space**
2. Specify the **operators**
3. Find a minimal sequence of moves to **solve the problem**
4. Find a good **heuristics** to be used by A∗
5. Draw the **search tree generated by A∗**. For each node indicate: the number (state), *f*, *g*, and *h* and an integer indicating the expansion order

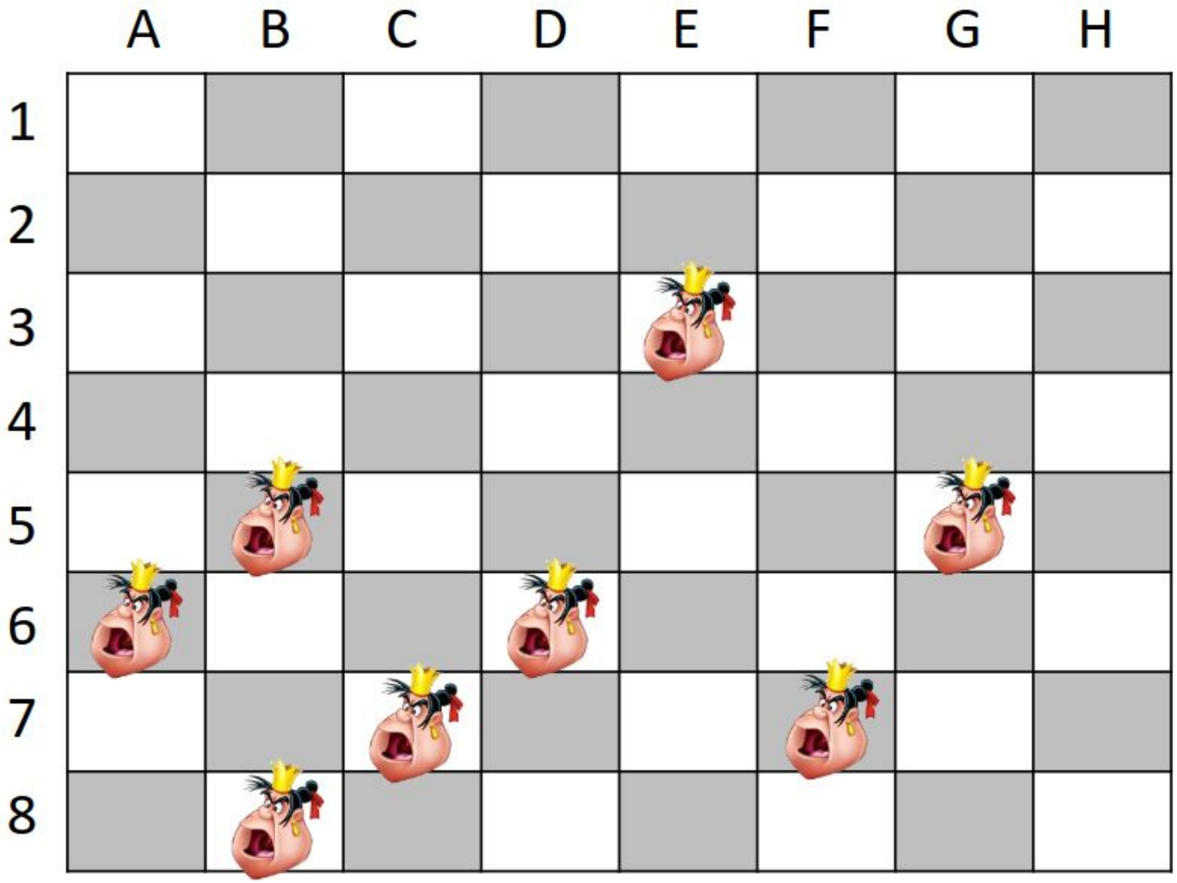
Consider the search problem represented in the following **graph**. It has start state **S** and goal state **G**. Transition **costs** are shown as numbers on the arrows. **Heuristic** values are shown below each state.

1. Draw the A* search tree
2. Mark the expansion order
3. Show the optimal solution path



Example: 8-Queens

Use a **hill-climbing** with the evaluation function “number of queens which are threatened by another queen” in the 8-Queens problem.



Example: 8-Queens

1. What is the current score for the evaluation function?
2. Write down 3 of the possible moves from this state to the goal one (queens can move anywhere)
3. Give an example of an illegal move (in the hill climbing search)
4. What do you do if there are no legal moves?

Example: 8-Queens

1. What is the current score for the evaluation function?

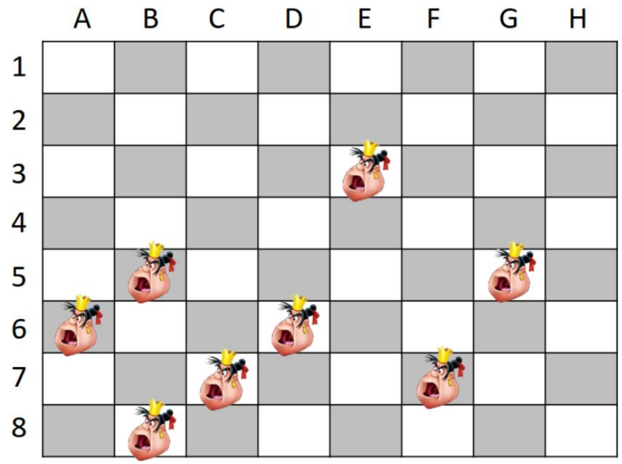
In this state, every queens is challenged by at least one other queen, and the total evaluation function is 17 (we consider attacking queens also the pairs of the type of B8-D6).

Example: 8-Queens

2. Write down 3 of the possible moves from this state to the goal one (queens can move anywhere)

We want to move one queen to any other place on the board where it is not threatened by any other queen, as this will reduce the number of threatened queens by at least one. If we can do so in such a way that it frees up another queen, then this will result as a bonus. Thus:

- G5 –> G4 : score after move is 17
- E3 –> H3 : score after move is 17
- B8 –> H8 : score after move is 11

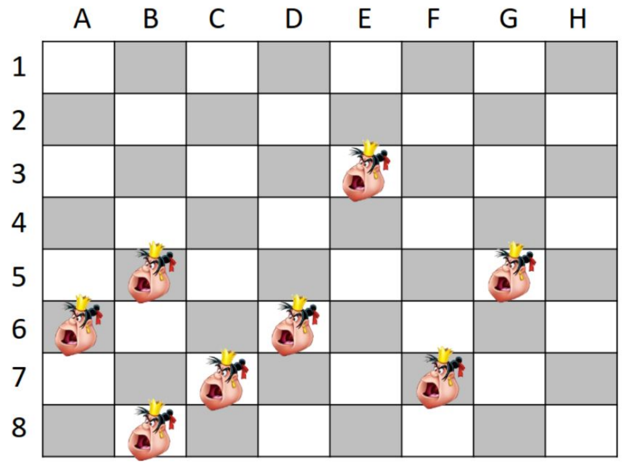


Example: 8-Queens

3. Give an example of an illegal move (in the hill climbing search)

Hill-climbing searches require the evaluation function to strictly decrease after every move.

Therefore an illegal move would be , for example:
B5 –> B7
since it would increase
the number of threatened queens



Example: 8-Queens

4. What do you do if there are no legal moves?

“Give up” when it happens. Start again by randomly putting your 8 queens on the board. This is called random-restart and, no backtracking is needed. Hence, it is very good on memory efficiency, as only the current board state has to be kept in memory.

Example: Tic-Tac-Toe

Tic-Tac-Toe is played on square grid of size 3 × 3 At each turn, players select an empty cell and place there its own symbol (i.e., O or X) A player wins when he places three of its own symbols in a line (vertical, horizontal or diagonal)

If the grid is filled without any player being able to place three symbols in a line, the game ends in a draw.

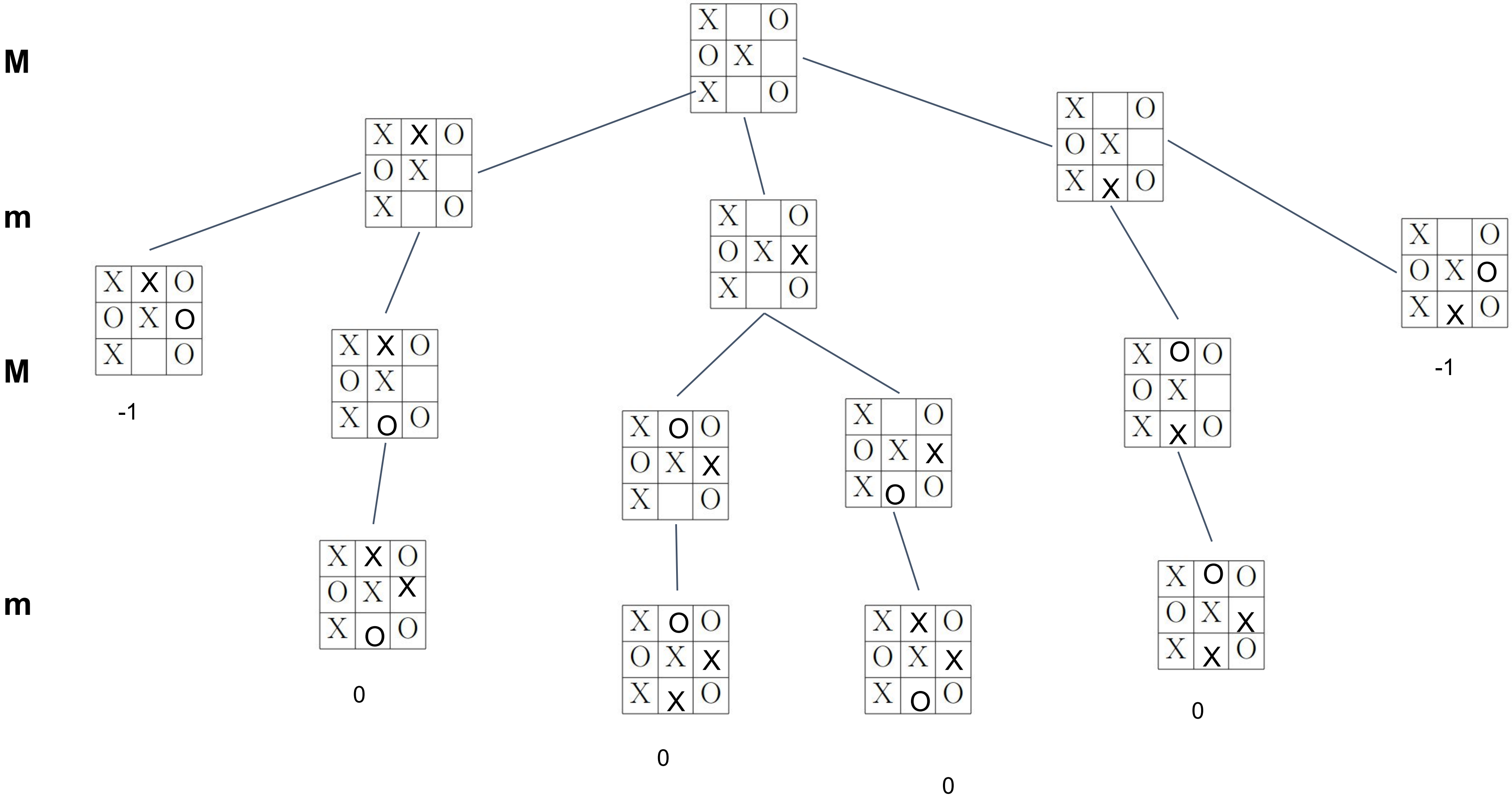
X		O
O	X	
X		O

1. Suppose that the game is in the state described by the previous picture and that X must move next. Draw the game tree of the rest of the match.

2. Show the solution of the game using mini-max, knowing that payoffs, are for each player, -1, 0, +1 depending on if he loses, draws or wins, respectively.

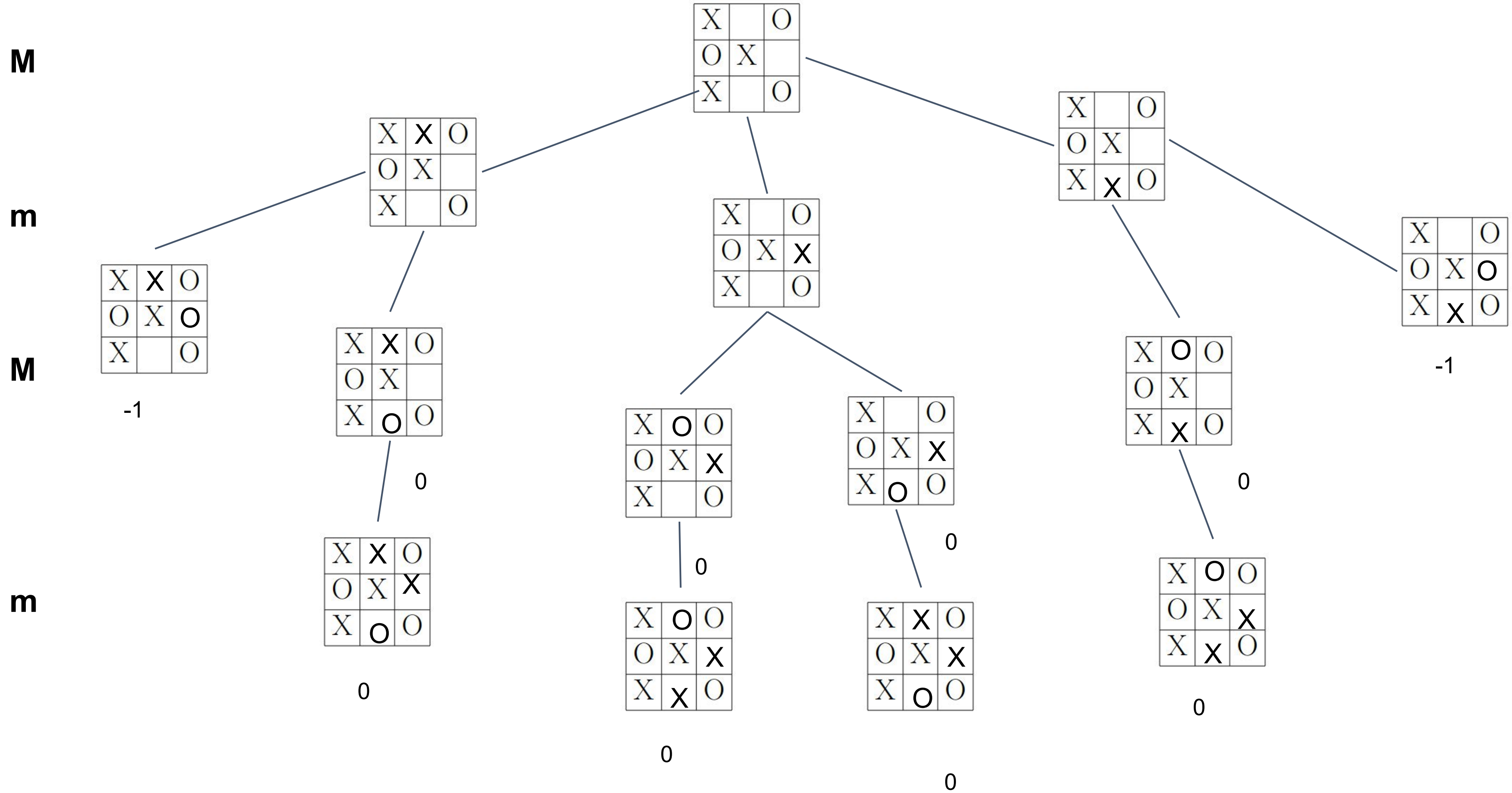
Example: Tic-Tac-Toe

Max -> X
Min -> O



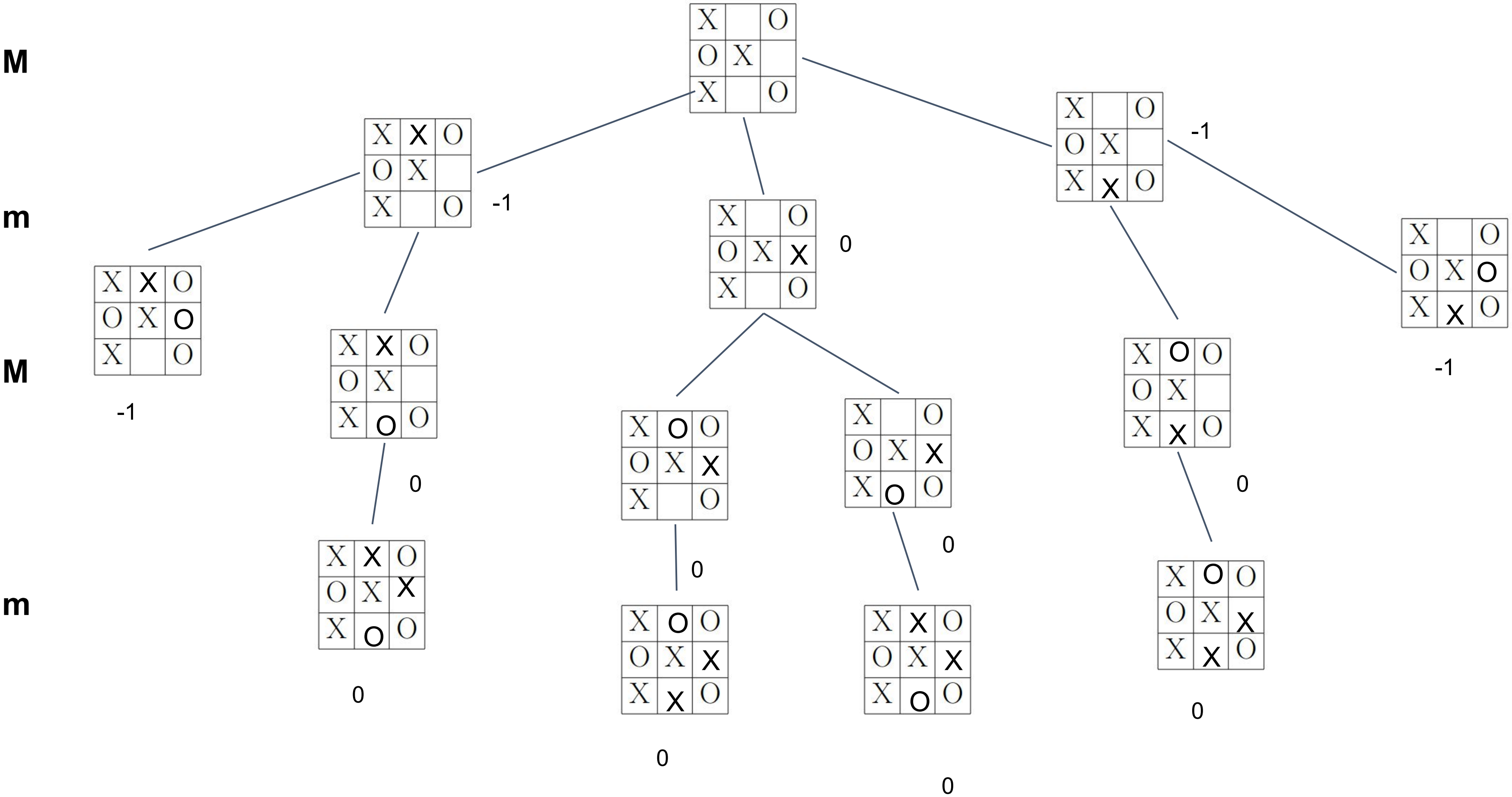
Example: Tic-Tac-Toe

Max -> X
Min -> O



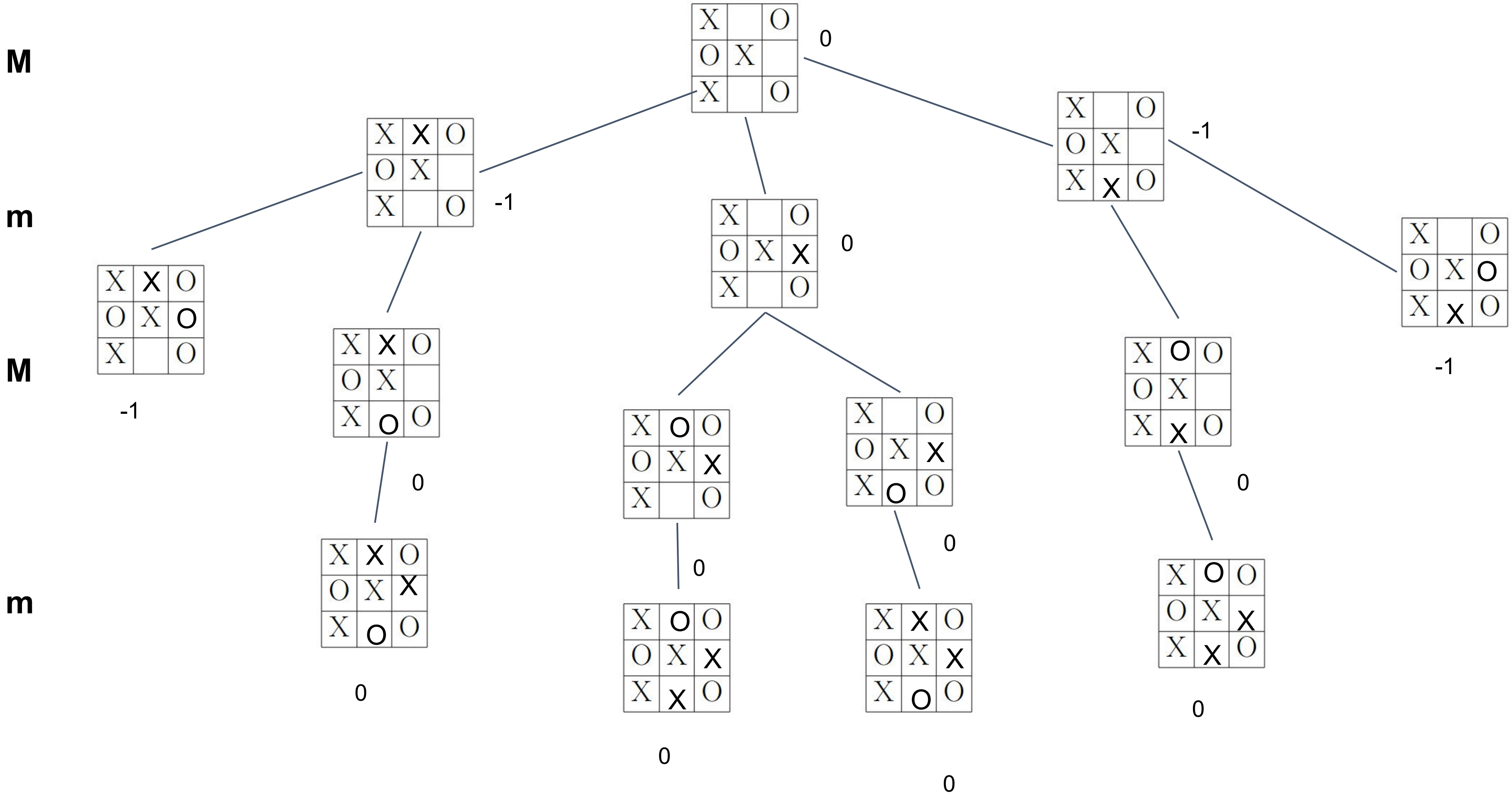
Example: Tic-Tac-Toe

Max -> X
Min -> O



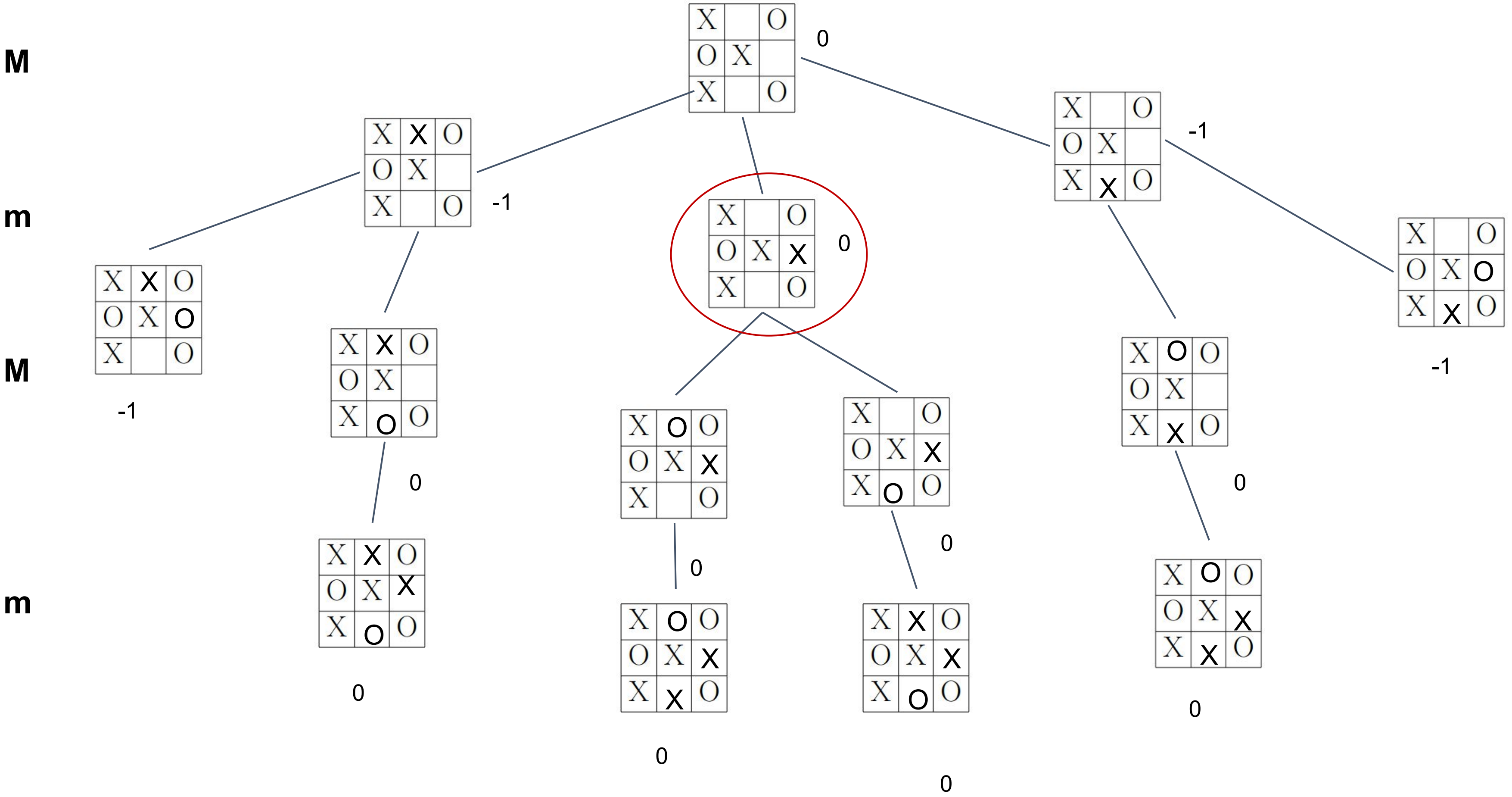
Example: Tic-Tac-Toe

Max -> X
Min -> O

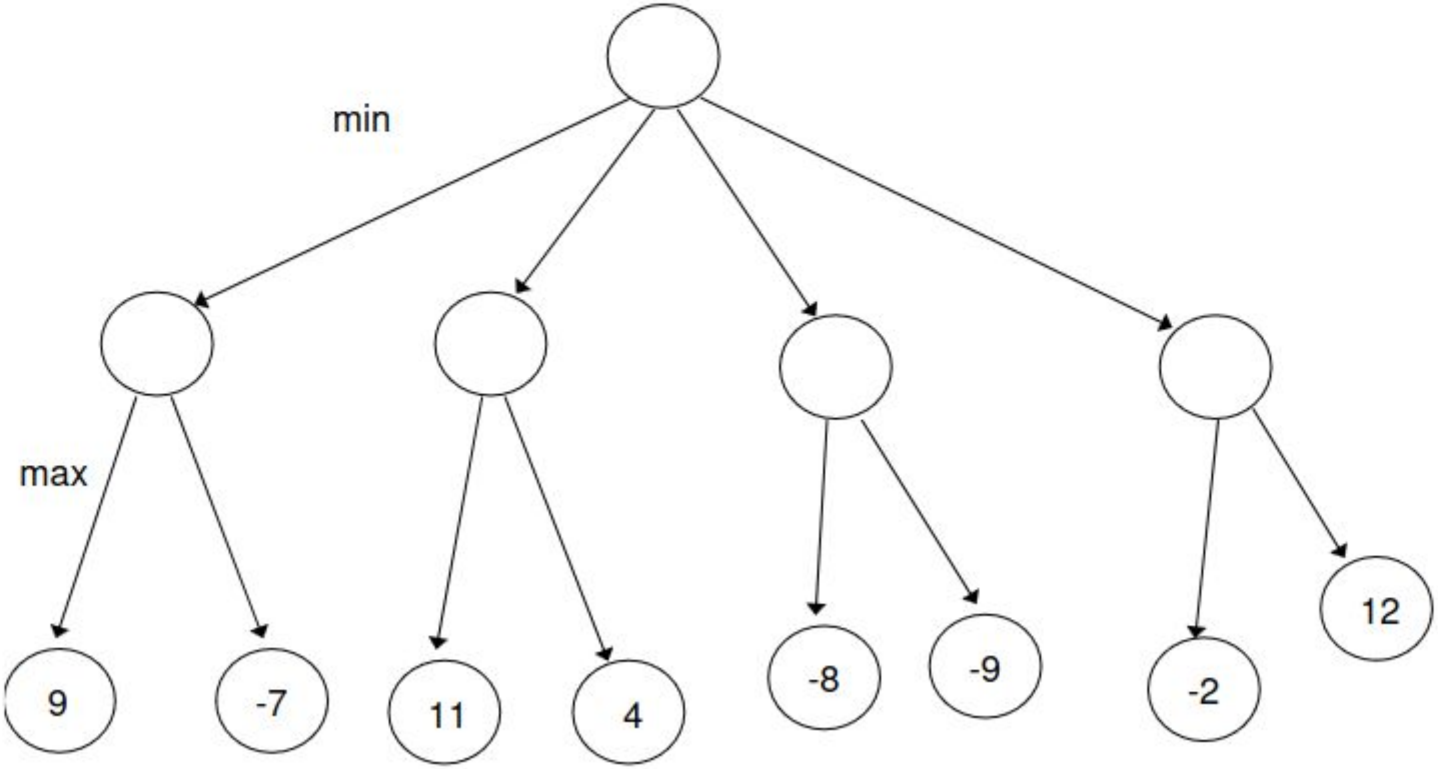


Example: Tic-Tac-Toe

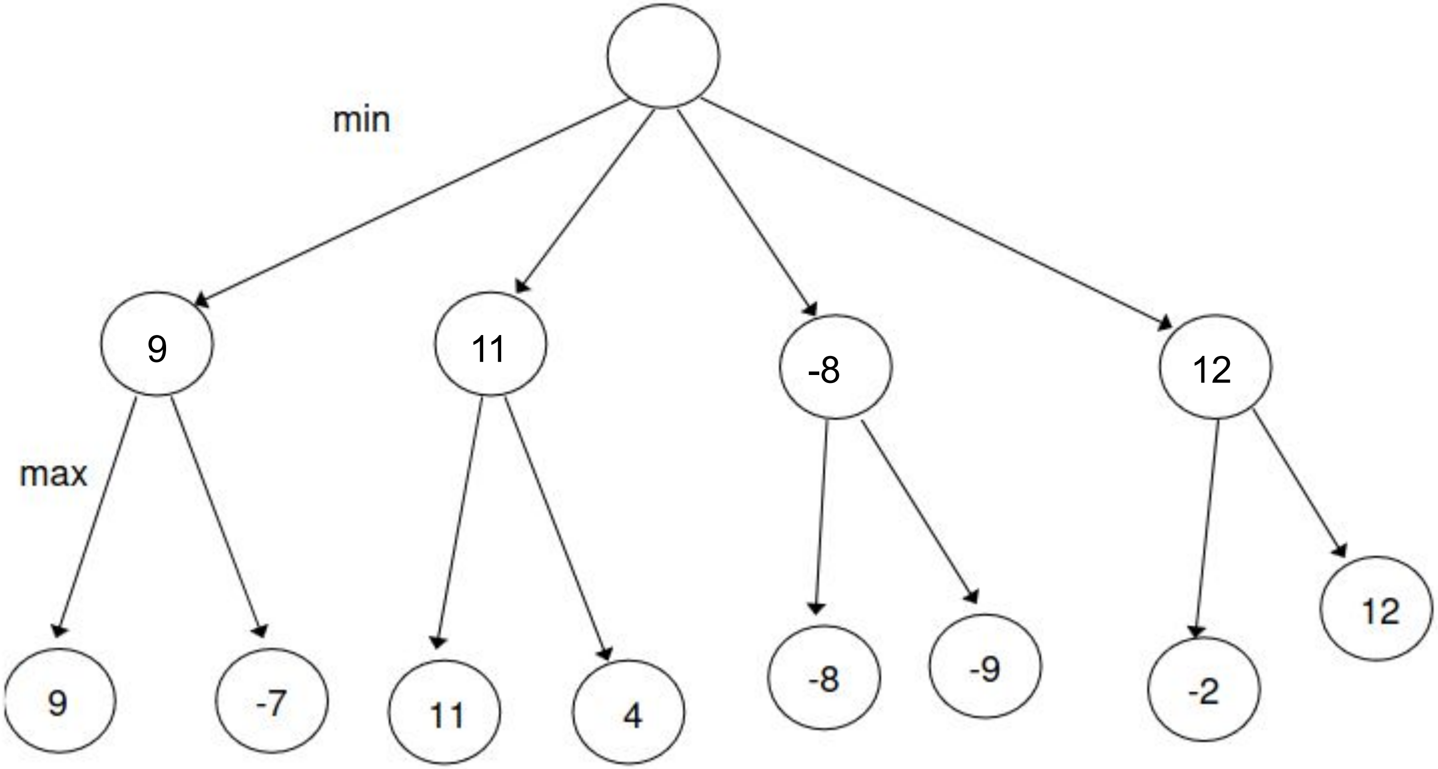
Max -> X
Min -> O



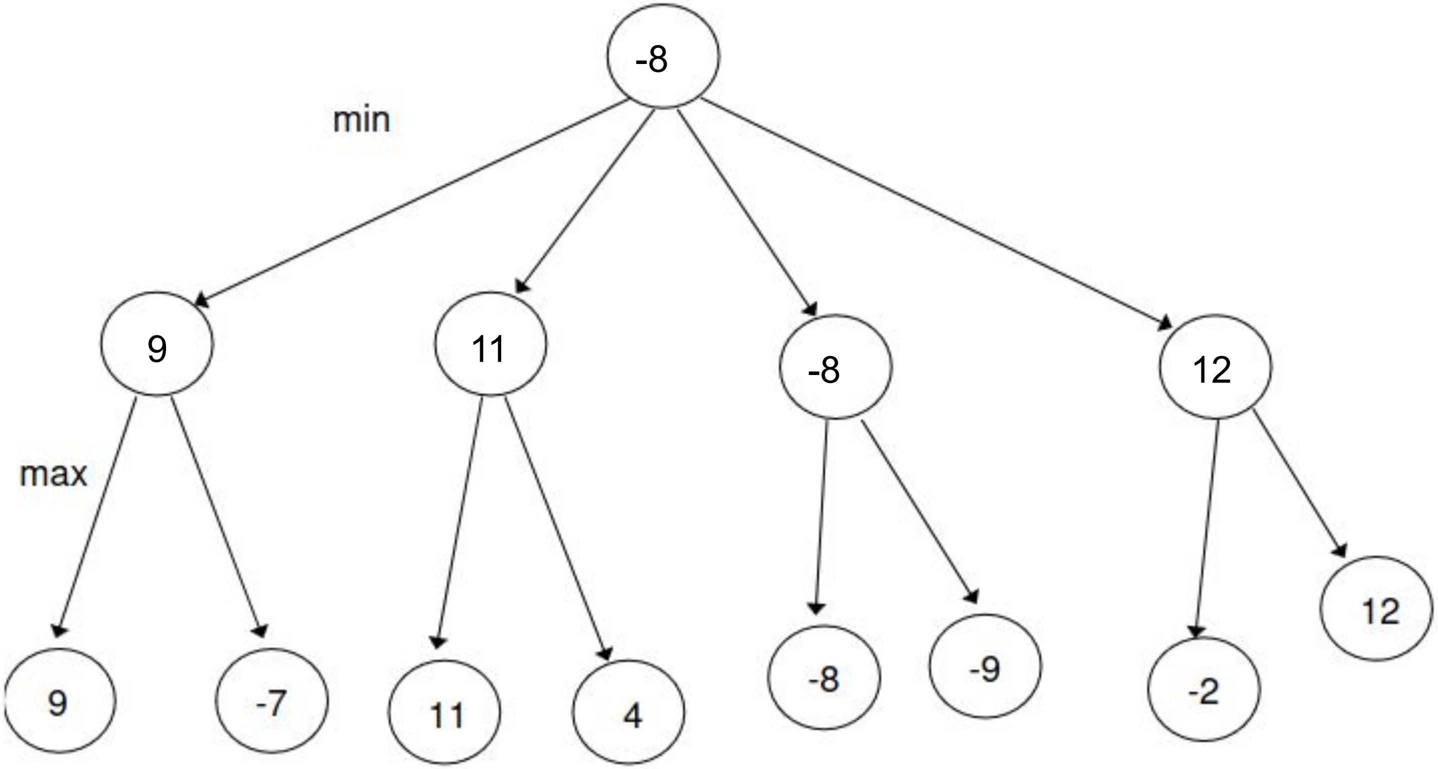
Example: Minimax 1



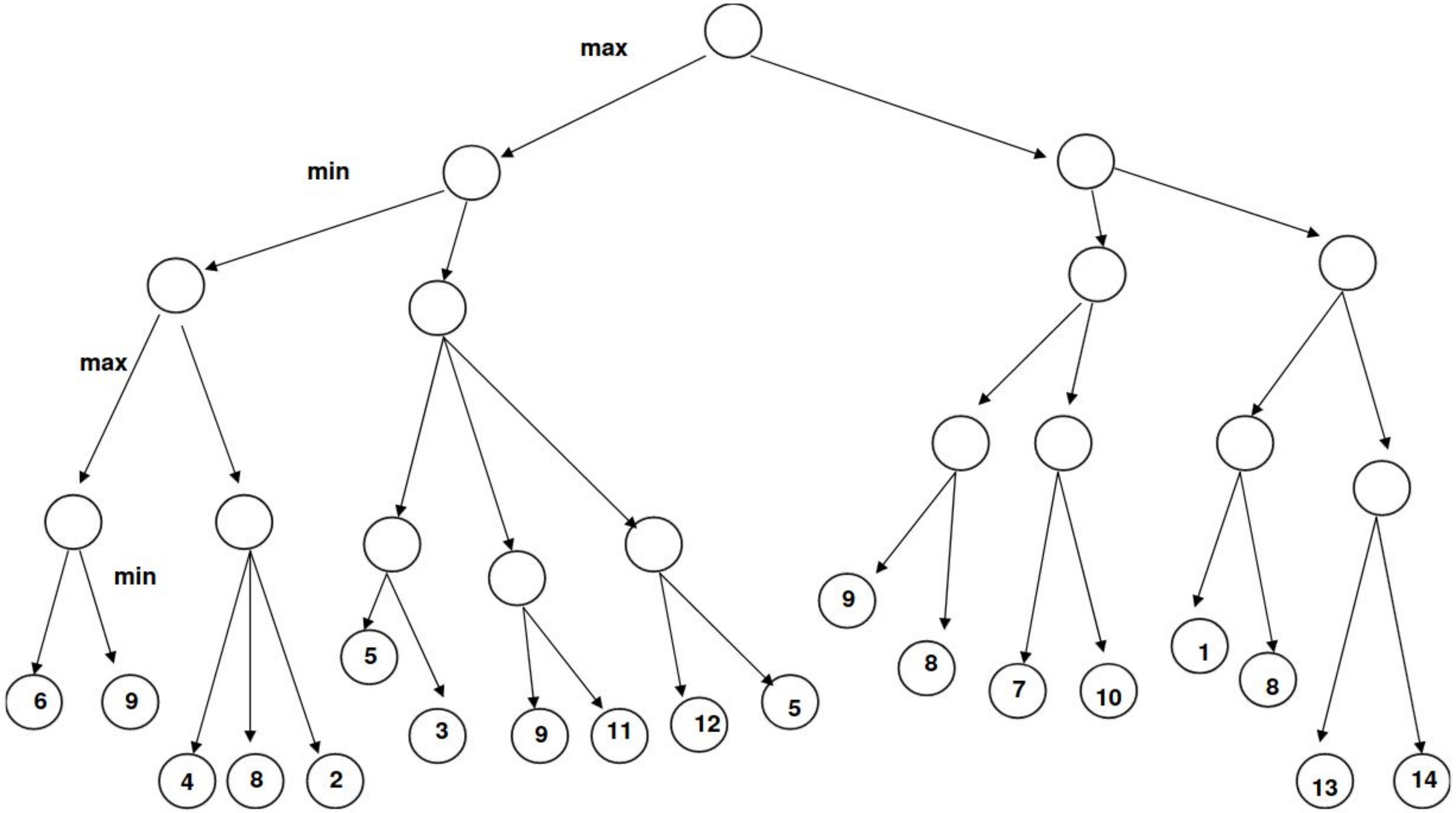
Example: Minimax 1



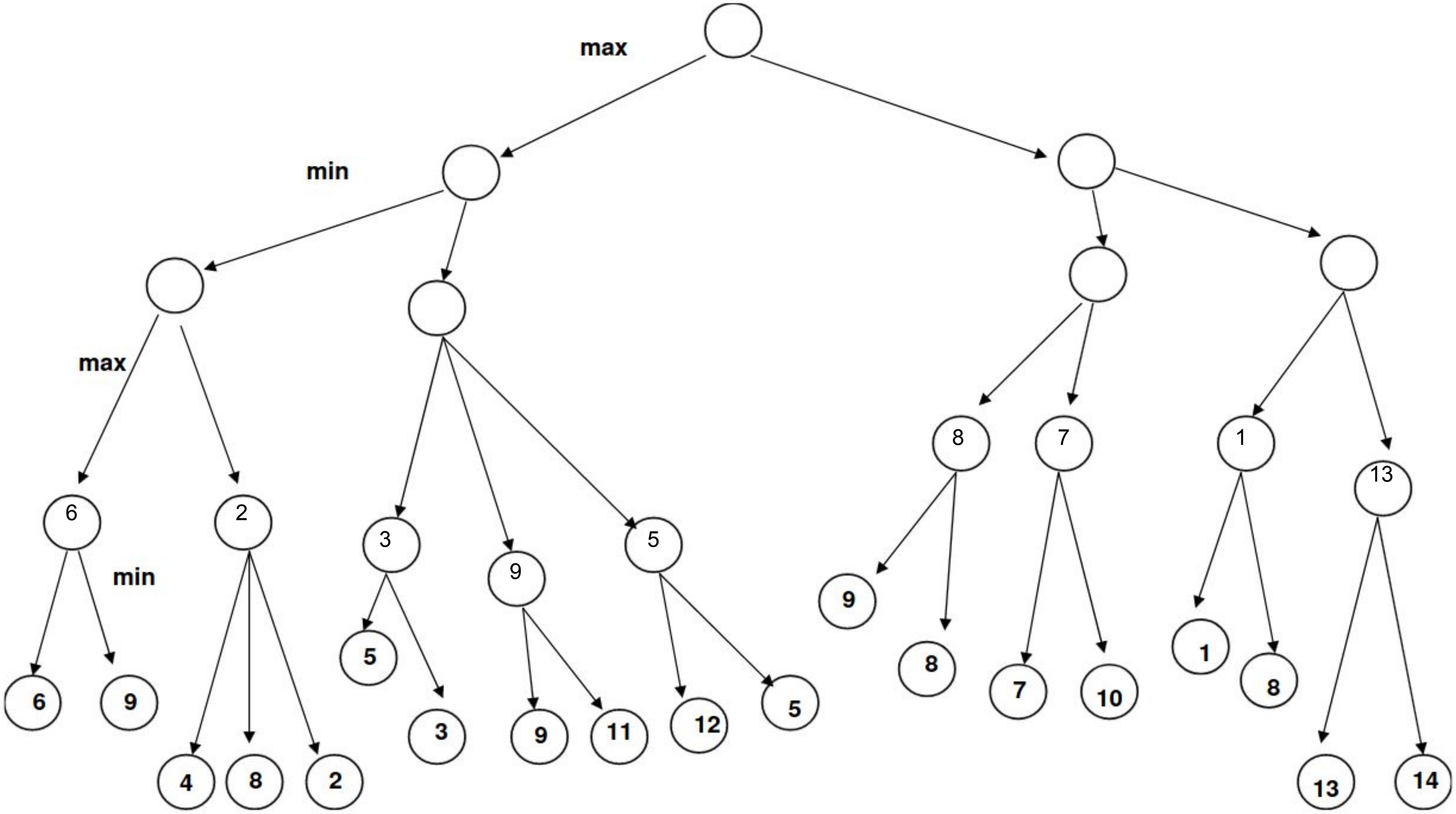
Example: Minimax 1



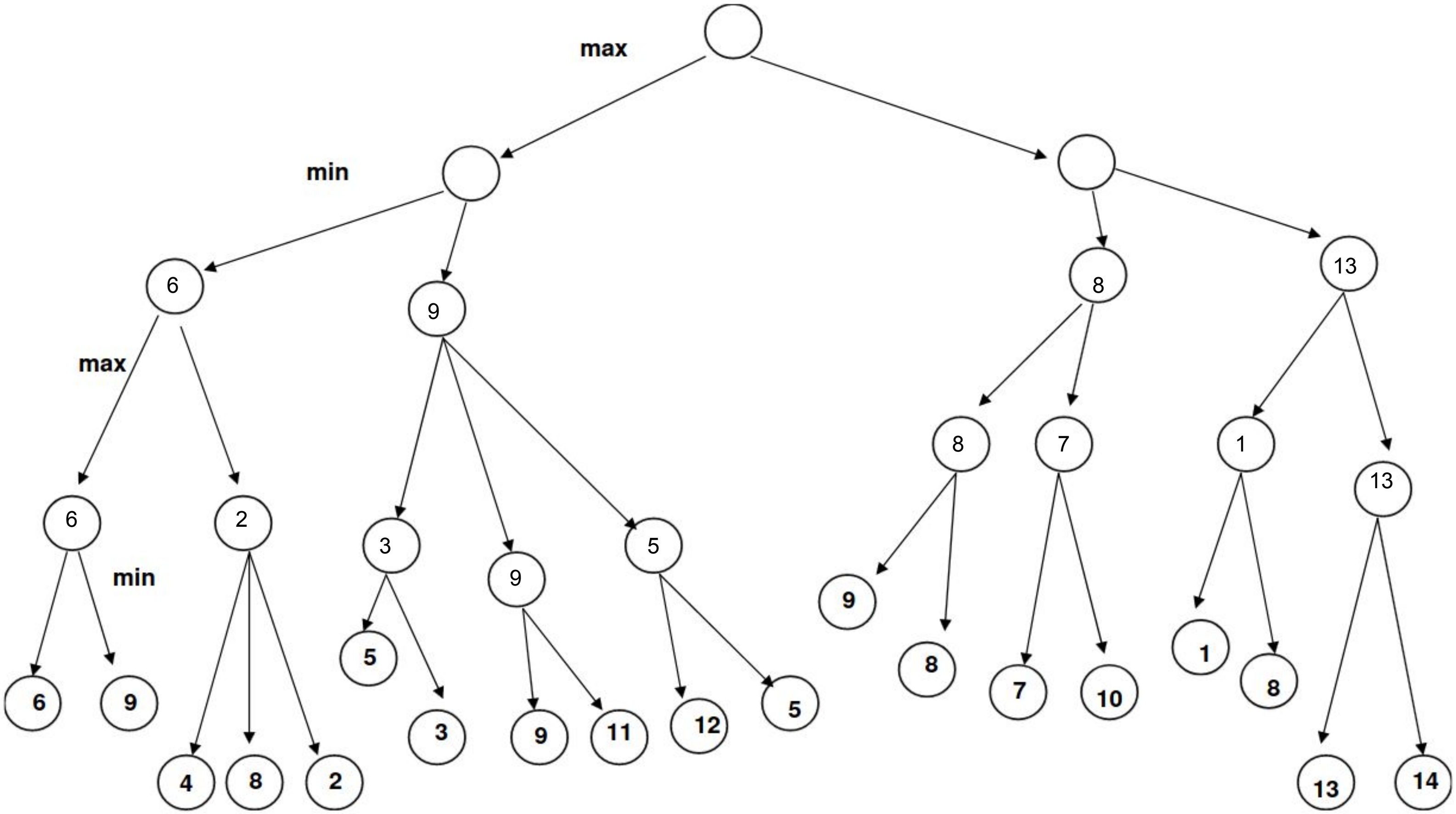
Example: Minimax 2



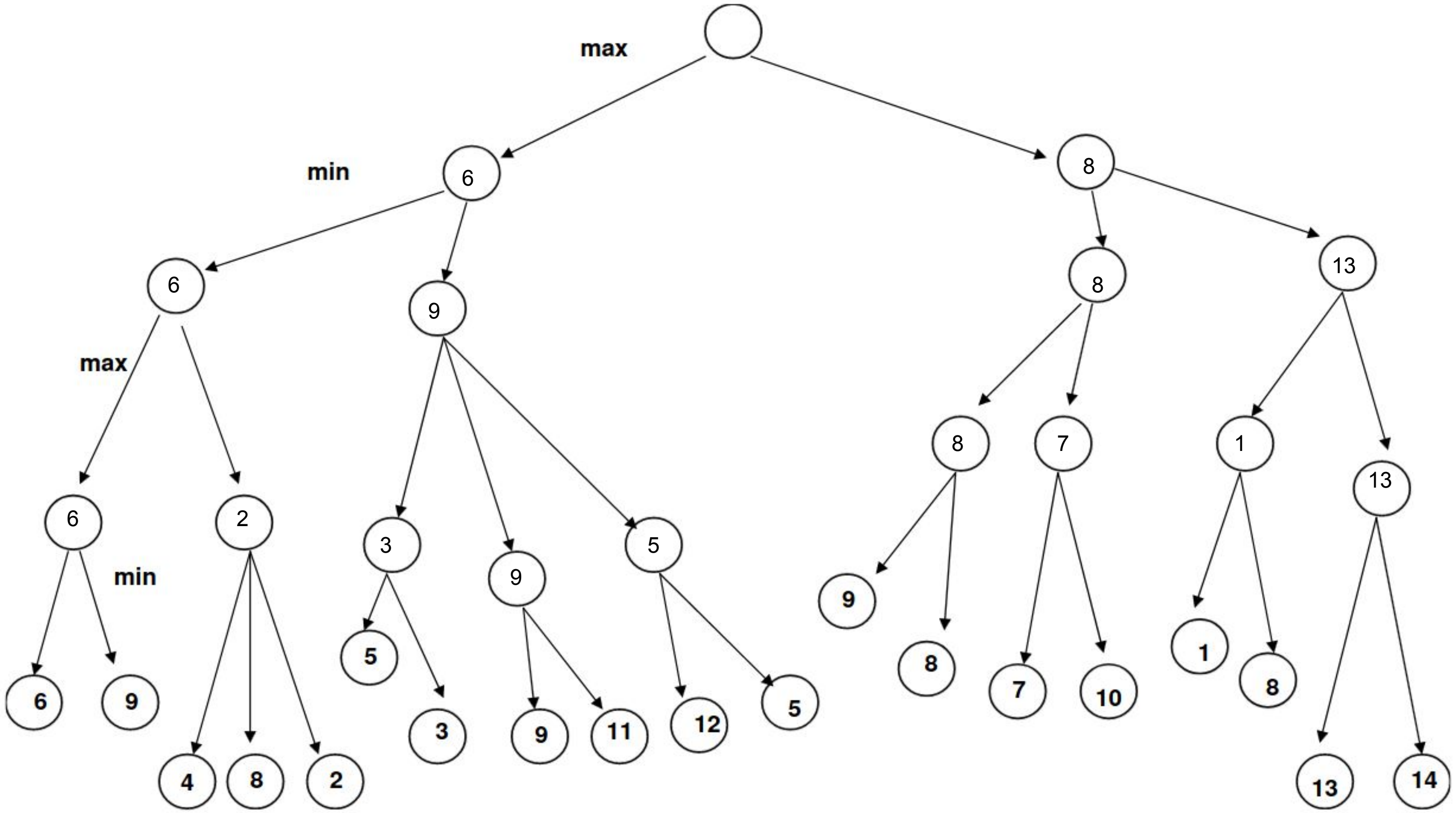
Example: Minimax 2



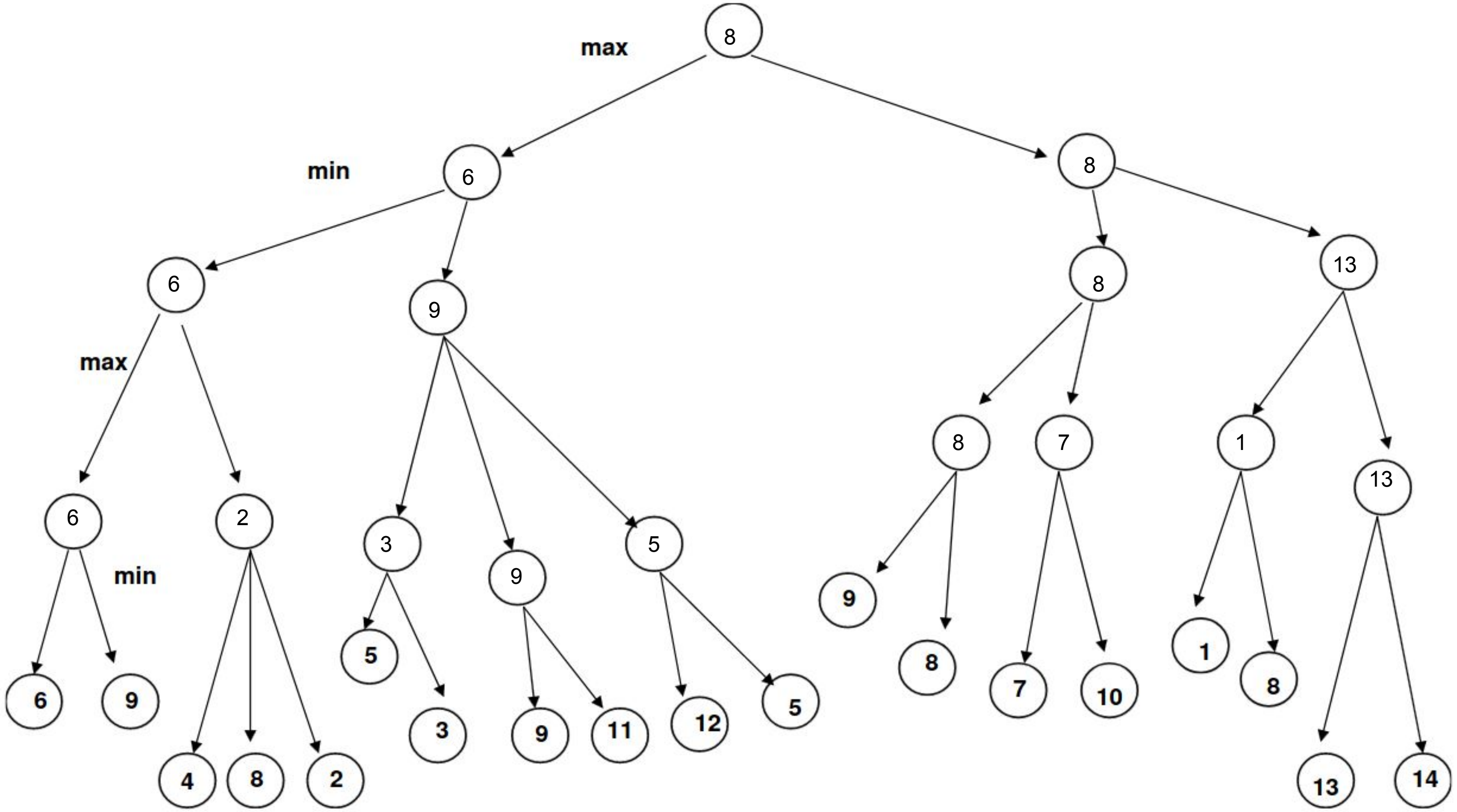
Example: Minimax 2



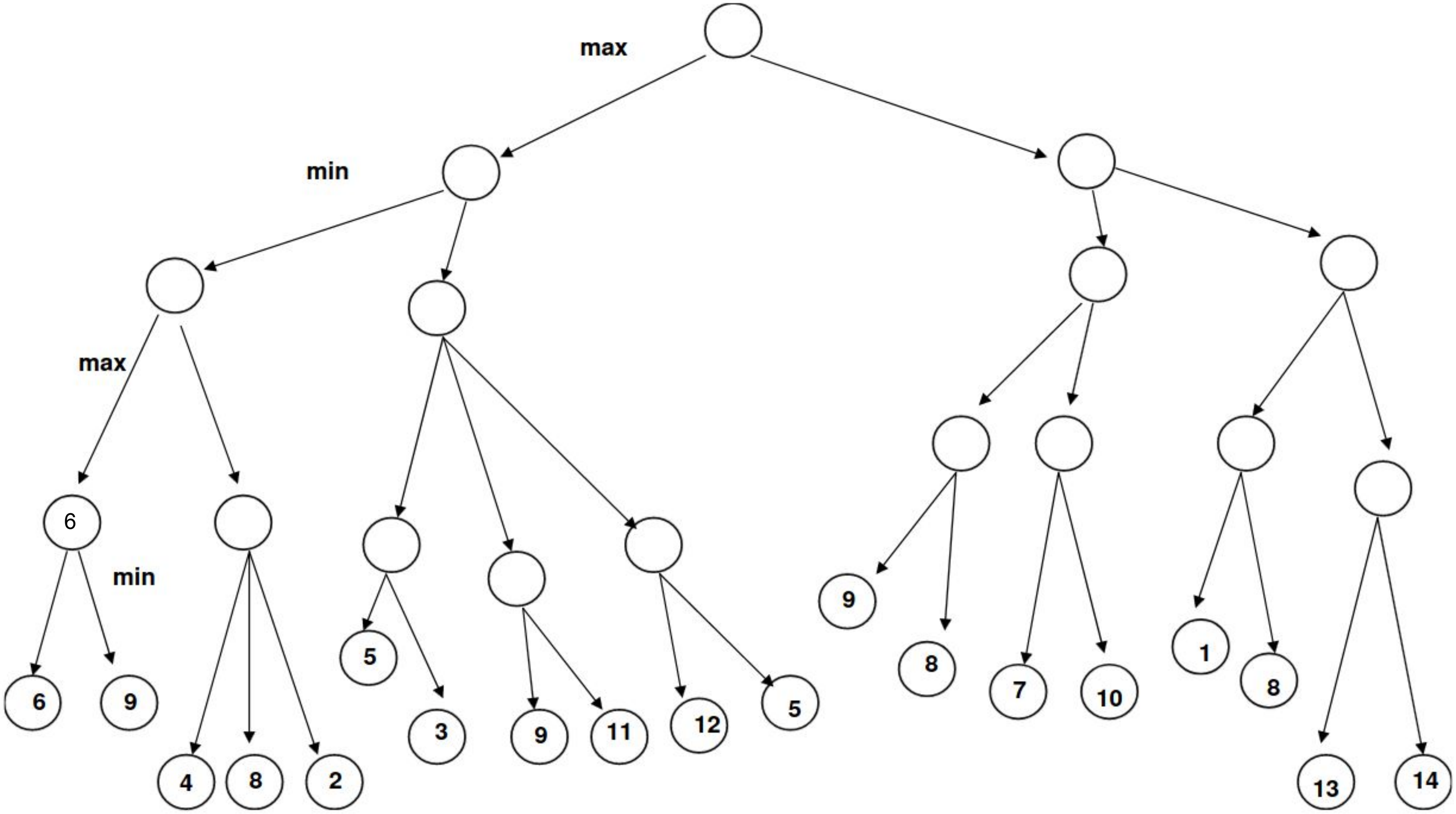
Example: Minimax 2



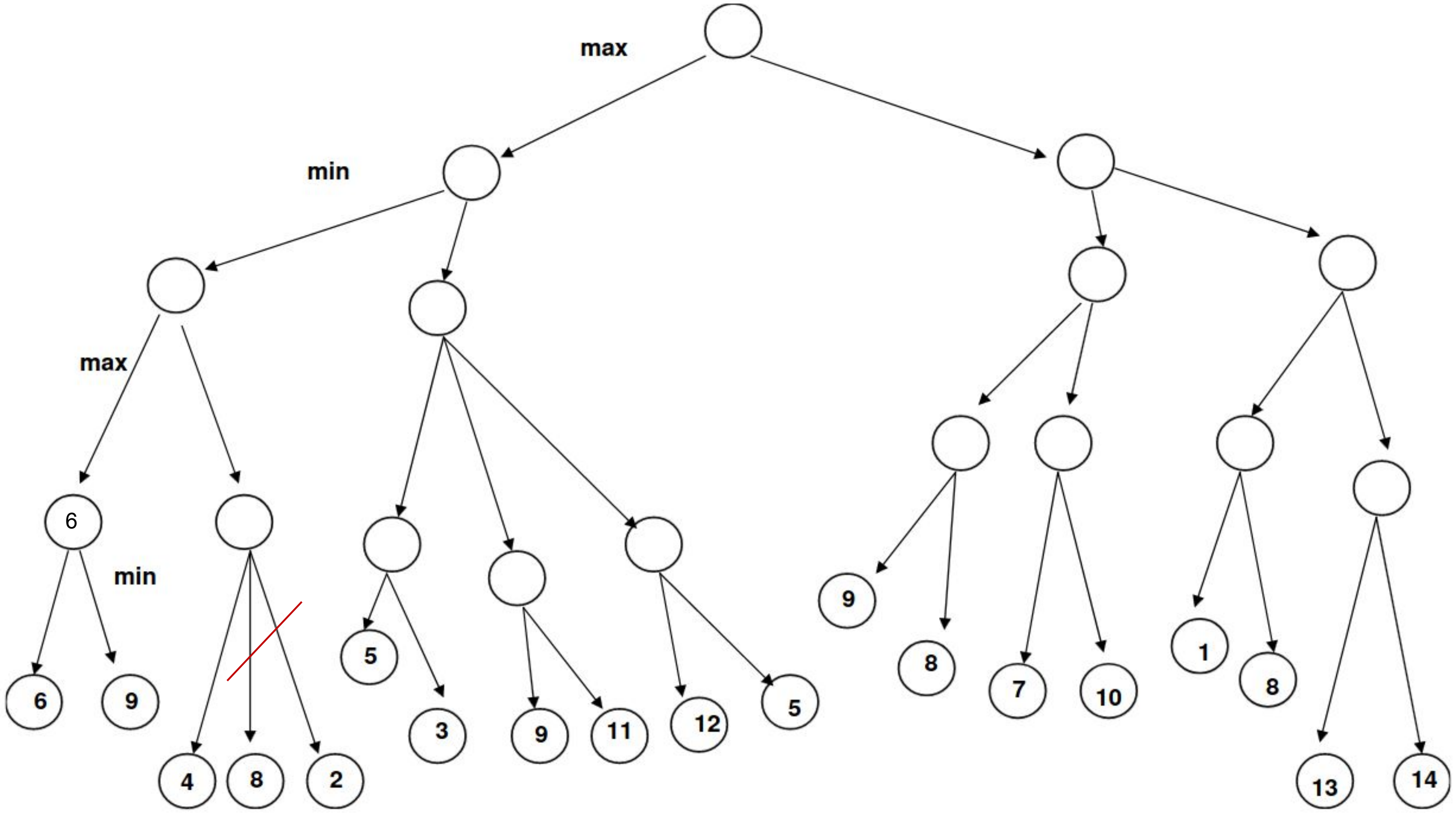
Example: Minimax 2



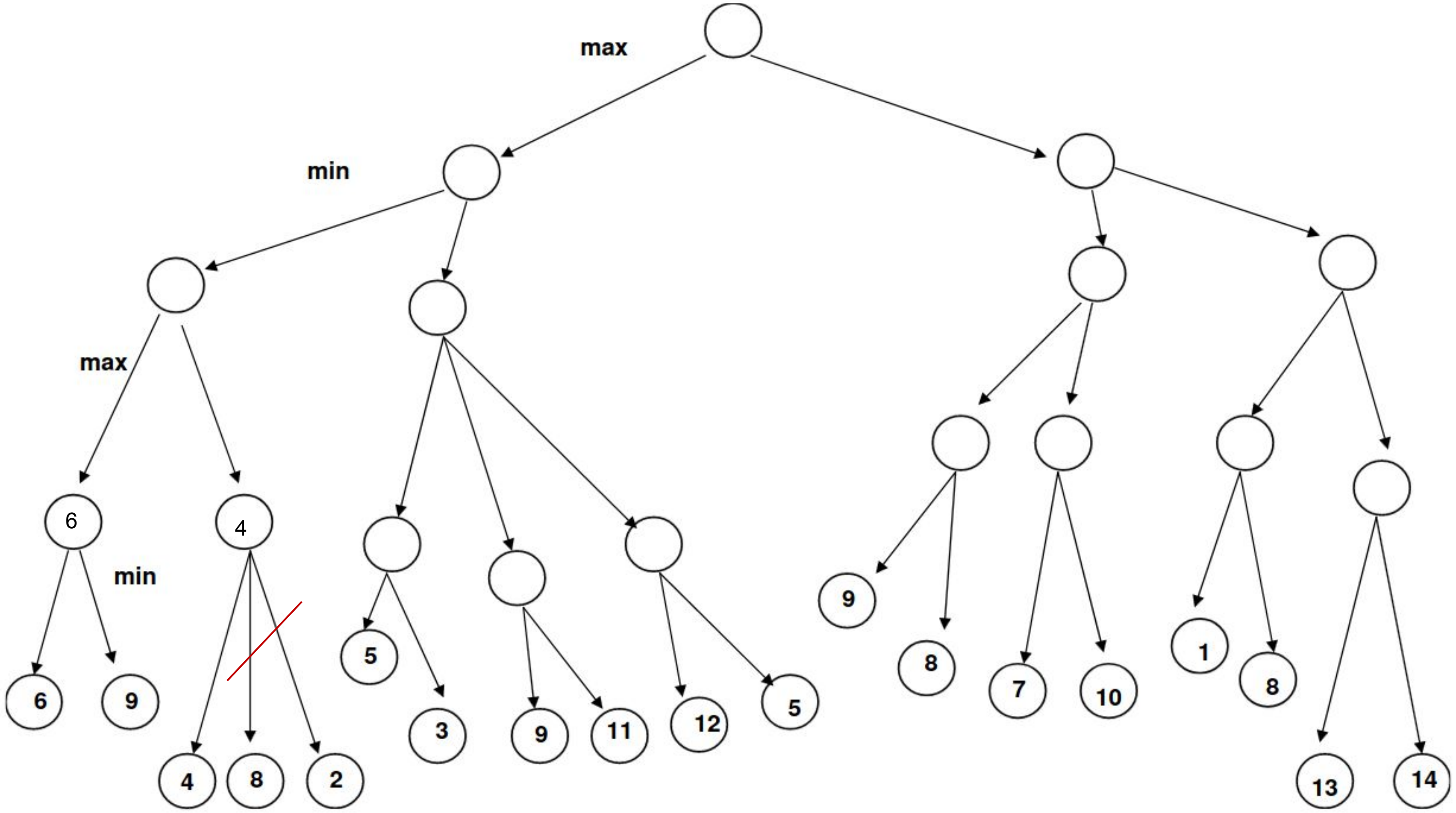
Example: Minimax 2



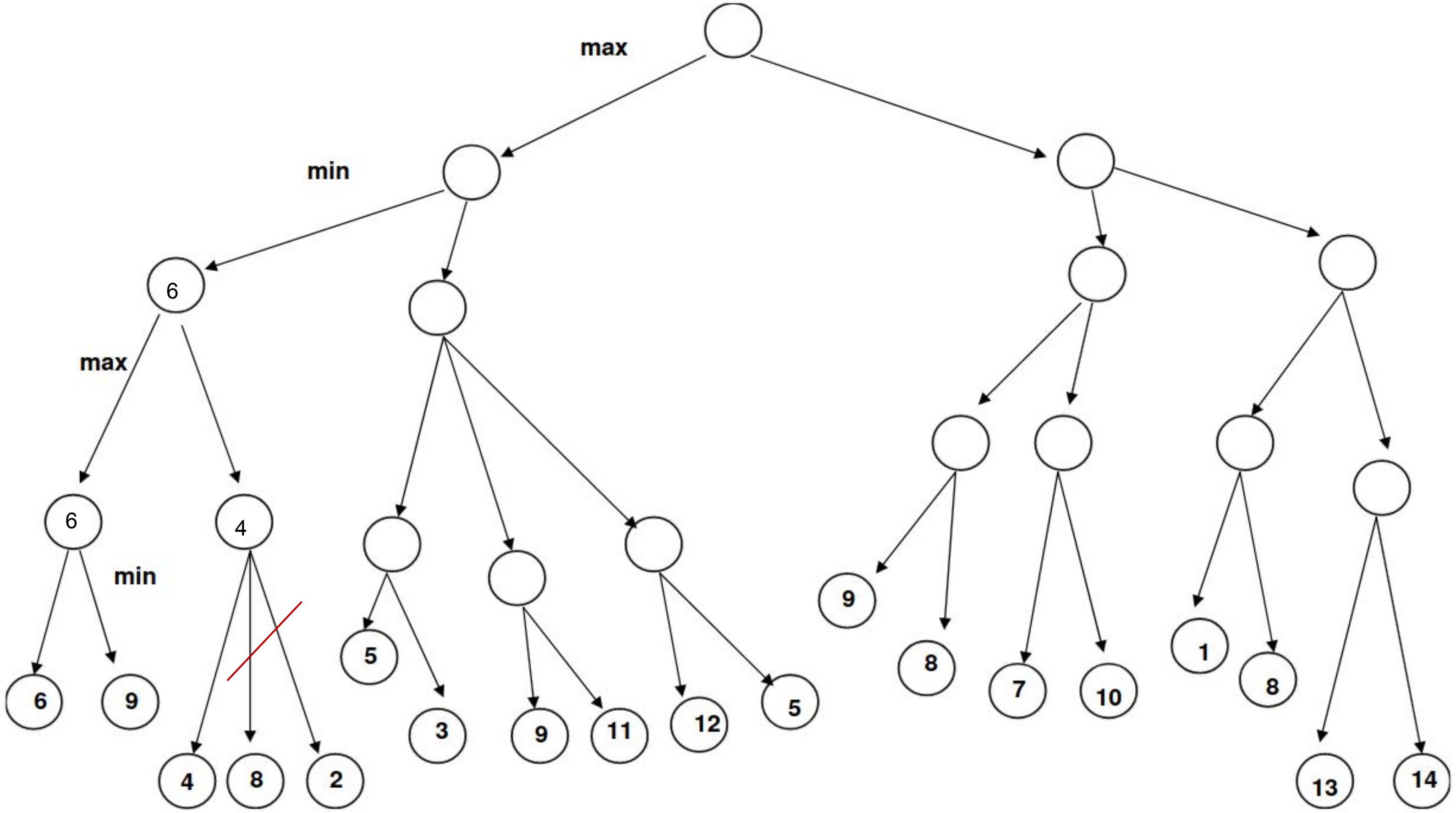
Example: Minimax 2



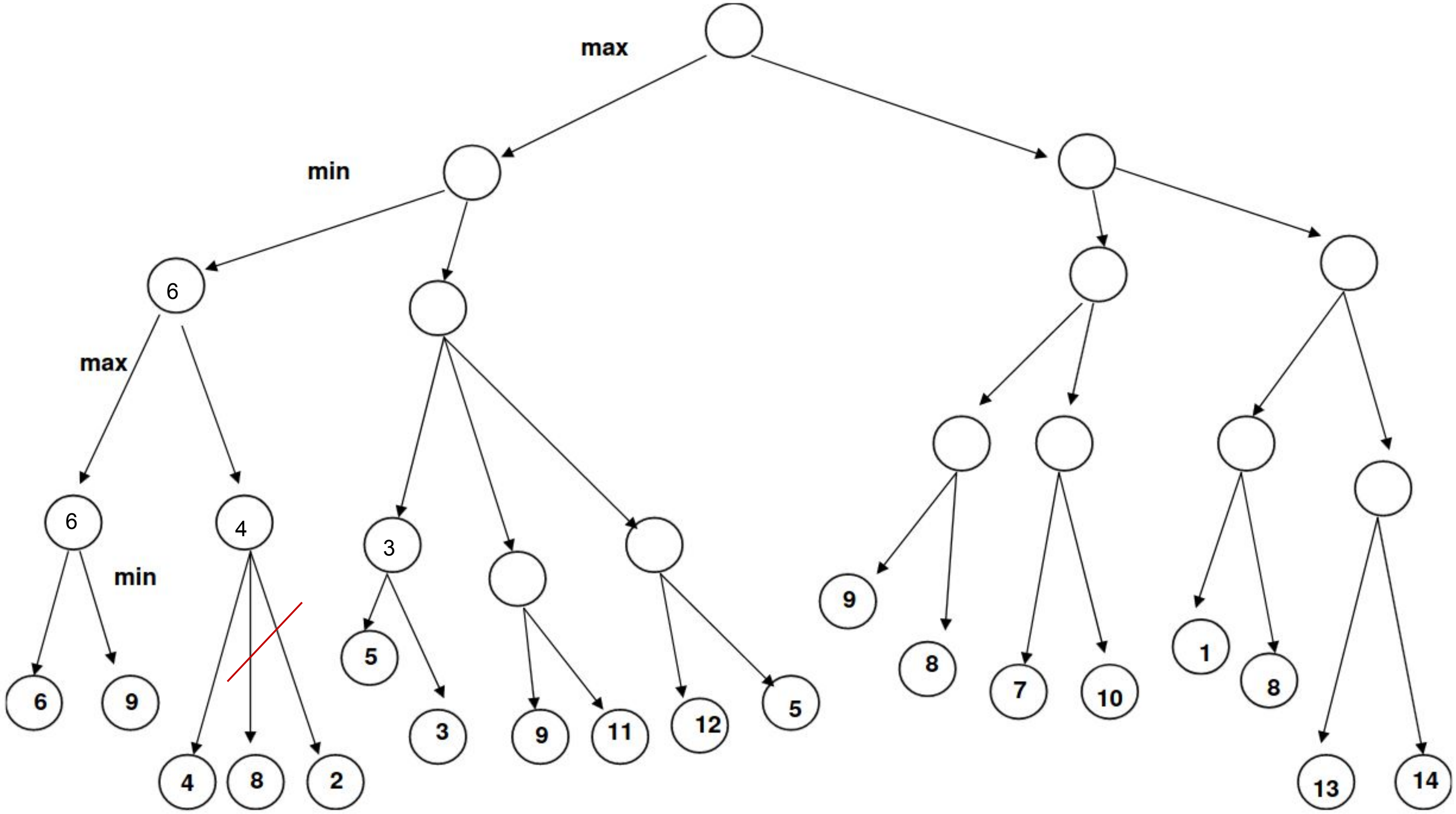
Example: Minimax 2



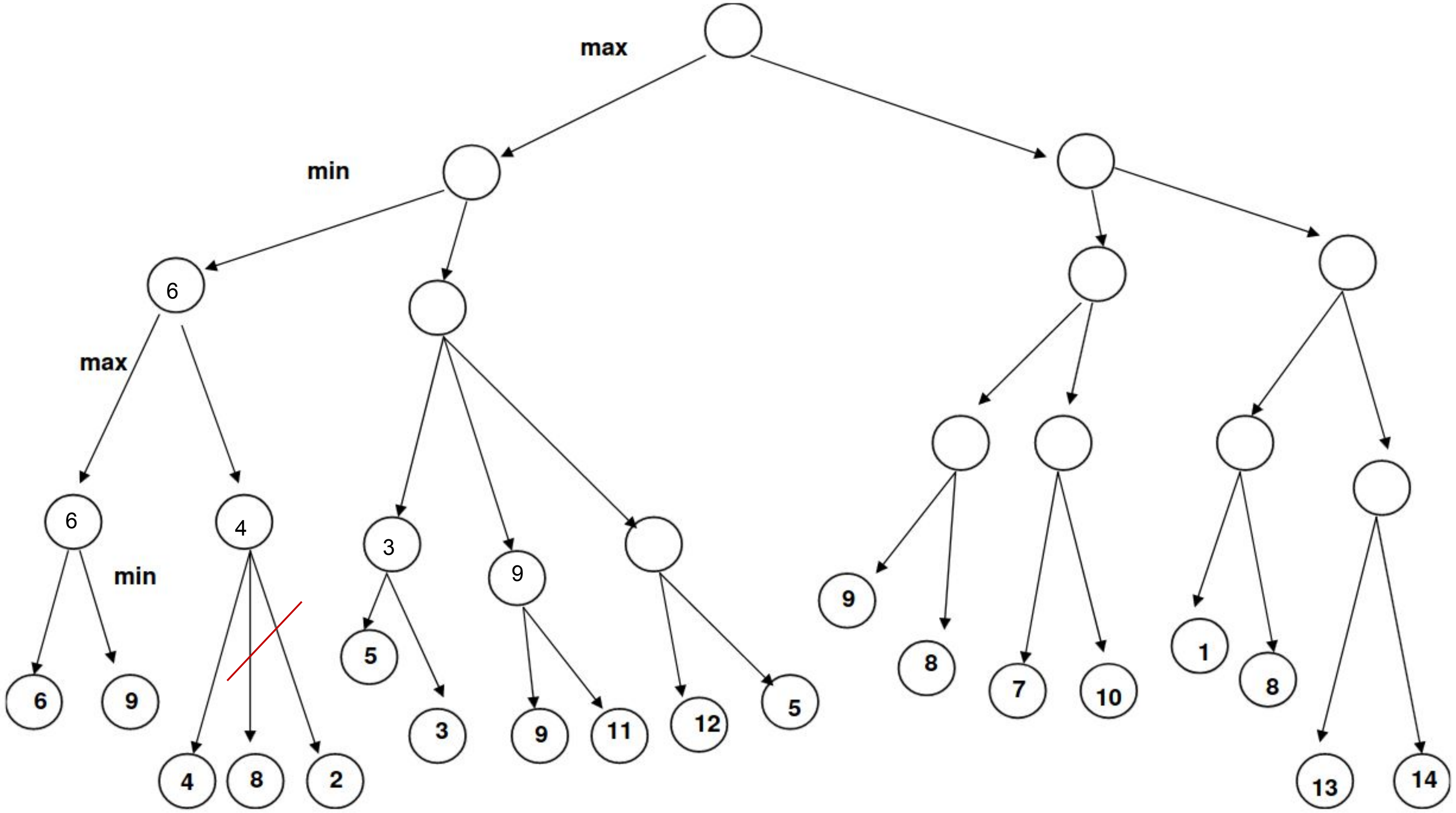
Example: Minimax 2



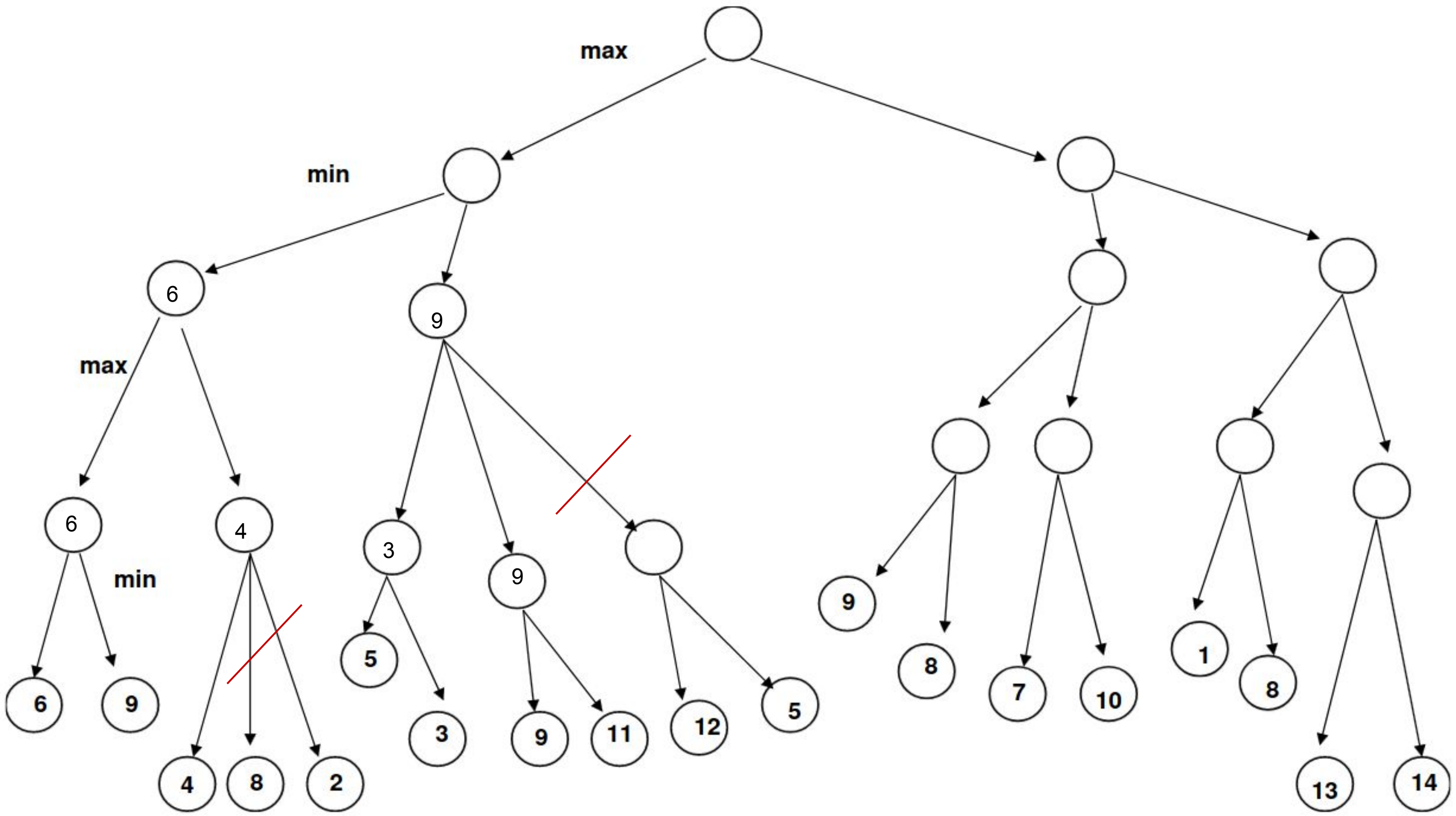
Example: Minimax 2



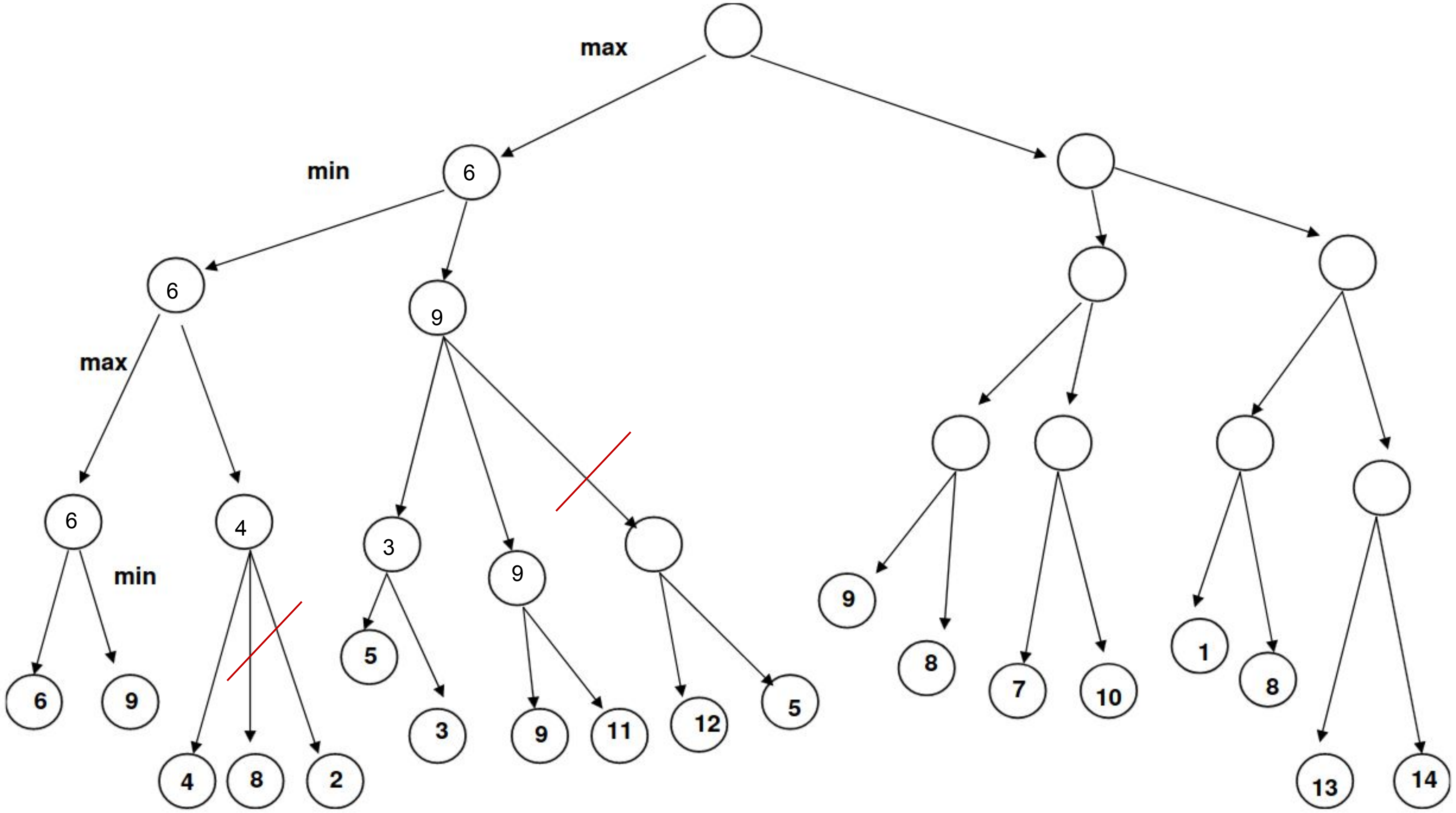
Example: Minimax 2



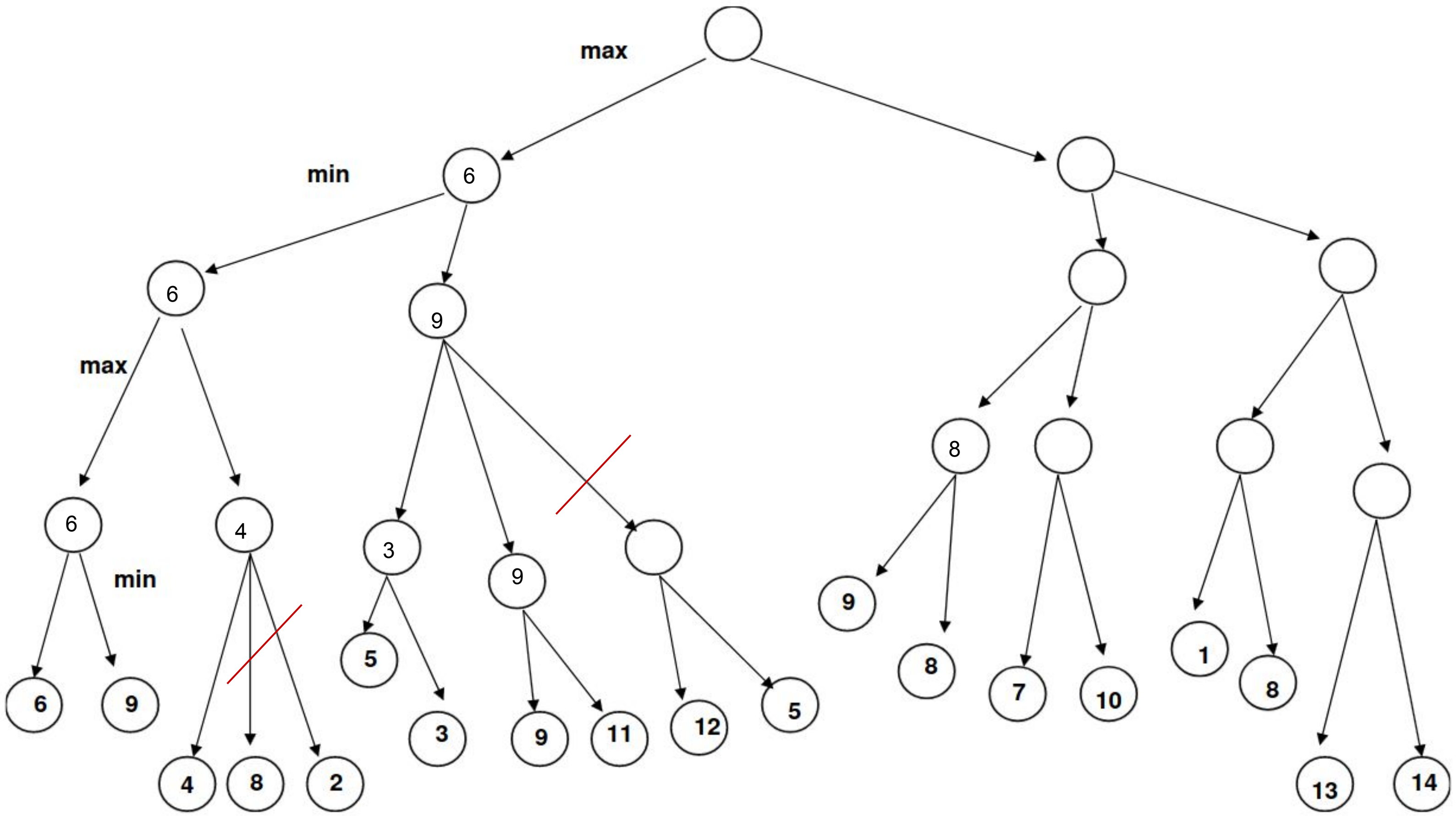
Example: Minimax 2



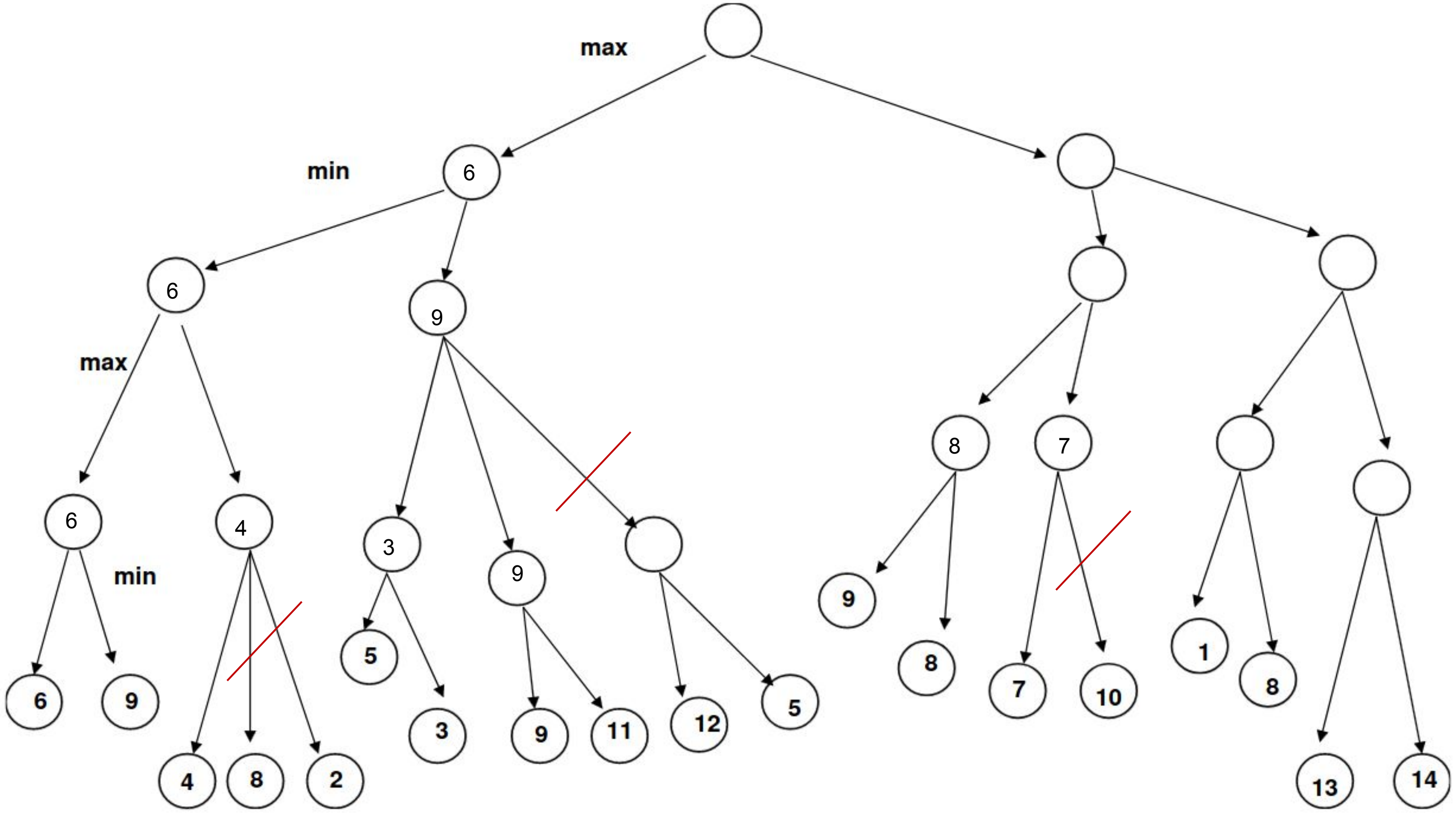
Example: Minimax 2



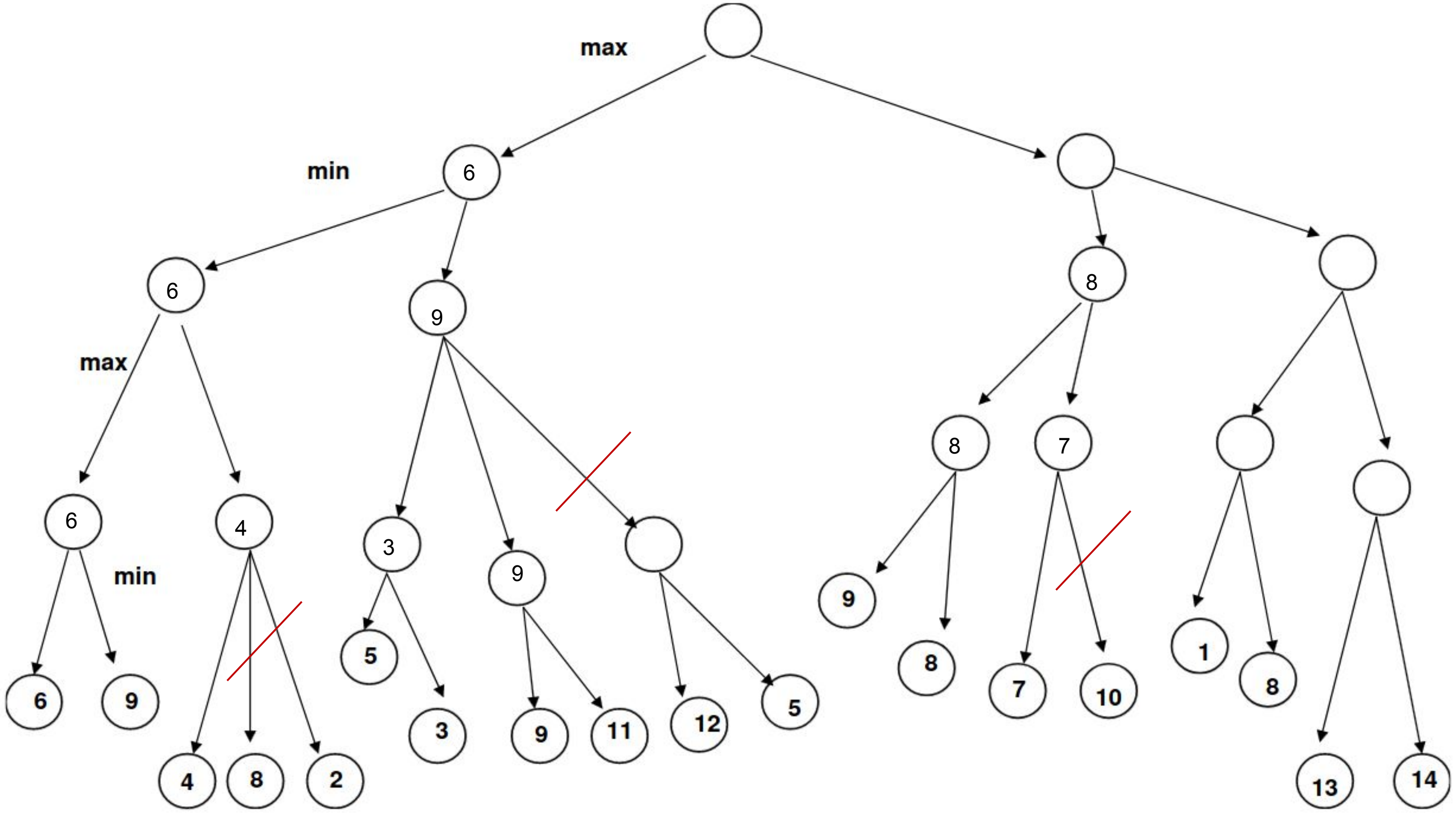
Example: Minimax 2



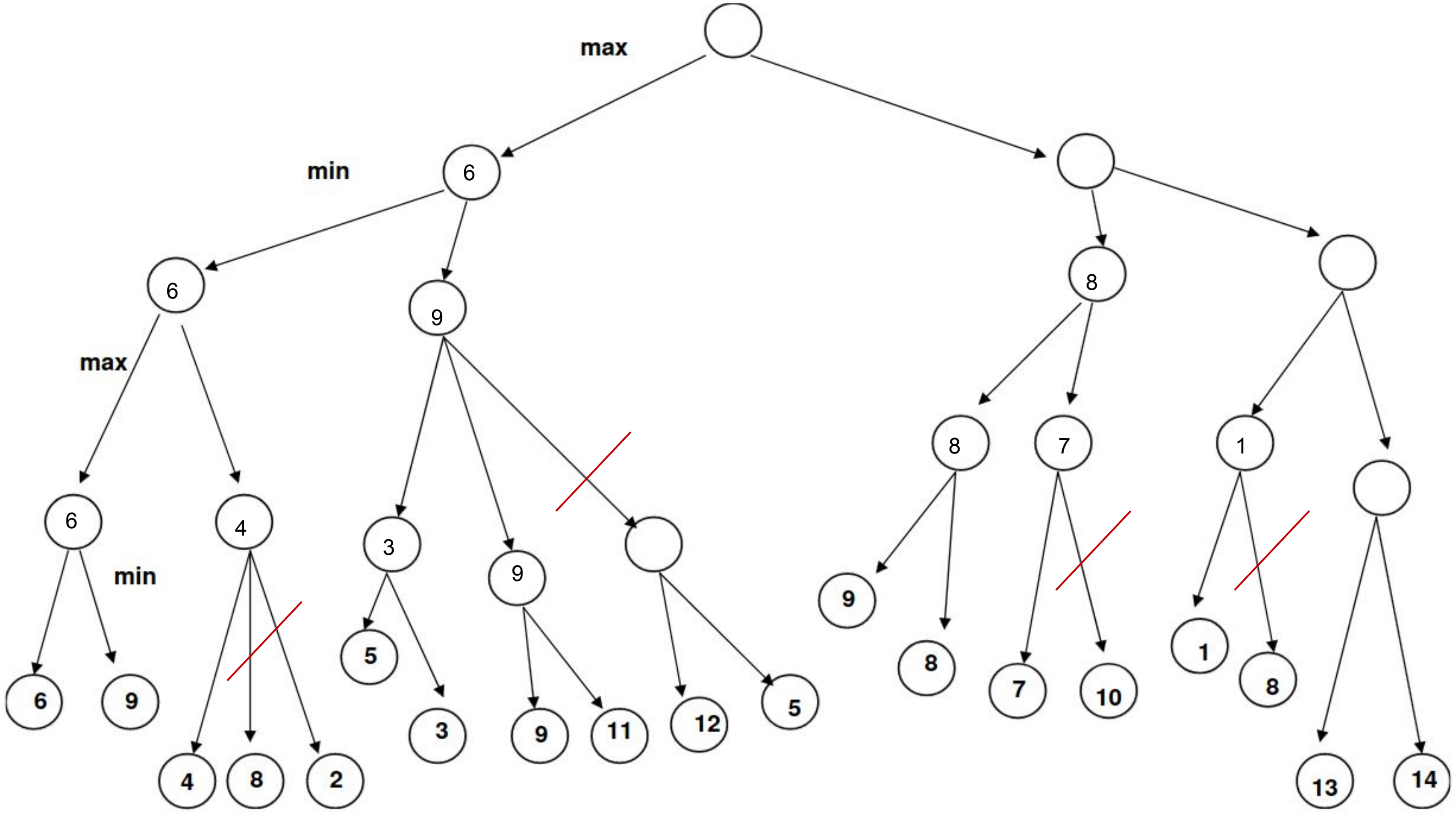
Example: Minimax 2



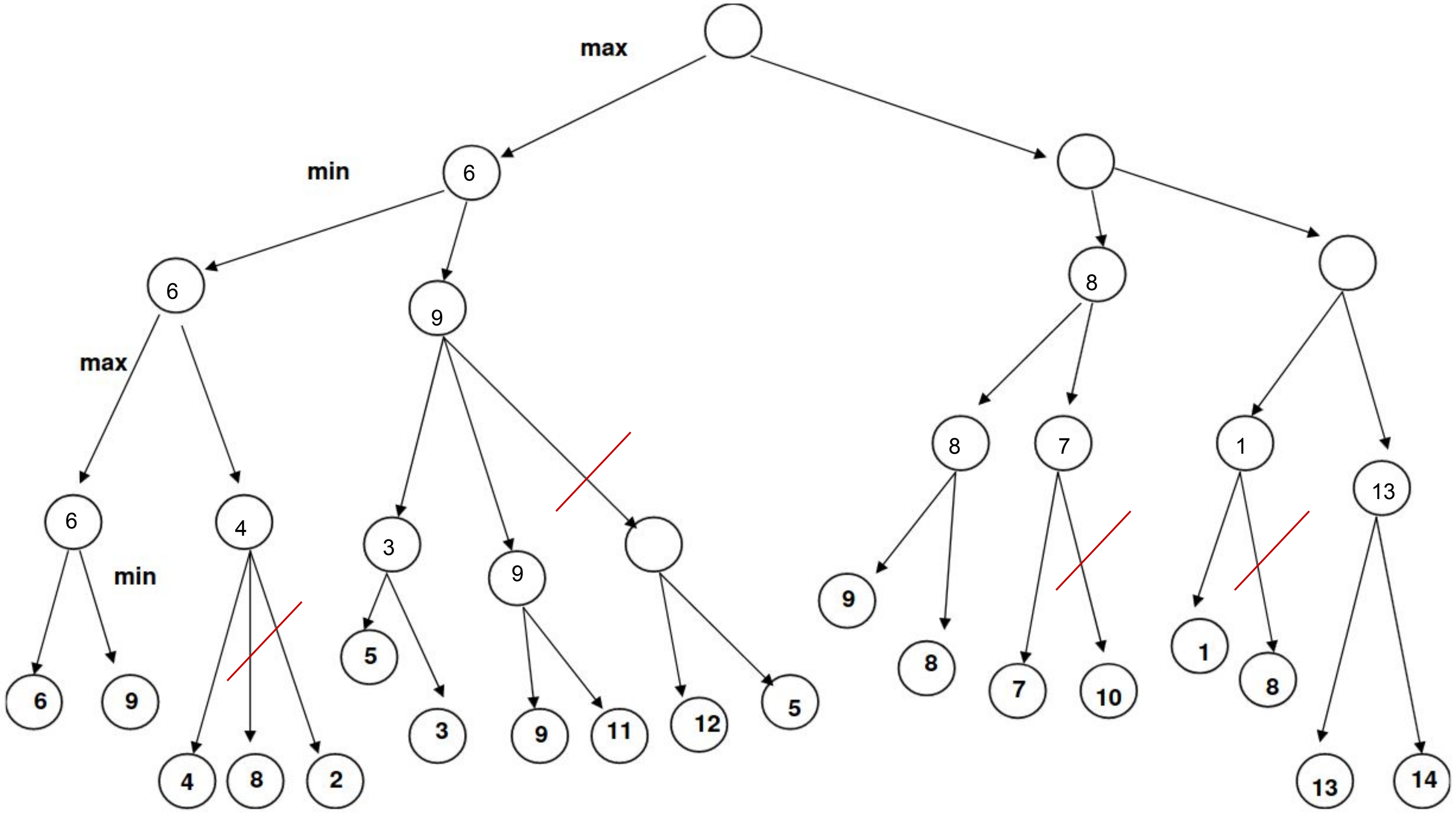
Example: Minimax 2



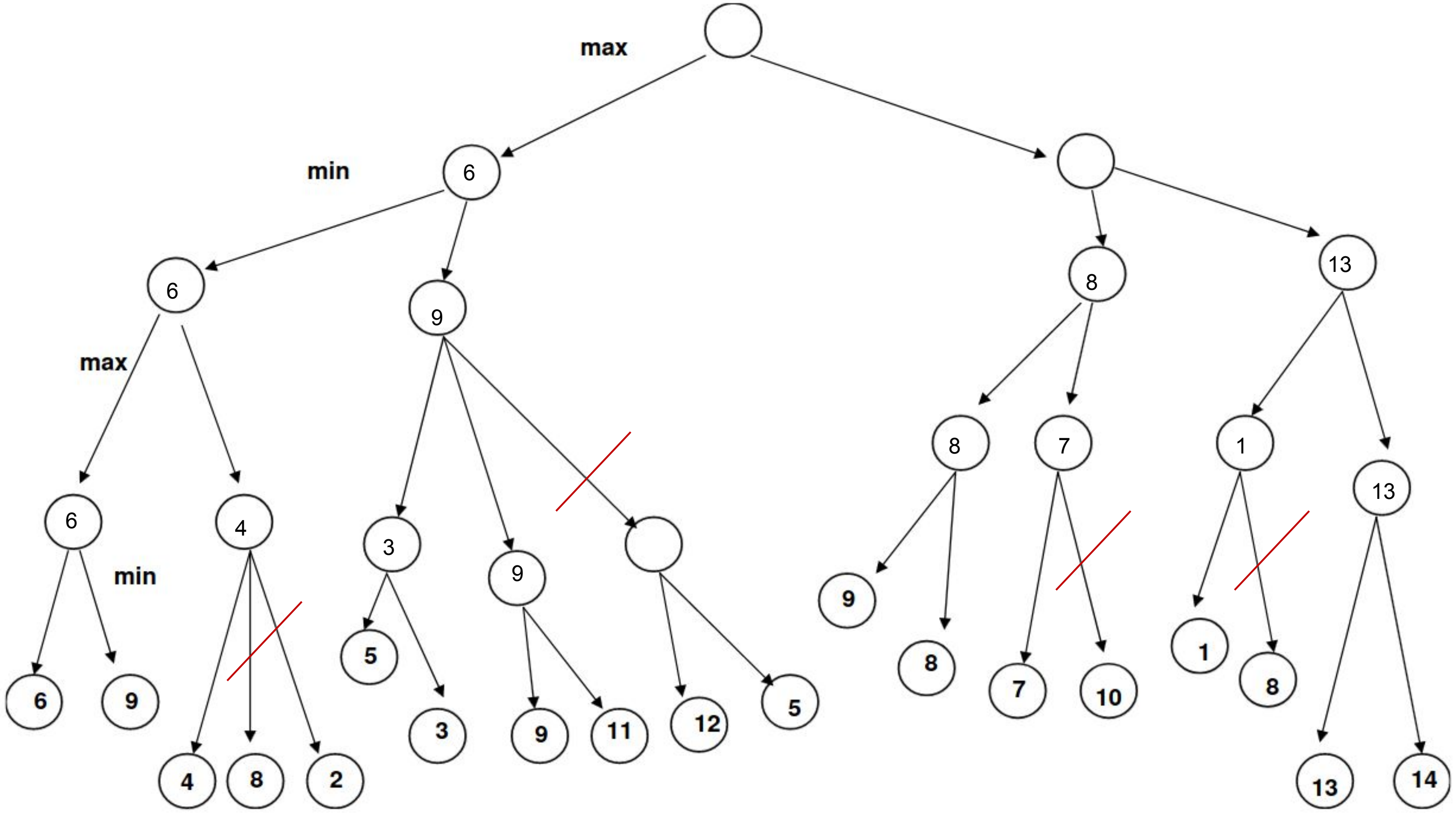
Example: Minimax 2



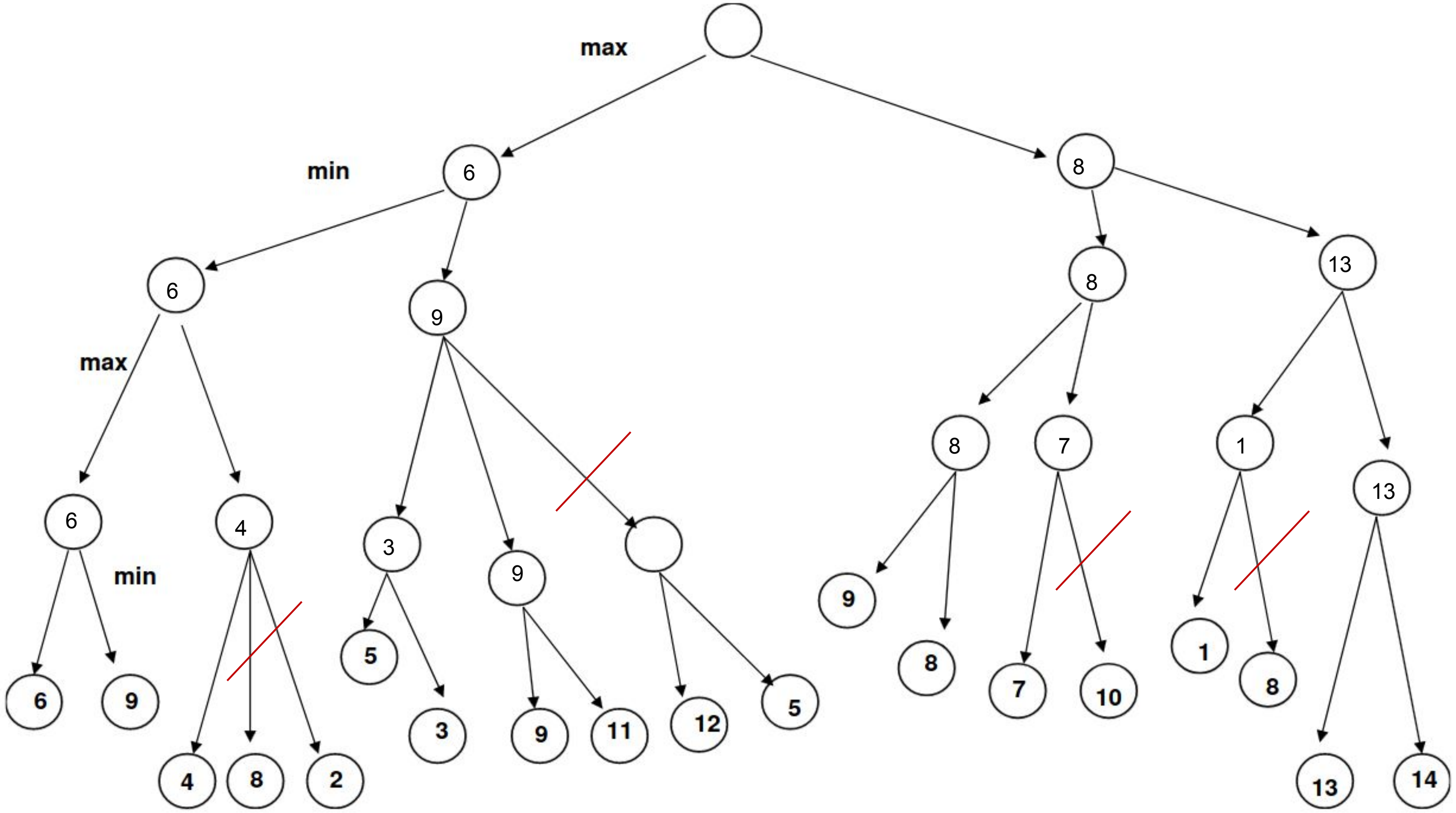
Example: Minimax 2



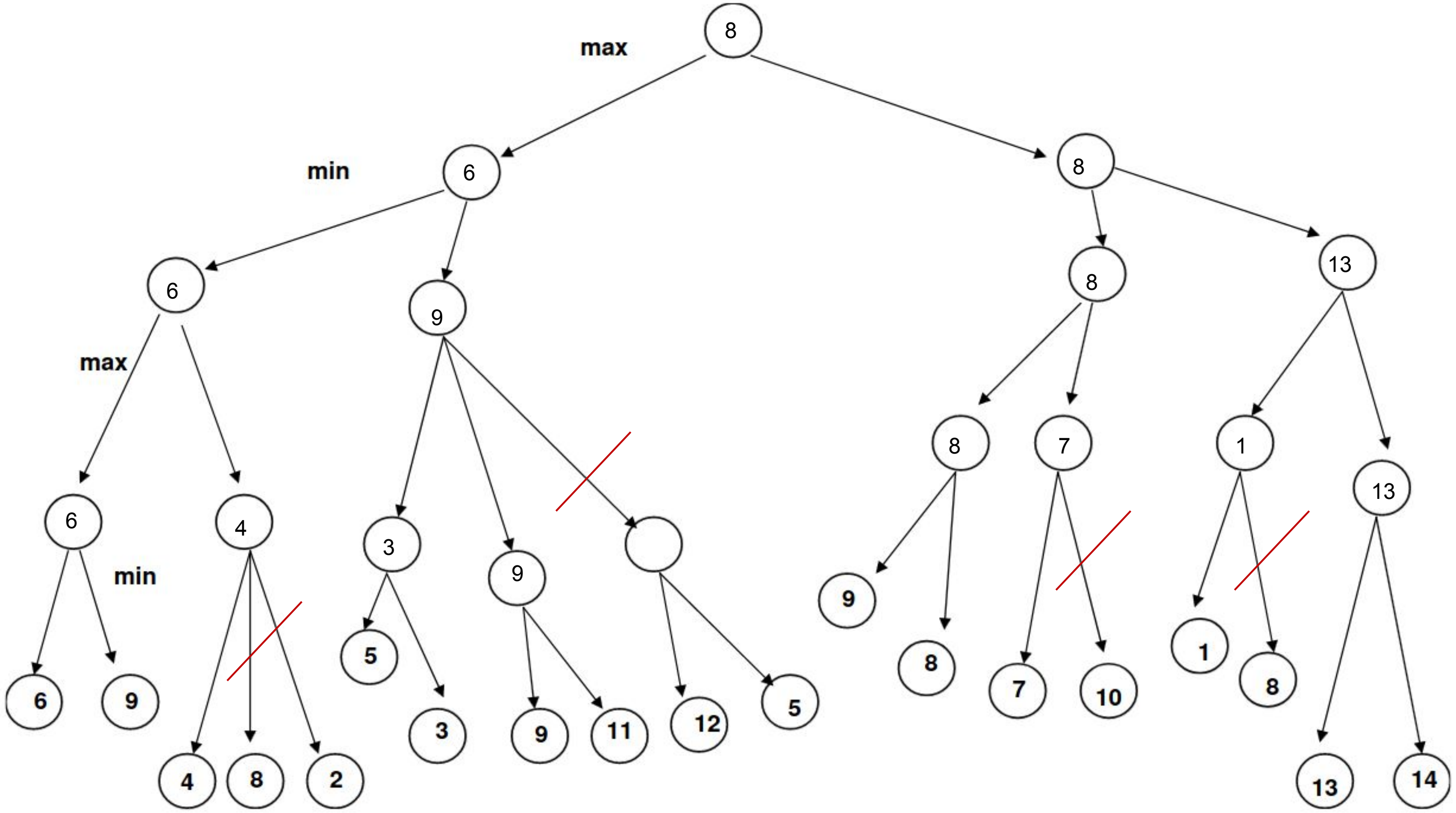
Example: Minimax 2



Example: Minimax 2



Example: Minimax 2



Tools

https://raphsilva.github.io/utilities/minimax_simulator/#

https://inst.eecs.berkeley.edu/~cs61b/fa14/ta-materials/apps/ab_tree_practice/