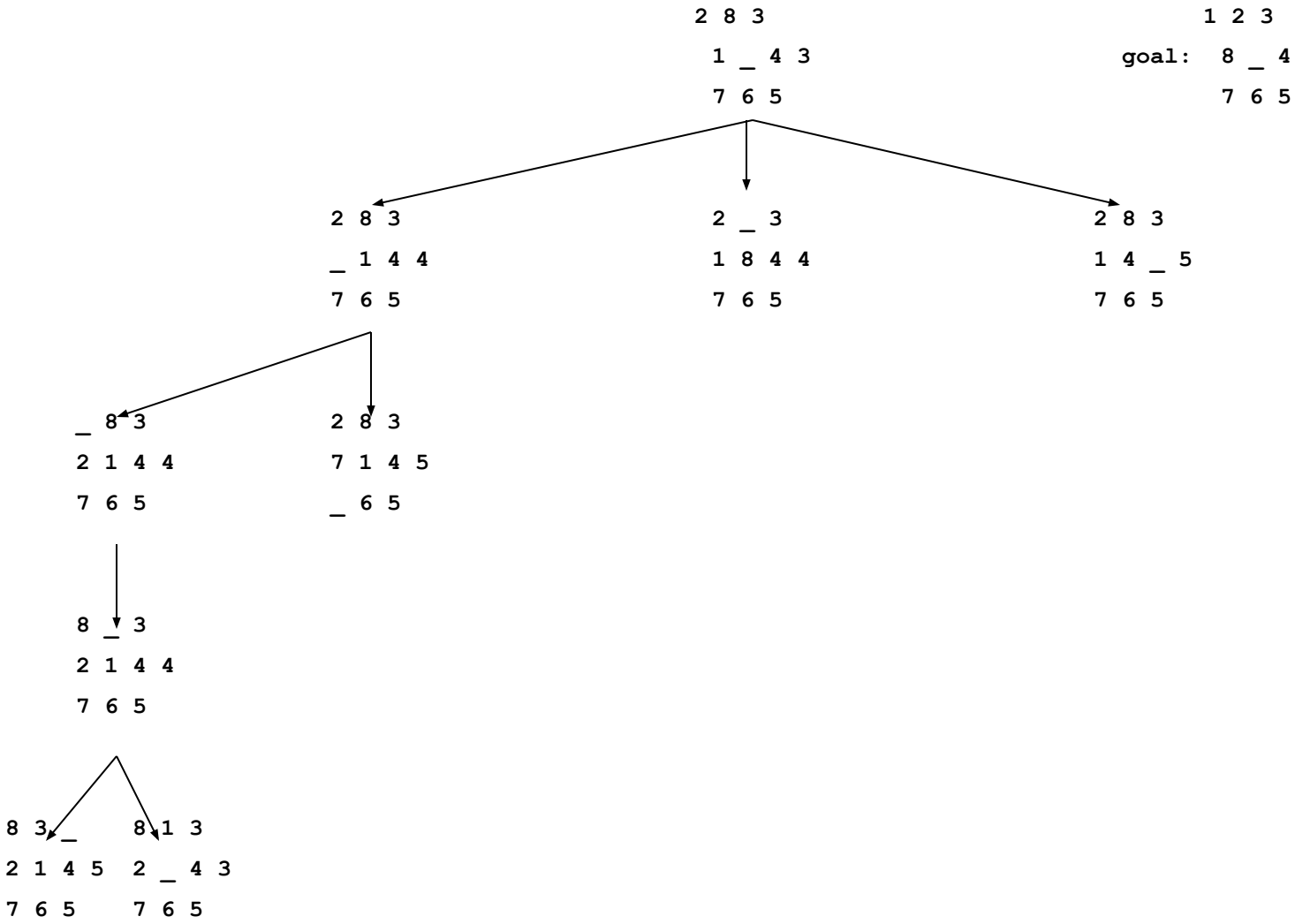
CPSC 312

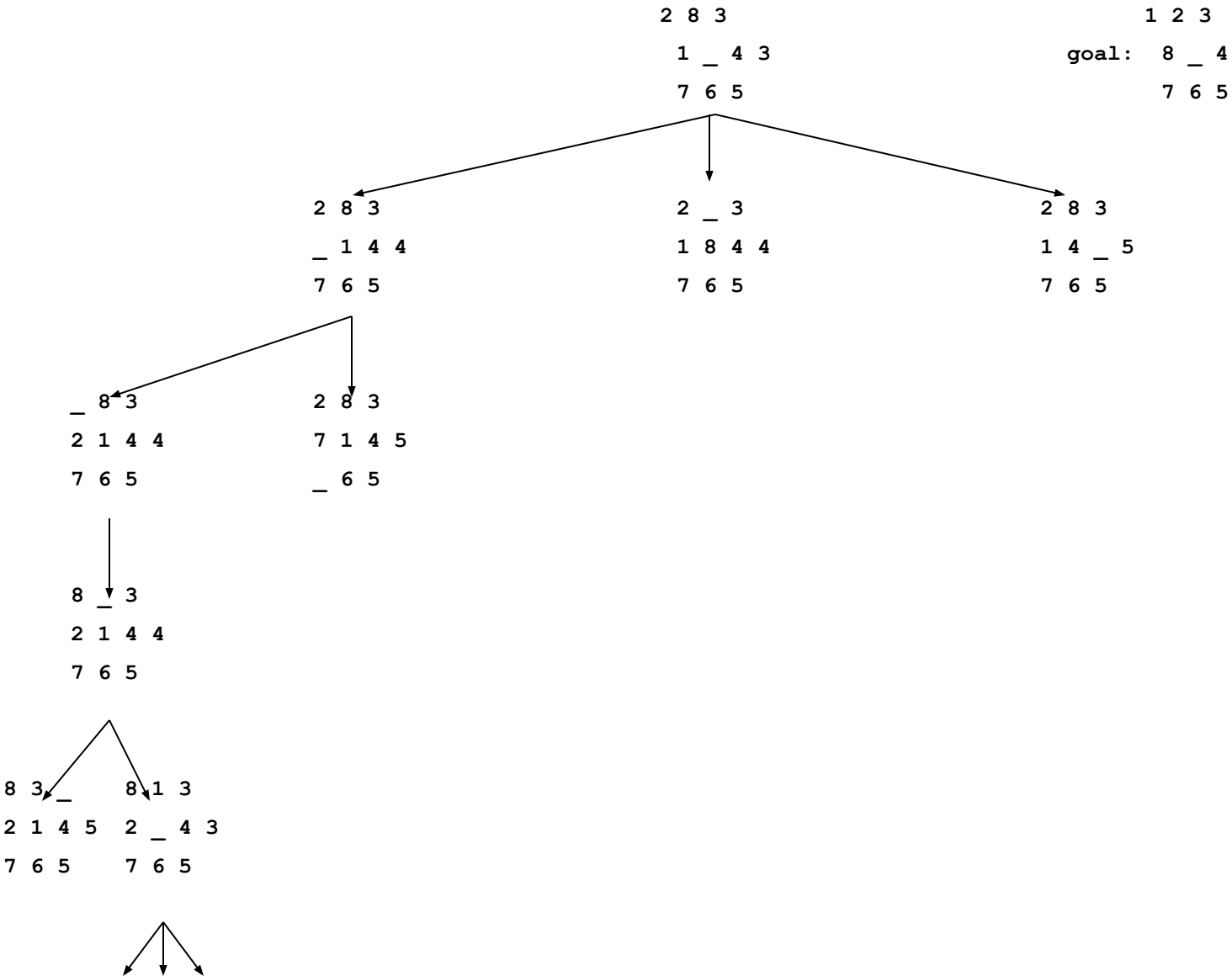
Functional and Logic Programming

November 24, 2015

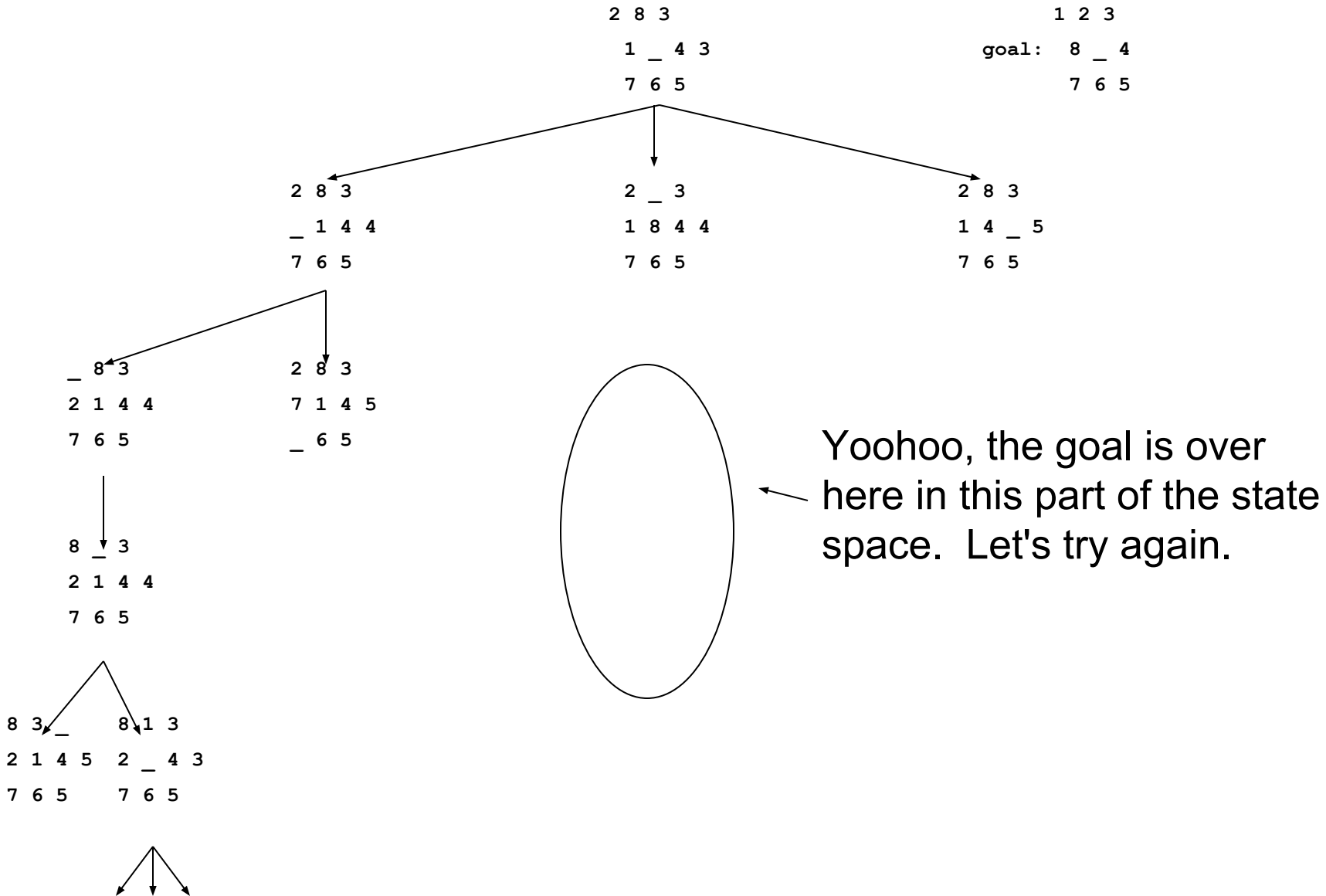
Best-first search - tiles out of place



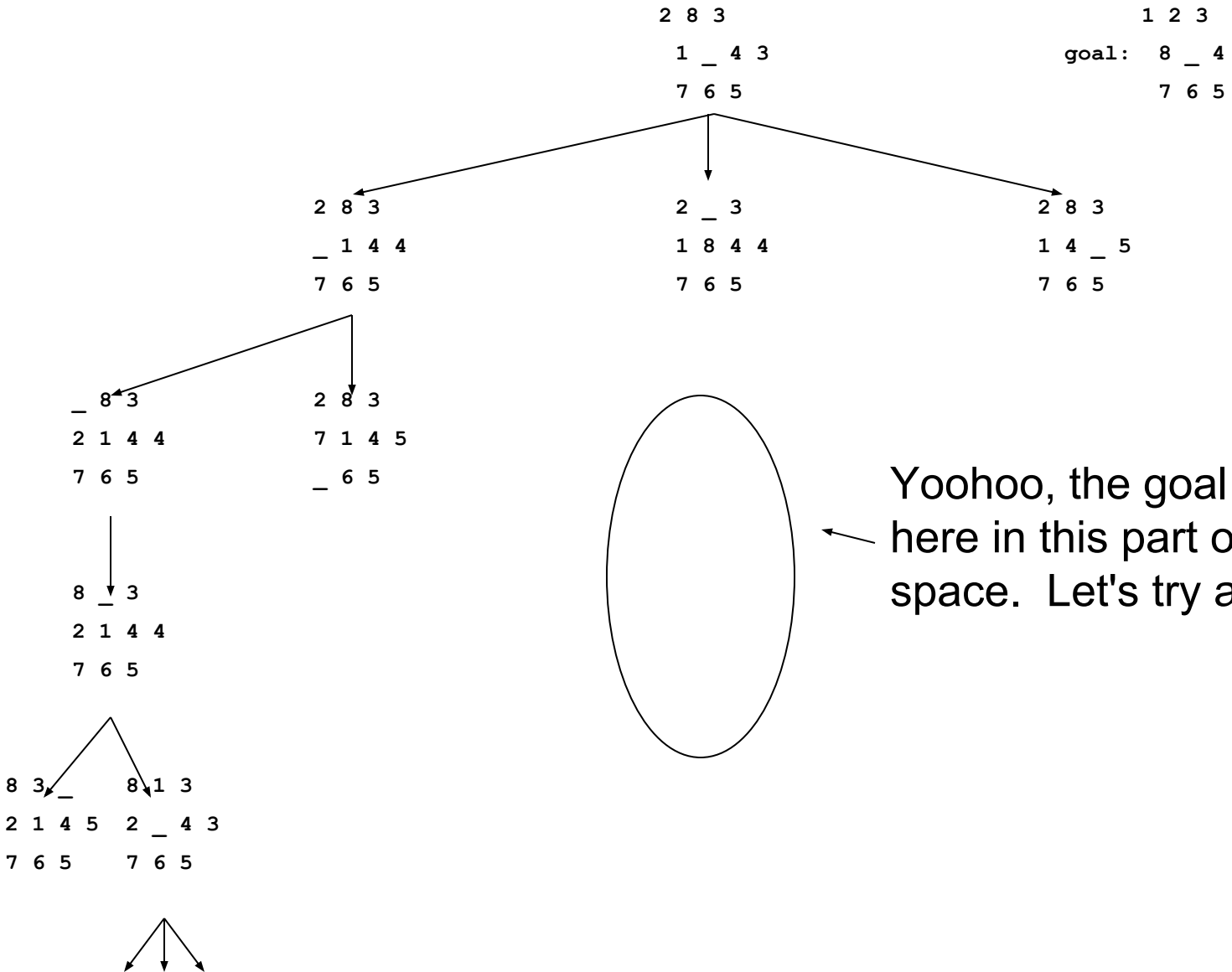
Best-first search - tiles out of place



Best-first search - tiles out of place

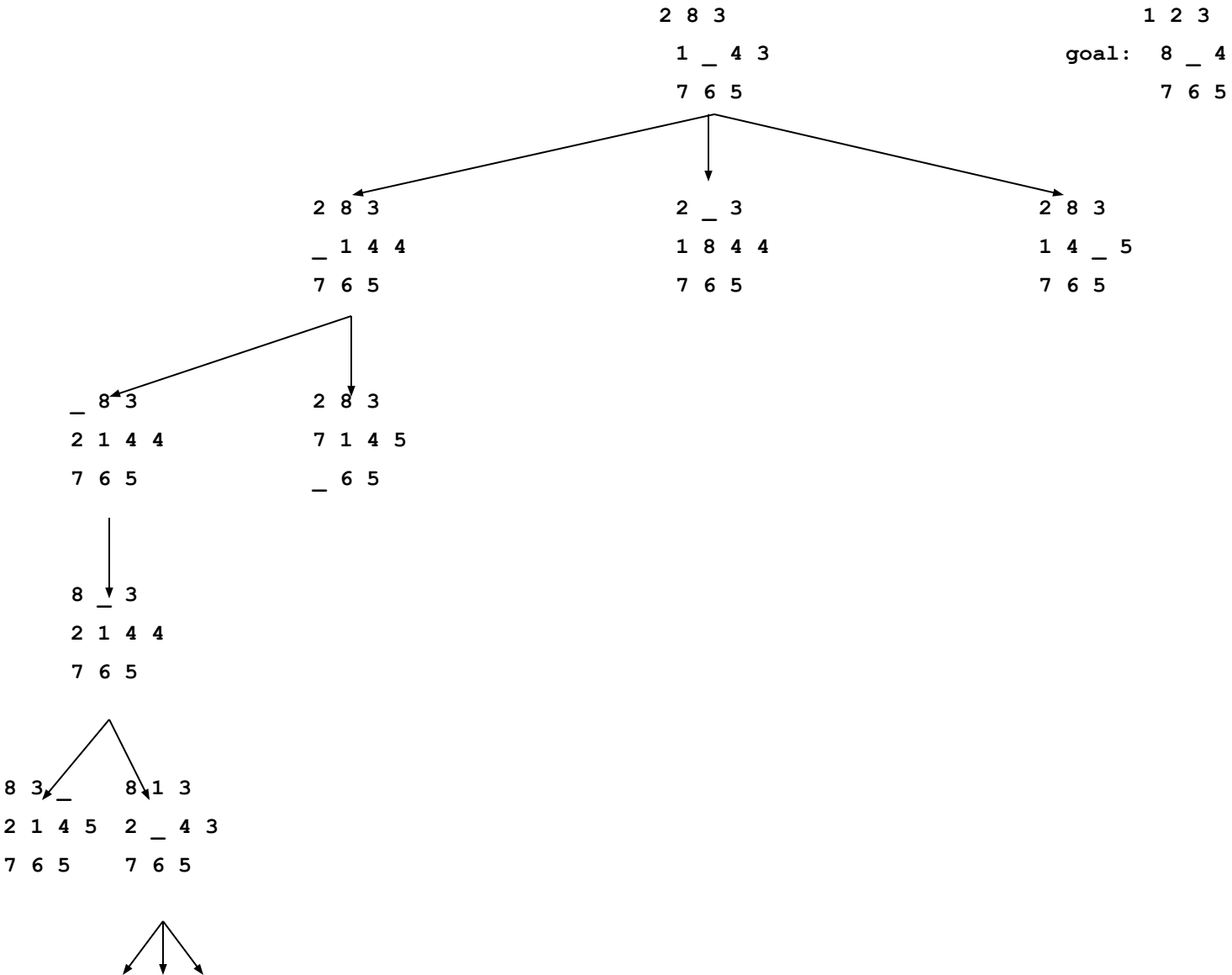


How about a better heuristic?



Yoohoo, the goal is over here in this part of the state space. Let's try again.

How about a better heuristic?



Best-first search - Manhattan distance

```
2 8 3
1 _ 4
7 6 5
```

```
      1 2 3
goal: 8 _ 4
      7 6 5
```

Best-first search - Manhattan distance

Manhattan district in New York City: streets based on grid system of roughly equal-size blocks

shortest distance between two points by taxicab is the sum of the absolute values of the differences of their coordinates

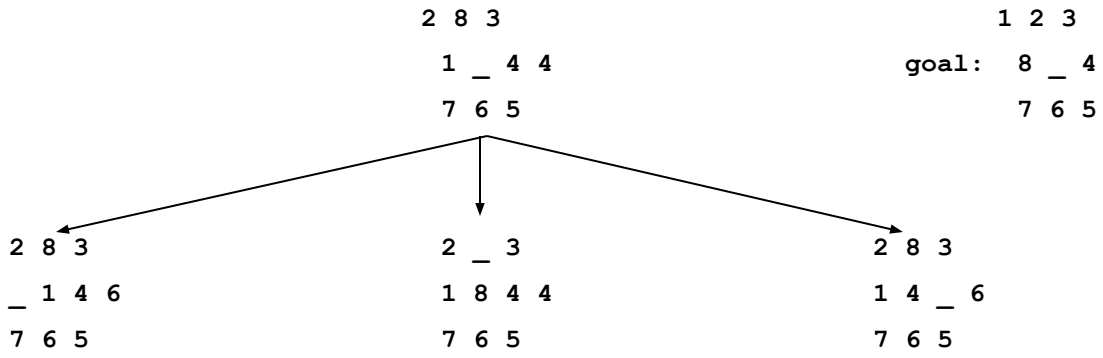
the distance from 8th Avenue and 42nd Street to 5th Avenue and 55th Street is 3 + 13 = 16 city blocks

also called rectilinear distance or city block distance

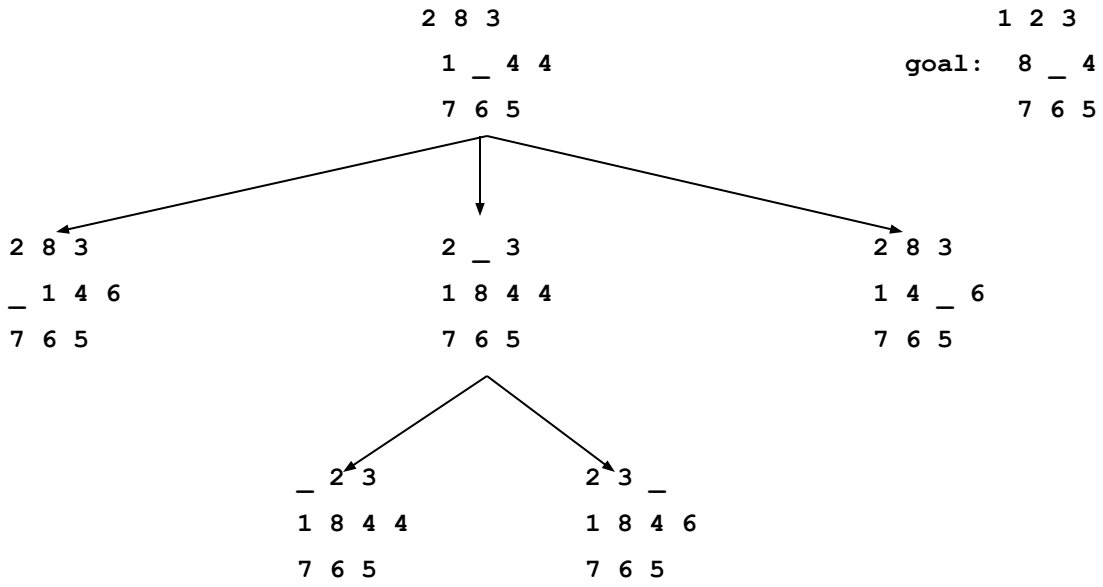
Best-first search - Manhattan distance

For the tile puzzle, we want to know, for each tile, what's the "Manhattan distance" between where the tile is now and where it needs to be to satisfy the goal state. In the example above, three tiles are out of place. The '1' and '2' tiles are a Manhattan distance of 1 away from where they need to be. The '8' tile is a Manhattan distance of 2 away. So the total Manhattan distance is 4.

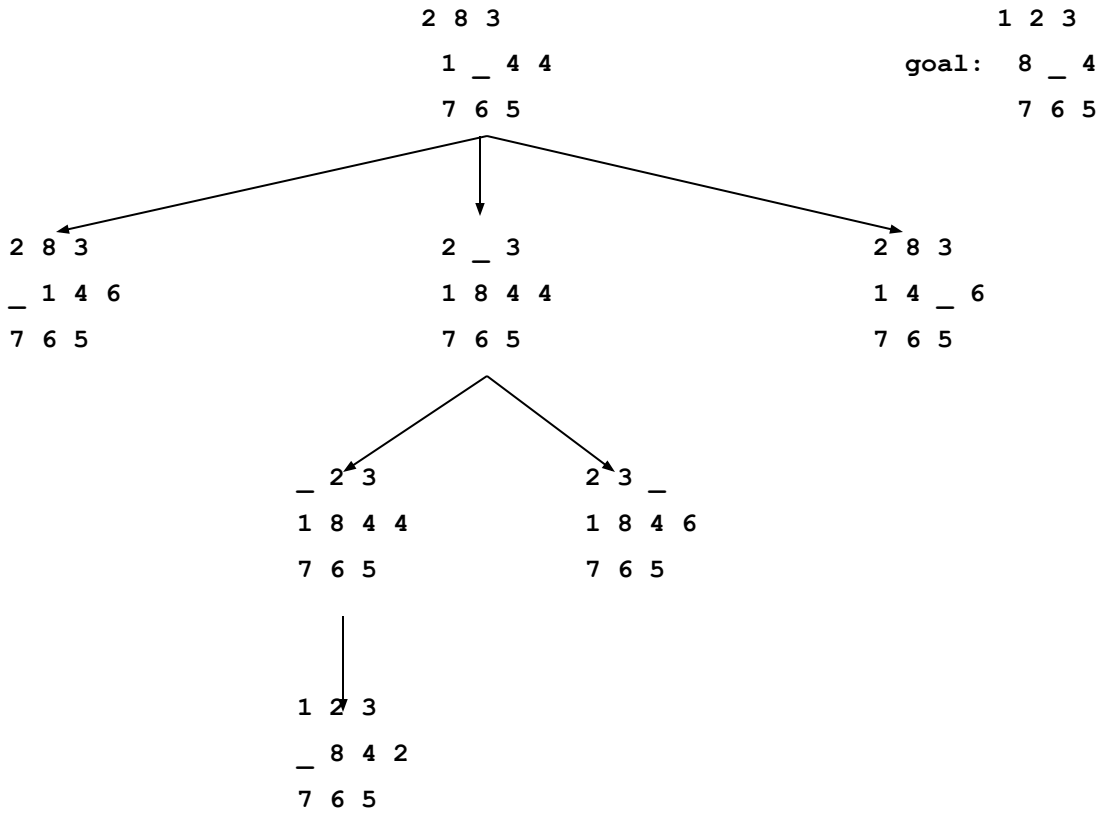
Best-first search - Manhattan distance



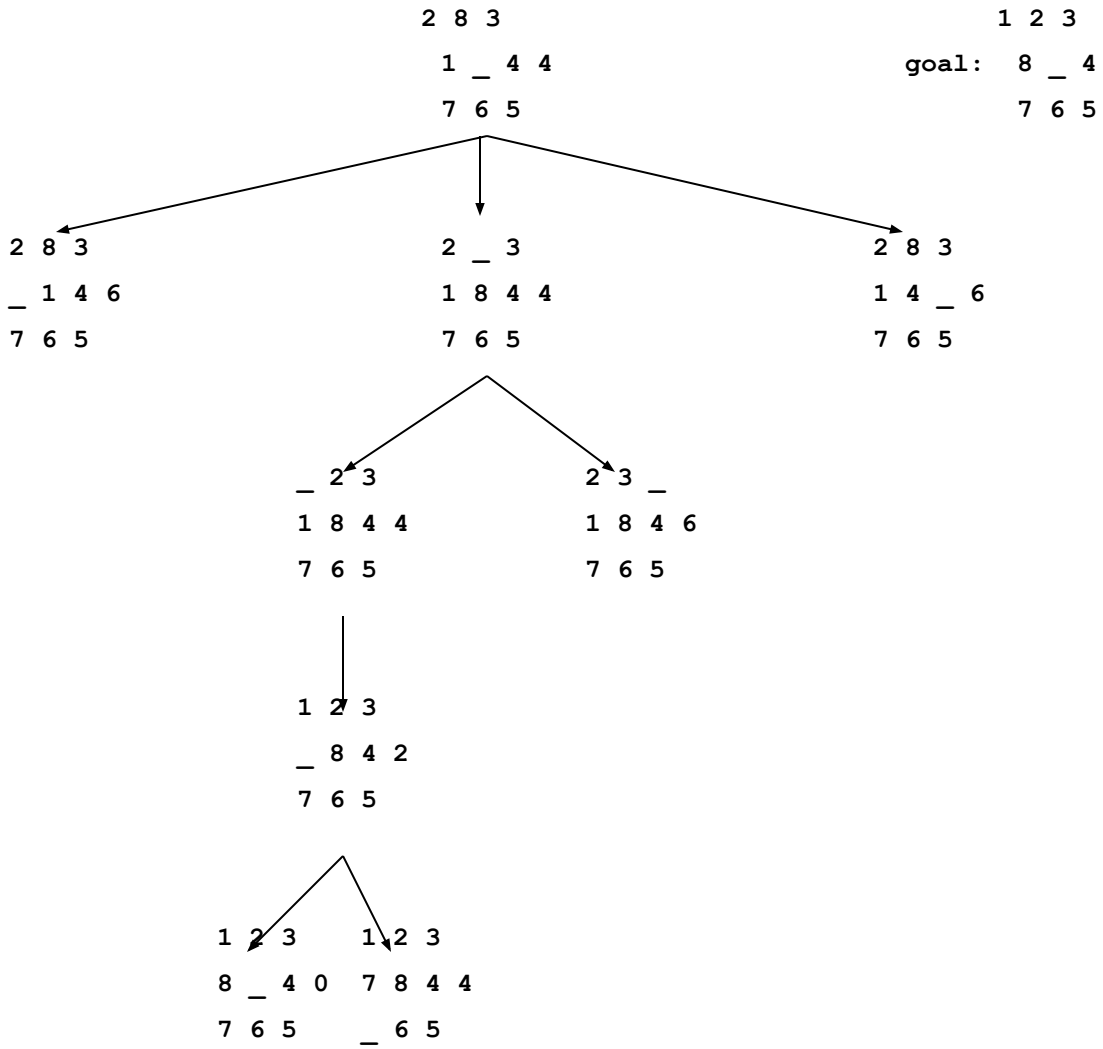
Best-first search - Manhattan distance



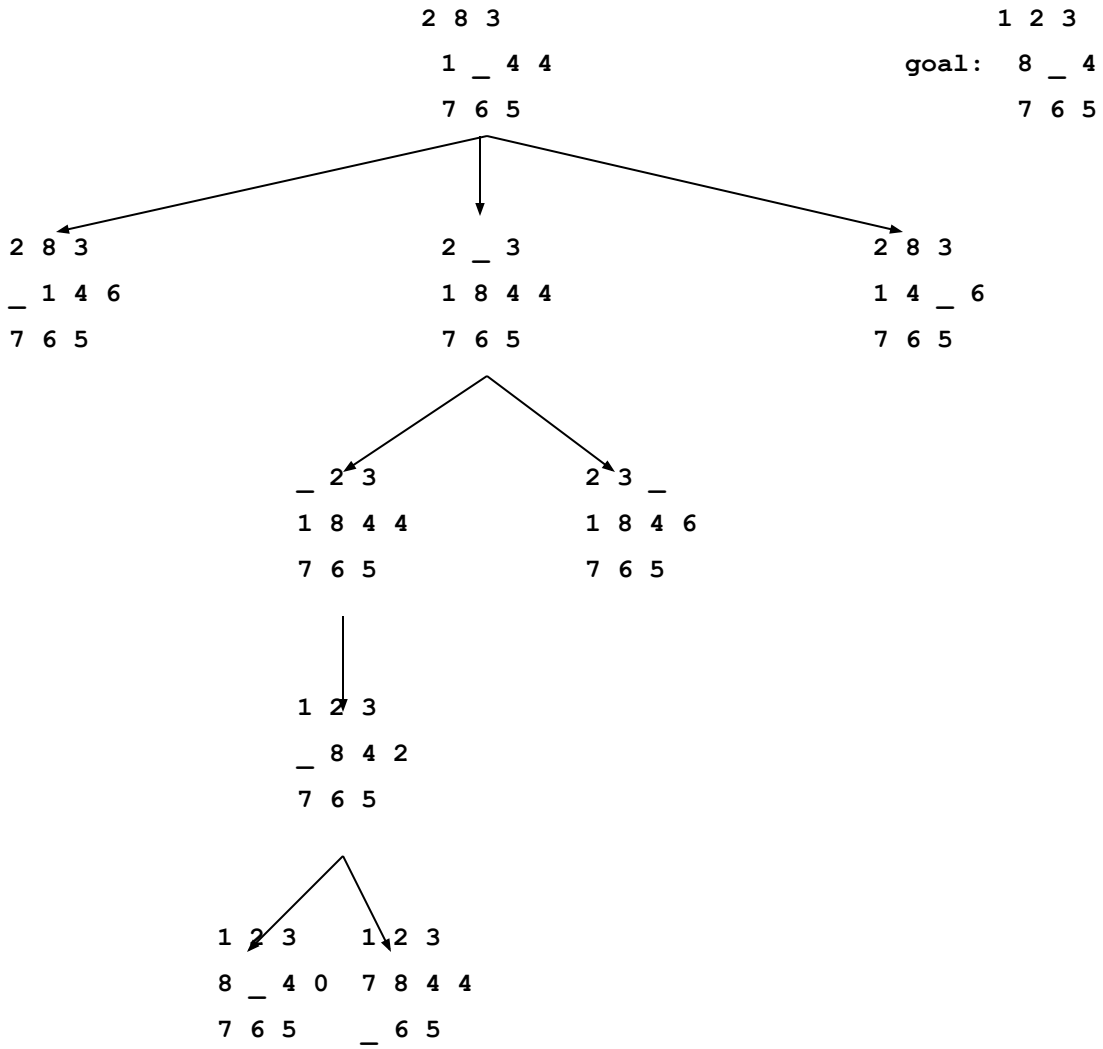
Best-first search - Manhattan distance



Best-first search - Manhattan distance



Best-first search - Manhattan distance



Quantifying goodness

You've now seen a couple of real examples of attempts to quantify the nearness of a state to the goal state - the goodness of the state. Crafting these heuristics is a skill that takes practice, and even with lots of practice your heuristics will still be wrong some of the time.

But it's something to think about. Ponder some heuristics for other puzzles you might be familiar with. Your opportunity to put heuristics in your Haskell programs is coming up real soon.

Questions?

The moral of the story so far...

We can write an evaluation function that, when applied to some state, uses knowledge about the problem domain to calculate a quantitative value that represents an estimate of that state's nearness to a goal.

That quantitative value can then be used to answer that question: Now what do I do?

Search in the real world

The sort of heuristic state-space search we've seen only gets us so far in the real world, because the real world can be a hostile place.

Consequently, we often find ourselves in situations where we're trying to find the path to our goal while somebody else is trying to prevent us from getting there.

Search in the real world

Real examples include

- the world of commerce
- the athletic field
- the battlefield
- organic chem lab when pre-med students are enrolled

Search in the real world

Real examples include

- the world of commerce
- the athletic field
- the battlefield
- organic chem lab when pre-med students are enrolled
- the simplest example is the two-player game, which leads us to...

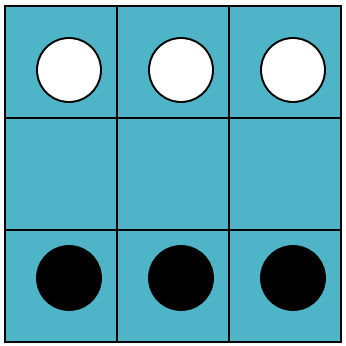
The Joy of Hex

The game of hexapawn

The Joy of Hex

The game of hexapawn

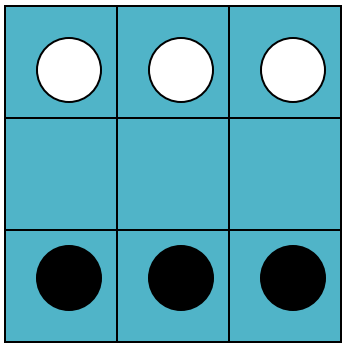
- 3 x 3 board
- 3 pawns on each side



The Joy of Hex

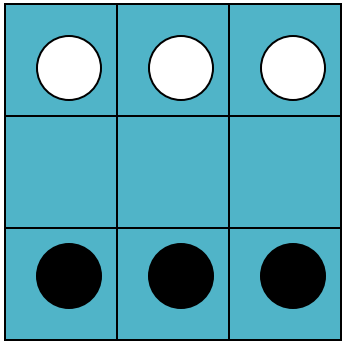
The game of hexapawn

- 3 x 3 board
- 3 pawns on each side
- movement of pawns:



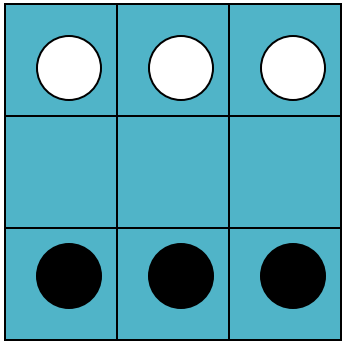
The Joy of Hex

The game of hexapawn



- 3 x 3 board
- 3 pawns on each side
- movement of pawns:
  - white moves first

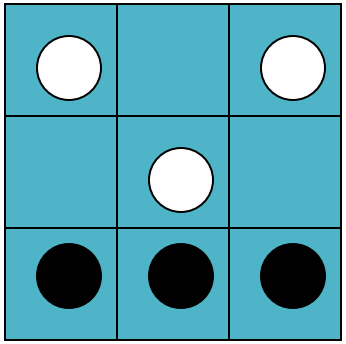
The Joy of Hex



The game of hexapawn

- 3 x 3 board
- 3 pawns on each side
- movement of pawns:
  - white moves first
  - pawn can move straight ahead one space if that space is empty

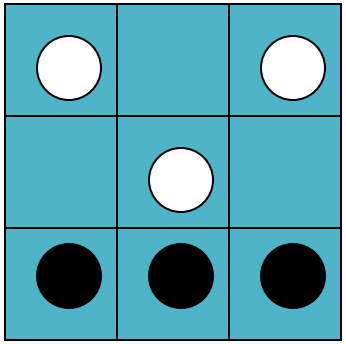
The Joy of Hex



The game of hexapawn

- 3 x 3 board
- 3 pawns on each side
- movement of pawns:
  - white moves first
  - pawn can move straight ahead one space if that space is empty

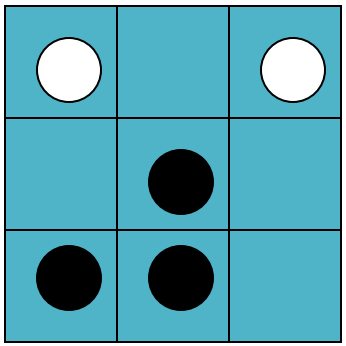
The Joy of Hex



The game of hexapawn

- 3 x 3 board
- 3 pawns on each side
- movement of pawns:
  - white moves first
  - pawn can move straight ahead one space if that space is empty
  - pawn can move diagonally one space forward to capture opponent's pawn occupying that space

The Joy of Hex

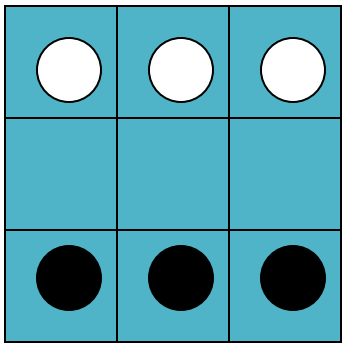


The game of hexapawn

- 3 x 3 board
- 3 pawns on each side
- movement of pawns:
  - white moves first
  - pawn can move straight ahead one space if that space is empty
  - pawn can move diagonally one space forward to capture opponent's pawn occupying that space

The Joy of Hex

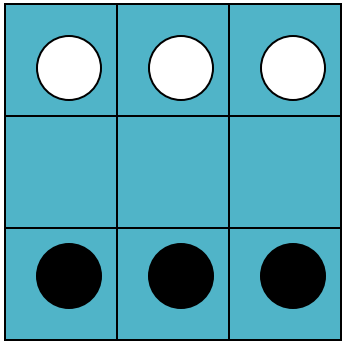
The game of hexapawn



- 3 ways to win:

The Joy of Hex

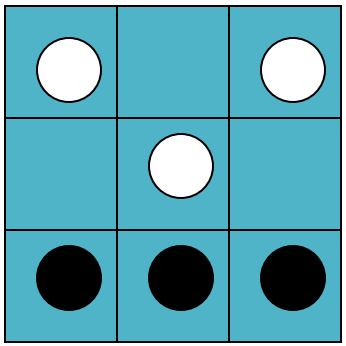
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns

The Joy of Hex

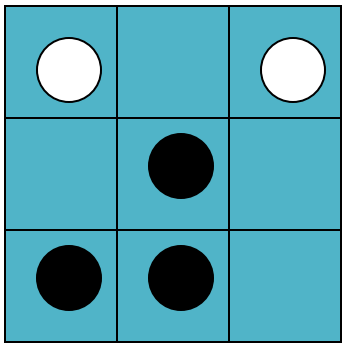
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns

The Joy of Hex

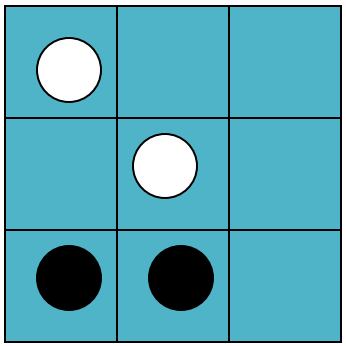
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns

The Joy of Hex

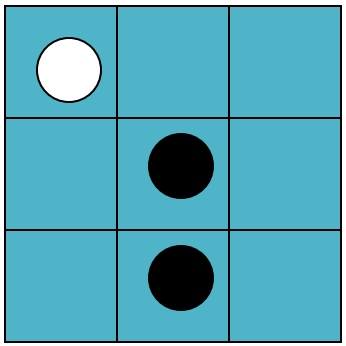
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns

The Joy of Hex

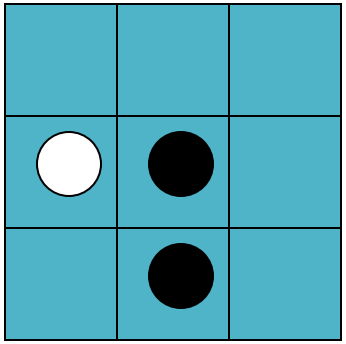
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns

The Joy of Hex

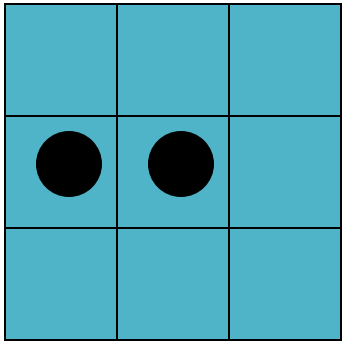
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns

The Joy of Hex

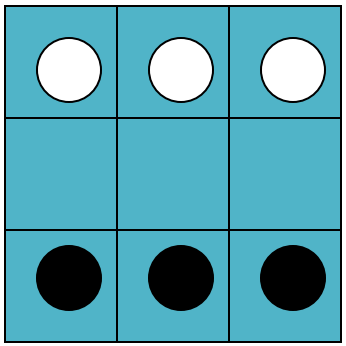
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns

The Joy of Hex

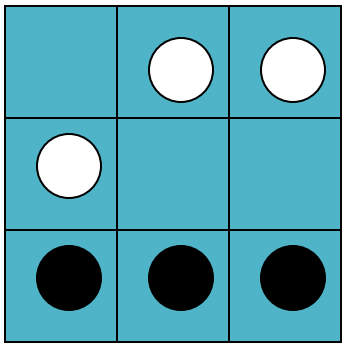
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board

The Joy of Hex

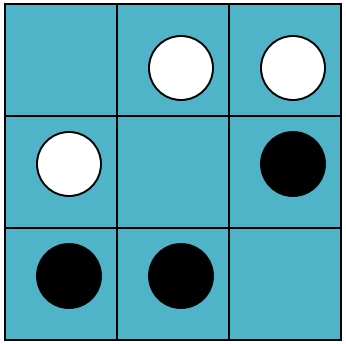
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board

The Joy of Hex

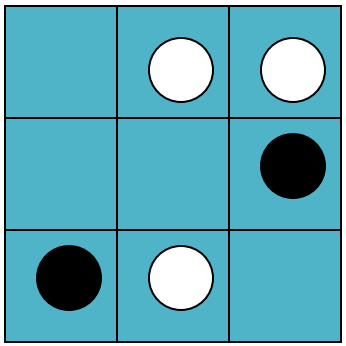
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board

The Joy of Hex

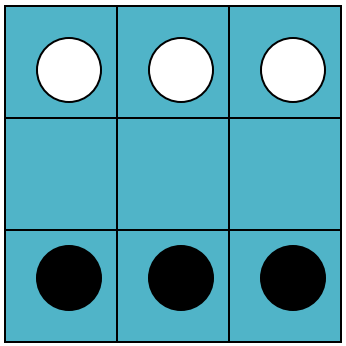
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board

The Joy of Hex

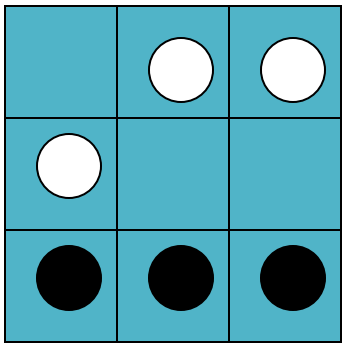
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board
  - it's your opponent's turn but your opponent can't move

The Joy of Hex

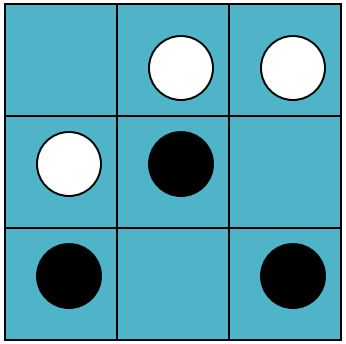
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board
  - it's your opponent's turn but your opponent can't move

The Joy of Hex

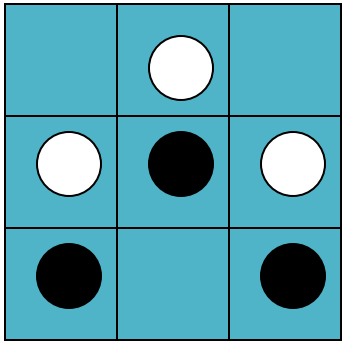
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board
  - it's your opponent's turn but your opponent can't move

The Joy of Hex

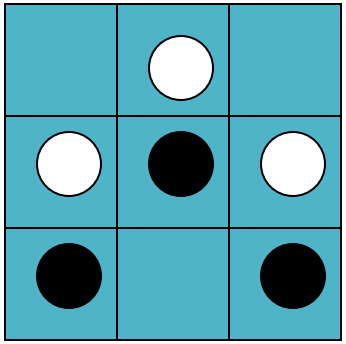
The game of hexapawn



- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board
  - it's your opponent's turn but your opponent can't move

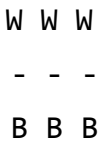
The Joy of Hex

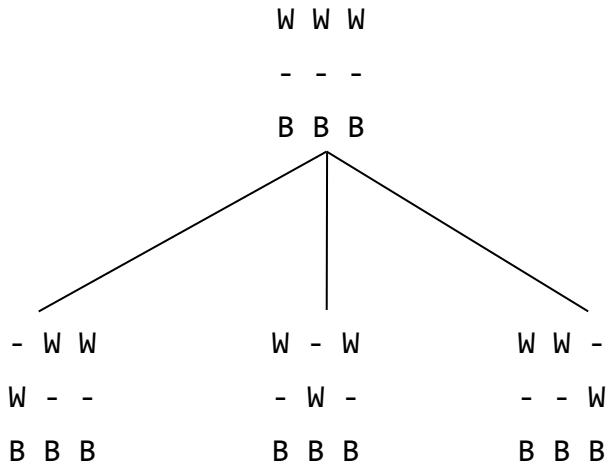
The game of hexapawn

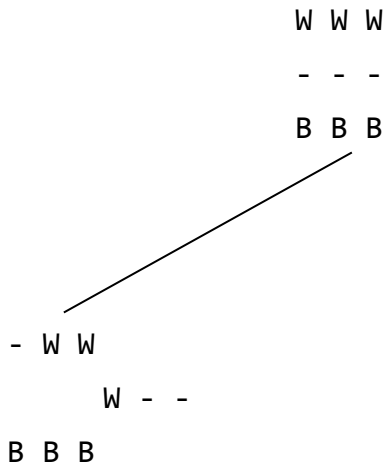


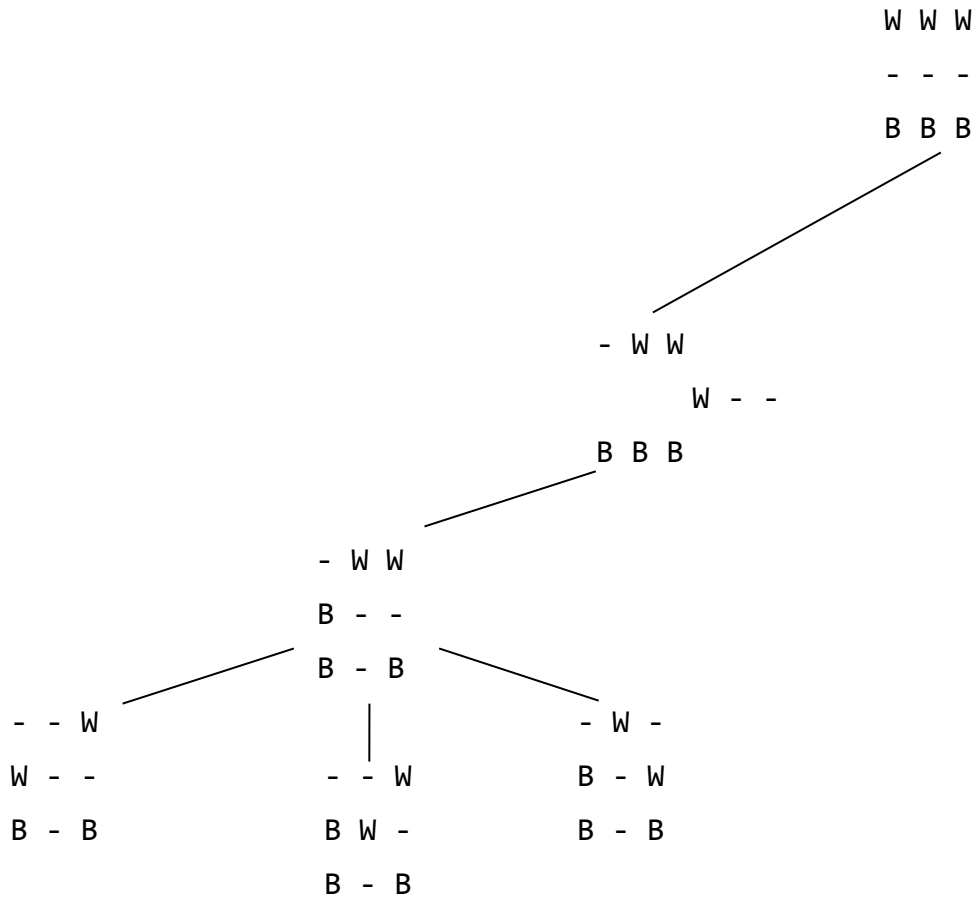
- 3 ways to win:
  - capture all your opponent's pawns
  - one of your pawns reaches the opposite end of the board
  - it's your opponent's turn but your opponent can't move

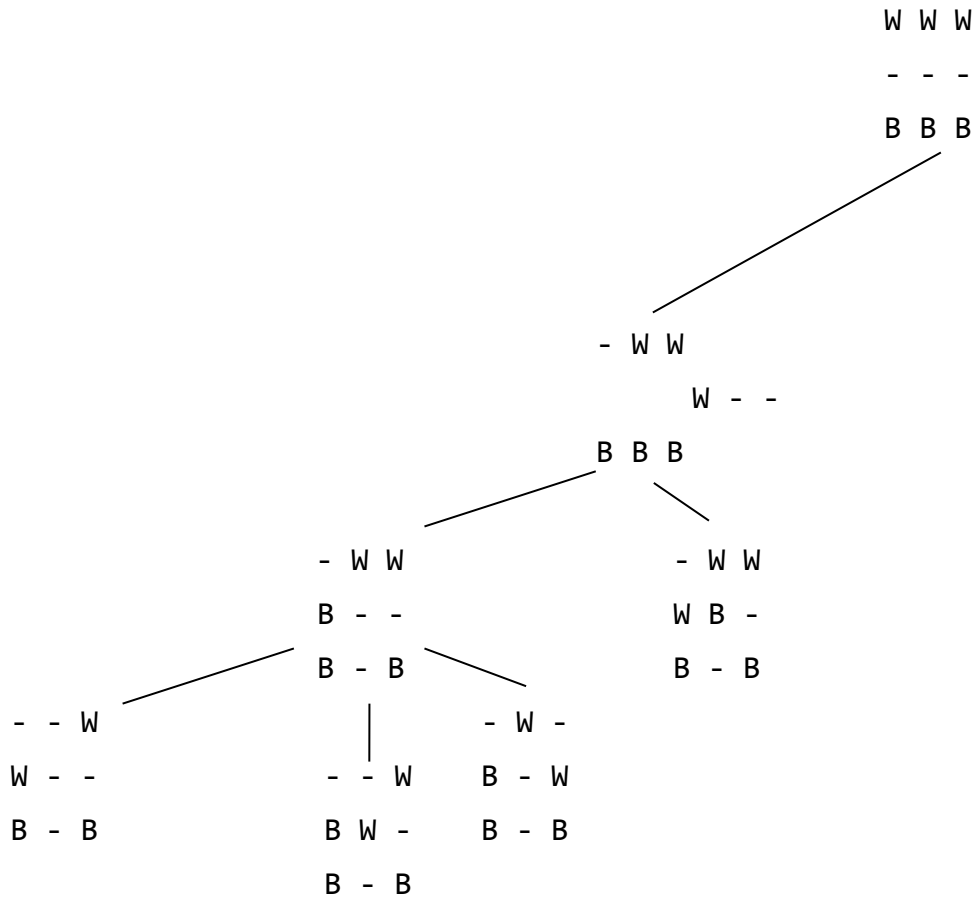
Now let's look at the search tree...
(we're pushing the black pawns)

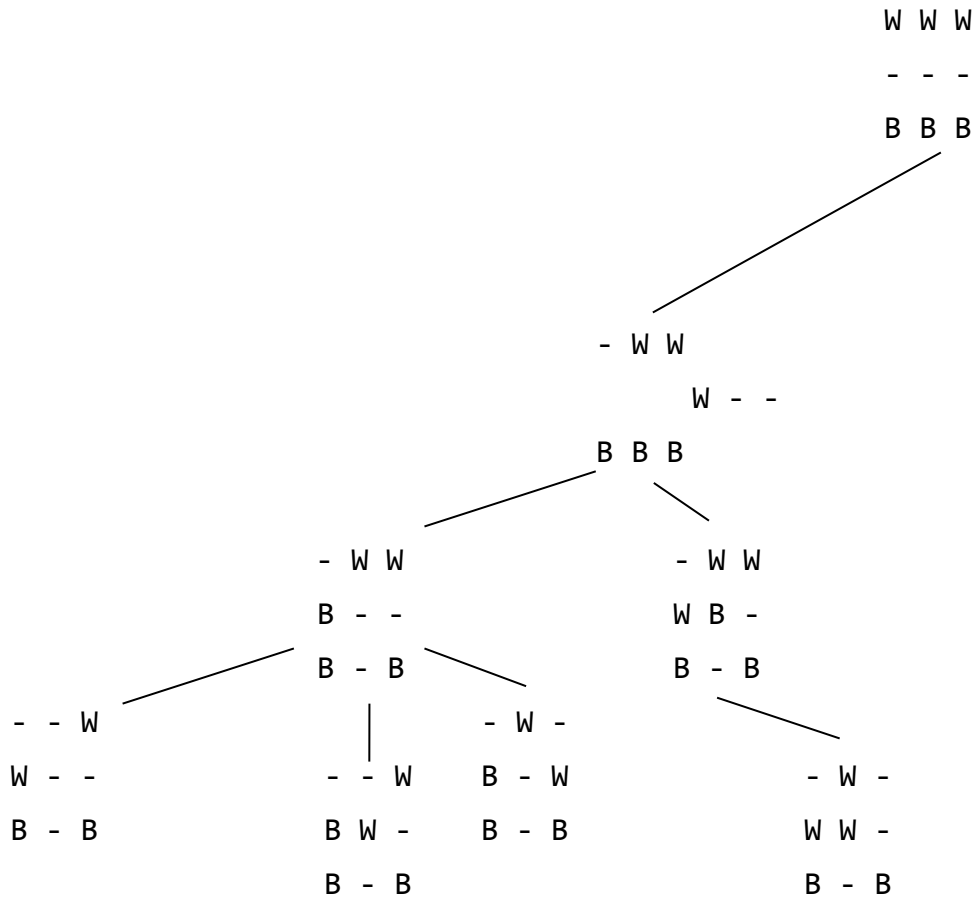

W W W
- - -
B B B

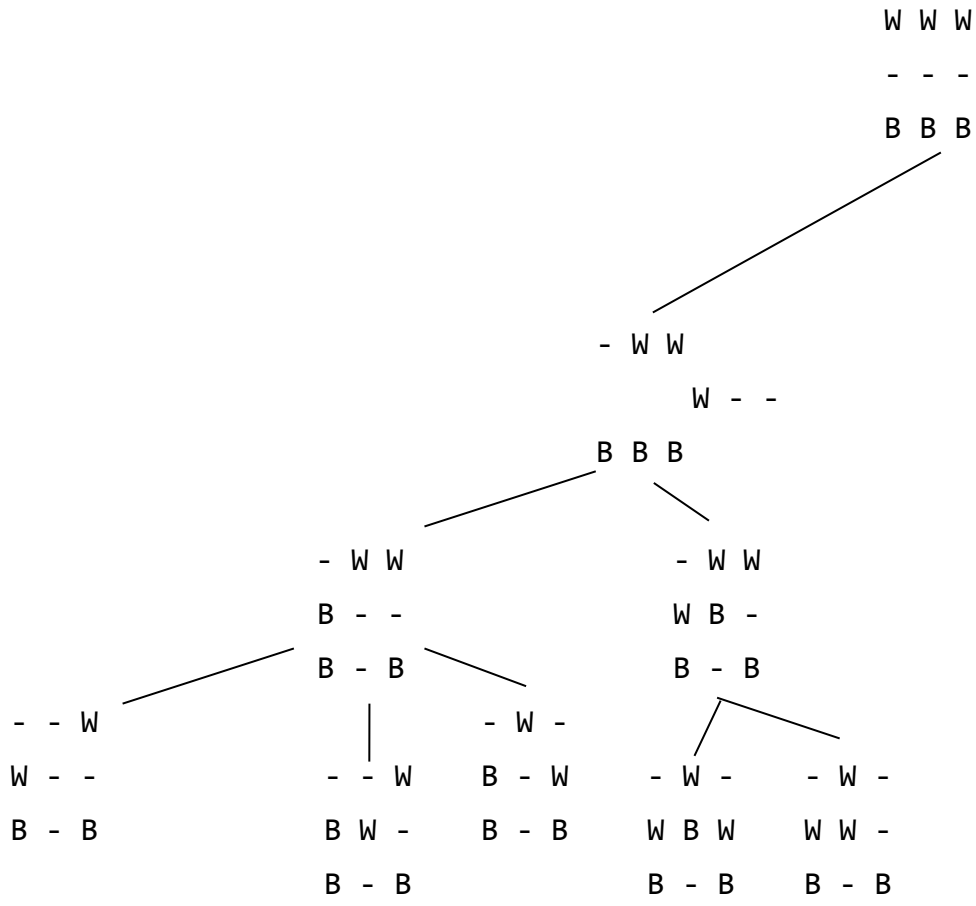

W W W
- - -
B B B
- W W
W - -
B B B
W - W
- W -
B B B
W W -
- - W
B B B

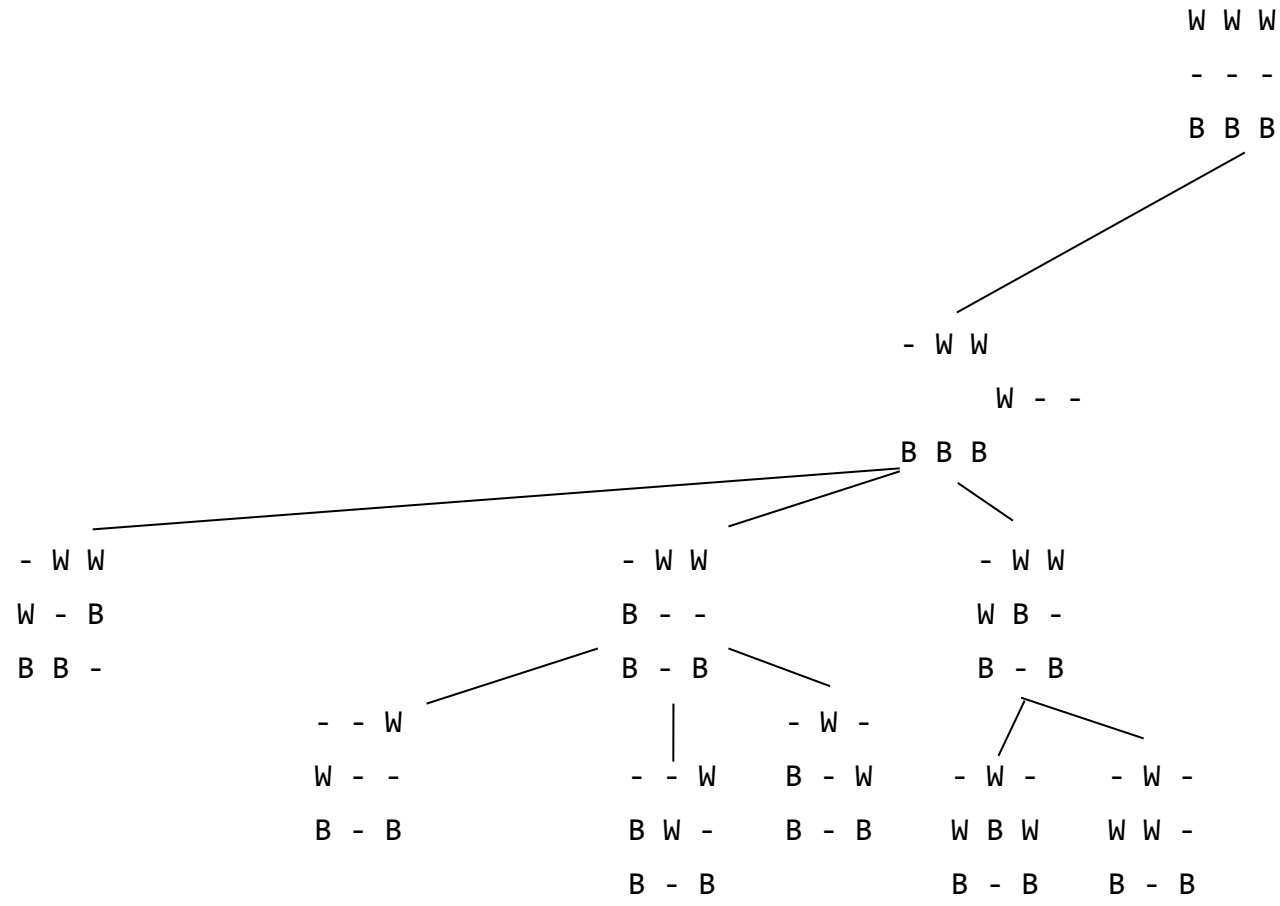

W W W
- - -
B B B
- W W
W - -
B B B

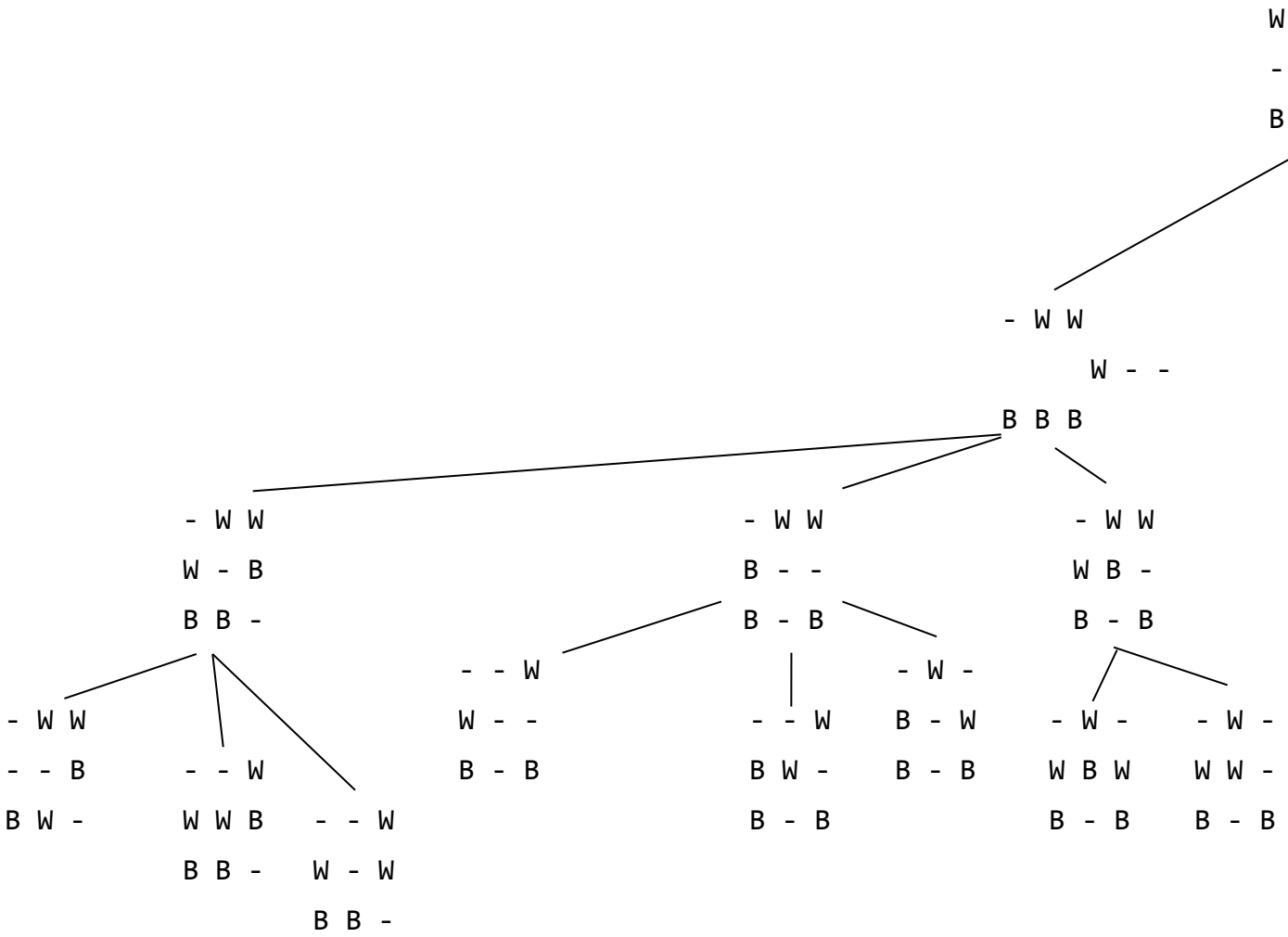

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B

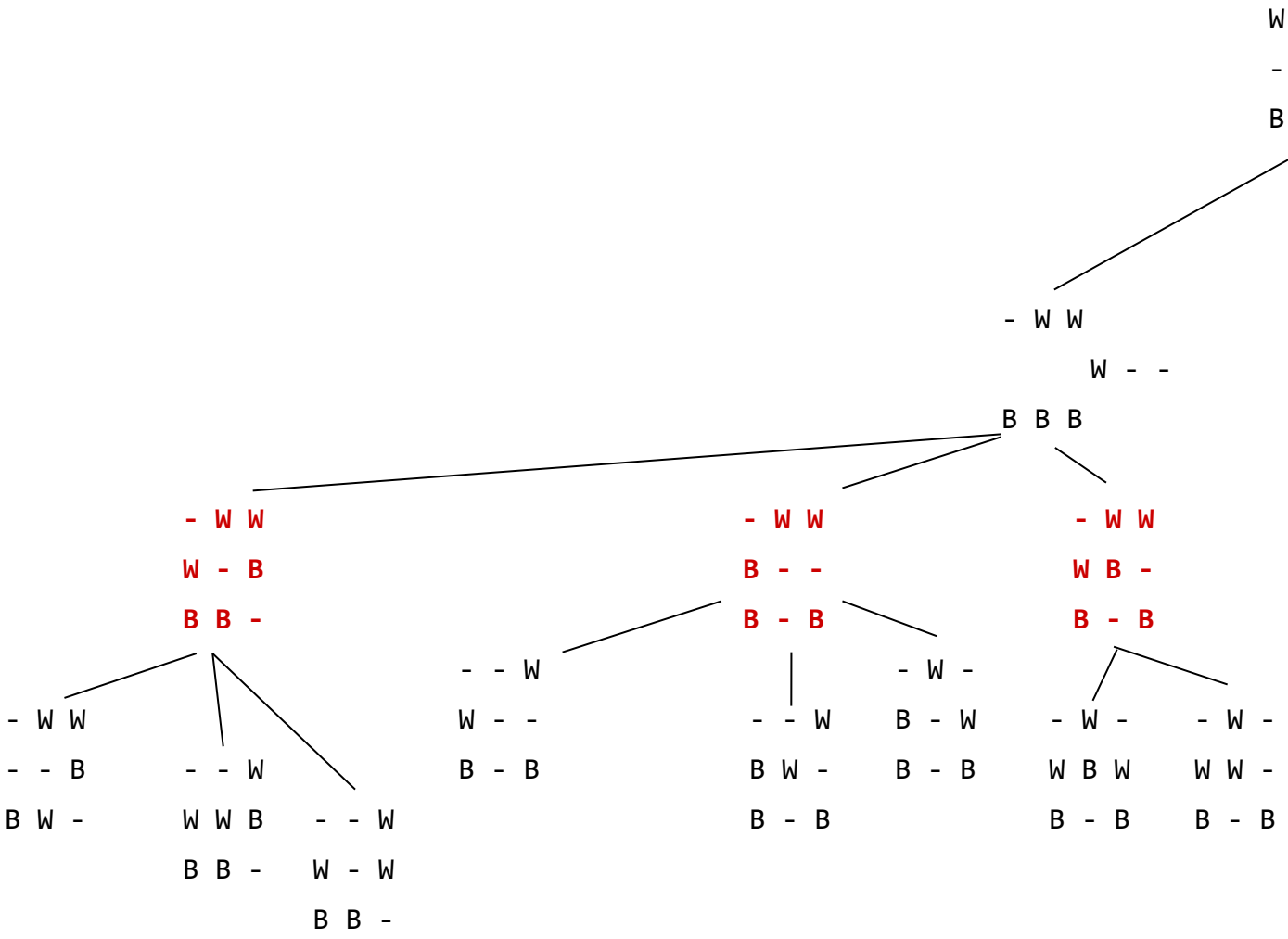

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B


W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B


W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W B W
B - B
- W -
W W -
B - B


W W W
- - -
B B B
- W W
W - -
B B B
- W W
W - B
B B -
- W W
B - -
B - B
- W W
W B -
B - B
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W B W
B - B
- W -
W W -
B - B


W W W
- - -
B B B
- W W
W - -
B B B
- W W
W - B
B B -
- W W
B - -
B - B
- W W
W B -
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
- - W
W - -
B - B
- W -
- - W
B - B
- W -
B - W
B - B
- W -
W B W
B - B
- W -
W W -
B - B



Do you have a sense of which of our possible moves we should make?

Our intuition says...

To figure out which move to make

look ahead some number of moves (search)

evaluate the game boards (quantify goodness)

make the move that leads to the most promising game board

Three Questions

First, how deep do you search?

Three Questions

First, how deep do you search?

As deep as you can within computational constraints:

- time
- memory
- space on powerpoint slide

The deeper the search, the more informed is your answer to the next question…

Three Questions

Second, how do you know which move to make?

Three Questions

Second, how do you know which move to make?

Use heuristic knowledge. In this case, we could apply this very crude (but not very effective) static board evaluation function:

if you have won then board value = +10
else if opponent has won then board value = -10
else board value = number of your pawns -
number of opponent's pawns

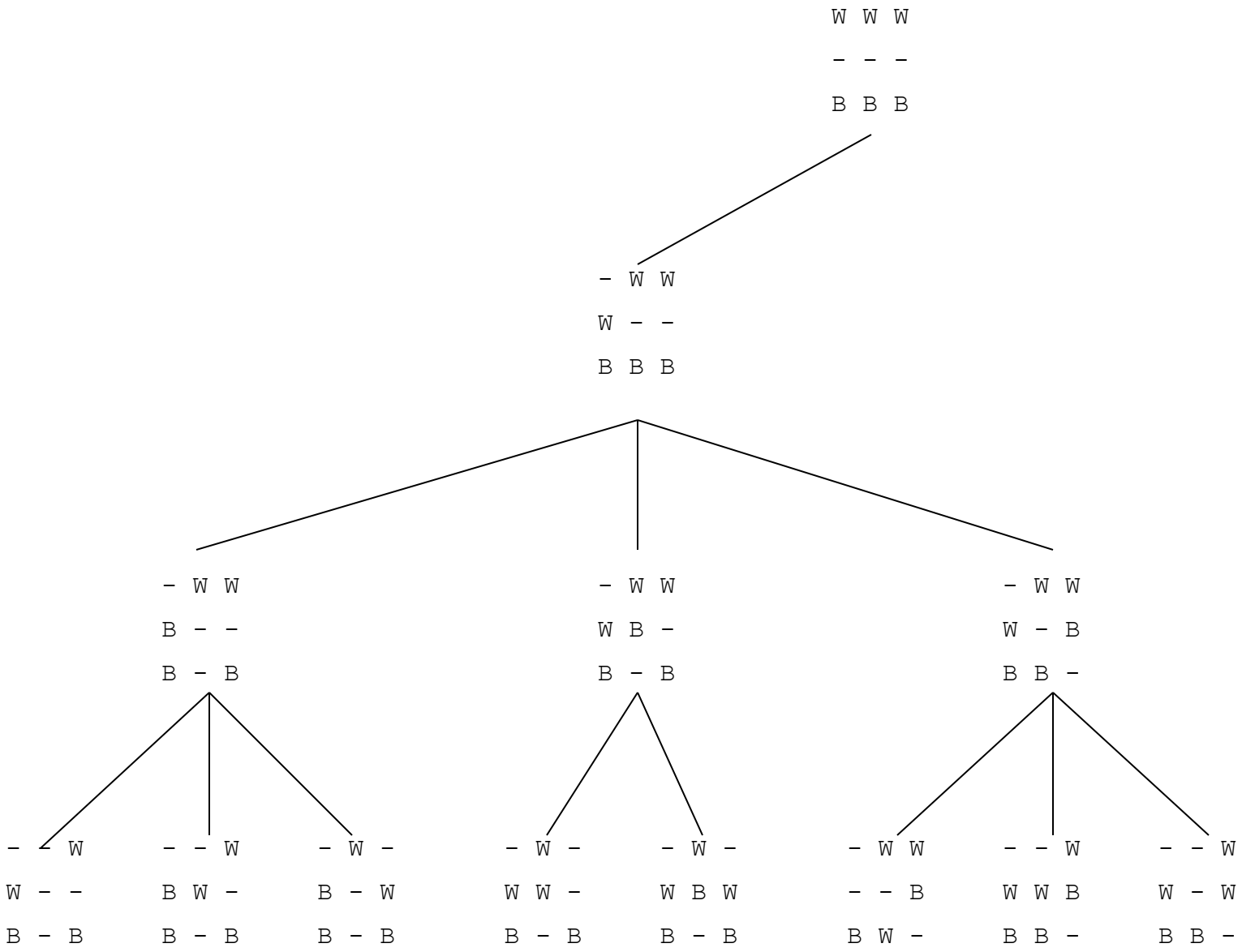
Three Questions

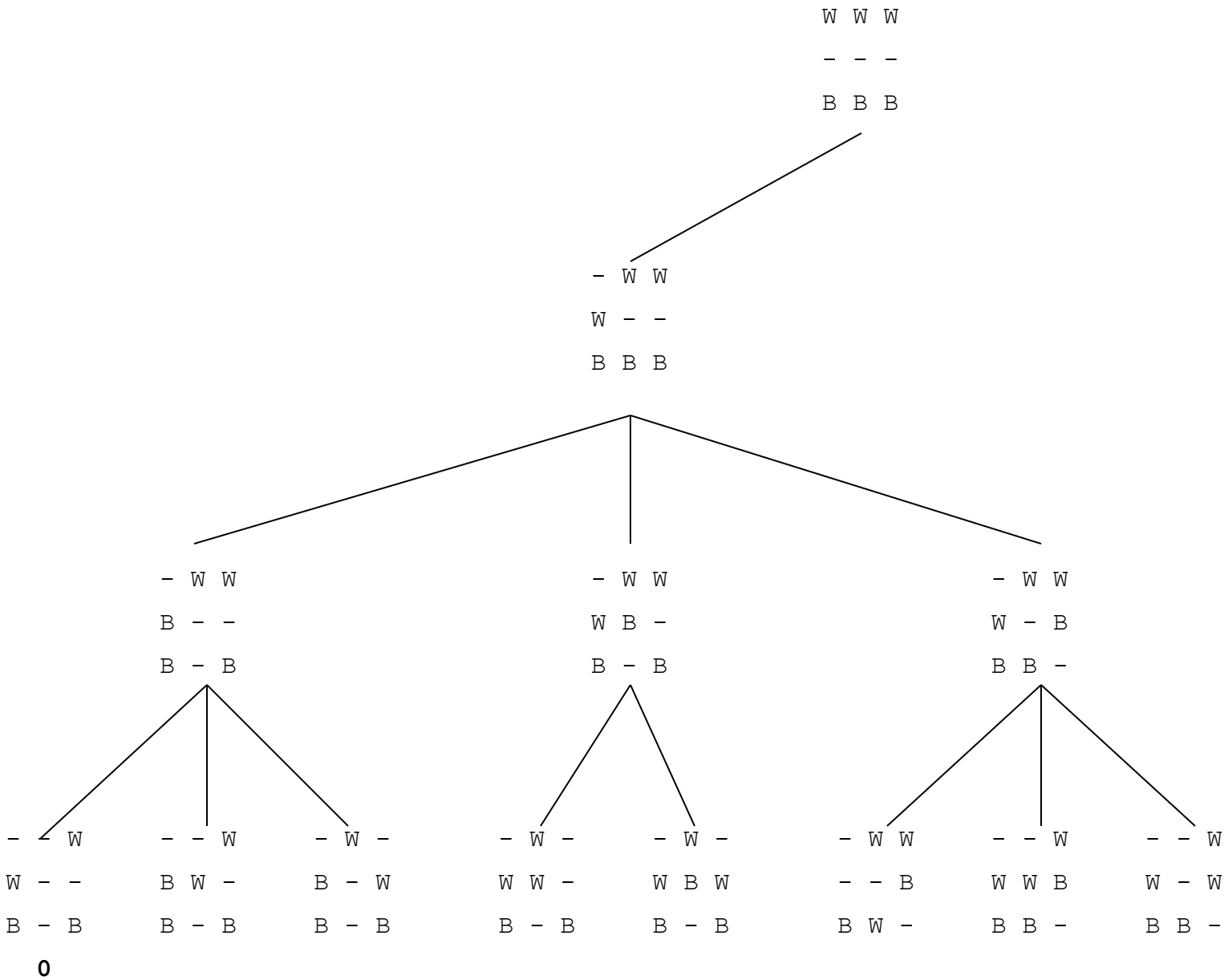
Second, how do you know which move to make?

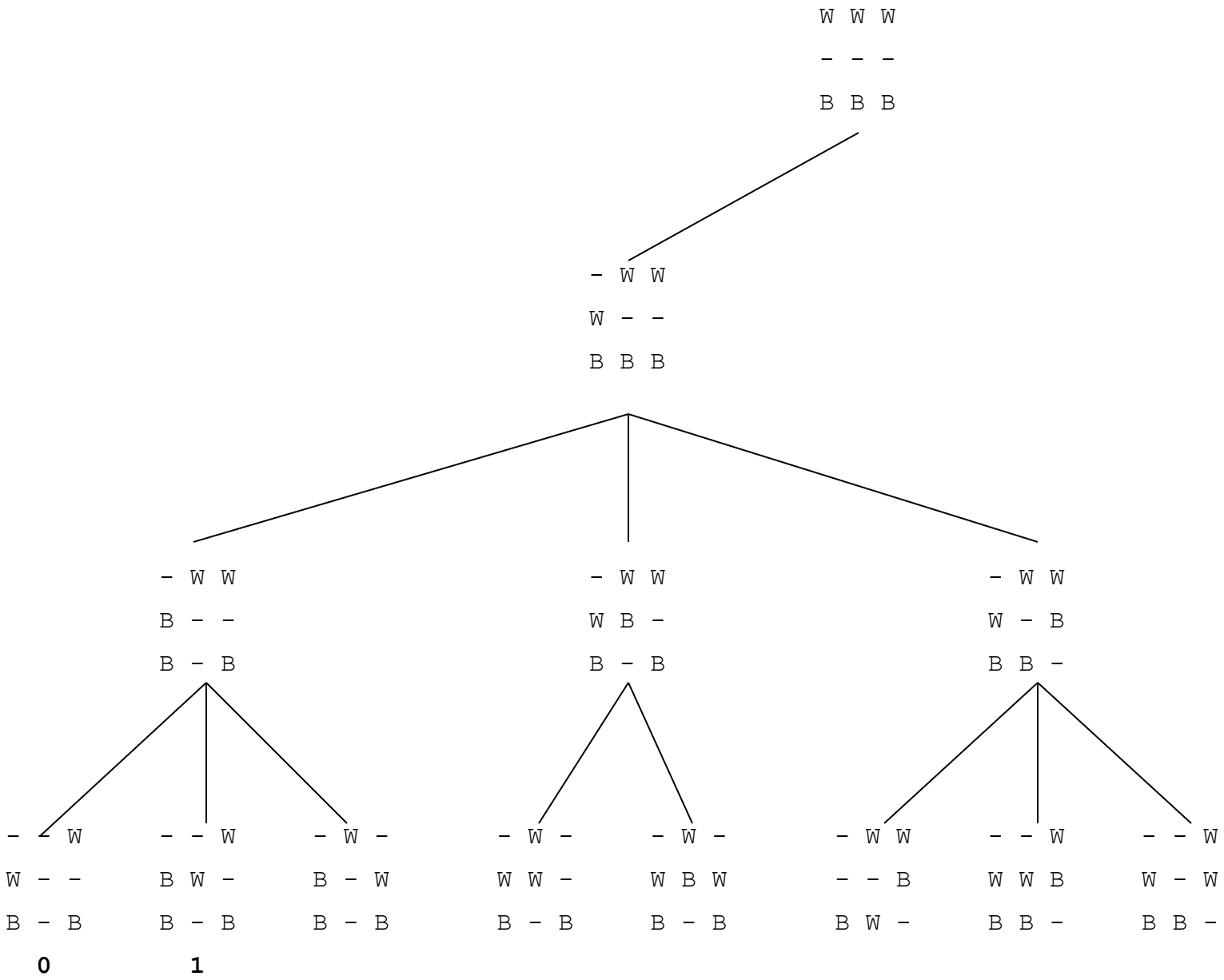
Use heuristic knowledge. In this case, we could apply this very crude (but not very effective) static board evaluation function:

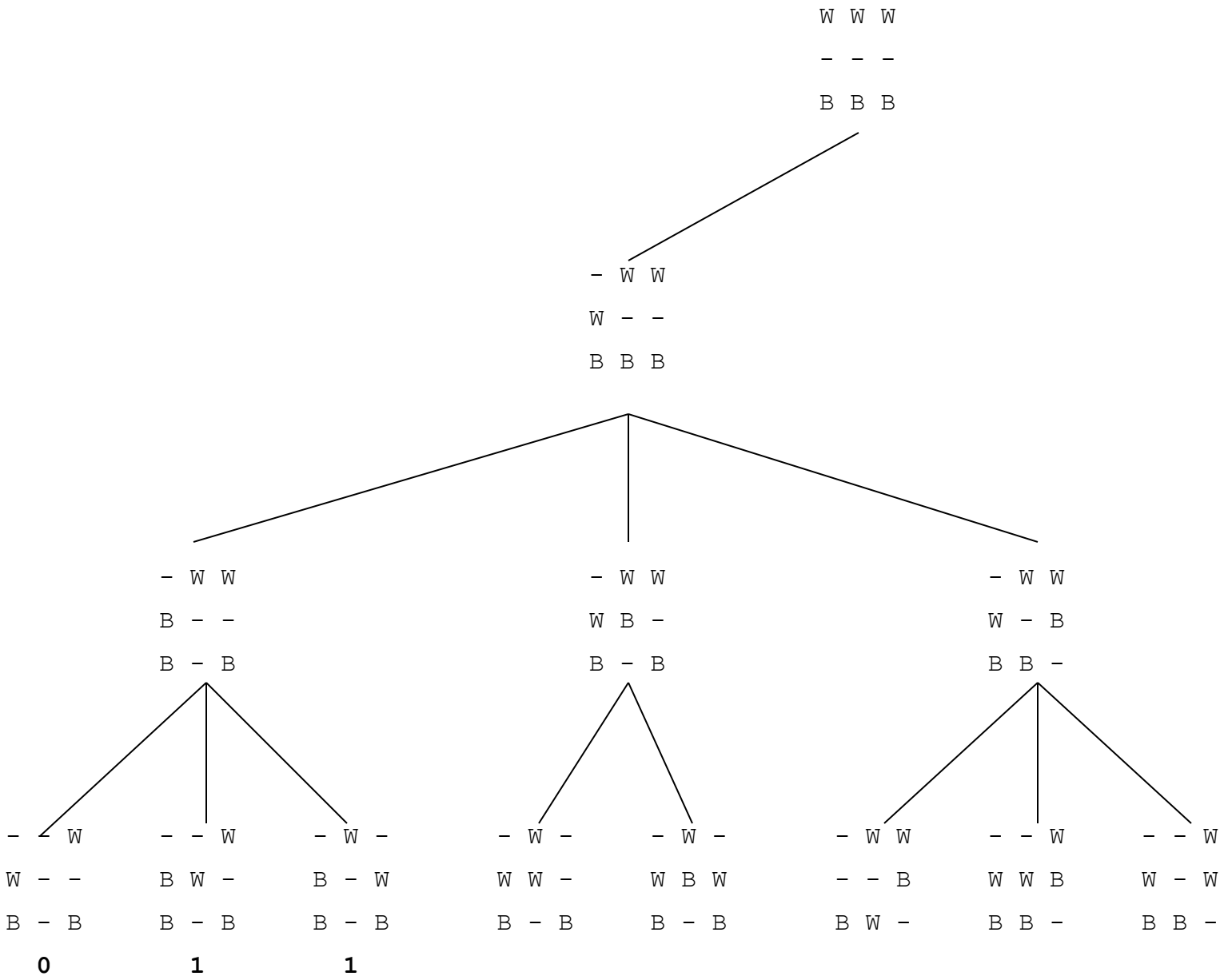
if you have won then board value = +10
else if opponent has won then board value = -10
else board value = number of your pawns -
number of opponent's pawns

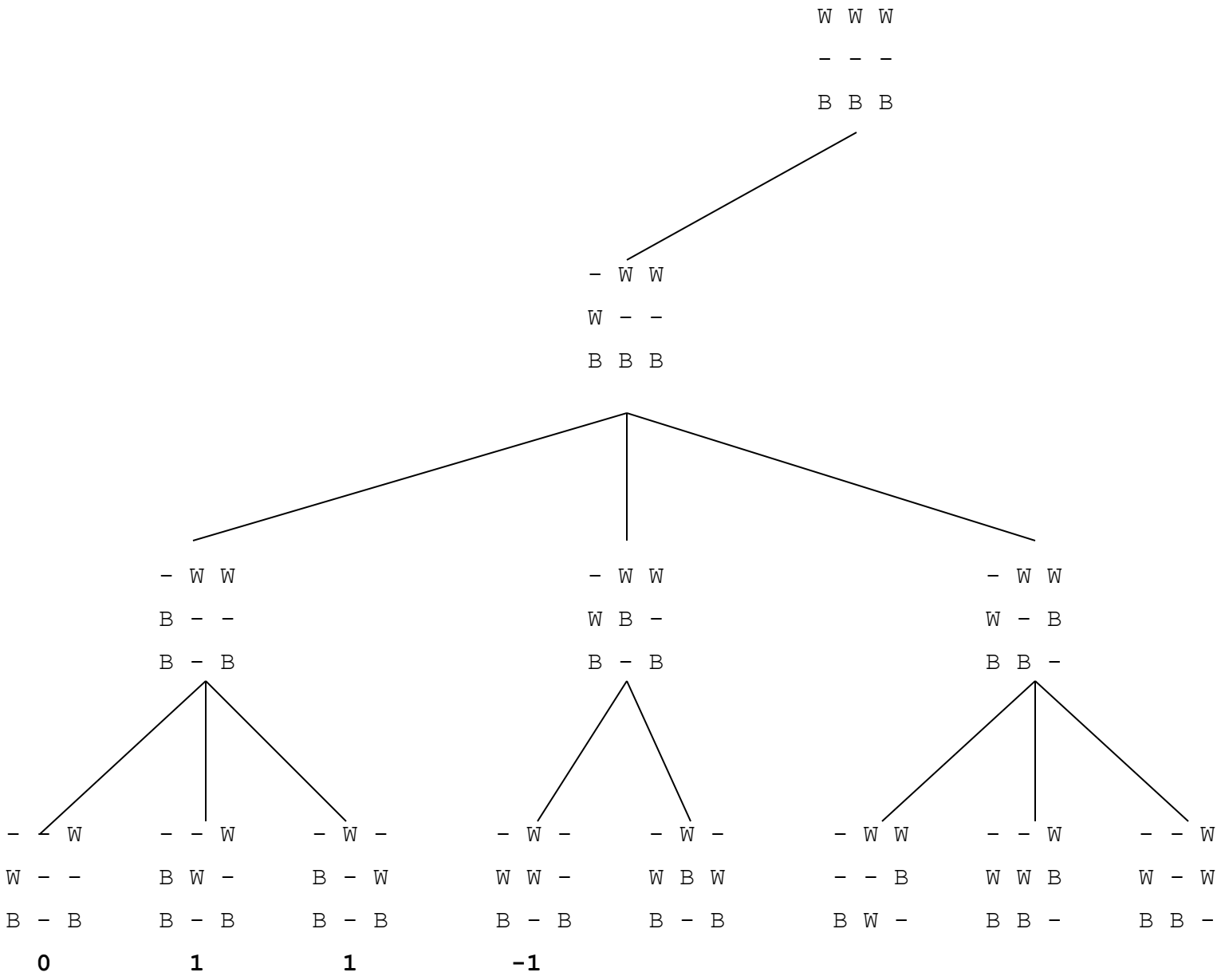
The board evaluation function is applied like this....

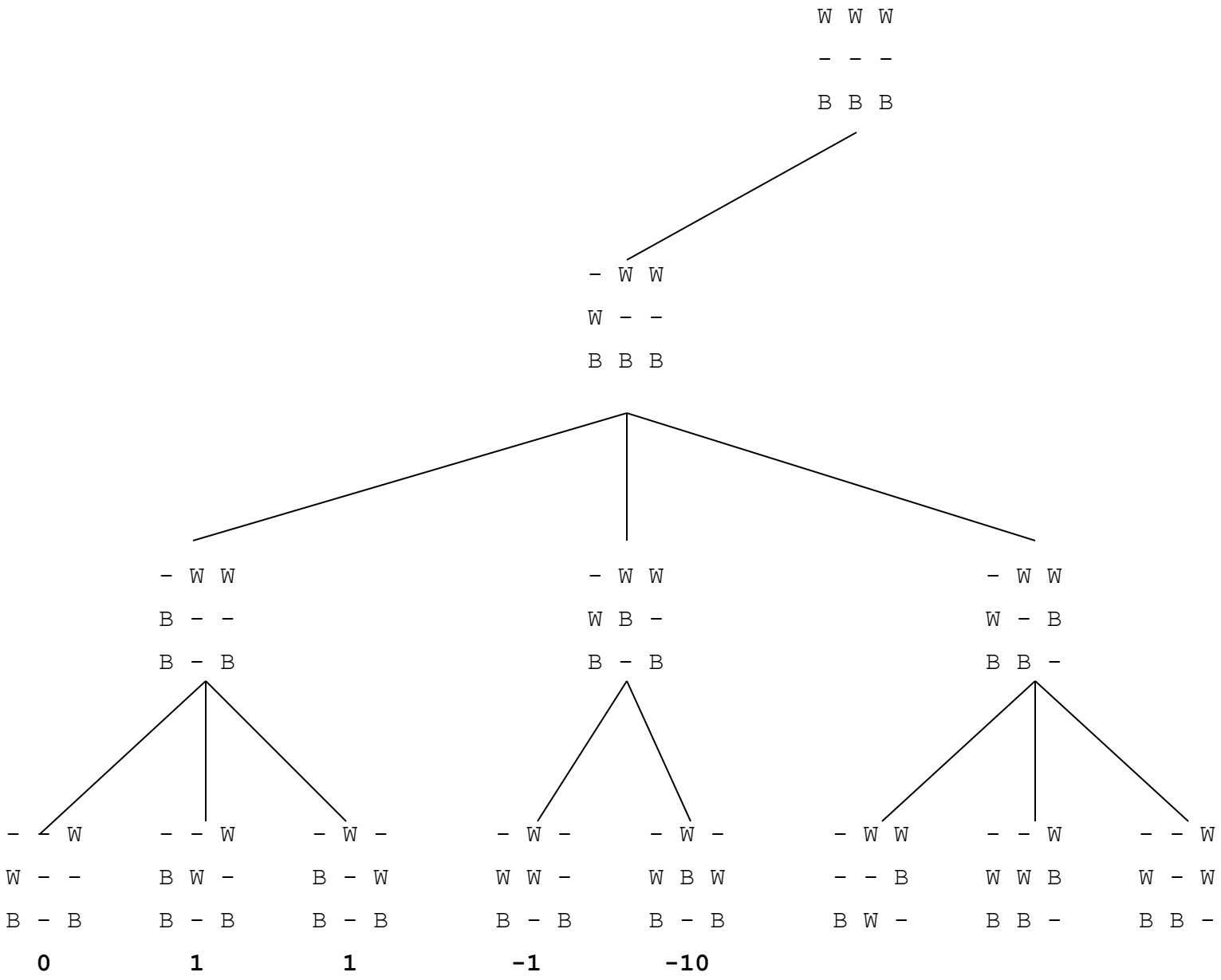

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -

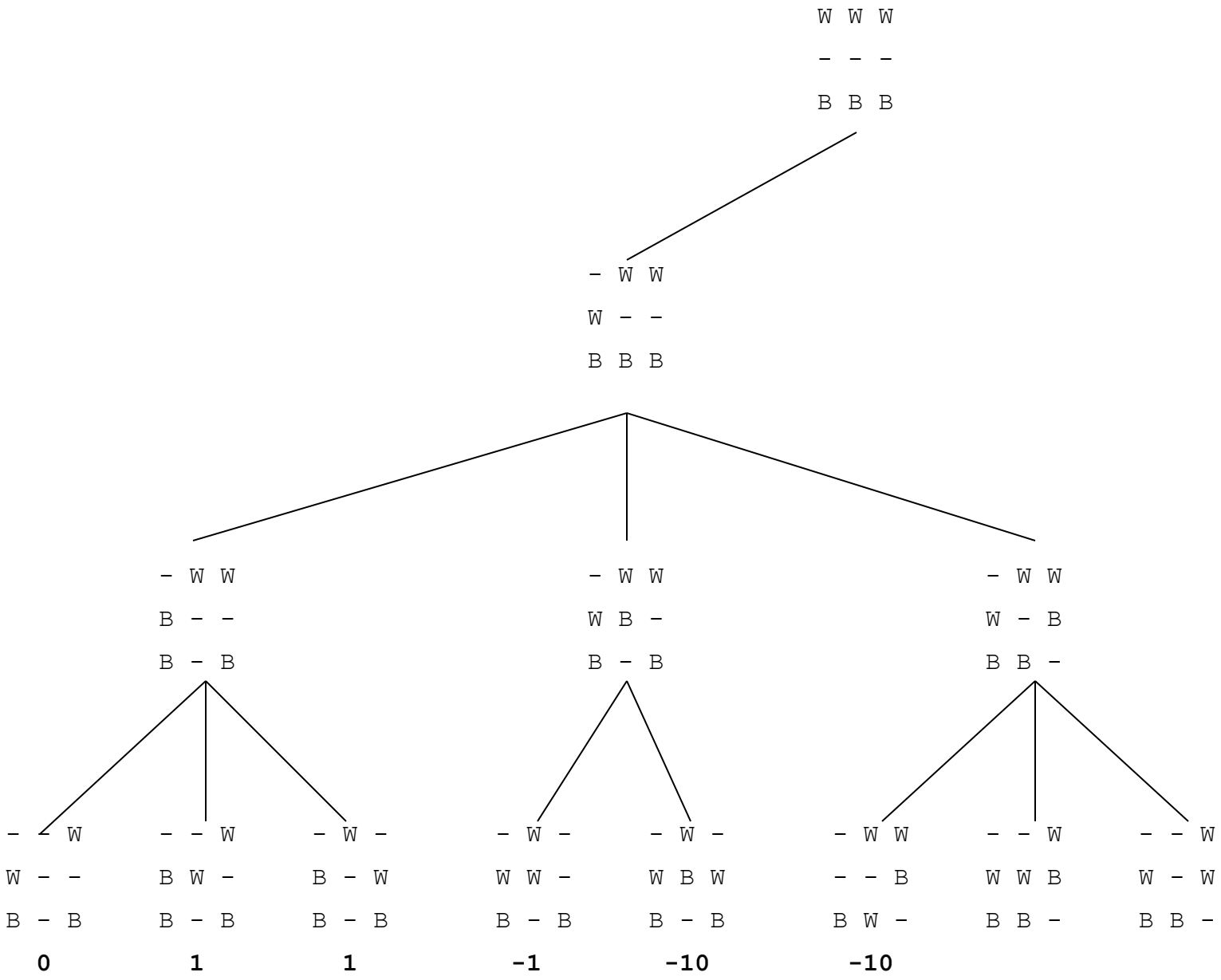

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0

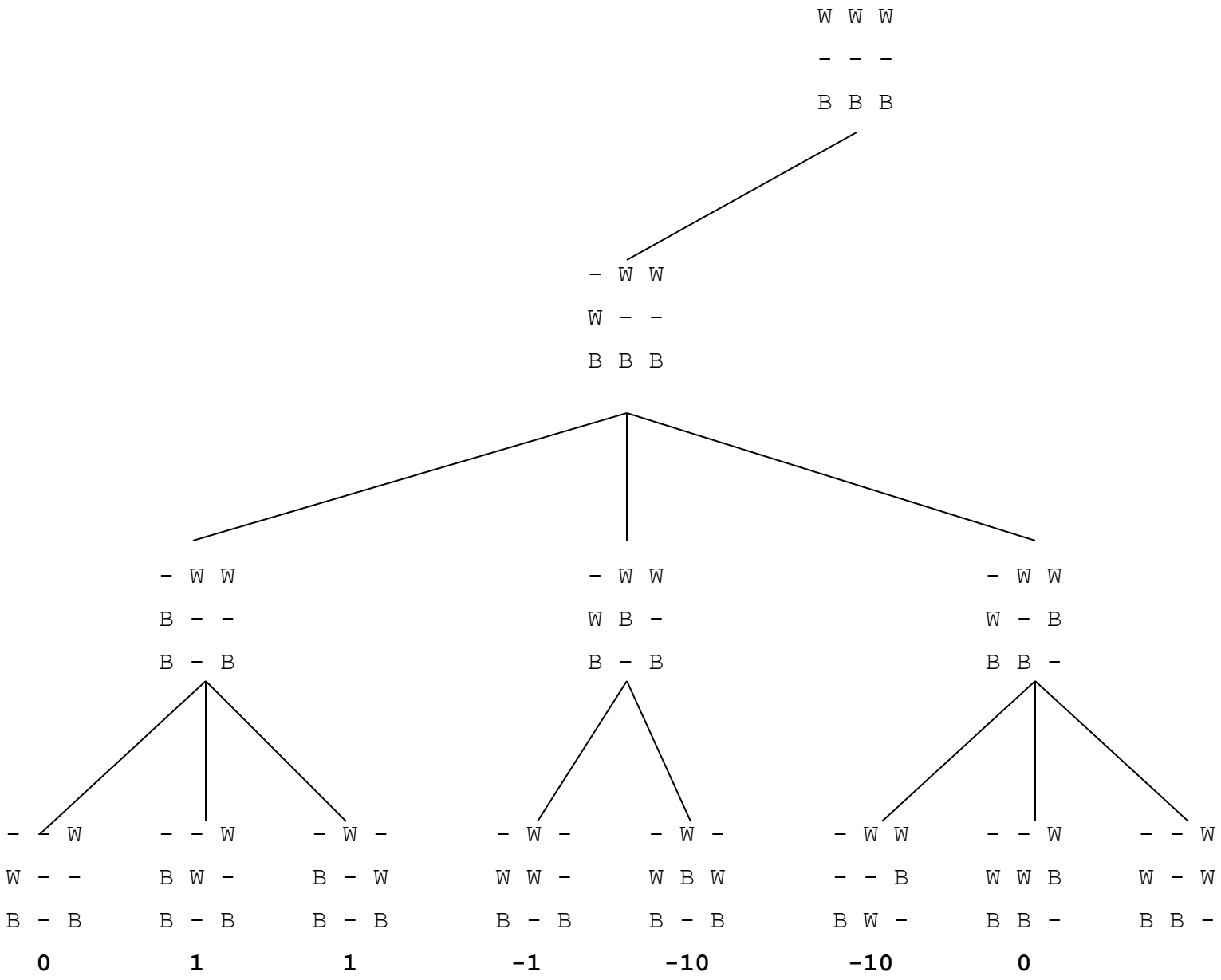

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1

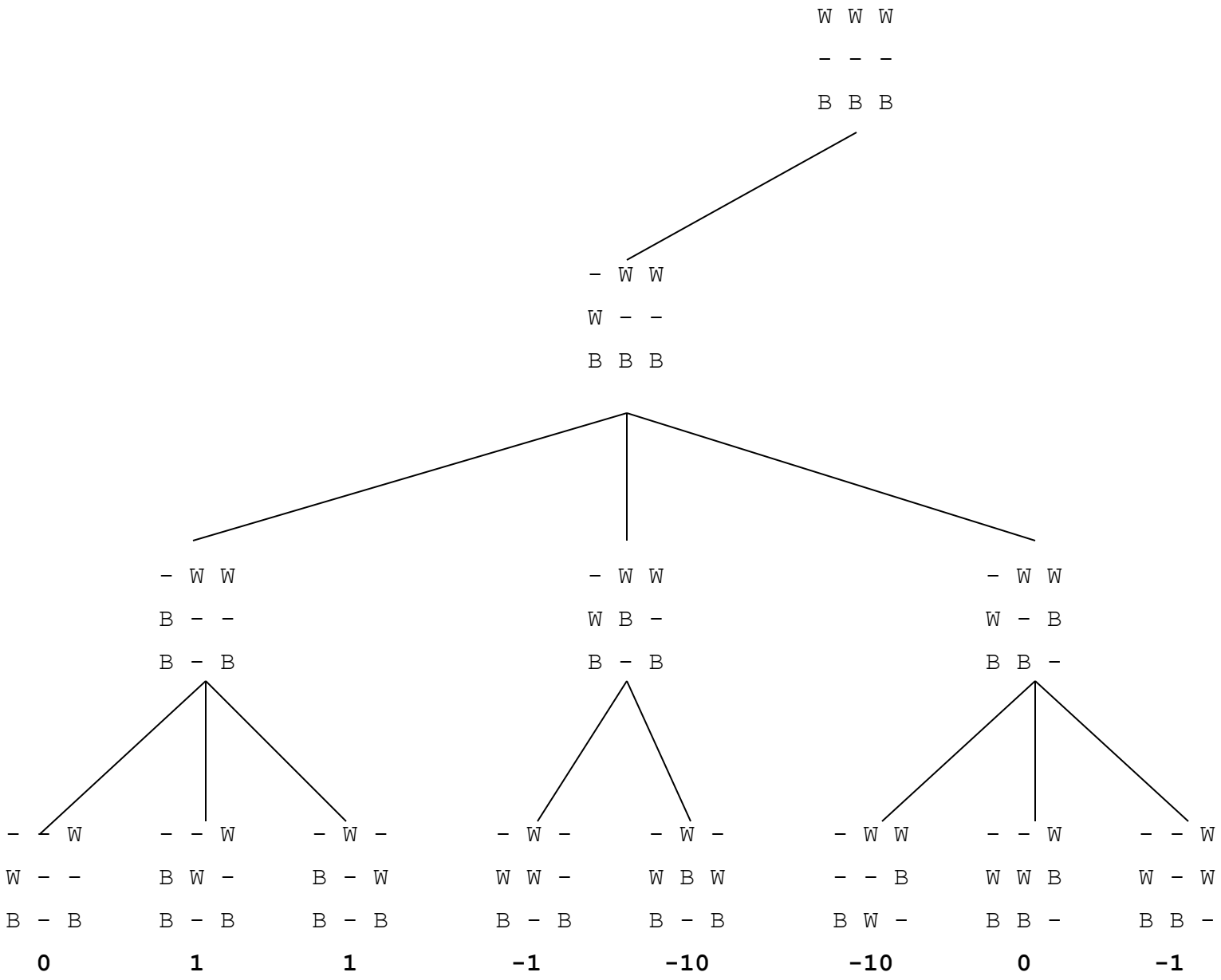

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1

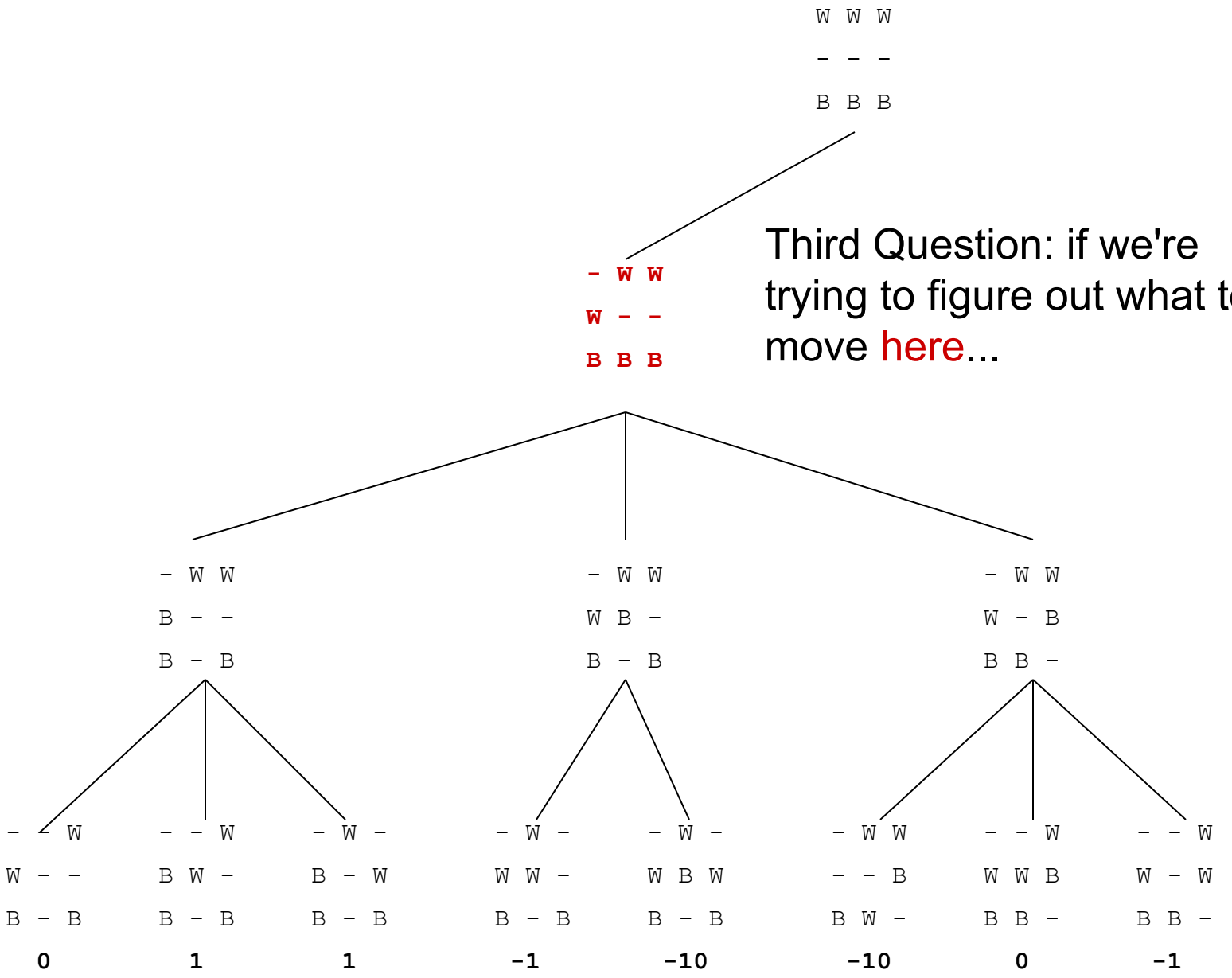

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1

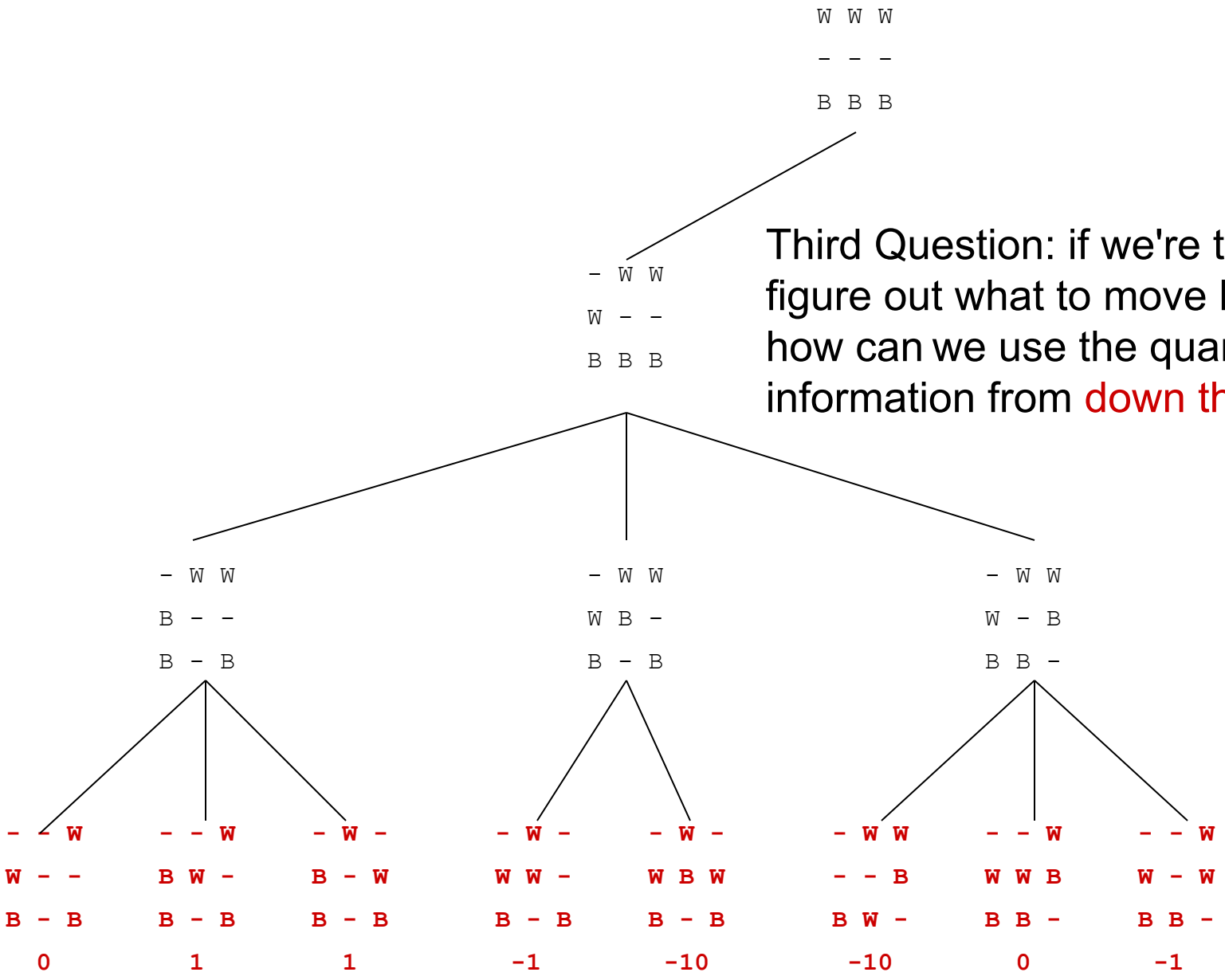

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10


W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10

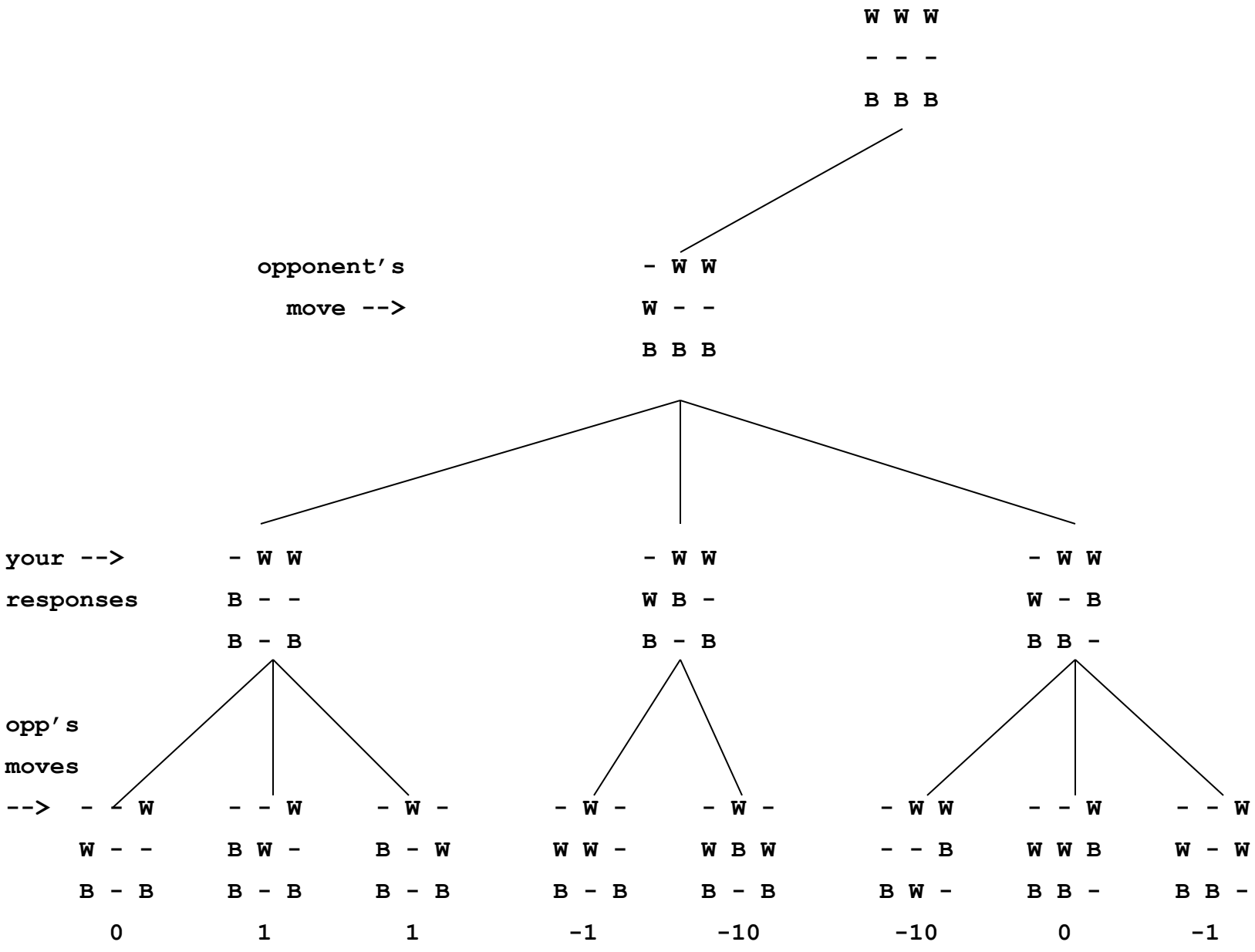

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10
0

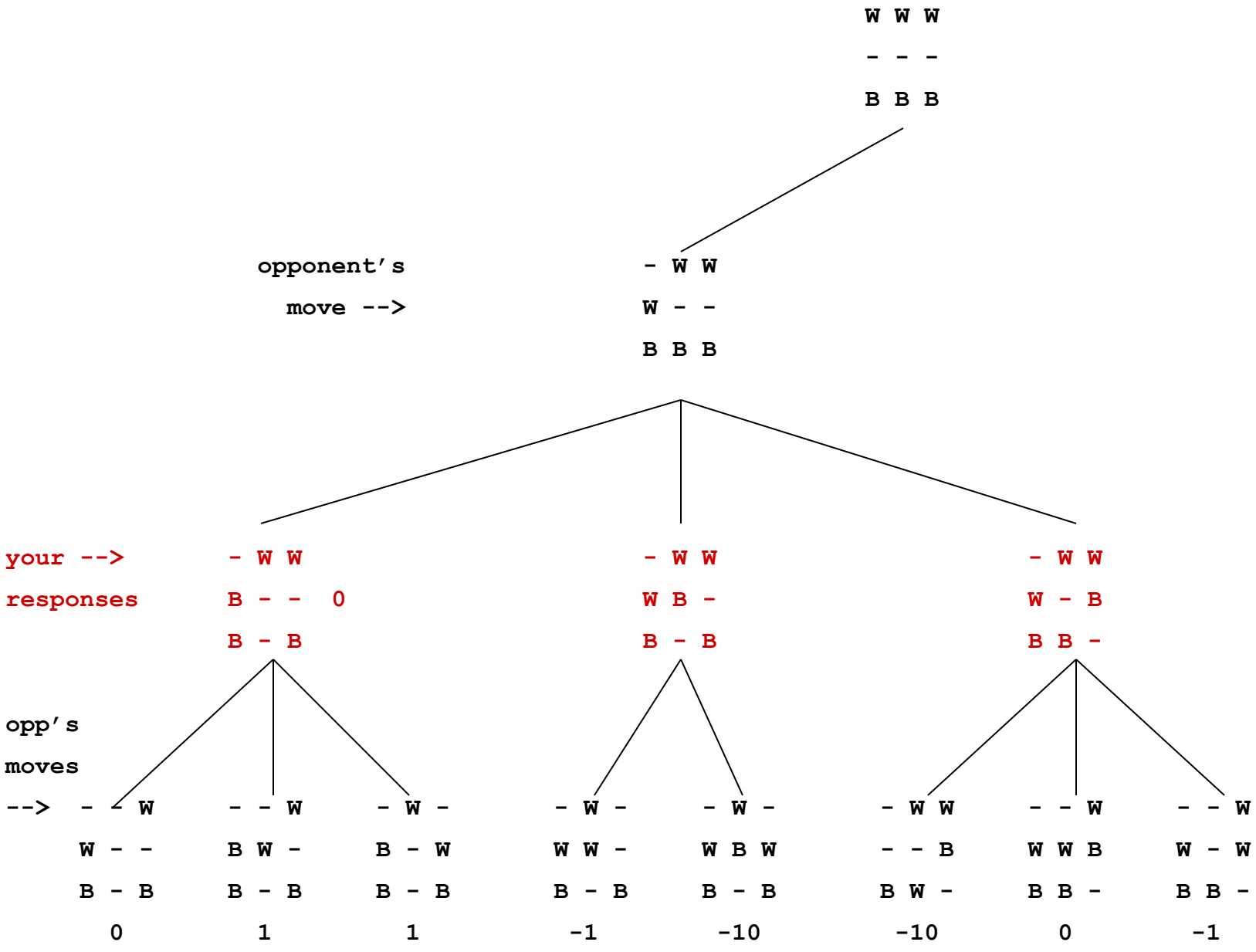

W W W
- - -
B B B
- W W
W - -
B B B
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10
0
-1

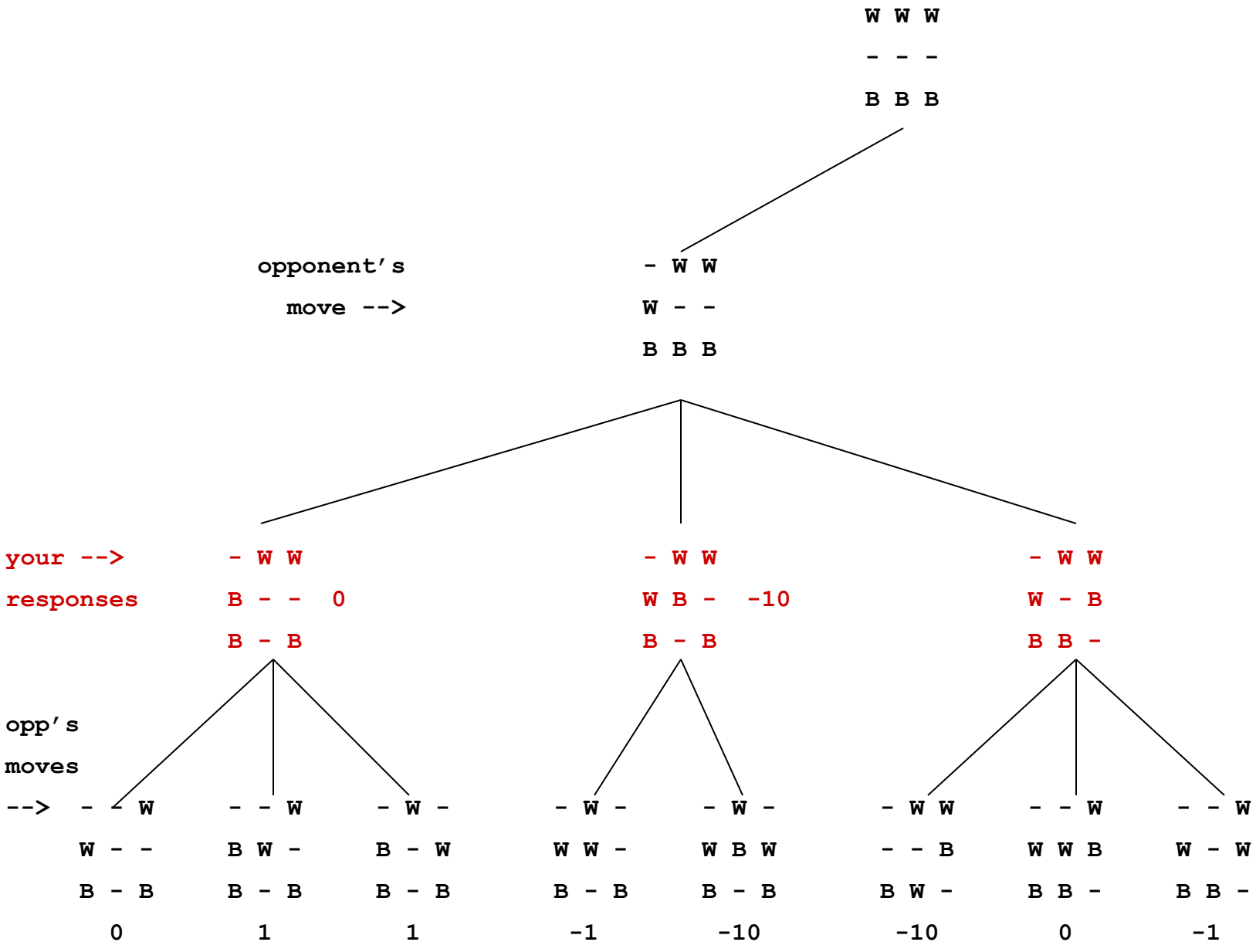

W W W
- - -
B B B
- W W
W - -
B B B
Third Question: if we're trying to figure out what to move here...
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10
0
-1

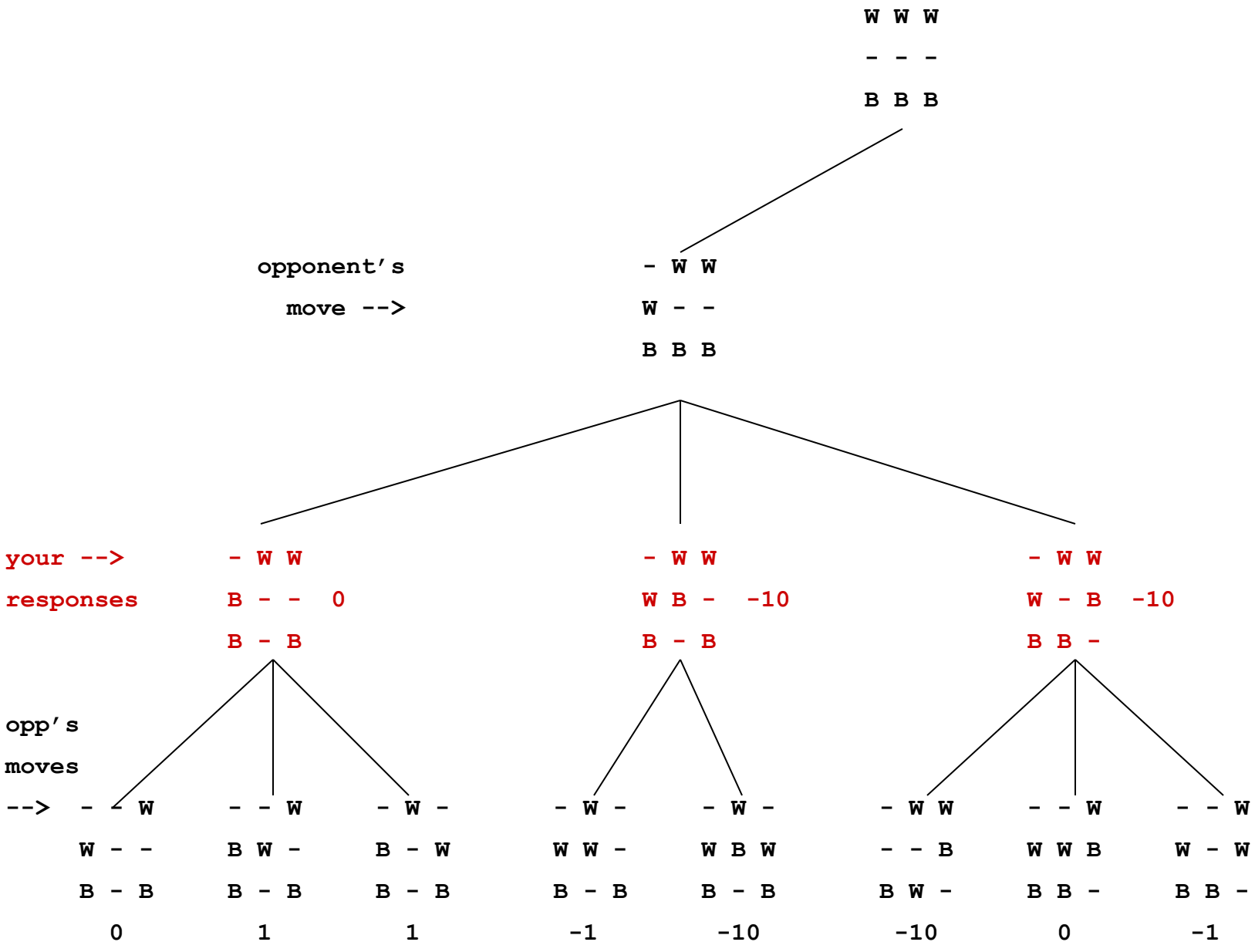


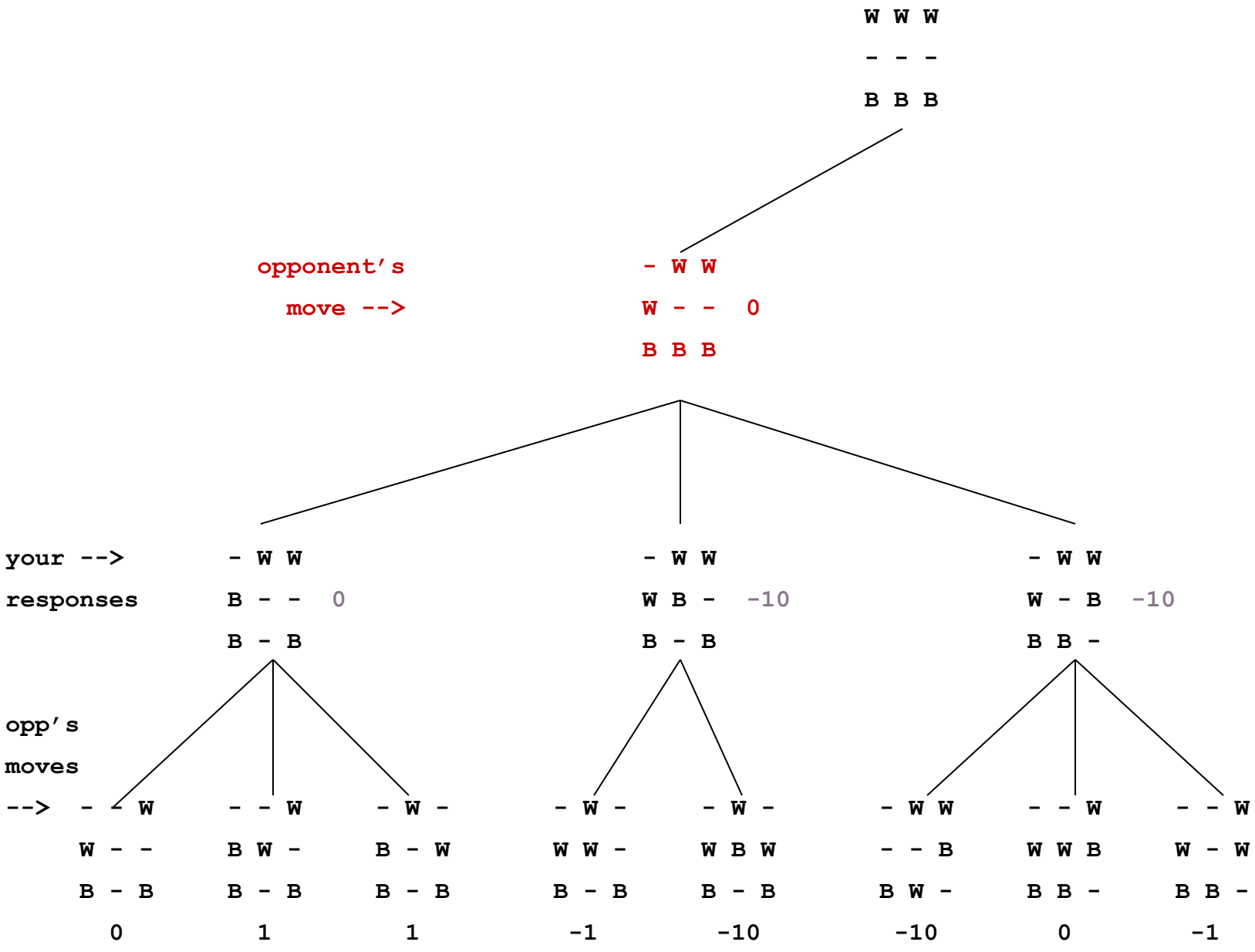
Third Question: if we're trying to figure out what to move here, how can we use the quantitative information from down there?

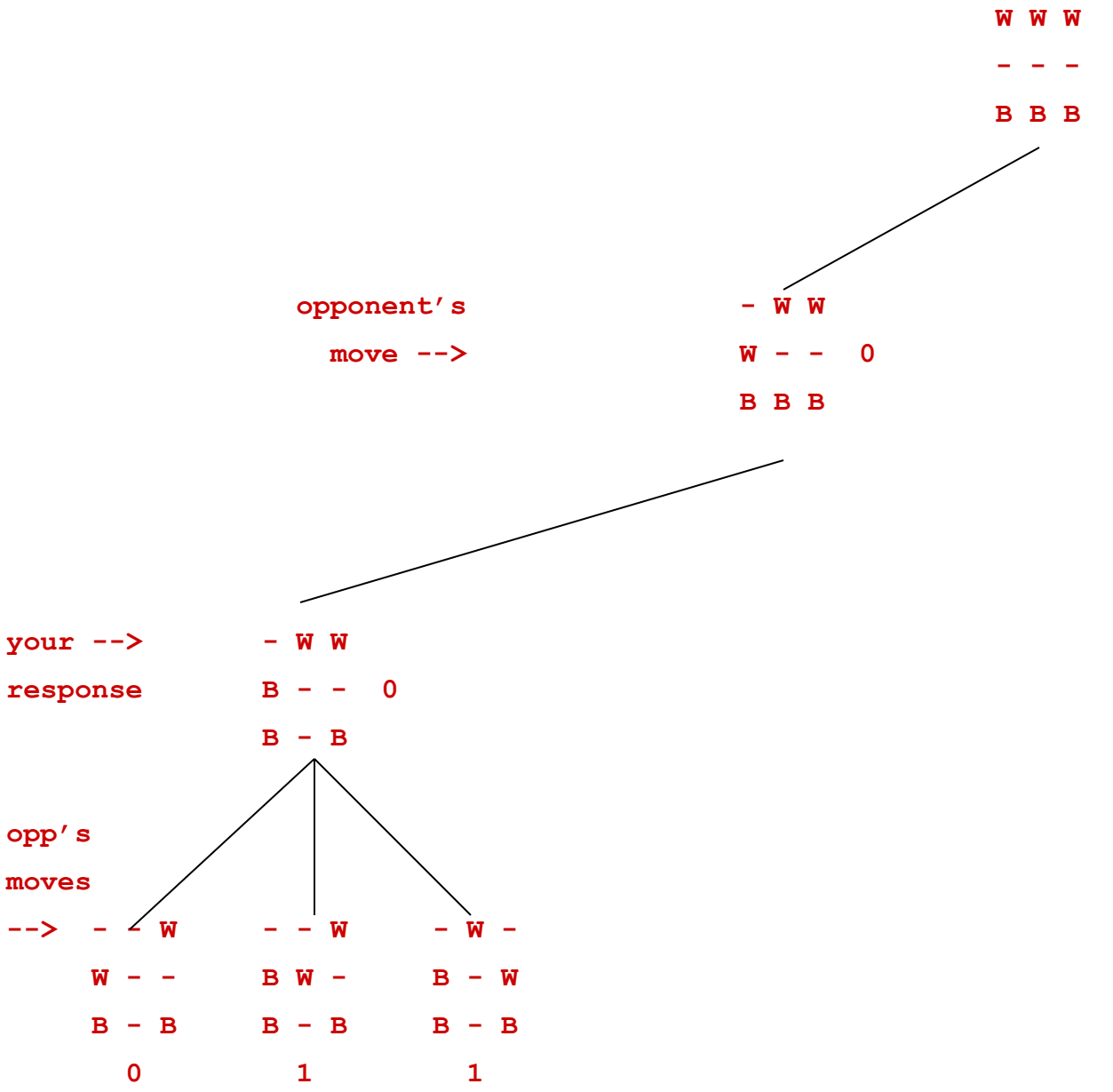

W W W
- - -
B B B
opponent's
move -->
- W W
W - -
B B B
your -->
responses
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
opp's
moves
-->
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10
0
-1

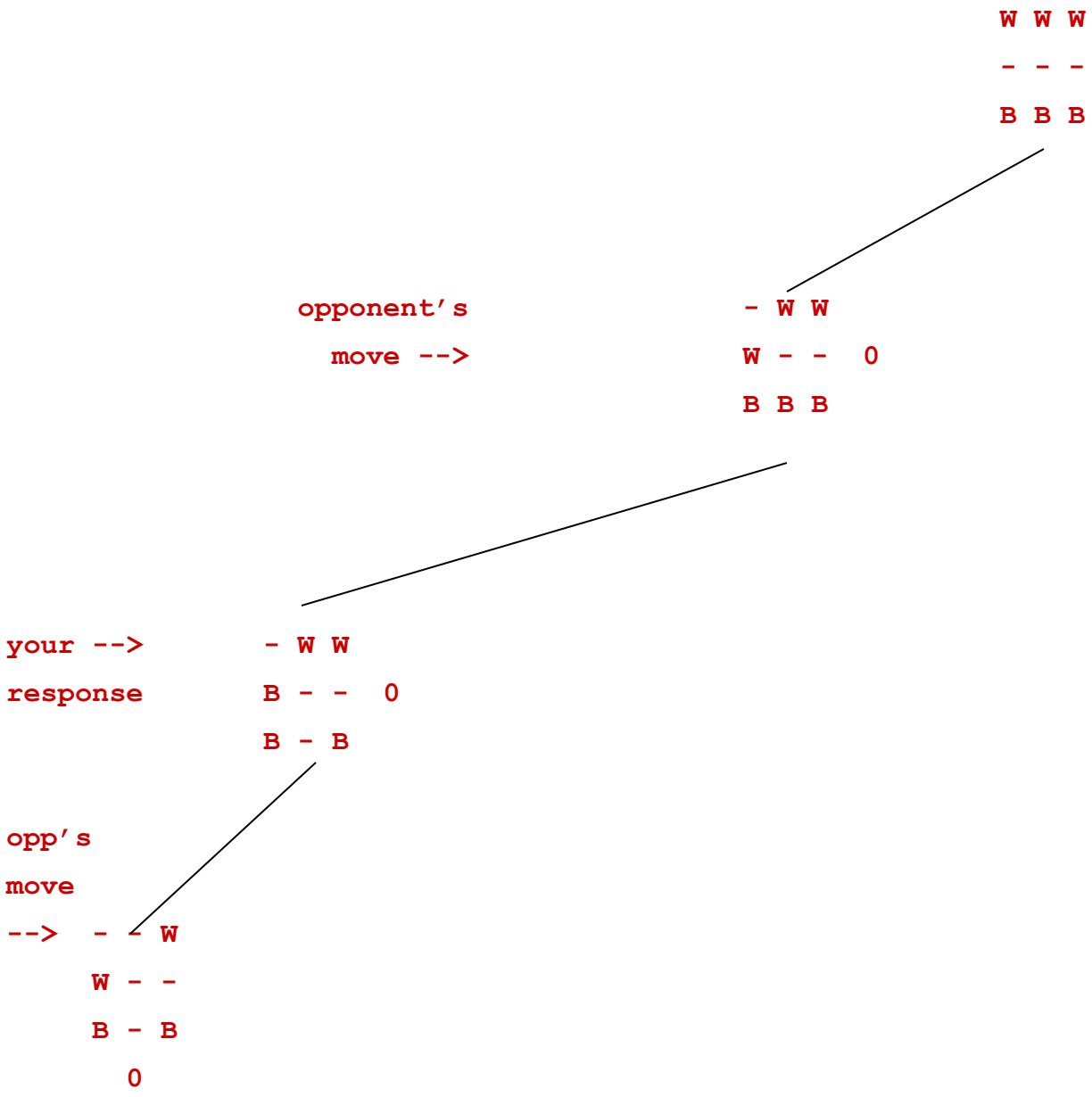

W W W
- - -
B B B
opponent's
move -->
- W W
W - -
B B B
your -->
responses
- W W
B - - 0
B - B
- W W
W B -
B - B
- W W
W - B
B B -
opp's
moves
--> - - W
W - -
B - B
0
- - W
B W -
B - B
1
- W -
B - W
B - B
1
- W -
W W -
B - B
-1
- W -
W B W
B - B
-10
- W W
- - B
B W -
-10
- - W
W W B
B B -
0
- - W
W - W
B B -
-1


W W W
- - -
B B B
opponent's
move -->
- W W
W - -
B B B
your -->
responses
- W W
B - - 0
B - B
- W W
W B - -10
B - B
- W W
W - B
B B -
opp's
moves
-->
- - W
W - -
B - B
0
- - W
B W -
B - B
1
- W -
B - W
B - B
1
- W -
W W -
B - B
-1
- W -
W B W
B - B
-10
- W W
- - B
B W -
-10
- - W
W W B
B B -
0
- - W
W - W
B B -
-1

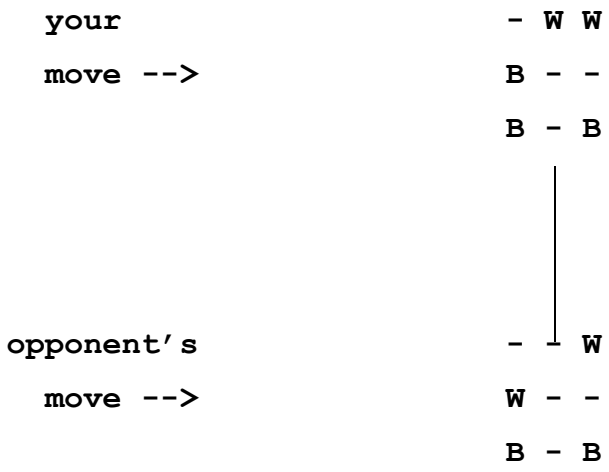

W W W
- - -
B B B
opponent's
move -->
- W W
W - -
B B B
your -->
responses
- W W
B - - 0
B - B
- W W
W B - -10
B - B
- W W
W - B -10
B B -
opp's
moves
-->
- - W
W - -
B - B
0
- - W
B W -
B - B
1
- W -
B - W
B - B
1
- W -
W W -
B - B
-1
- W -
W B W
B - B
-10
- W W
- - B
B W -
-10
- - W
W W B
B B -
0
- - W
W - W
B B -
-1

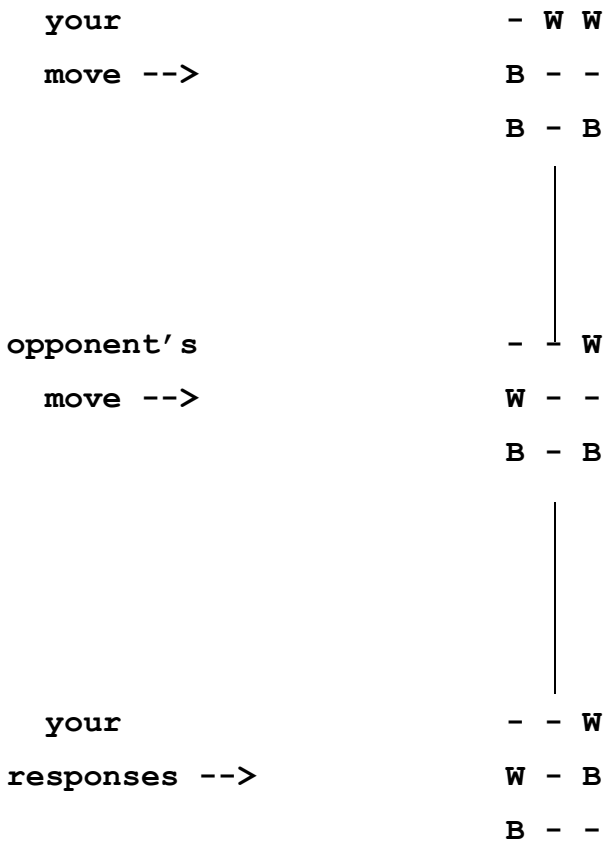

W W W
- - -
B B B
opponent's
move -->
- W W
W - - 0
B B B
your -->
responses
- W W
B - - 0
B - B
- W W
W B - -10
B - B
- W W
W - B -10
B B -
opp's
moves
-->
- - W
W - -
B - B
0
- - W
B W -
B - B
1
- W -
B - W
B - B
1
- W -
W W -
B - B
-1
- W -
W B W
B - B
-10
- W W
- - B
B W -
-10
- - W
W W B
B B -
0
- - W
W - W
B B -
-1

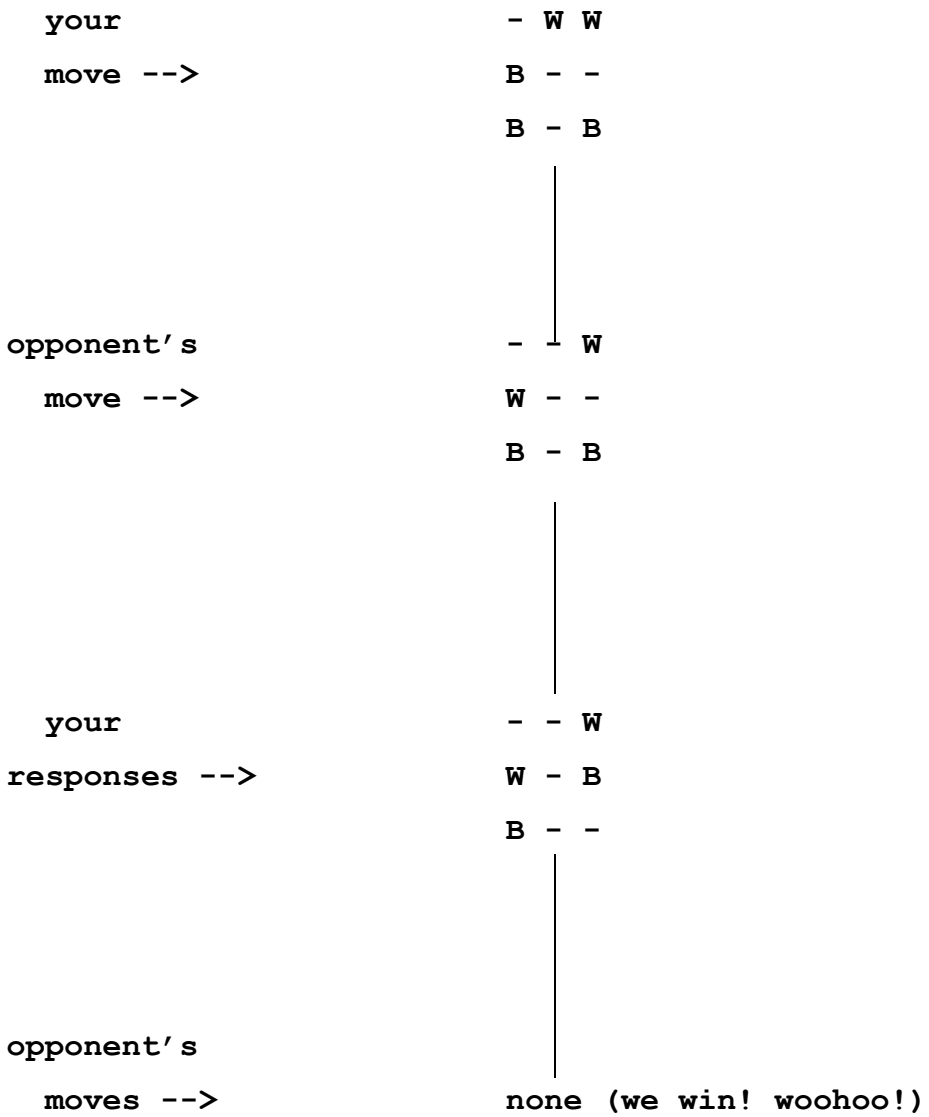

W W W
- - -
B B B
opponent's
move -->
- W W
W - - 0
B B B
your -->
response
- W W
B - - 0
B - B
opp's
moves
--> - - W
W - -
B - B
0
- - W
B W -
B - B
1
- W -
B - W
B - B
1


W W W
- - -
B B B
opponent's
move -->
- W W
W - - 0
B B B
your -->
response
- W W
B - - 0
B - B
opp's
move
--> - - W
W - -
B - B
0

What happens next?

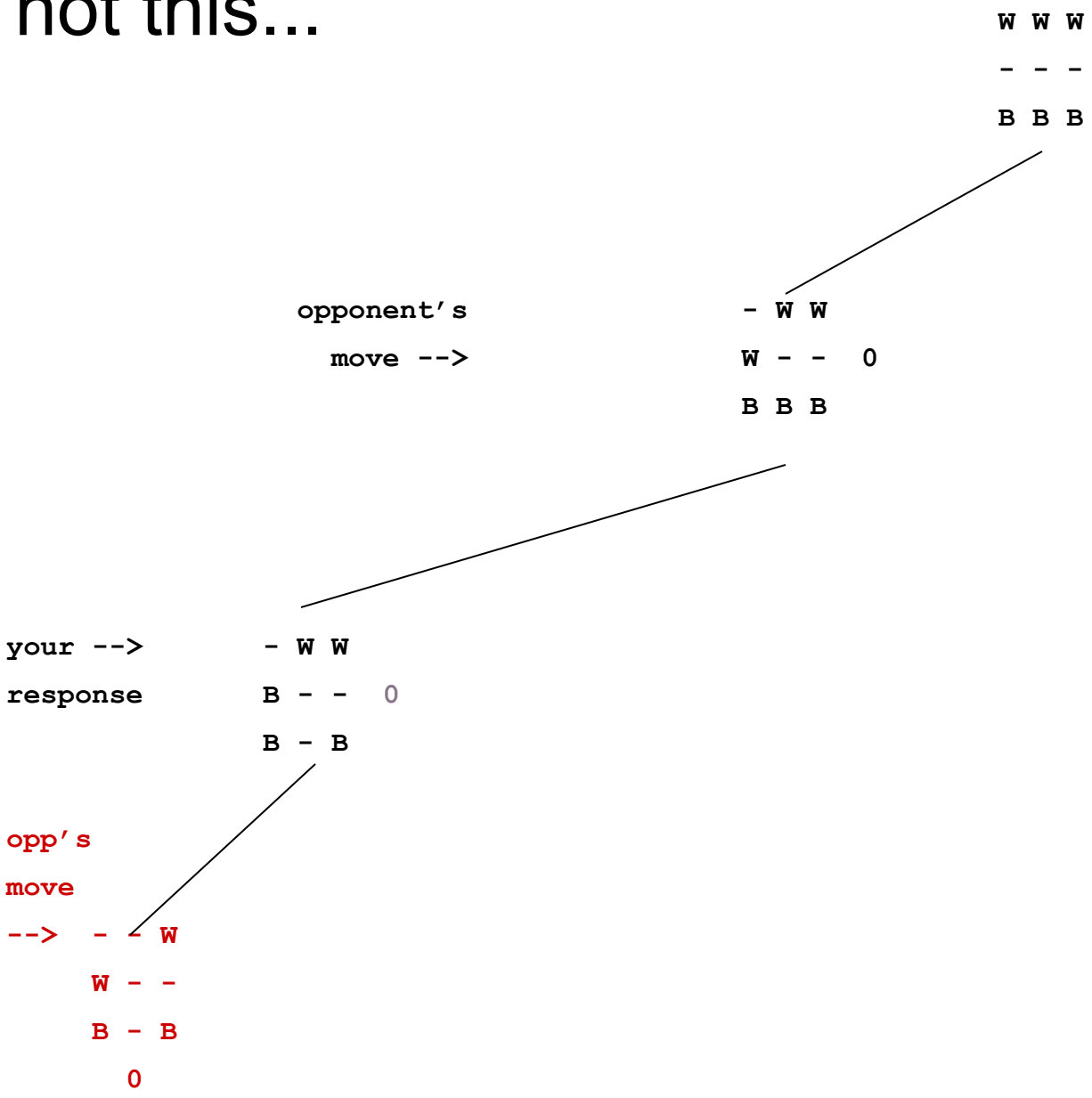

your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
W - -
B - B


your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
W - -
B - B
your
responses -->
- - W
W - B
B - -

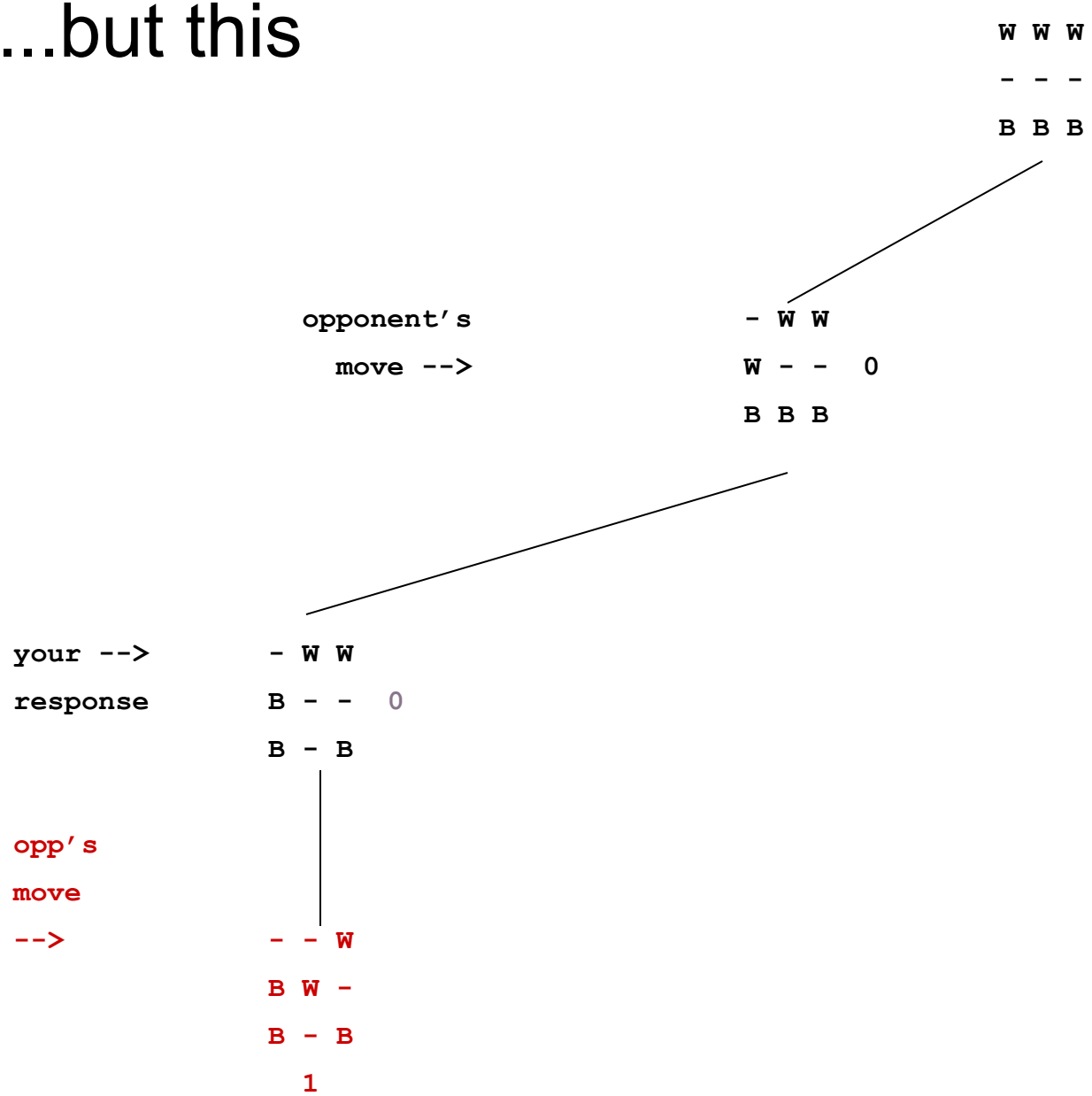

your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
W - -
B - B
your
responses -->
- - W
W - B
B - -
opponent's
moves -->
none (we win! woohoo!)

What if white makes a different move?

not this...

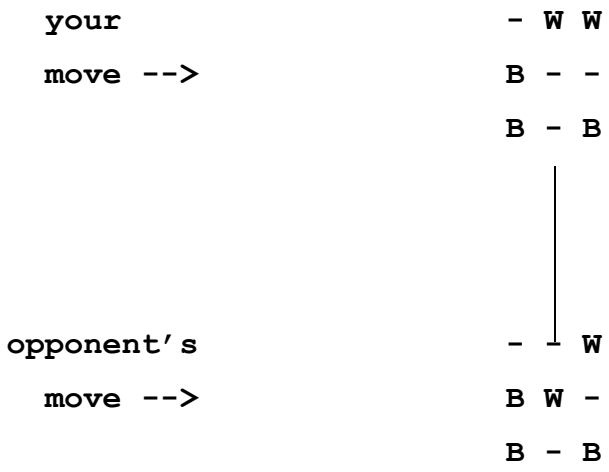

W W W
- - -
B B B
opponent's
move -->
- W W
W - - 0
B B B
your -->
response
- W W
B - - 0
B - B
opp's
move
--> - - W
W - -
B - B
0

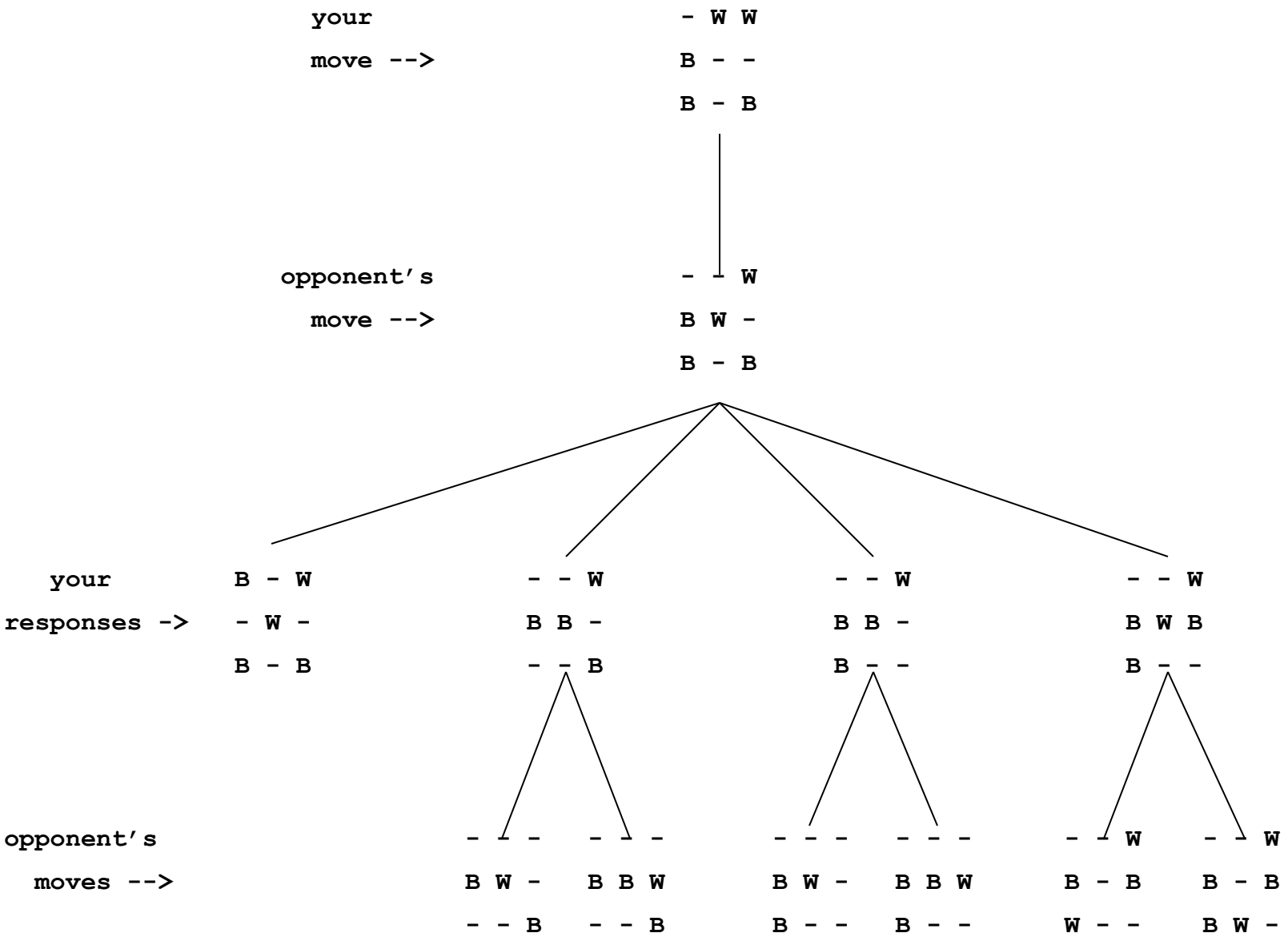
...but this

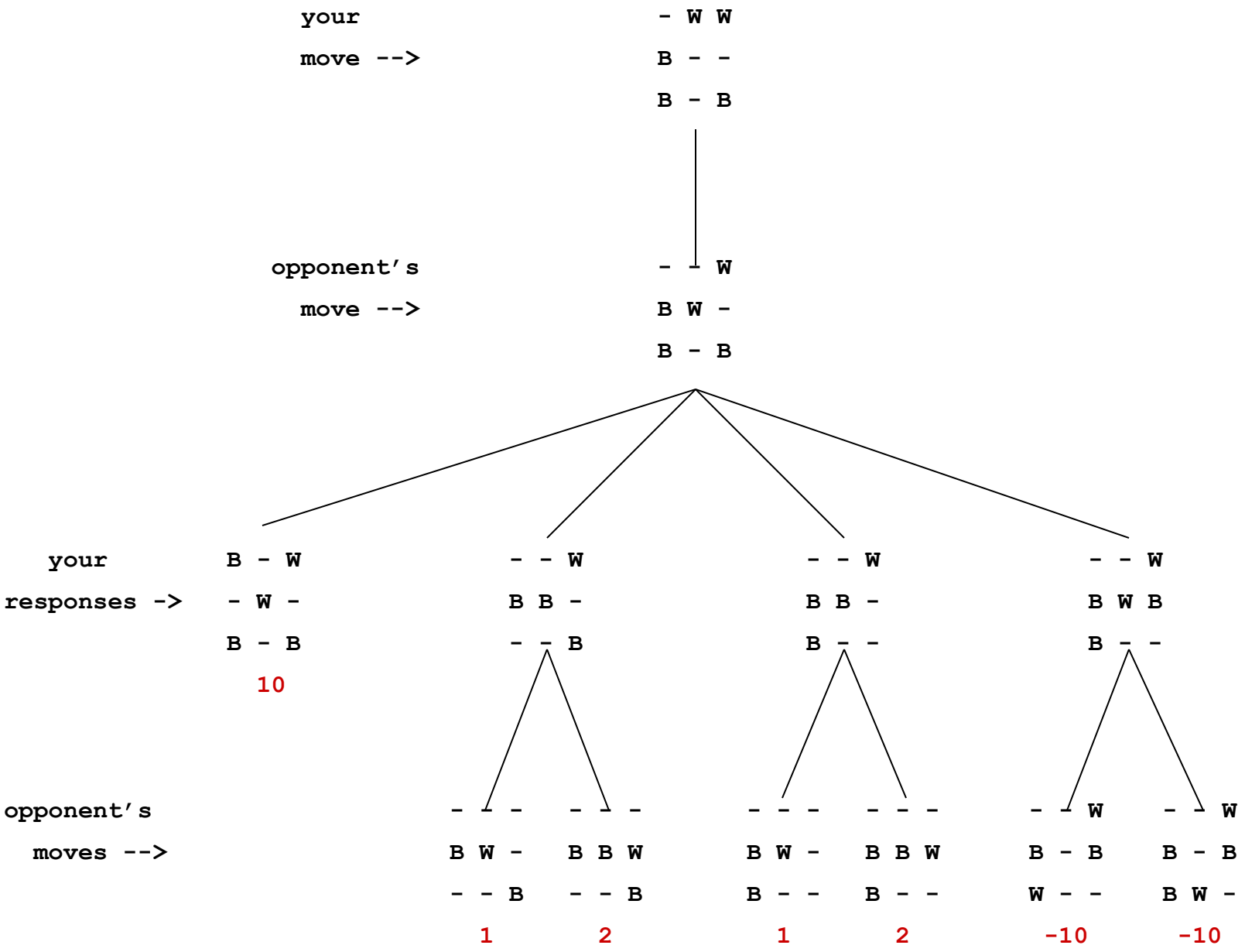

W W W
- - -
B B B
opponent's
move -->
- W W
W - - 0
B B B
your -->
response
- W W
B - - 0
B - B
opp's
move
-->
- - W
B W -
B - B
1

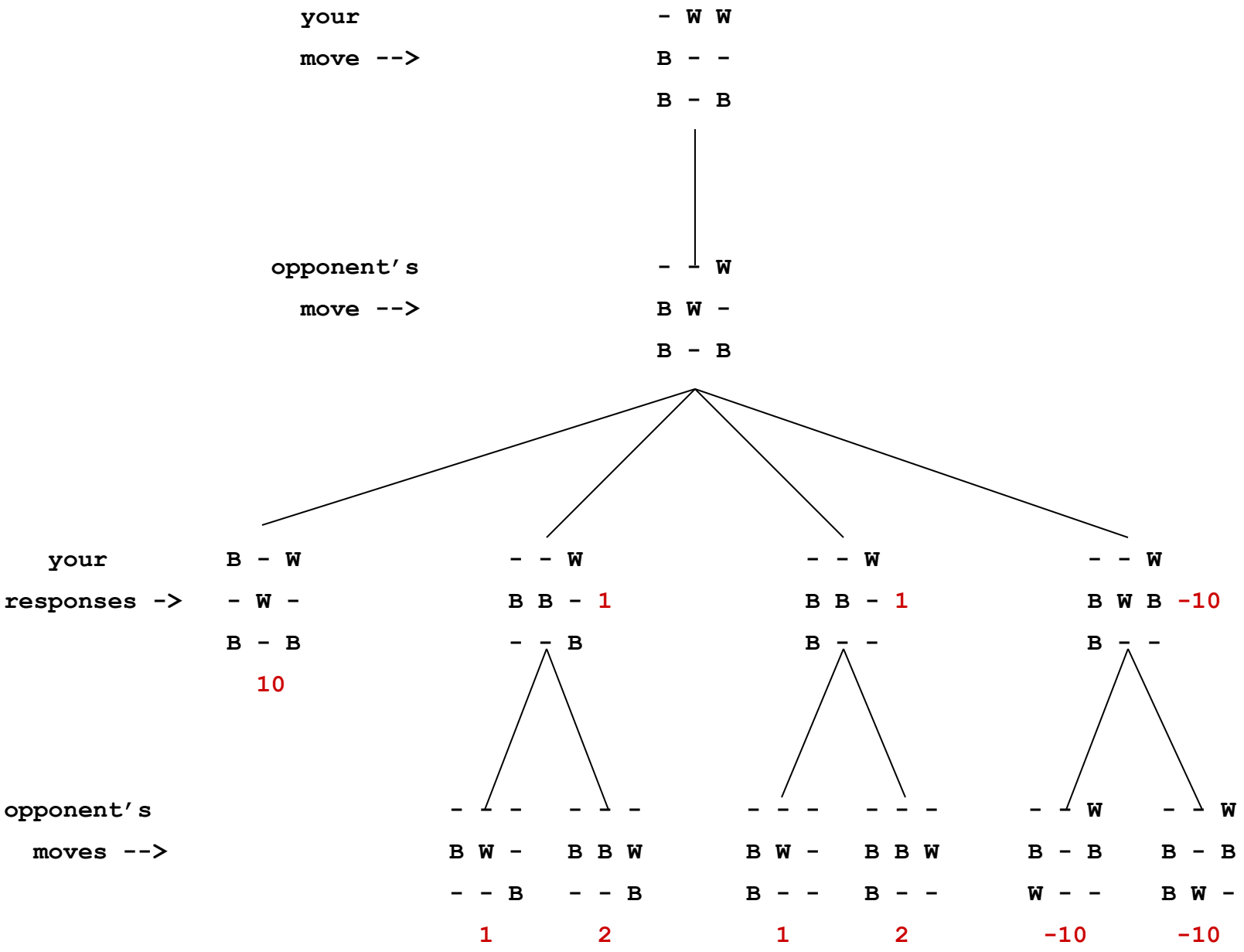
What if the white makes a different move?

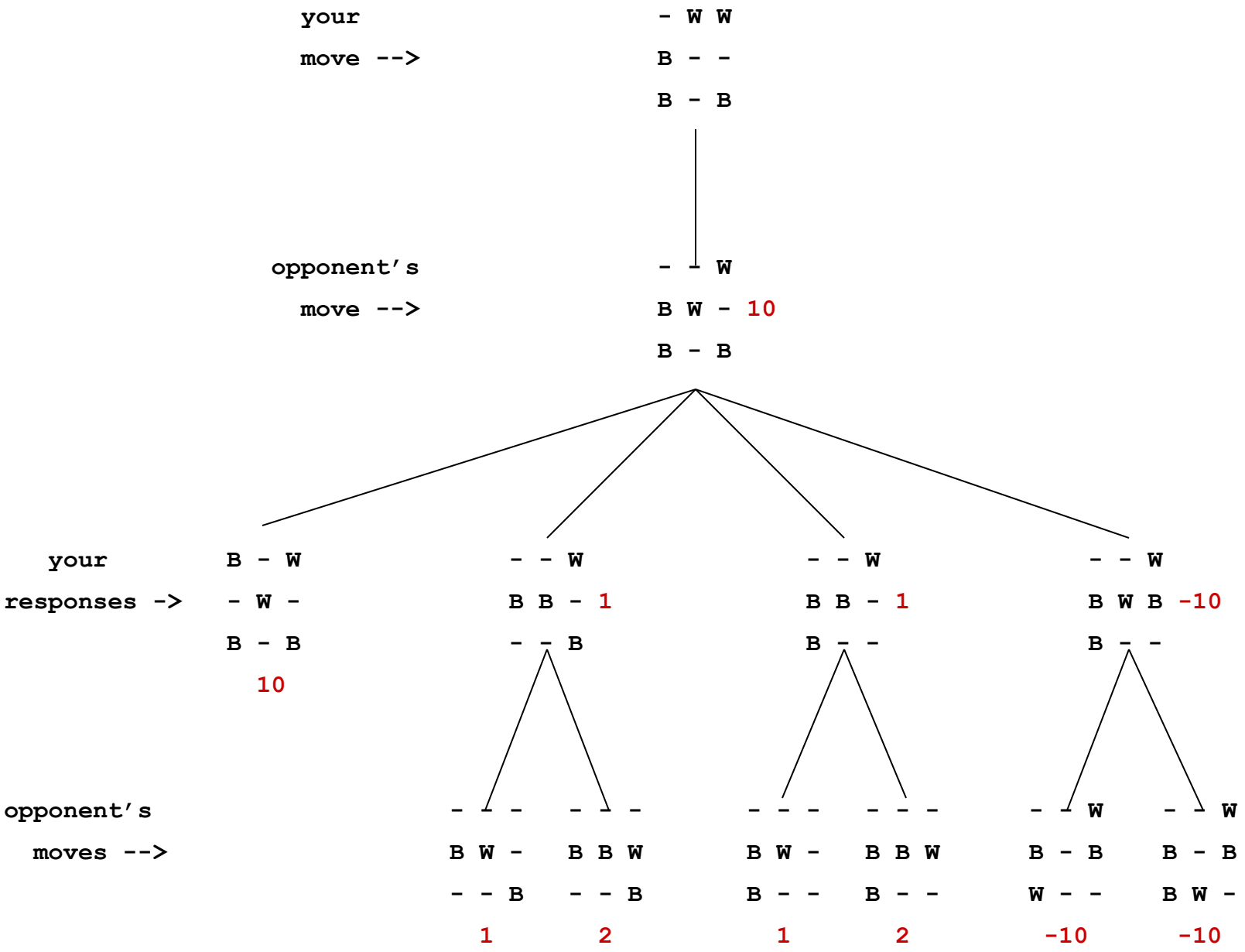
Then apply the same search technique again to white’ s move and make your next move accordingly

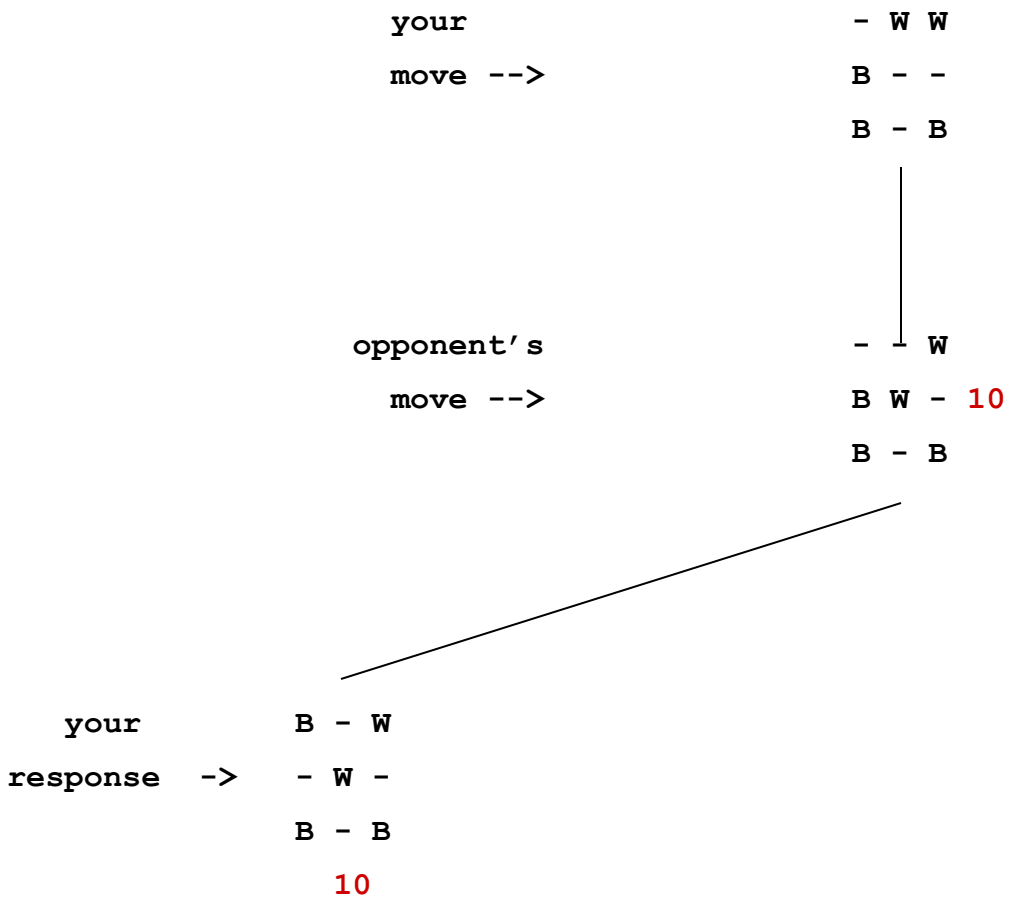

your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
B W -
B - B


your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
B W -
B - B
your
responses ->
B - W
- W -
B - B
- - W
B B -
- - B
- - W
B B -
B - -
- - W
B W B
B - -
opponent's
moves -->
- - -
B W -
- - B
- - -
B B W
- - B
- - -
B W -
B - -
- - -
B B W
B - -
- - W
B - B
W - -
- - W
B - B
B W -


your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
B W -
B - B
your
responses ->
B - W
- W -
B - B
10
- - W
B B -
- - B
- - W
B B -
B - -
- - W
B W B
B - -
opponent's
moves -->
- - - - - -
B W - B B W
- - B - - B
1 2
- - - - - -
B W - B B W
B - - B - -
1 2
- - W - - W
B - B B - B
W - - B W -
-10 -10

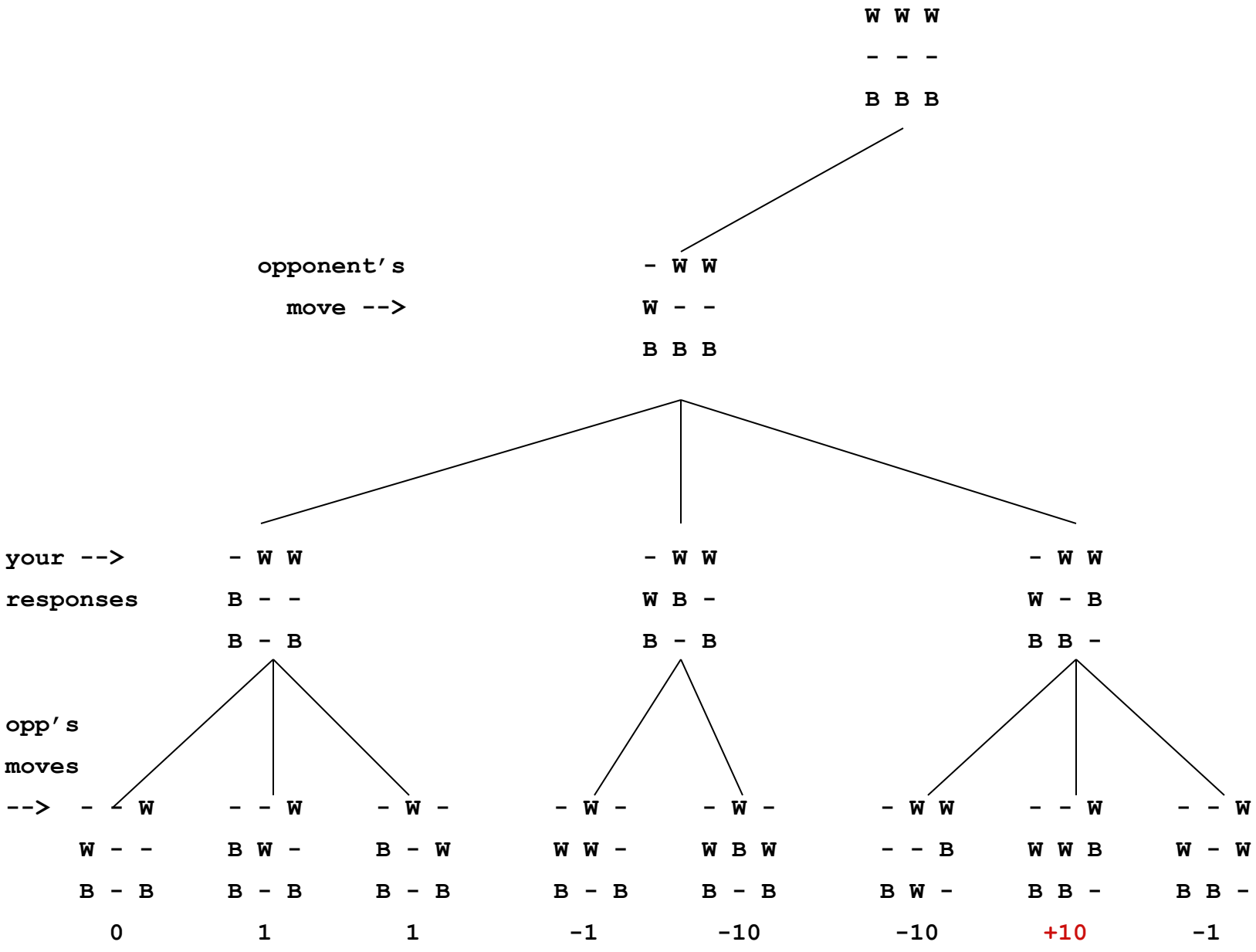

your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
B W -
B - B
your
responses ->
B - W
- W -
B - B
10
- - W
B B - 1
- - B
- - W
B B - 1
B - -
- - W
B W B -10
B - -
opponent's
moves -->
- - - - - -
B W - B B W
- - B - - B
1 2
- - - - - -
B W - B B W
B - - B - -
1 2
- - W - - W
B - B B - B
W - - B W -
-10 -10

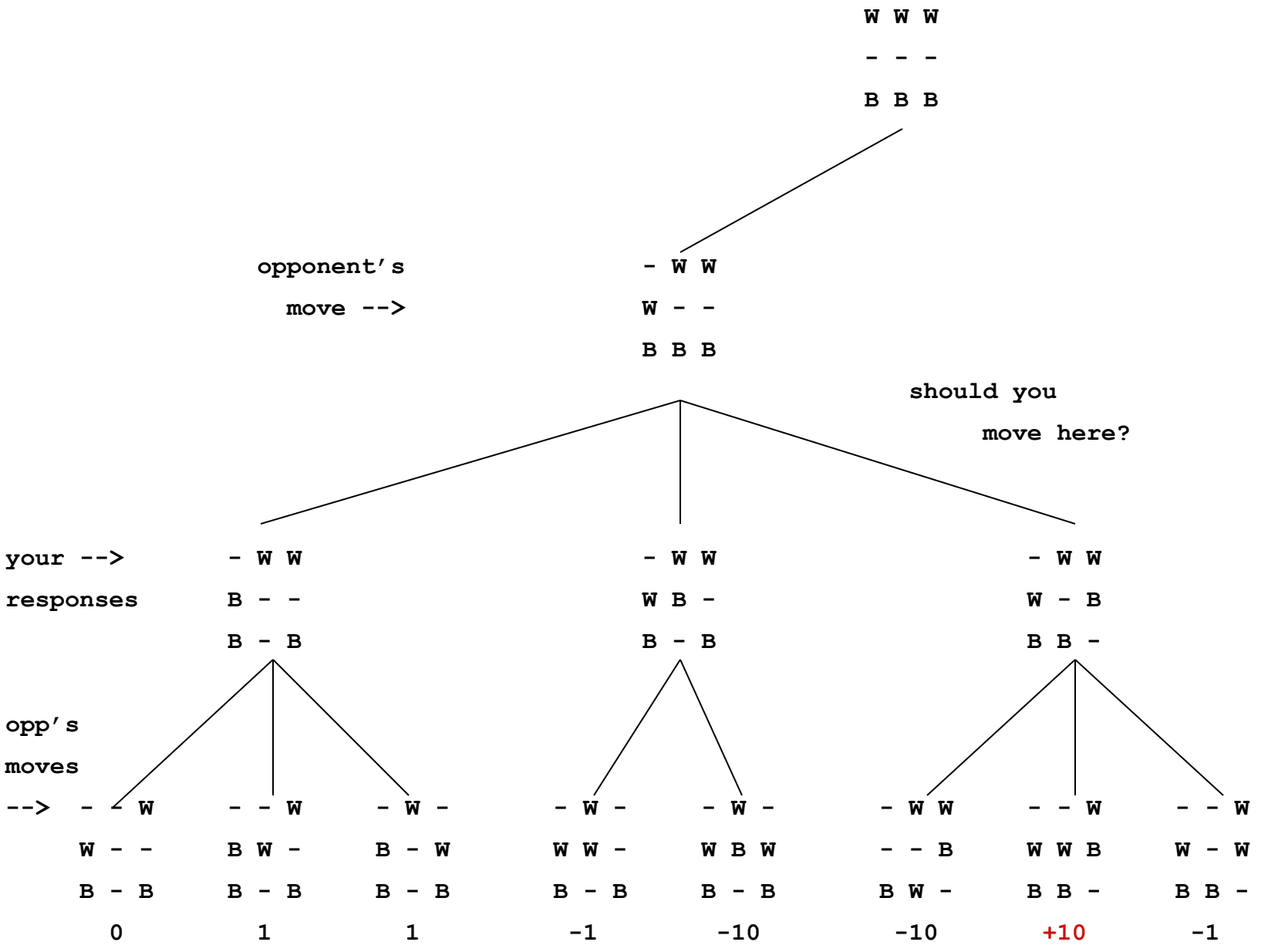

your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
B W - 10
B - B
your
responses ->
B - W
- W -
B - B
10
- - W
B B - 1
- - B
- - W
B B - 1
B - -
- - W
B W B -10
B - -
opponent's
moves -->
- - -
B W -
- - B
1
- - -
B B W
- - B
2
- - -
B W -
B - -
1
- - -
B B W
B - -
2
- - W
B - B
W - -
-10
- - W
B - B
B W -
-10

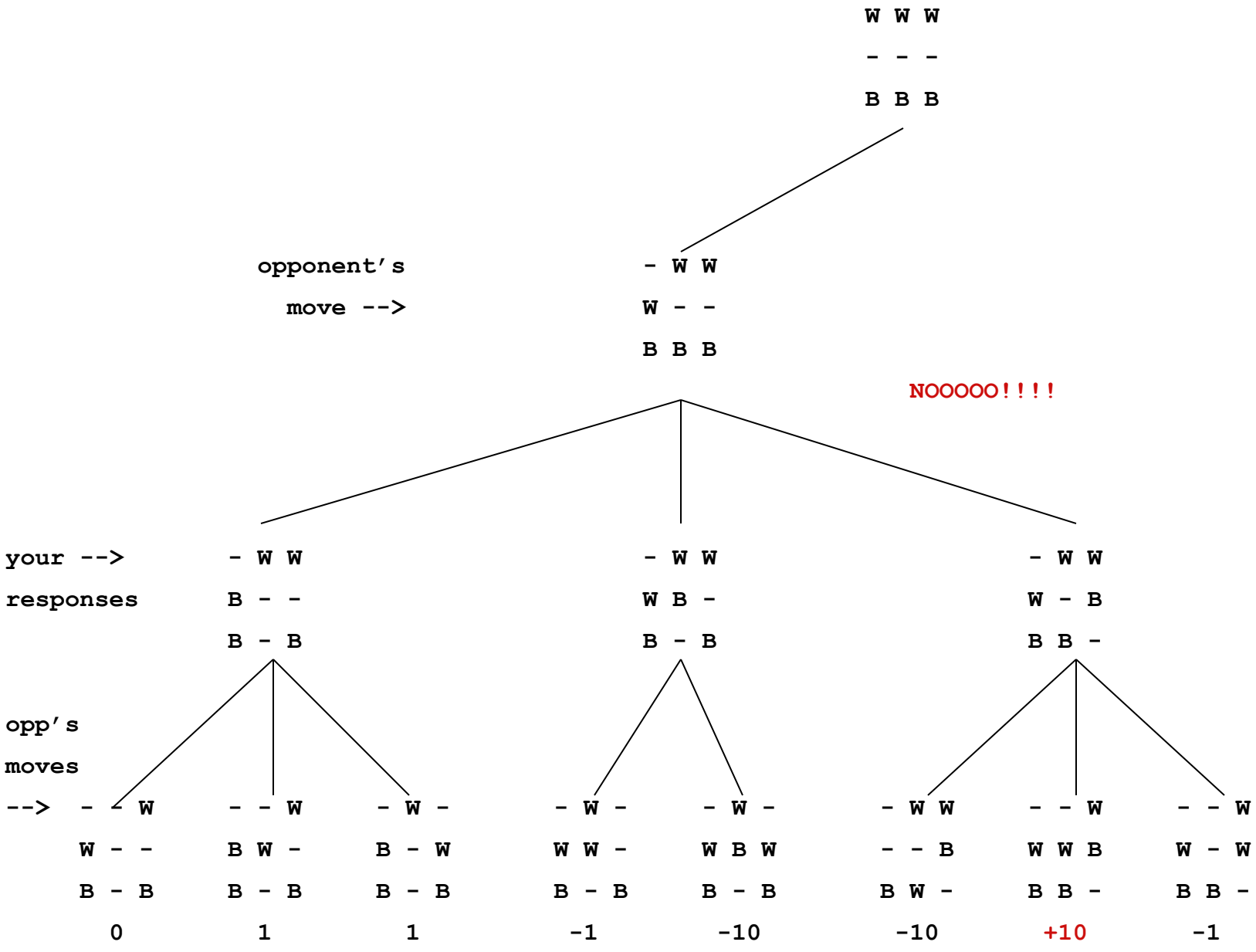

your
move -->
- W W
B - -
B - B
opponent's
move -->
- - W
B W - 10
B - B
your
response ->
B - W
- W -
B - B
10

What if ...?

...way back at the beginning there was an obvious win for you in your search space?


W W W
- - -
B B B
opponent's
move -->
- W W
W - -
B B B
your -->
responses
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
opp's
moves
-->
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10
+10
-1


W W W
- - -
B B B
opponent's
move -->
- W W
W - -
B B B
should you
move here?
your -->
responses
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
opp's
moves
-->
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10
+10
-1


W W W
- - -
B B B
opponent's
move -->
- W W
W - -
B B B
NOOOOO!!!!
your -->
responses
- W W
B - -
B - B
- W W
W B -
B - B
- W W
W - B
B B -
opp's
moves
-->
- - W
W - -
B - B
- - W
B W -
B - B
- W -
B - W
B - B
- W -
W W -
B - B
- W -
W B W
B - B
- W W
- - B
B W -
- - W
W W B
B B -
- - W
W - W
B B -
0
1
1
-1
-10
-10
+10
-1

Questions?

Game search

(also known as adversarial search) has these components:

move (or board) generator

static board evaluation function (this is the heuristic part...and it doesn't generate moves)

minimax algorithm to alternately propagate minima and maxima upward from "bottom"

Minimax algorithm

Start with the following:

a) there are two players, MAX and MIN

Minimax algorithm

Start with the following:

a) there are two players, MAX and MIN

b) it’s MAX’s turn to move

Minimax algorithm

Start with the following:

a) there are two players, MAX and MIN

b) it's MAX's turn to move

c) MAX has a static board evaluation function that returns bigger values if a board is favorable to MAX

Minimax algorithm

Start with the following:

a) there are two players, MAX and MIN

b) it's MAX's turn to move

c) MAX has a static board evaluation function that returns bigger values if a board is favorable to MAX

d) the evaluation function gets better as the game gets closer to a goal state (else why bother to generate the game space?)

Minimax algorithm

Start with the following:

a) there are two players, MAX and MIN

b) it's MAX's turn to move

c) MAX has a static board evaluation function that returns bigger values if a board is favorable to MAX

d) the evaluation function gets better as the game gets closer to a goal state (else why bother to generate the game space?)

e) MAX believes that MIN's evaluation function is no better than MAX's (if that's not true, then MAX should at least avoid betting money on this game)

Minimax algorithm

1. Generate the game tree to as many levels (plies) that time and space constraints allow. The top level is called MAX (as in it's now MAX's turn to move), the next level is called MIN, the next level is MAX, and so on.

2. Apply the evaluation function to all the terminal (leaf) states/boards to get "goodness" values

3. Use those terminal board values to determine the values to be assigned to the immediate parents:
   a) if the parent is at a MIN level, then the value is the minimum of the values of its children
   b) if the parent is at a MAX level, then the value is the maximum of the values of its children

4. Keep propagating values upward as in step 3

5. When the values reach the top of the game tree, MAX chooses the move indicated by the highest value

Minimax algorithm

1. Generate the game tree to as many levels (plies) that time and space constraints allow. The top level is called MAX (as in it's now MAX's turn to move), the next level is called MIN, the next level is MAX, and so on.

2. Apply the evaluation function to all the terminal (leaf) states/boards to get "goodness" values

3. Use those terminal board values to determine the values to be assigned to the immediate parents:
   a) if the parent is at a MIN level, then the value is the minimum of the values of its children
   b) if the parent is at a MAX level, then the value is the maximum of the values of its children

4. Keep propagating values upward as in step 3

5. When the values reach the top of the game tree, MAX chooses the move indicated by the highest value

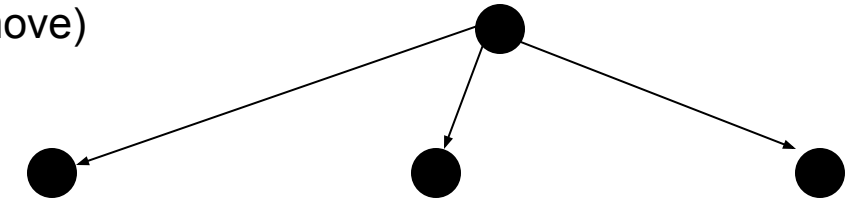
Minimax algorithm

MAX (it’s MAX’s turn to move)

Minimax algorithm

MAX (it's MAX's turn to move)

MIN

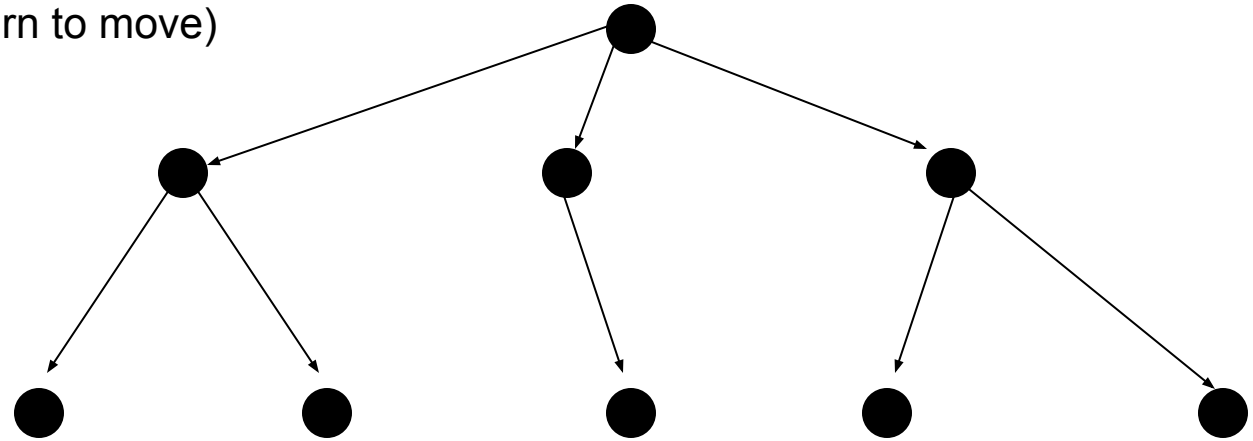


Minimax algorithm

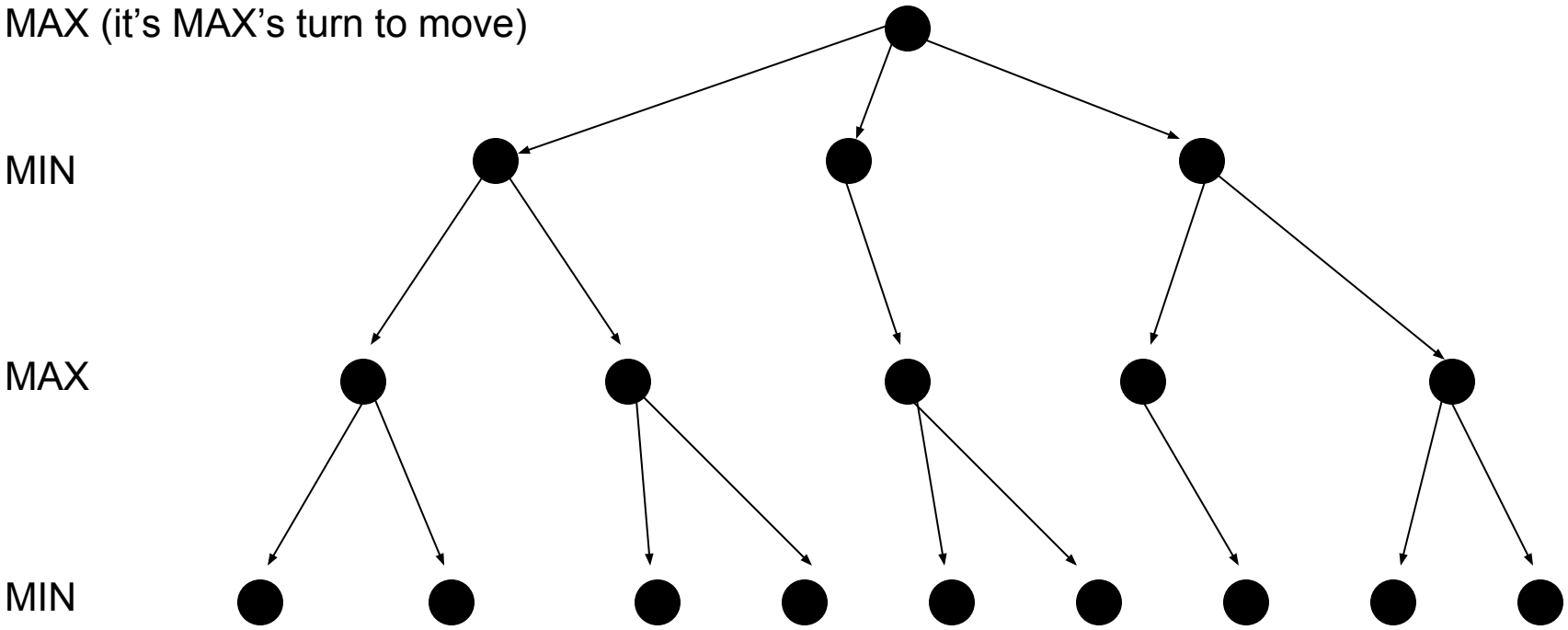
MAX (it's MAX's turn to move)

MIN

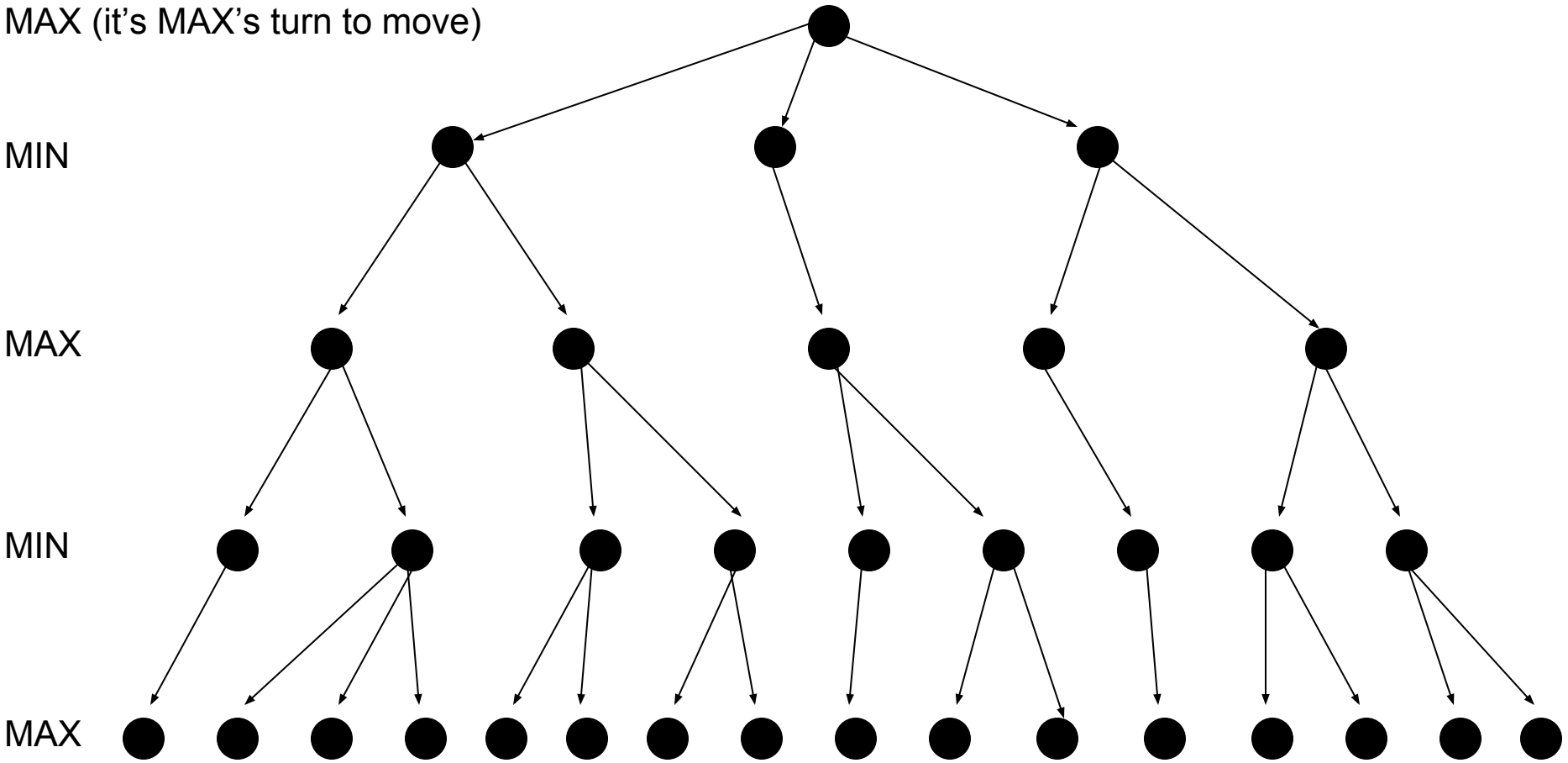
MAX



Minimax algorithm



Minimax algorithm



Minimax algorithm

1. Generate the game tree to as many levels (plies) that time and space constraints allow. The top level is called MAX (as in it's now MAX's turn to move), the next level is called MIN, the next level is MAX, and so on.

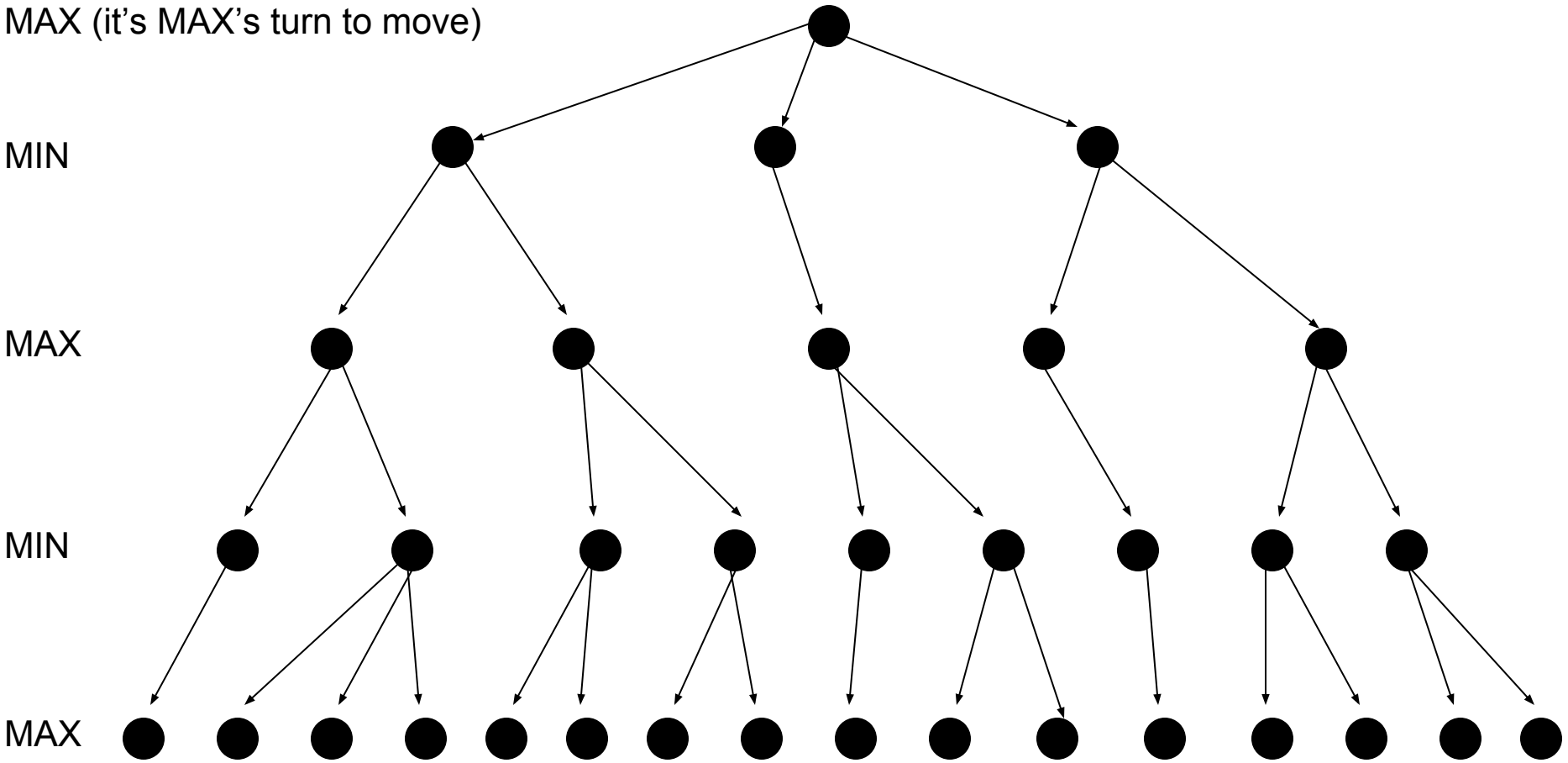
2. Apply the evaluation function to all the terminal (leaf) states/boards to get "goodness" values

3. Use those terminal board values to determine the values to be assigned to the immediate parents:
   a) if the parent is at a MIN level, then the value is the minimum of the values of its children
   b) if the parent is at a MAX level, then the value is the maximum of the values of its children

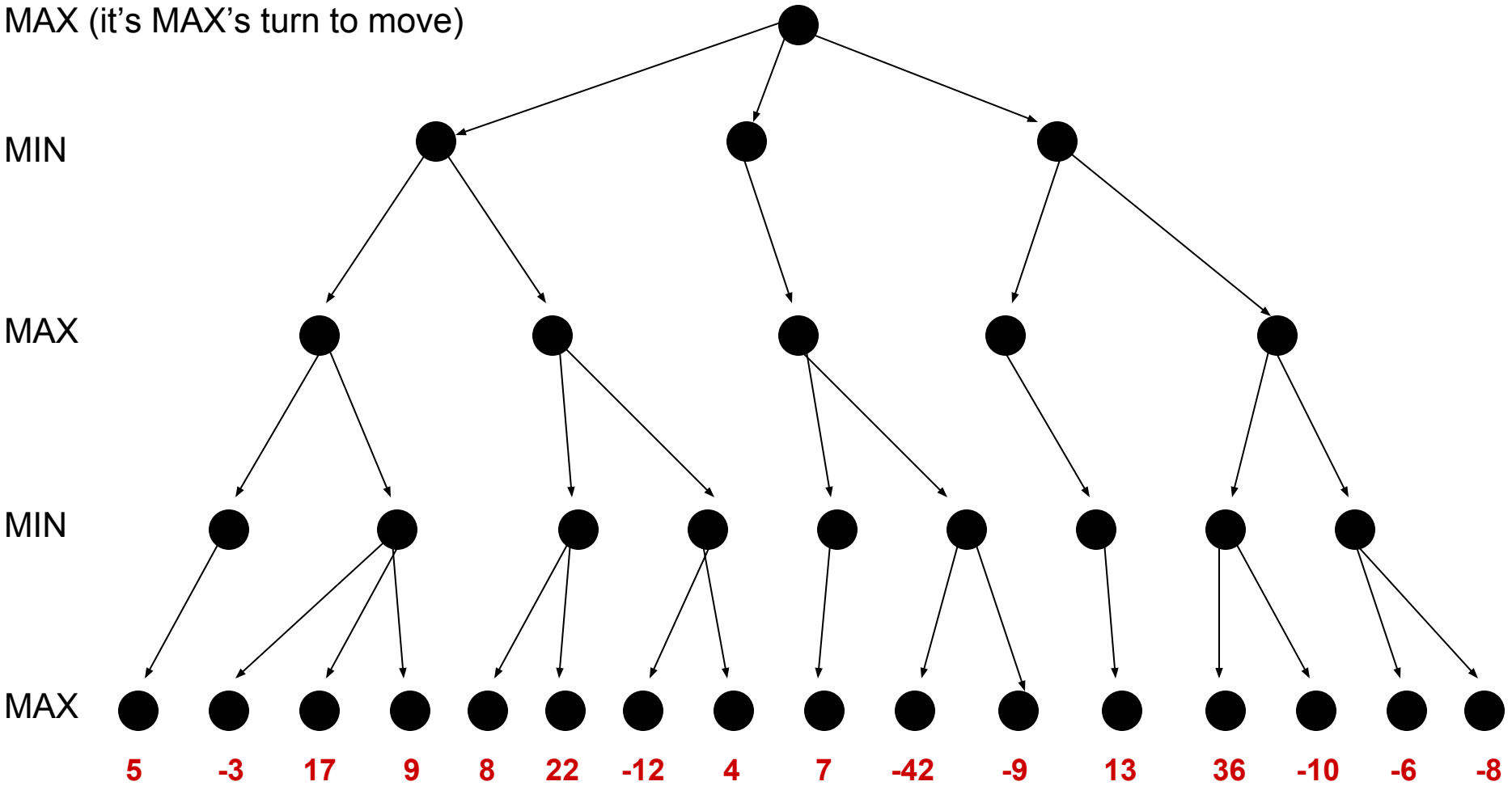
4. Keep propagating values upward as in step 3

5. When the values reach the top of the game tree, MAX chooses the move indicated by the highest value

Minimax algorithm



Minimax algorithm



Minimax algorithm

1. Generate the game tree to as many levels (plies) that time and space constraints allow. The top level is called MAX (as in it's now MAX's turn to move), the next level is called MIN, the next level is MAX, and so on.

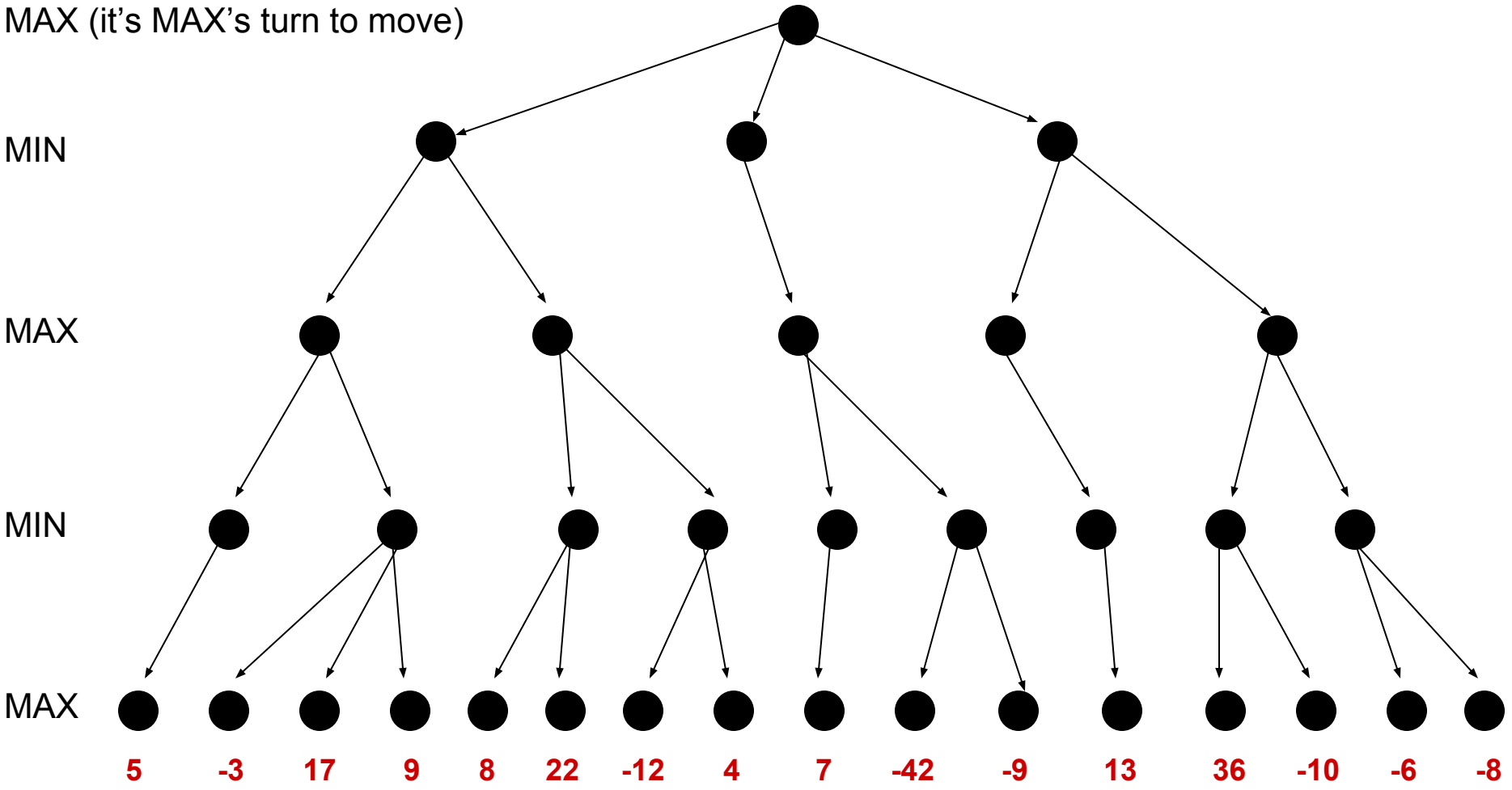
2. Apply the evaluation function to all the terminal (leaf) states/boards to get "goodness" values

3. Use those terminal board values to determine the values to be assigned to the immediate parents:
   a) if the parent is at a MIN level, then the value is the minimum of the values of its children
   b) if the parent is at a MAX level, then the value is the maximum of the values of its children

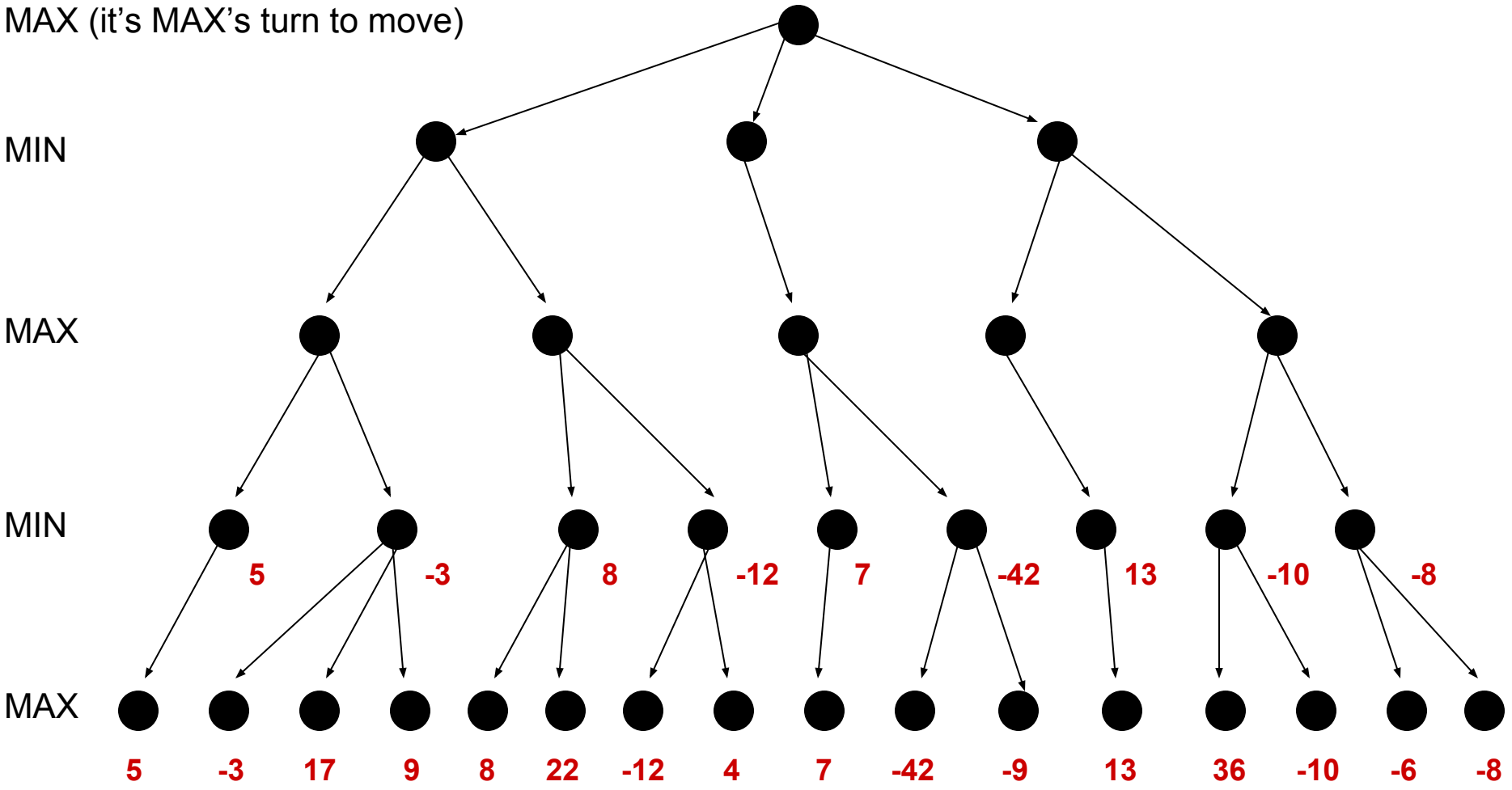
4. Keep propagating values upward as in step 3

5. When the values reach the top of the game tree, MAX chooses the move indicated by the highest value

Minimax algorithm



Minimax algorithm



Minimax algorithm

1. Generate the game tree to as many levels (plies) that time and space constraints allow. The top level is called MAX (as in it's now MAX's turn to move), the next level is called MIN, the next level is MAX, and so on.

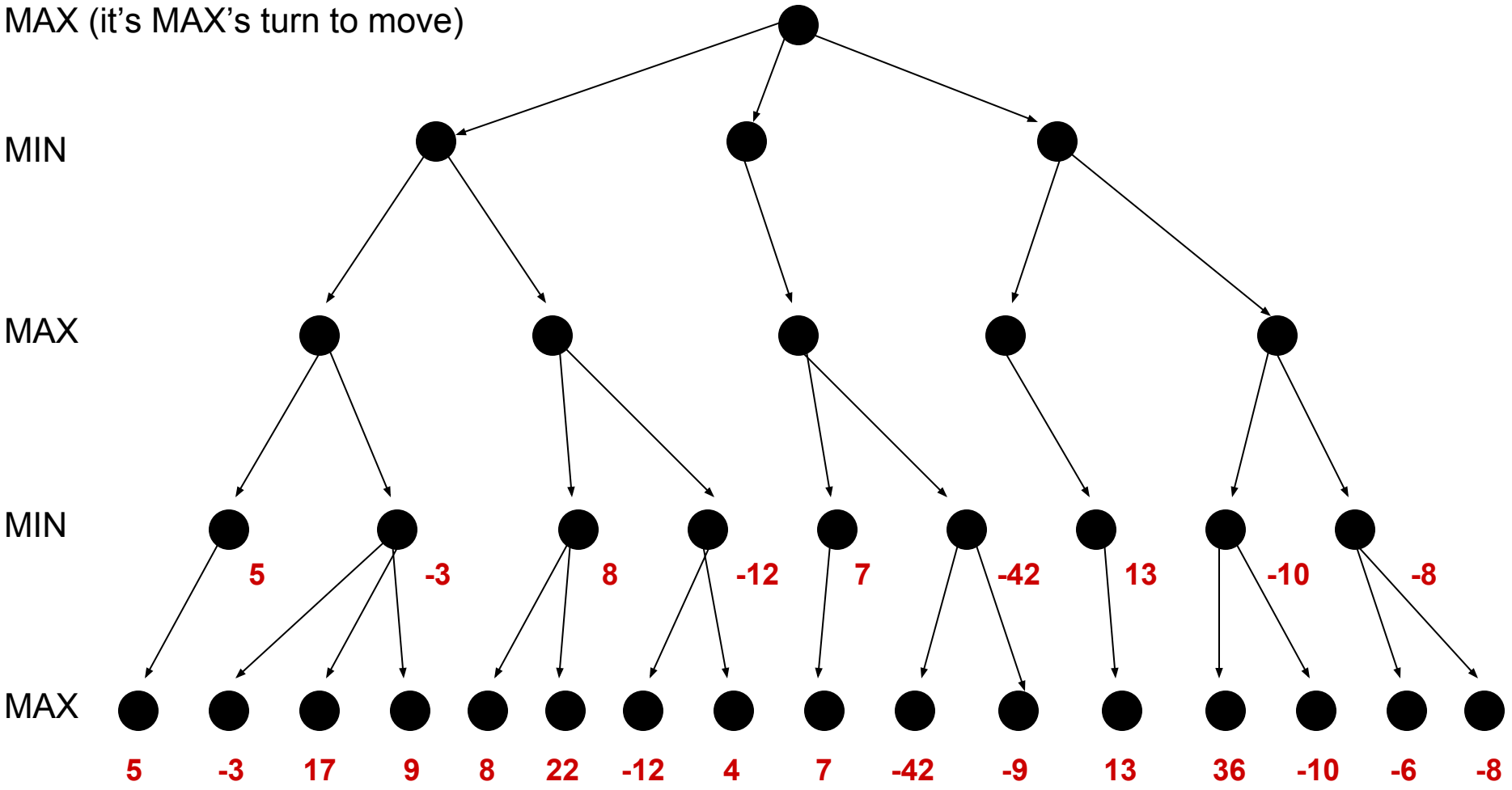
2. Apply the evaluation function to all the terminal (leaf) states/boards to get "goodness" values

3. Use those terminal board values to determine the values to be assigned to the immediate parents:
   a) if the parent is at a MIN level, then the value is the minimum of the values of its children
   b) if the parent is at a MAX level, then the value is the maximum of the values of its children

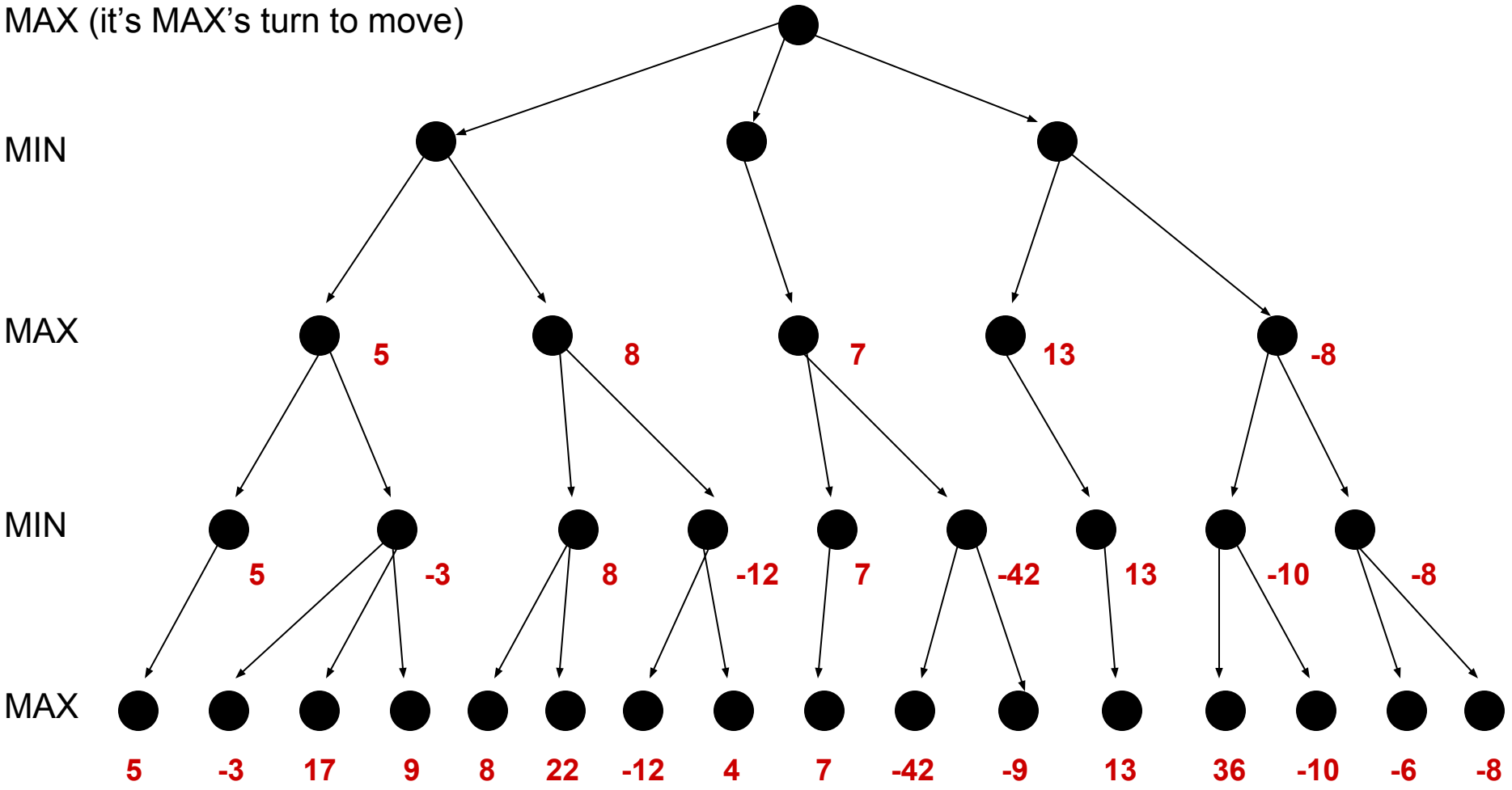
4. Keep propagating values upward as in step 3

5. When the values reach the top of the game tree, MAX chooses the move indicated by the highest value

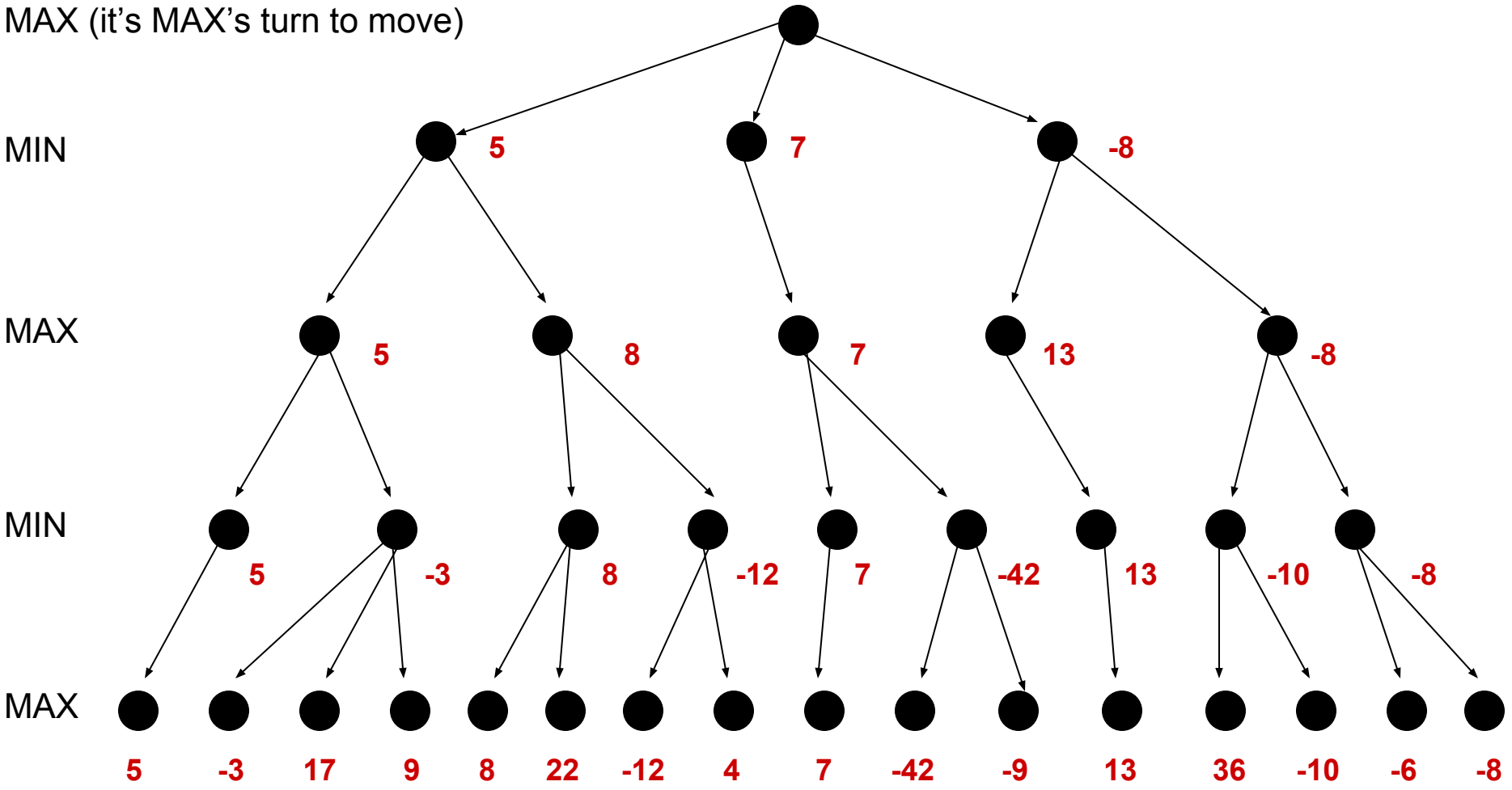
Minimax algorithm



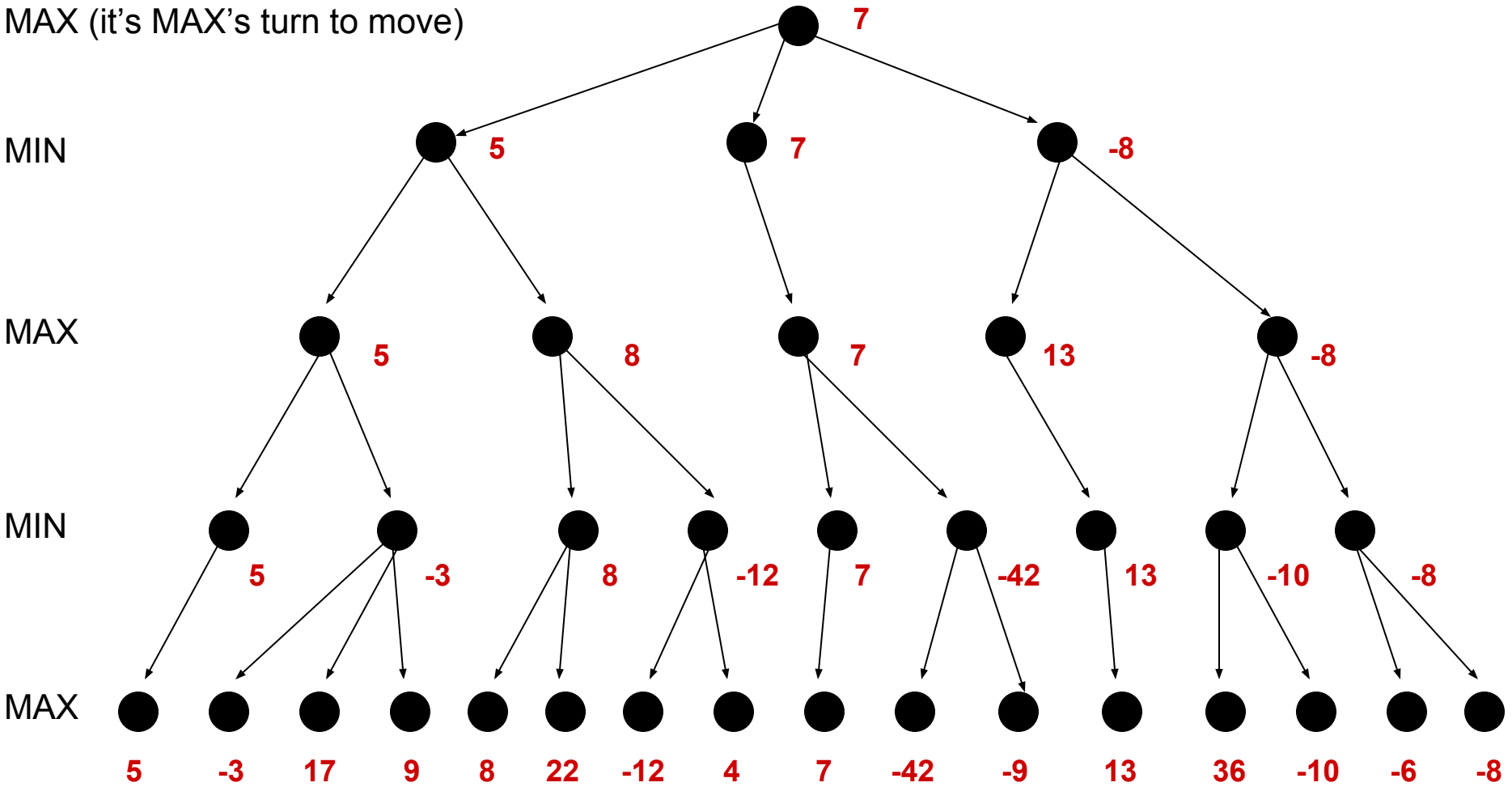
Minimax algorithm



Minimax algorithm



Minimax algorithm



Minimax algorithm

1. Generate the game tree to as many levels (plies) that time and space constraints allow. The top level is called MAX (as in it's now MAX's turn to move), the next level is called MIN, the next level is MAX, and so on.

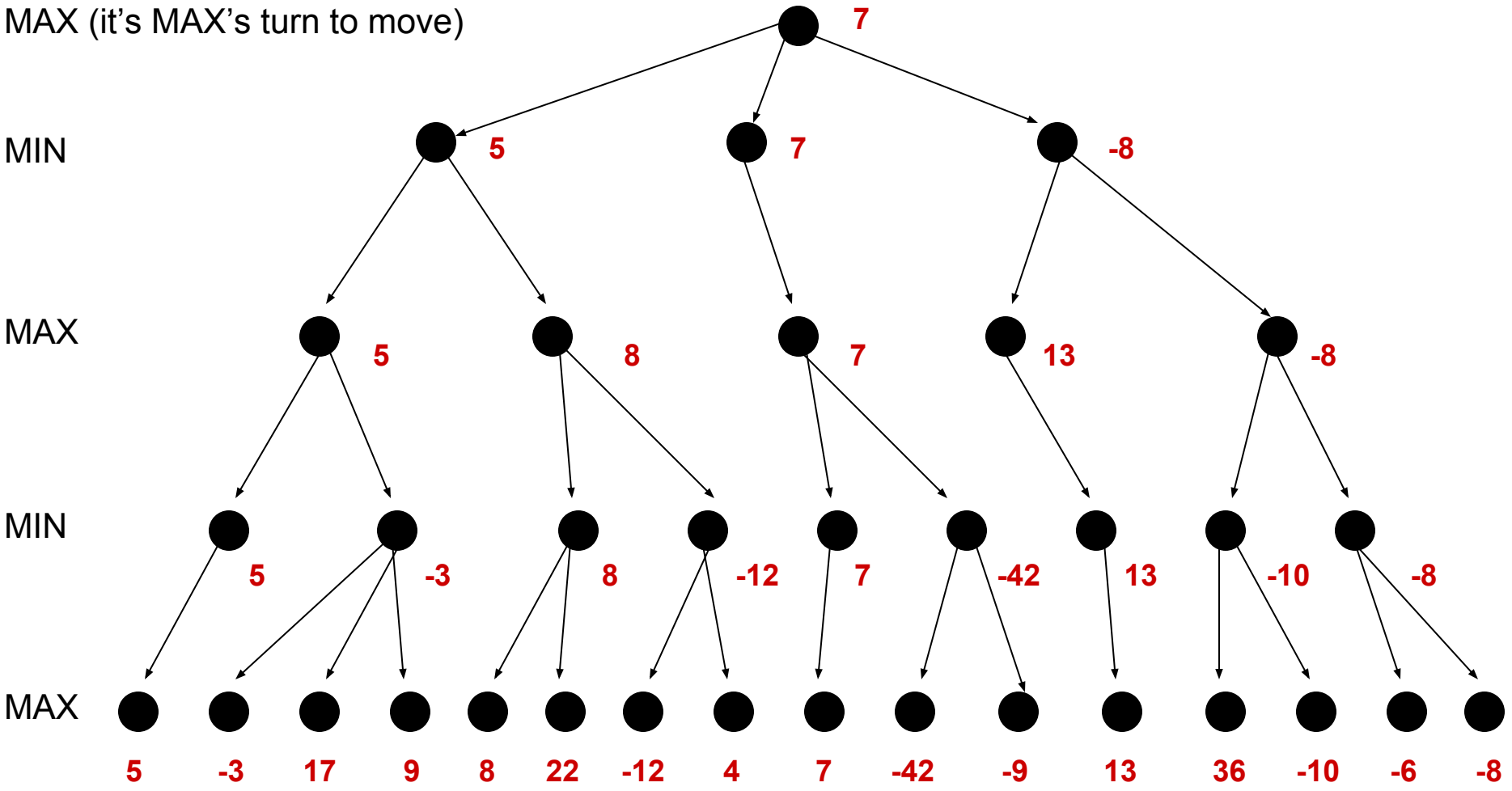
2. Apply the evaluation function to all the terminal (leaf) states/boards to get "goodness" values

3. Use those terminal board values to determine the values to be assigned to the immediate parents:
   a) if the parent is at a MIN level, then the value is the minimum of the values of its children
   b) if the parent is at a MAX level, then the value is the maximum of the values of its children

4. Keep propagating values upward as in step 3

5. When the values reach the top of the game tree, MAX chooses the move indicated by the highest value

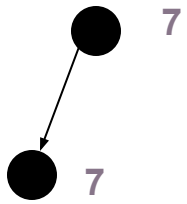
Minimax algorithm



Minimax algorithm

MAX (it's MAX's turn to move)

MIN



Minimax algorithm

A simple but powerful technique...

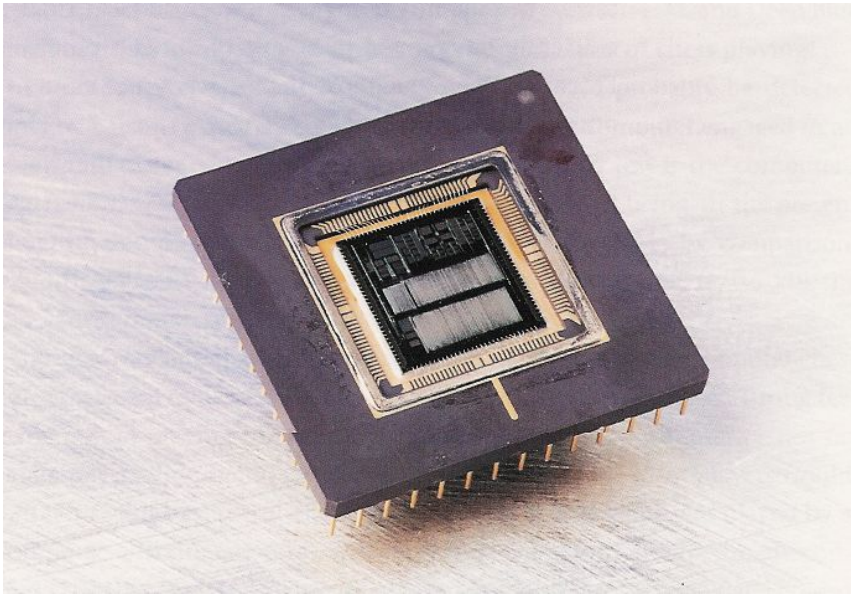
...how powerful is it?

1997

Deep Blue
vital statistics:
• 200,000,000 moves per second

• 480 custom chess-playing chips

1997



1997

Deep Blue
vital statistics:
- 200,000,000 moves per second
- 480 custom chess-playing chips

Garry Kasparov
vital statistics:
- 3 moves per second
- meat

1997

1997

Employing minimax, Deep Blue
defeats Garry Kasparov
3.5 games to 2.5 games

Kasparov wins $400,000

Deep Blue wins $700,000

The human race is humiliated

Questions