Algorithmic Game Theory and Applications

Extensive Form Games

The formal definition of a (strategic) game

Definition: A game in normal or strategic form is a tuple $(N, S_1, S_2, \ldots, S_n, u_1, u_2, \ldots, u_n)$ where

1. $N = \{1, \ldots, n\}$ is a set of players (sometimes called "agents").

2. For each player $i \in N$, there is a set S_i of (pure) strategies.

A vector $(s_1, s_2, \ldots, s_n) \in S_1 \times S_2 \times \ldots \times S_n = S$ is called a strategy profile.

3. For each player $i \in N$, there is a payoff (or utility) function $u_i : S \to \mathbb{R}$ which assigns a numerical value $u_i(s_1, s_2, \ldots, s_n)$ to player i for a given strategy profile $(s_1, s_2, \ldots, s_n)$.

The formal definition of a (strategic) game

Definition: A game in normal or strategic form is a tuple $(N, S_1, S_2, \ldots, S_n, u_1, u_2, \ldots, u_n)$ where

1. $N = \{1, \ldots, n\}$ is a set of players (sometimes called “agents”).

2. For each player $i \in N$, there is a set S_i of (pure) strategies.

A vector $(s_1, s_2, \ldots, s_n) \in S_1 \times S_2 \times \ldots \times S_n = S$ is called a strategy profile.

3. For each player $i \in N$, there is a payoff (or utility) function $u_i : S \to \mathbb{R}$ which assigns a numerical value $u_i(s_1, s_2, \ldots, s_n)$ to player i for a given strategy profile $(s_1, s_2, \ldots, s_n)$.

Games Played Sequentially

Games Played Sequentially

In normal form games, the assumption is that the players pick their entire strategies independently (“simultaneously”).

Games Played Sequentially

In normal form games, the assumption is that the players pick their entire strategies independently ("simultaneously").

In reality, games are often played in "turns": a player chooses a "move" and another player reacts with another "move" and so on.

Games Played Sequentially

In normal form games, the assumption is that the players pick their entire strategies independently (“simultaneously”).

In reality, games are often played in “turns”: a player chooses a “move” and another player reacts with another “move” and so on.

Examples: Chess, Poker, Tichu, most board games

Games Played Sequentially

In normal form games, the assumption is that the players pick their entire strategies independently (“simultaneously”).

In reality, games are often played in “turns”: a player chooses a “move” and another player reacts with another “move” and so on.

Examples: Chess, Poker, Tichu, most board games

More examples: Negotiations, bargaining, politics, life

Games Played Sequentially

In normal form games, the assumption is that the players pick their entire strategies independently ("simultaneously").

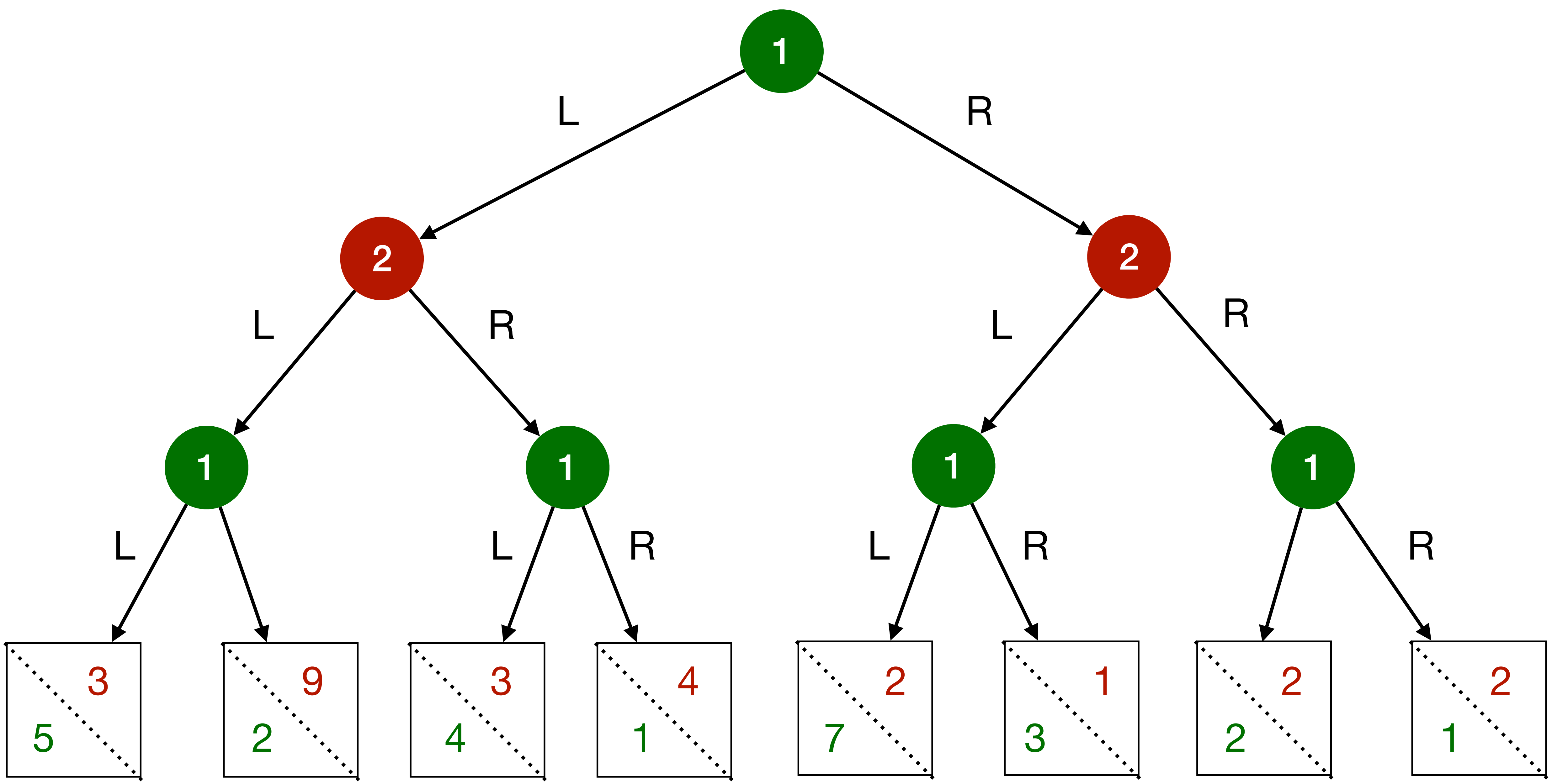
In reality, games are often played in "turns": a player chooses a "move" and another player reacts with another "move" and so on.

Examples: Chess, Poker, Tichu, most board games

More examples: Negotiations, bargaining, politics, life

We would like a mathematical model that captures these situations.

Game Trees



Extensive Form Games

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

4. Z is a set of terminal nodes, $Z \cup H = \emptyset$

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

4. Z is a set of terminal nodes, $Z \cup H = \varnothing$

5. $\chi : A \to 2^A$ is the action function, mapping to each choice node a set of possible actions.

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

4. Z is a set of terminal nodes, $Z \cup H = \emptyset$

5. $\chi : A \to 2^A$ is the action function, mapping to each choice node a set of possible actions.

6. $\rho : H \to N$ is the player function, which determines which player takes an action at each choice node.

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

4. Z is a set of terminal nodes, $Z \cup H = \emptyset$

5. $\chi : A \rightarrow 2^A$ is the action function, mapping to each choice node a set of possible actions.

6. $\rho : H \rightarrow N$ is the player function, which determines which player takes an action at each choice node.

7. $\sigma : H \times A \rightarrow H \cup Z$ is the successor function, which maps a choice node and an action to another node (choice or terminal).

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

4. Z is a set of terminal nodes, $Z \cup H = \varnothing$

5. $\chi : A \to 2^A$ is the action function, mapping to each choice node a set of possible actions.

6. $\rho : H \to N$ is the player function, which determines which player takes an action at each choice node.

7. $\sigma : H \times A \to H \cup Z$ is the successor function, which maps a choice node and an action to another node (choice or terminal).

8. $u_i : Z \to \mathbb{R}$ is a utility function for each player i, mapping terminal nodes to real-valued utilities.

Example: Sharing Game

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both
- the sister keeps both

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both
- the sister keeps both
- they each keep one

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both
- the sister keeps both
- they each keep one

Then the sister chooses to

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both
- the sister keeps both
- they each keep one

Then the sister chooses to

- accept the split

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

<u>First the brother suggests a split</u>, either

- he keeps both
- the sister keeps both
- they each keep one

<u>Then the sister chooses to</u>

- accept the split
- reject the split

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both
- the sister keeps both
- they each keep one

Then the sister chooses to

- accept the split
- reject the split

If she accepts, they each get their allocated present. If she does not, neither of them gets any present. Both siblings get a value of i from keeping i presents.

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both
- the sister keeps both
- they each keep one

3 minute exercise: Model this as an extensive form game, by writing down the game tree.

Then the sister chooses to

- accept the split
- reject the split

If she accepts, they each get their allocated present. If she does not, neither of them gets any present. Both siblings get a value of i from keeping i presents.

Example: Sharing Game

Two siblings (brother and sister) receive two identical presents from their parents, and have decided to share them according to the following agreement:

First the brother suggests a split, either

- he keeps both
- the sister keeps both
- they each keep one

Then the sister chooses to

- accept the split
- reject the split

3 minute exercise: Model this as an extensive form game, by writing down the game tree.

Another 2 minutes: Identify all the elements of the extensive form game tuple.

If she accepts, they each get their allocated present. If she does not, neither of them gets any present. Both siblings get a value of i from keeping i presents.

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

4. Z is a set of terminal nodes, $Z \cup H = \emptyset$

5. $\chi : A \to 2^A$ is the action function, mapping to each choice node a set of possible actions.

6. $\rho : H \to N$ is the player function, which determines which player takes an action at each choice node.

7. $\sigma : H \times A \to H \cup Z$ is the successor function, which maps a choice node and an action to another node (choice or terminal).

8. $u_i : Z \to \mathbb{R}$ is a utility function for each player i, mapping terminal nodes to real-valued utilities.

Pure Strategies

Pure Strategies

A pure strategy of player i in an extensive form game is a choice of action for each time the player is the one making a choice at a choice node.

Pure Strategies

A pure strategy of player i in an extensive form game is a choice of action for each time the player is the one making a choice at a choice node.

Formally, the set of pure strategies S_i of player i is

$$S_i = \prod_{h \in H : \rho(h) = i} \chi(h)$$

Pure Strategies

A pure strategy of player i in an extensive form game is a choice of action for each time the player is the one making a choice at a choice node.

Formally, the set of pure strategies S_i of player i is

$$S_i = \prod_{h \in H : \rho(h) = i} \chi(h)$$

In the example before, we had

$S_1 = \{2-0, 1-1, 0-2\}$ and

$S_2 = \{\text{Yes}, \text{No}\}$

Pure Strategies

A pure strategy of player i in an extensive form game is a choice of action for each time the player is the one making a choice at a choice node.

Formally, the set of pure strategies S_i of player i is

$$S_i = \prod_{h \in H : \rho(h) = i} \chi(h)$$

In the example before, we had

$S_1 = \{2 - 0, 1 - 1, 0 - 2\}$ and

$S_2 = \{\text{Yes, No}\}$ Not quite, subtlety!

Pure Strategies

A pure strategy of player i in an extensive form game is a choice of action for each time the player is the one making a choice at a choice node.

Formally, the set of pure strategies S_i of player i is

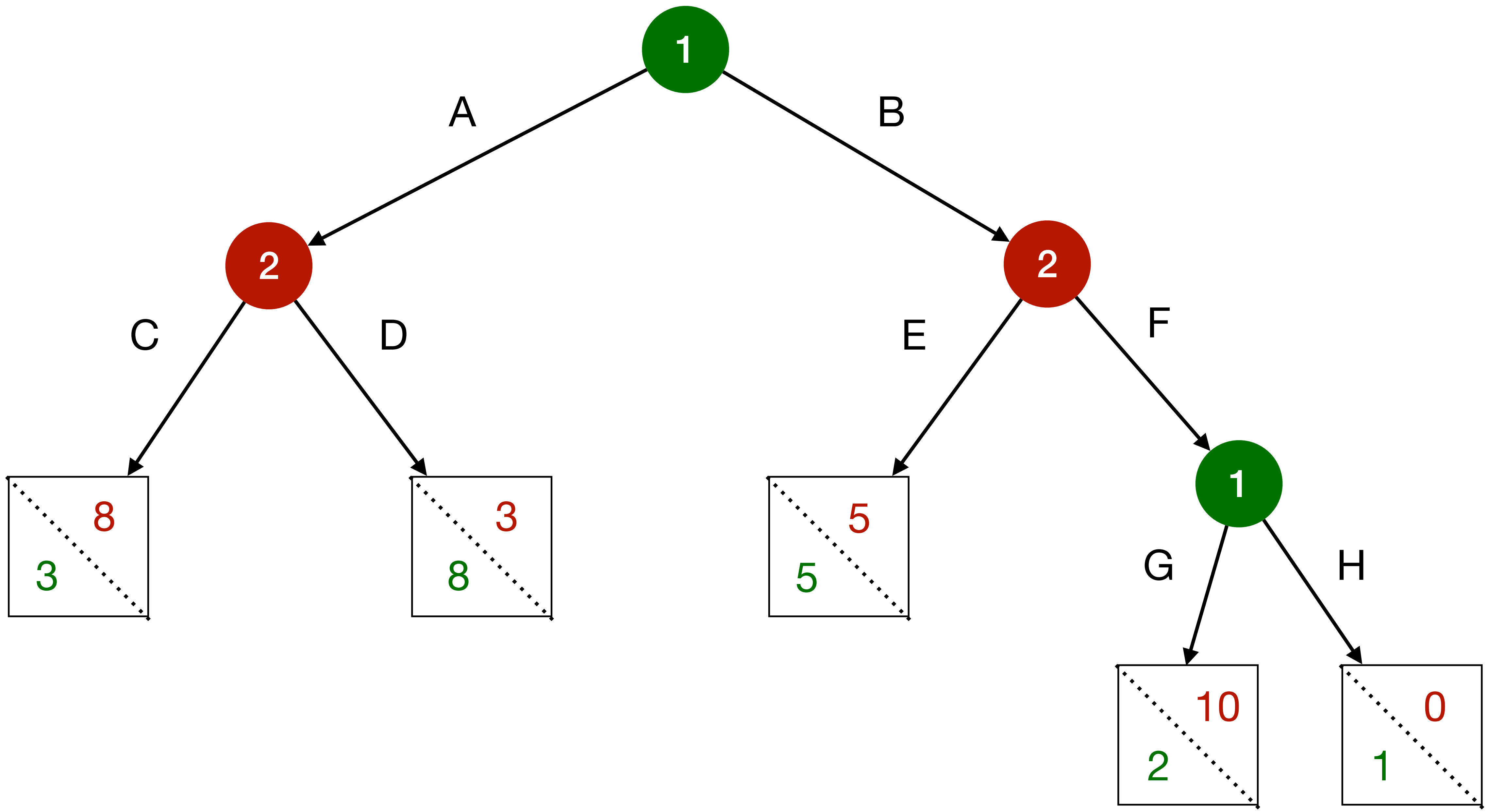
$$S_i = \prod_{h \in H : \rho(h) = i} \chi(h)$$

In the example before, we had

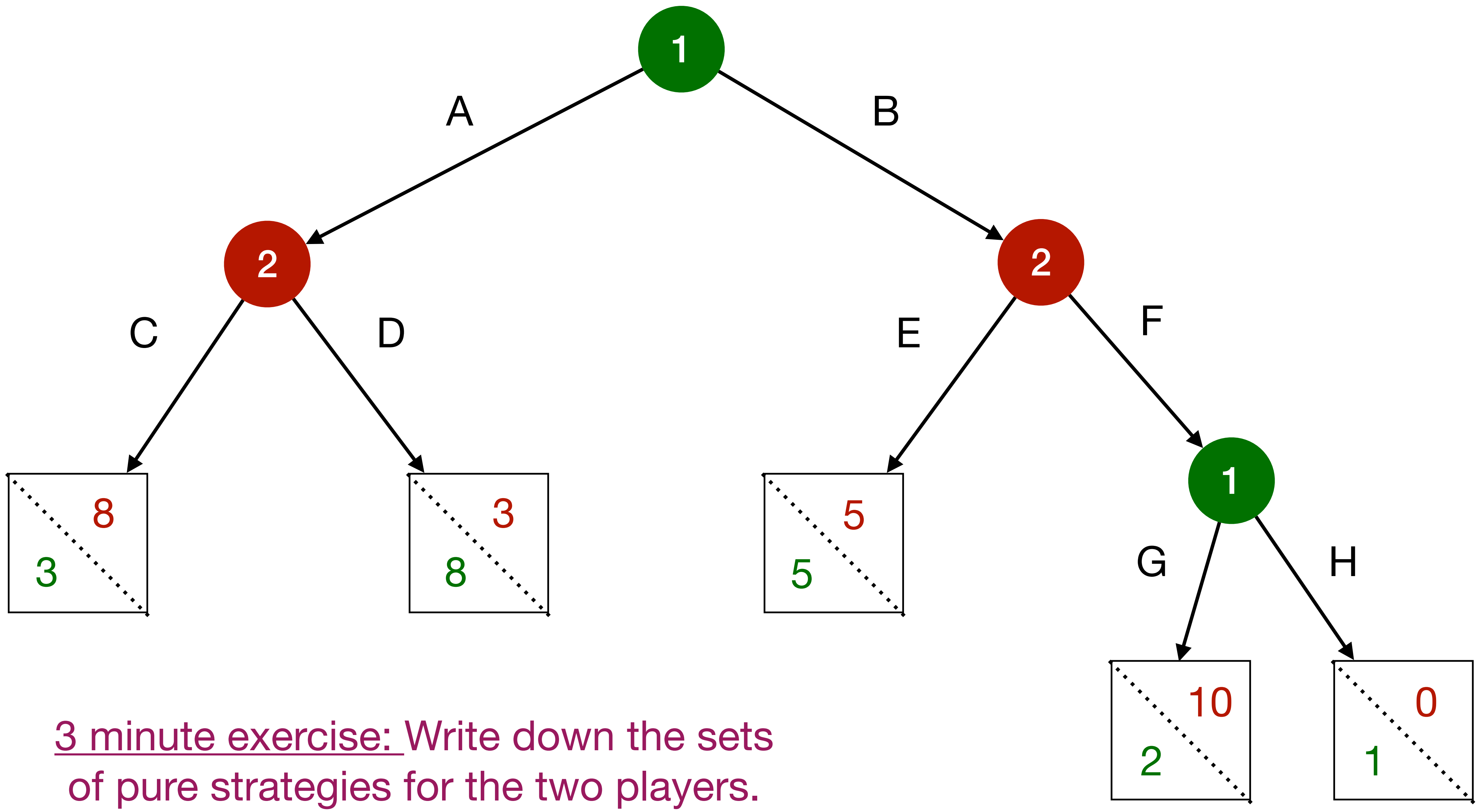
$S_1 = \{2 - 0, 1 - 1, 0 - 2\}$ and

$S_2 = \{(\text{Yes, Yes, Yes}), (\text{Yes, Yes, No}), (\text{Yes, No, Yes}), \ldots\}$

Example

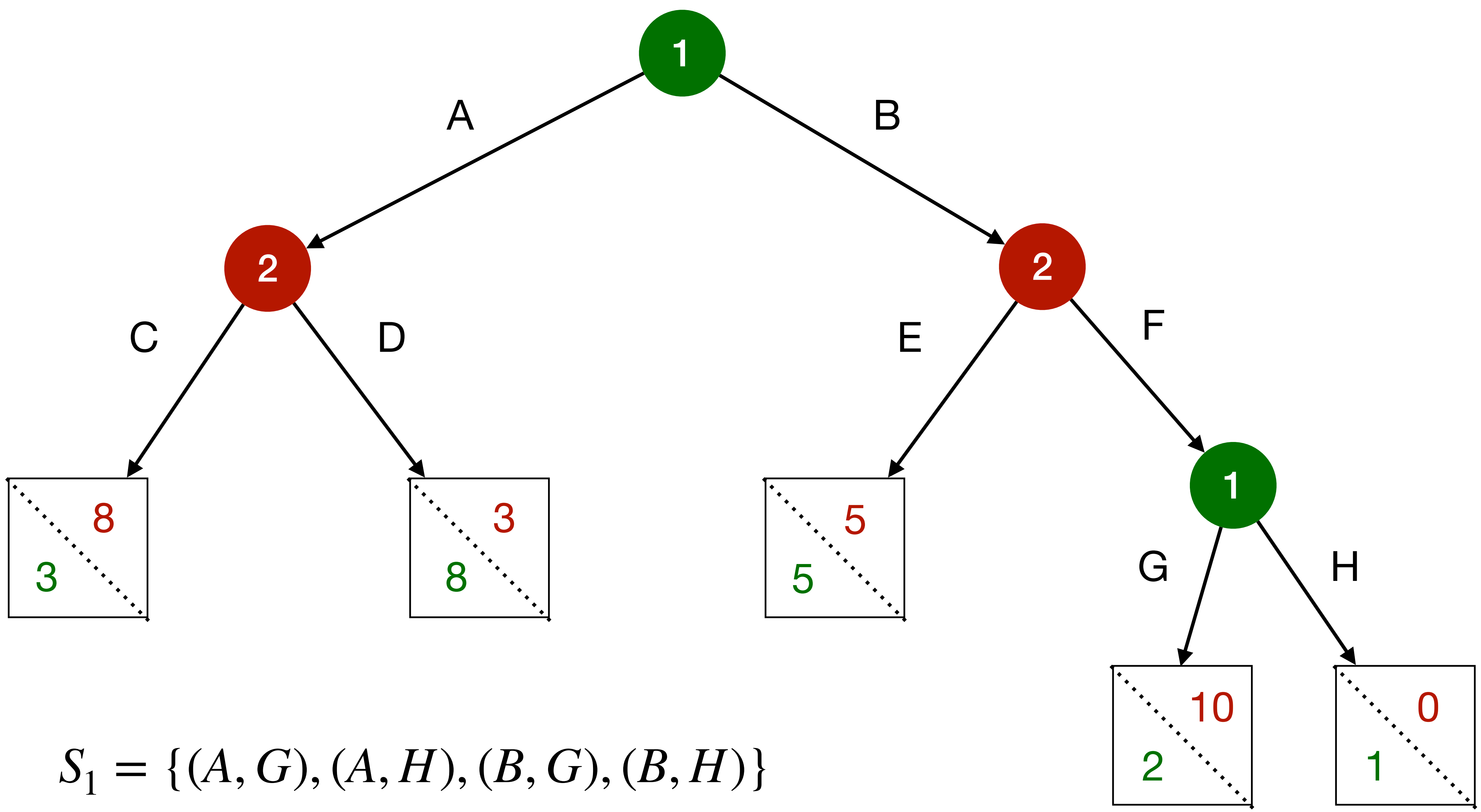


Example



<u>3 minute exercise:</u> Write down the sets of pure strategies for the two players.

Example



$$S_1 = \{(A, G), (A, H), (B, G), (B, H)\}$$
$$S_2 = \{(C, E), (C, F), (D, E), (D, F)\}$$

In normal form

	(C,E)	(C,F)	(D,E)	(D,F)
(A,G)	3, 8	3, 8	8, 3	8, 3
(A,H)	3, 8	3, 8	8, 3	8, 3
(B,G)	5, 5	2, 10	5, 5	2, 10
(B,H)	5, 5	1, 0	5, 5	1, 0

In normal form

	(C,E)	(C,F)	(D,E)	(D,F)
(A,G)	3, 8	3, 8	8, 3	8, 3
(A,H)	3, 8	3, 8	8, 3	8, 3
(B,G)	5, 5	2, 10	5, 5	2, 10
(B,H)	5, 5	1, 0	5, 5	1, 0

Some outcomes are redundant, as they appear more times in the NFG than in the EFG.

In normal form

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

Some outcomes are redundant, as they appear more times in the NFG than in the EFG.

We can always transform a perfect information EFG to a NFG.

In normal form

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

Some outcomes are redundant, as they appear more times in the NFG than in the EFG.

We can always transform a perfect information EFG to a NFG.

This transformation might result in an exponential blowup in the representation of the game!

In normal form

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

In normal form

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

Can you find a pure Nash equilibrium of this game?

In normal form

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

Can you find a pure Nash equilibrium of this game?

We can actually find three!

In normal form

	(C,E)	(C,F)	(D,E)	(D,F)
(A,G)	3, 8	3, 8	8, 3	8, 3
(A,H)	3, 8	3, 8	8, 3	8, 3
(B,G)	5, 5	2, 10	5, 5	2, 10
(B,H)	5, 5	1, 0	5, 5	1, 0

Can you find a pure Nash equilibrium of this game?

We can actually find three!

Is this a coincidence?

In normal form

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

Can you find a pure Nash equilibrium of this game?

We can actually find three!

Is this a coincidence?

In normal form

	(C,E)	(C,F)	(D,E)	(D,F)
(A,G)	3, 8	3, 8	8, 3	8, 3
(A,H)	3, 8	3, 8	8, 3	8, 3
(B,G)	5, 5	2, 10	5, 5	2, 10
(B,H)	5, 5	1, 0	5, 5	1, 0

Can you find a pure Nash equilibrium of this game?

We can actually find three!

Is this a coincidence?

In normal form

	(C,E)	(C,F)	(D,E)	(D,F)
(A,G)	3, 8	3, 8	8, 3	8, 3
(A,H)	3, 8	3, 8	8, 3	8, 3
(B,G)	5, 5	2, 10	5, 5	2, 10
(B,H)	5, 5	1, 0	5, 5	1, 0

Can you find a pure Nash equilibrium of this game?

We can actually find three!

Is this a coincidence?

Equilibrium in Extensive Form Games

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

Intuition: Players take turns, and everyone gets to see everything that has happened thus fur, so it does not seem that randomisation is necessary.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

Intuition: Players take turns, and everyone gets to see everything that has happened thus fur, so it does not seem that randomisation is necessary.

We also have a (naive, exponential time) algorithm for finding (pure) Nash equilibria in perfect information extensive form games.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

Intuition: Players take turns, and everyone gets to see everything that has happened thus fur, so it does not seem that randomisation is necessary.

We also have a (naive, exponential time) algorithm for finding (pure) Nash equilibria in perfect information extensive form games.

(Which one?)

A closer look

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

Can you find a pure Nash equilibrium of this game?

We can actually find three!

Is this a coincidence?

A closer look

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	3, 8	8, 3	8, 3
(A, H)	3, 8	3, 8	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	5, 5	1, 0	5, 5	1, 0

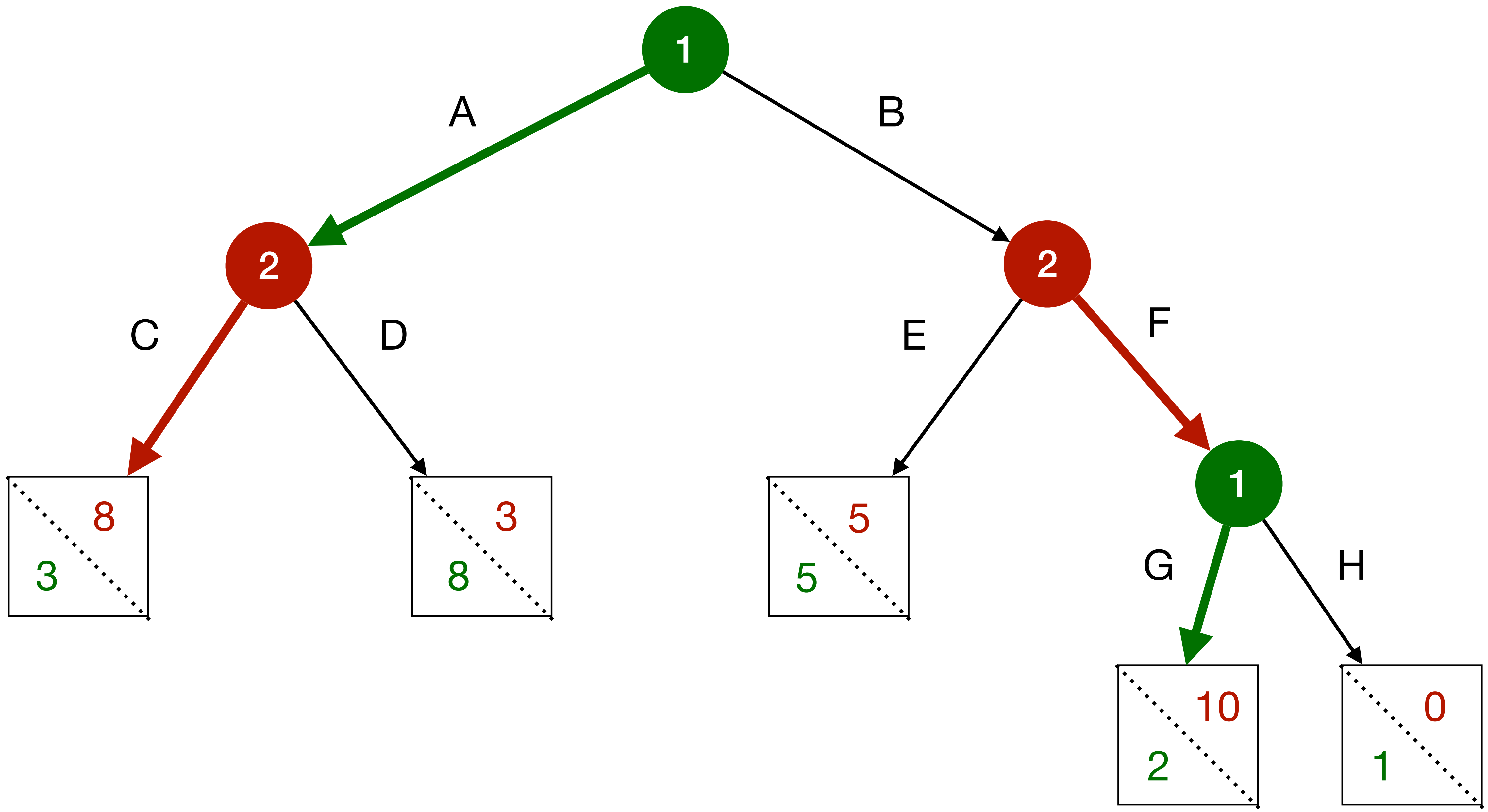
Can you find a pure Nash equilibrium of this game?

We can actually find three!

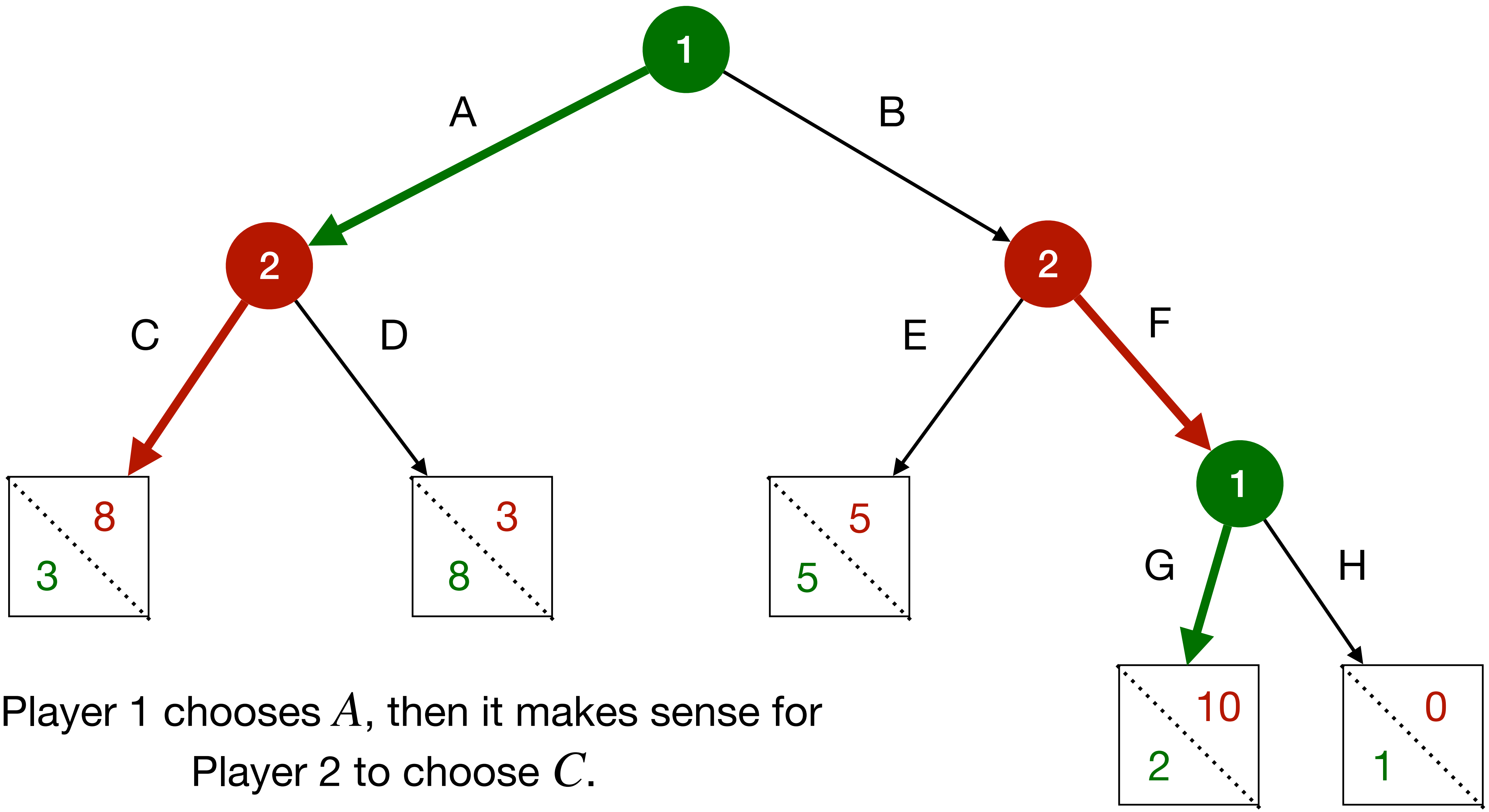
Is this a coincidence?

Let's look at those on the game tree.

Equilibrium 1: (A, G), (C, F)

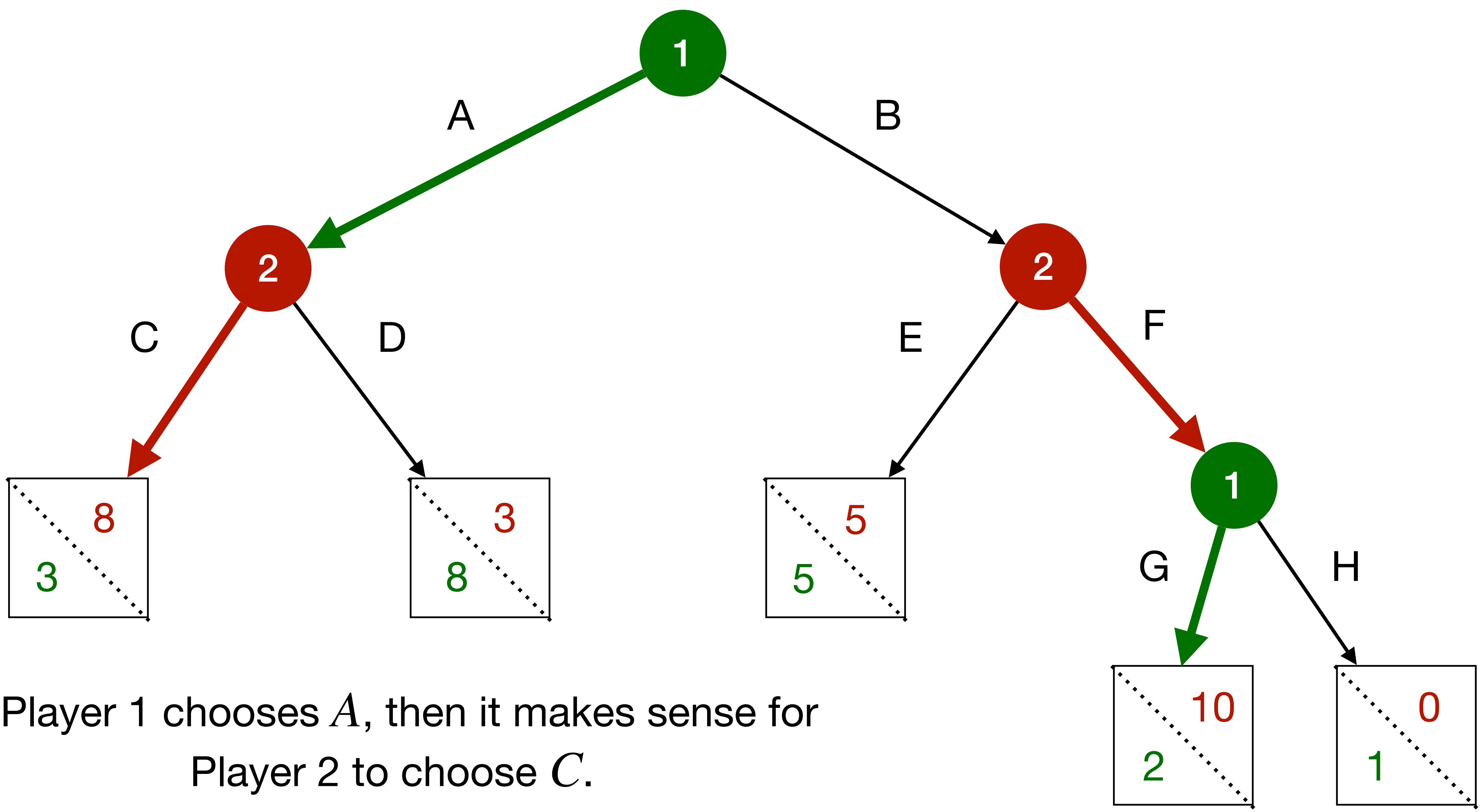


Equilibrium 1: (A, G), (C, F)



If Player 1 chooses A, then it makes sense for Player 2 to choose C.

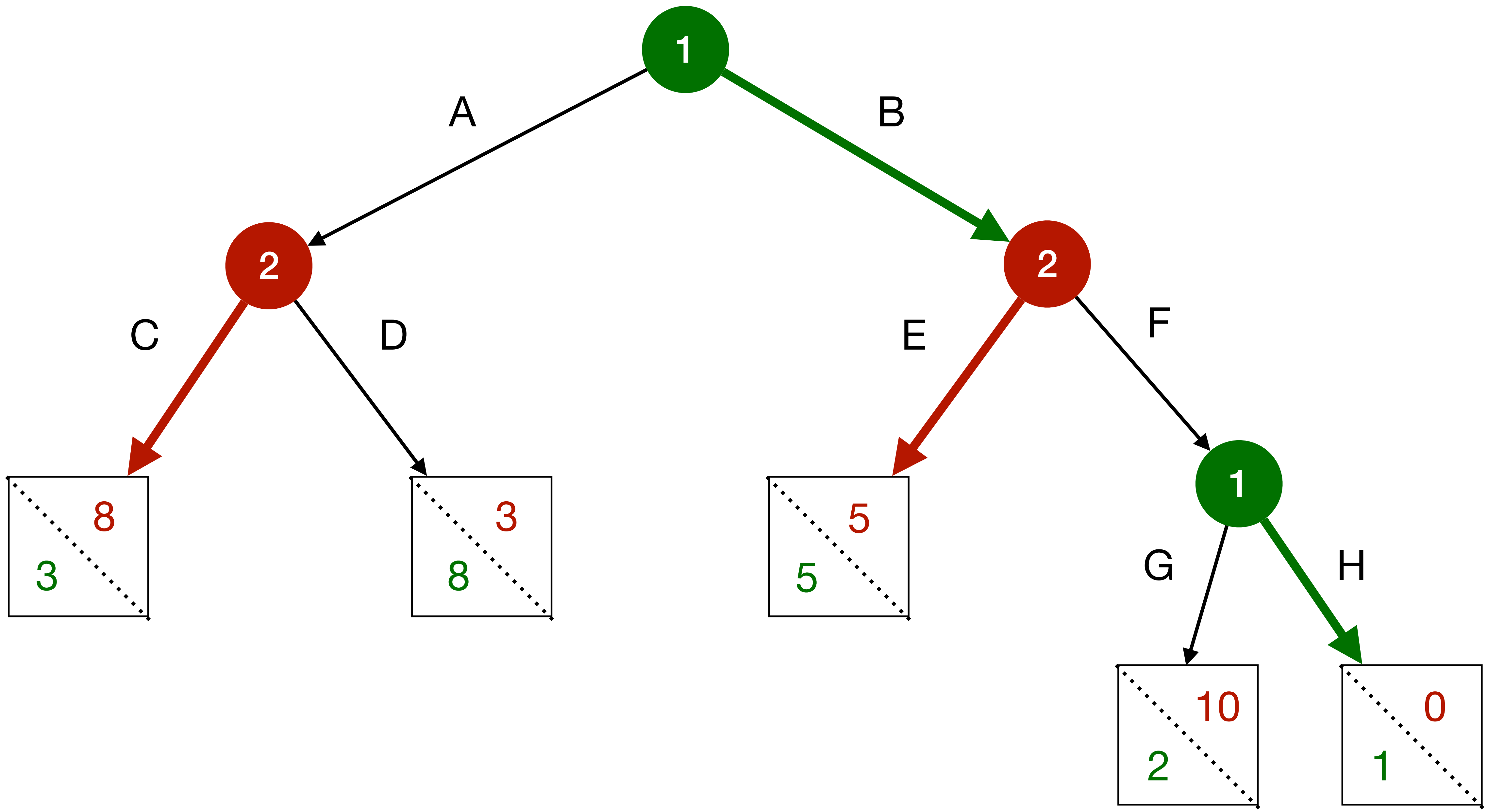
Equilibrium 1: $(A, G), (C, F)$



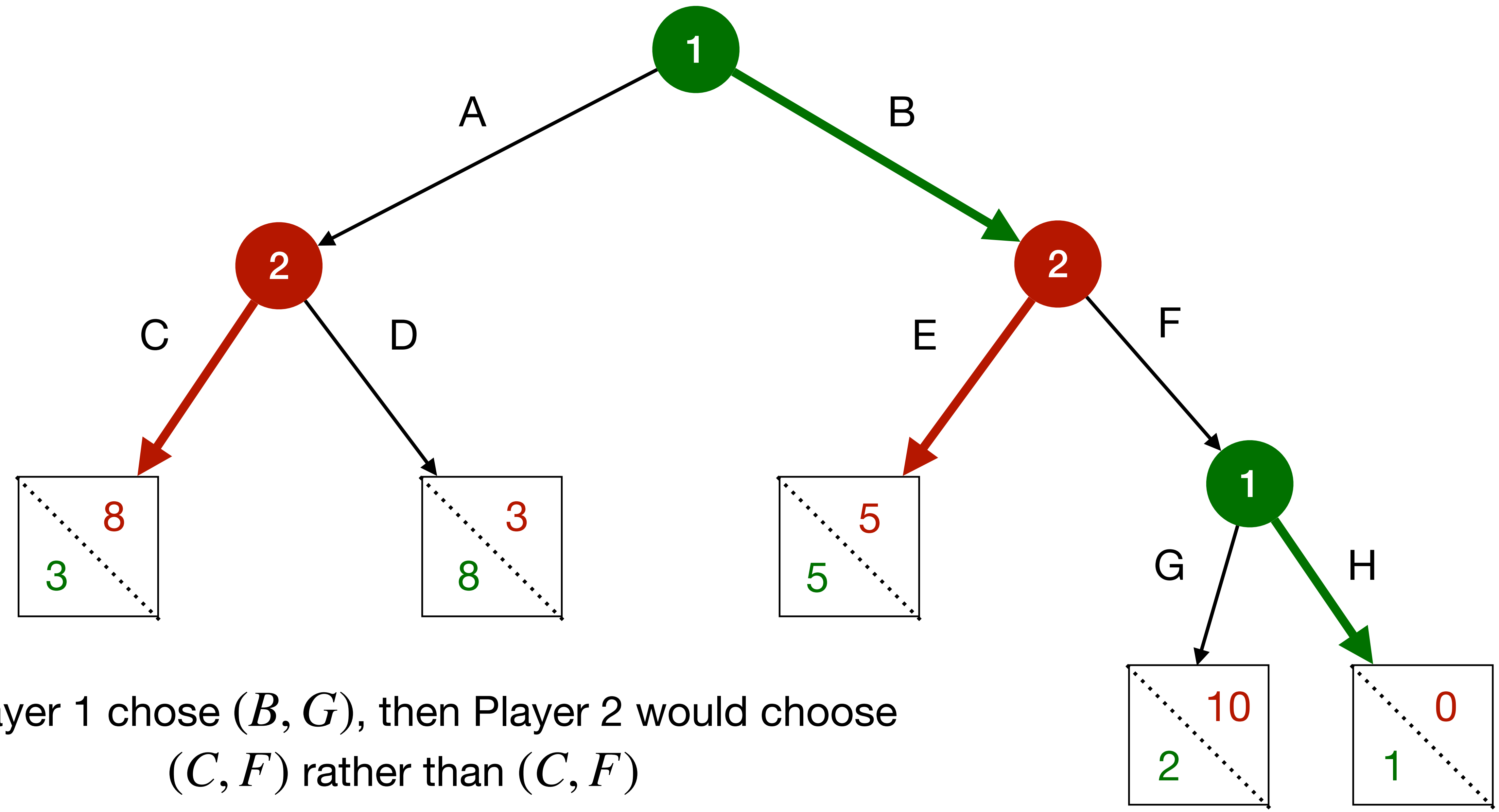
If Player 1 chooses A, then it makes sense for Player 2 to choose C.

If Player 2 played (C, E) instead, it would make sense for Player 1 to choose B instead.

Equilibrium 3: (B, H), (C, E)

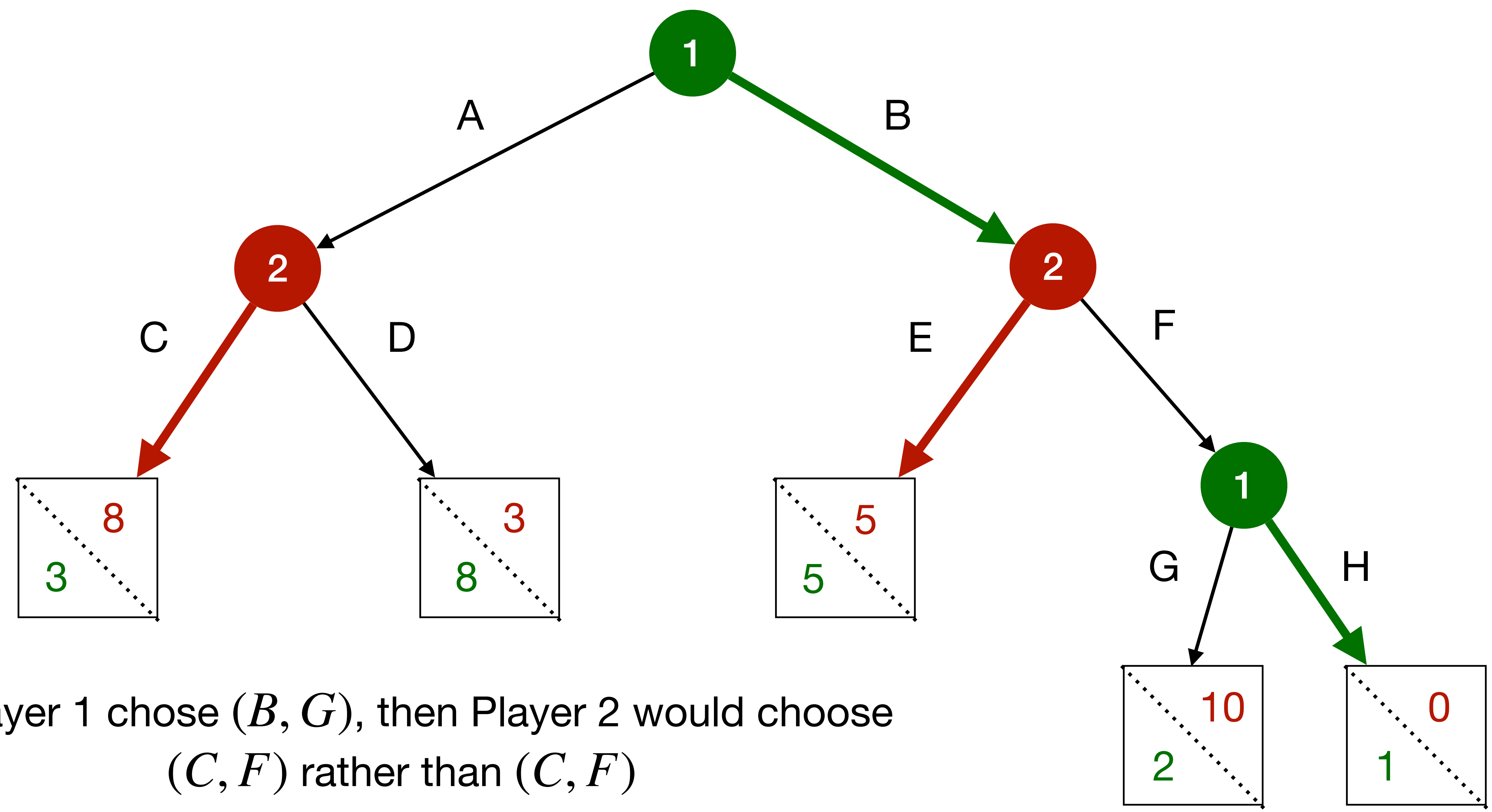


Equilibrium 3: (B, H), (C, E)



If Player 1 chose (B, G), then Player 2 would choose (C, F) rather than (C, F)

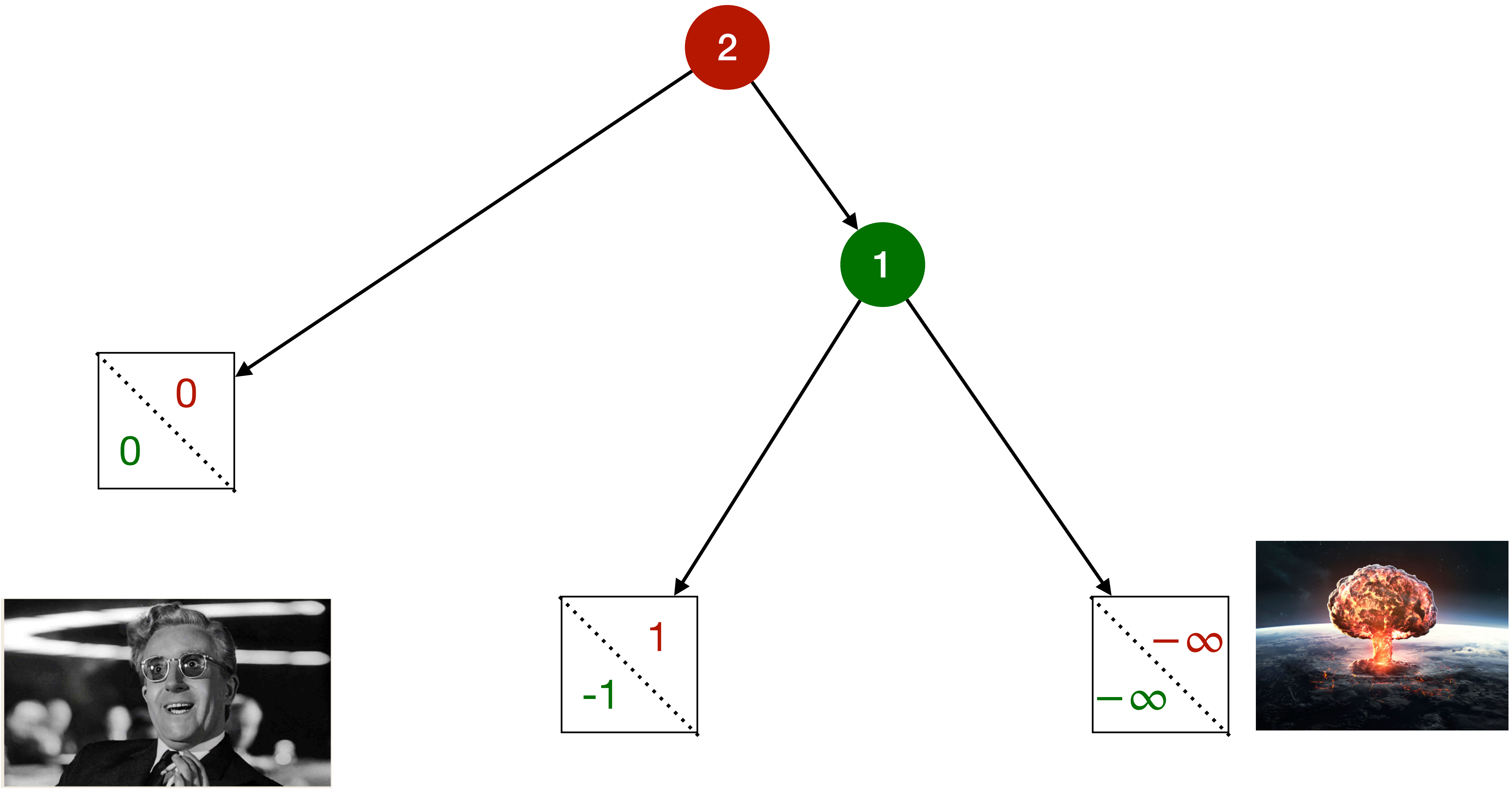
Equilibrium 3: (B, H), (C, E)



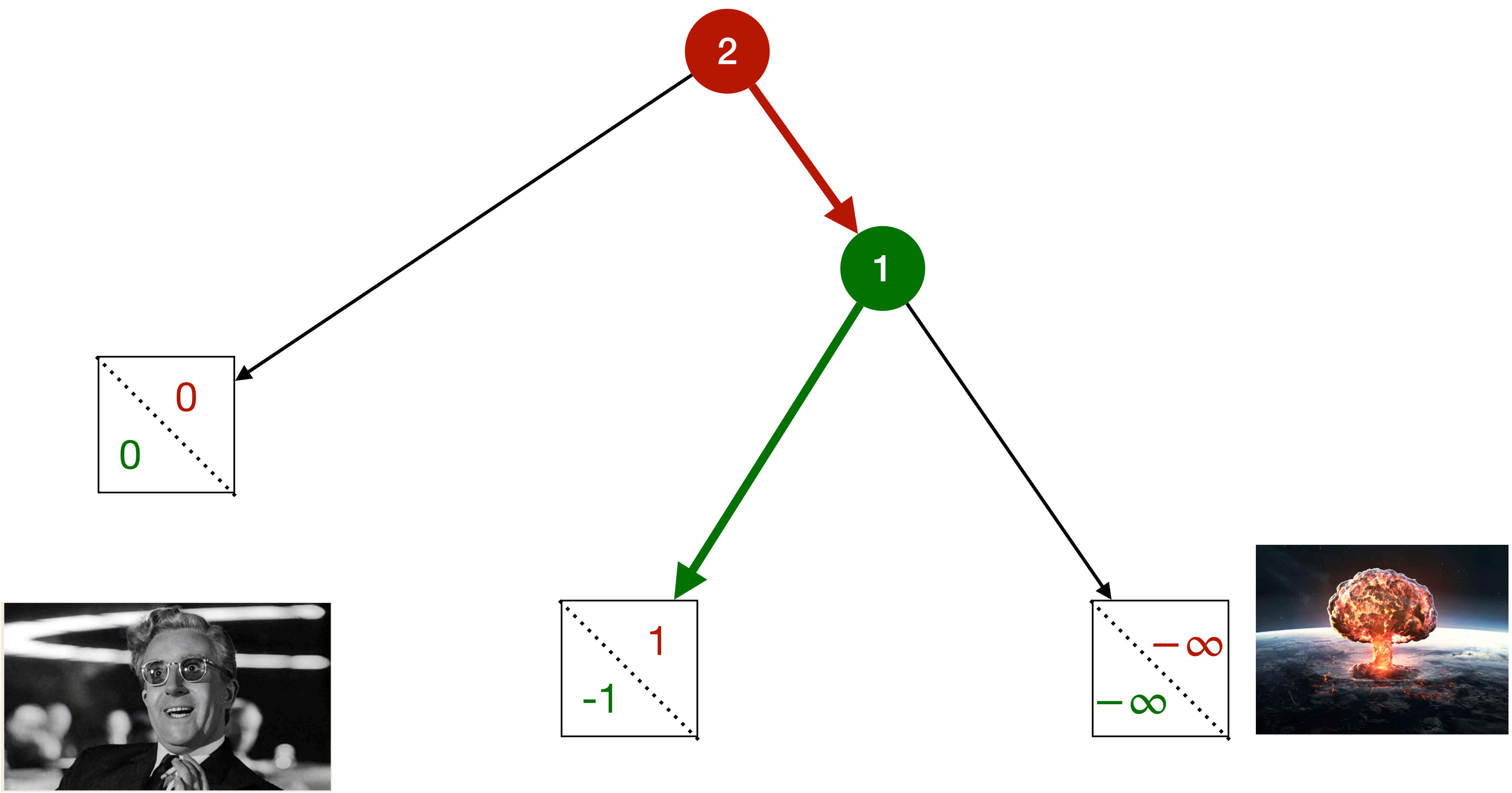
If Player 1 chose (B, G), then Player 2 would choose (C, F) rather than (C, F)

The only reason why Player 2 chooses (C, E) is because Player 1 is *threatening* with a worse action.

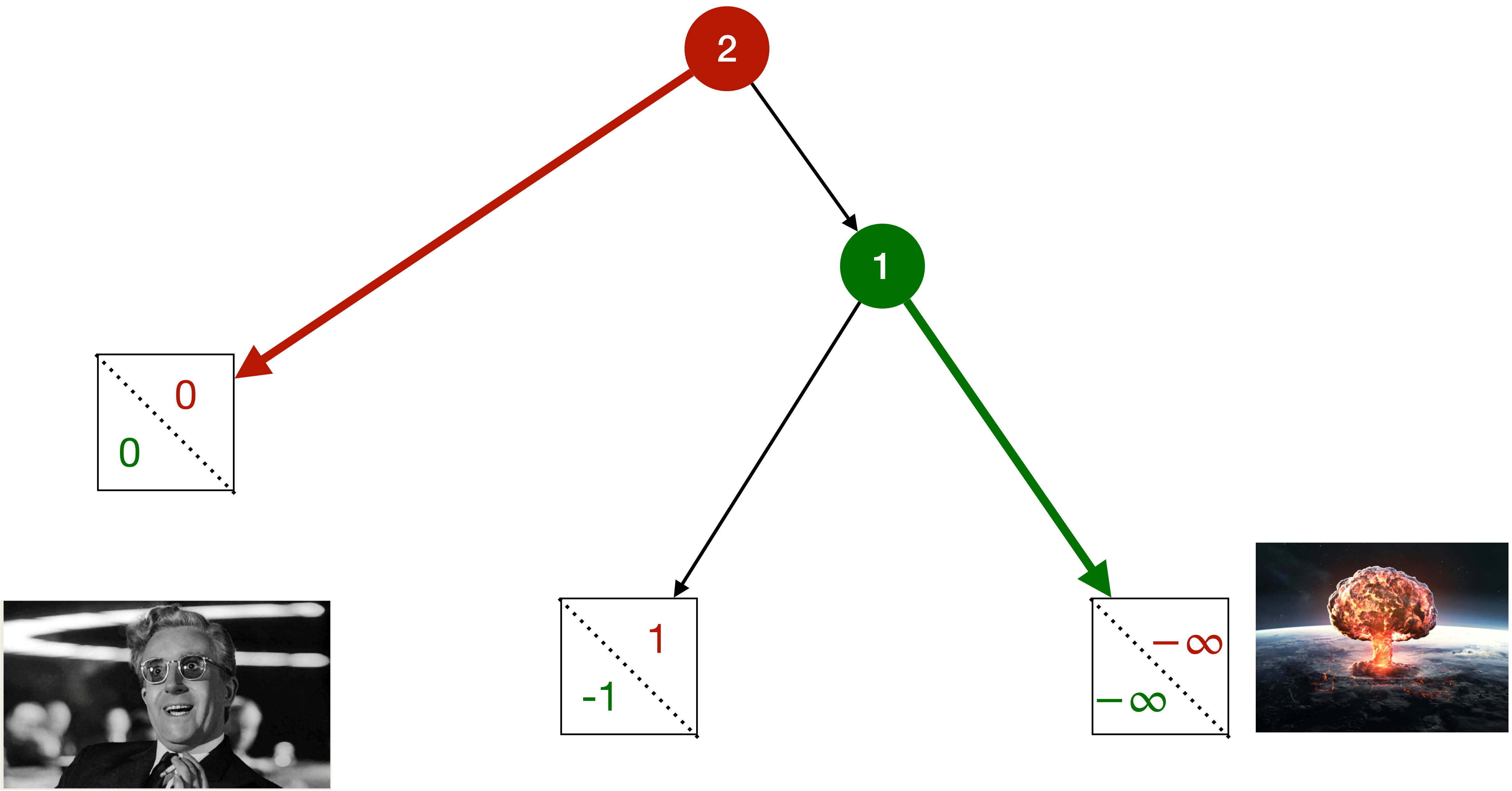
The Doomsday Game



This is a PNE

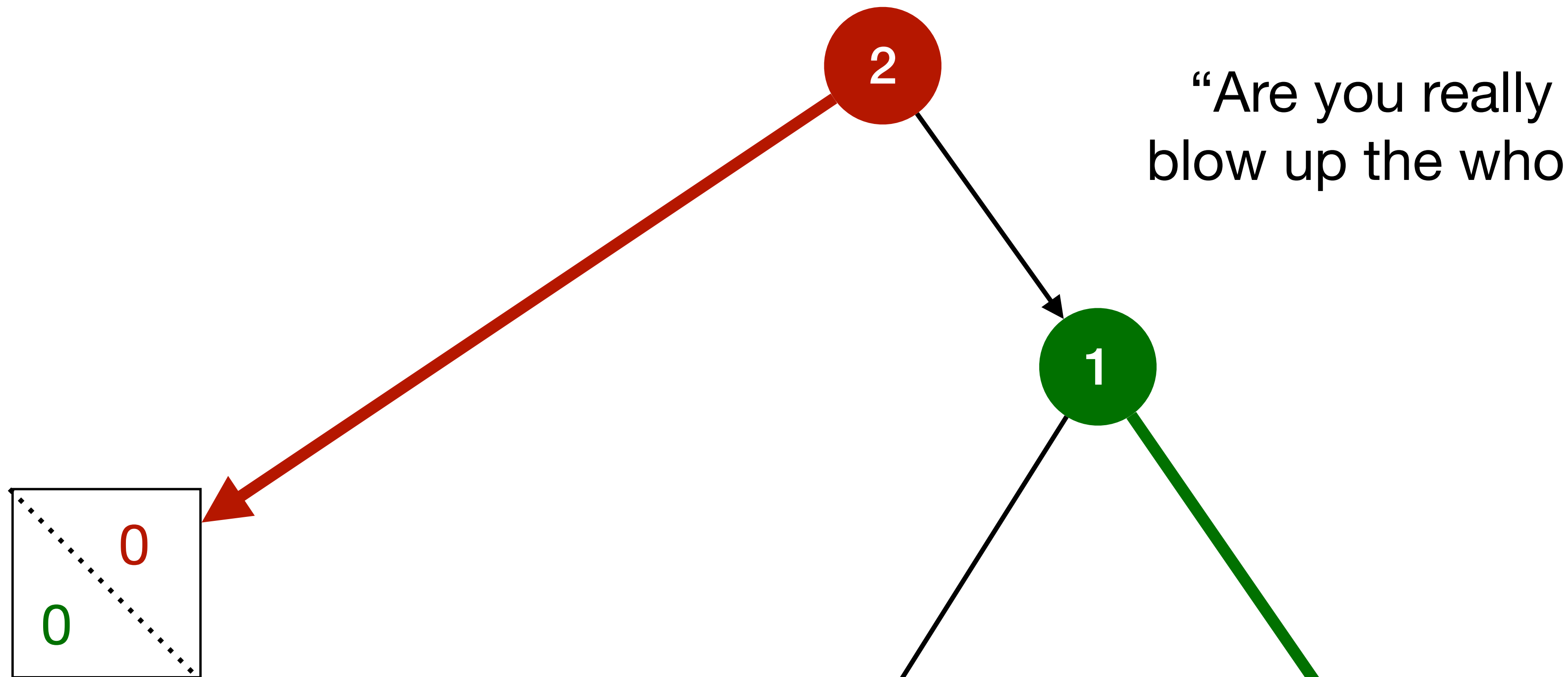


This is also PNE



This is also PNE

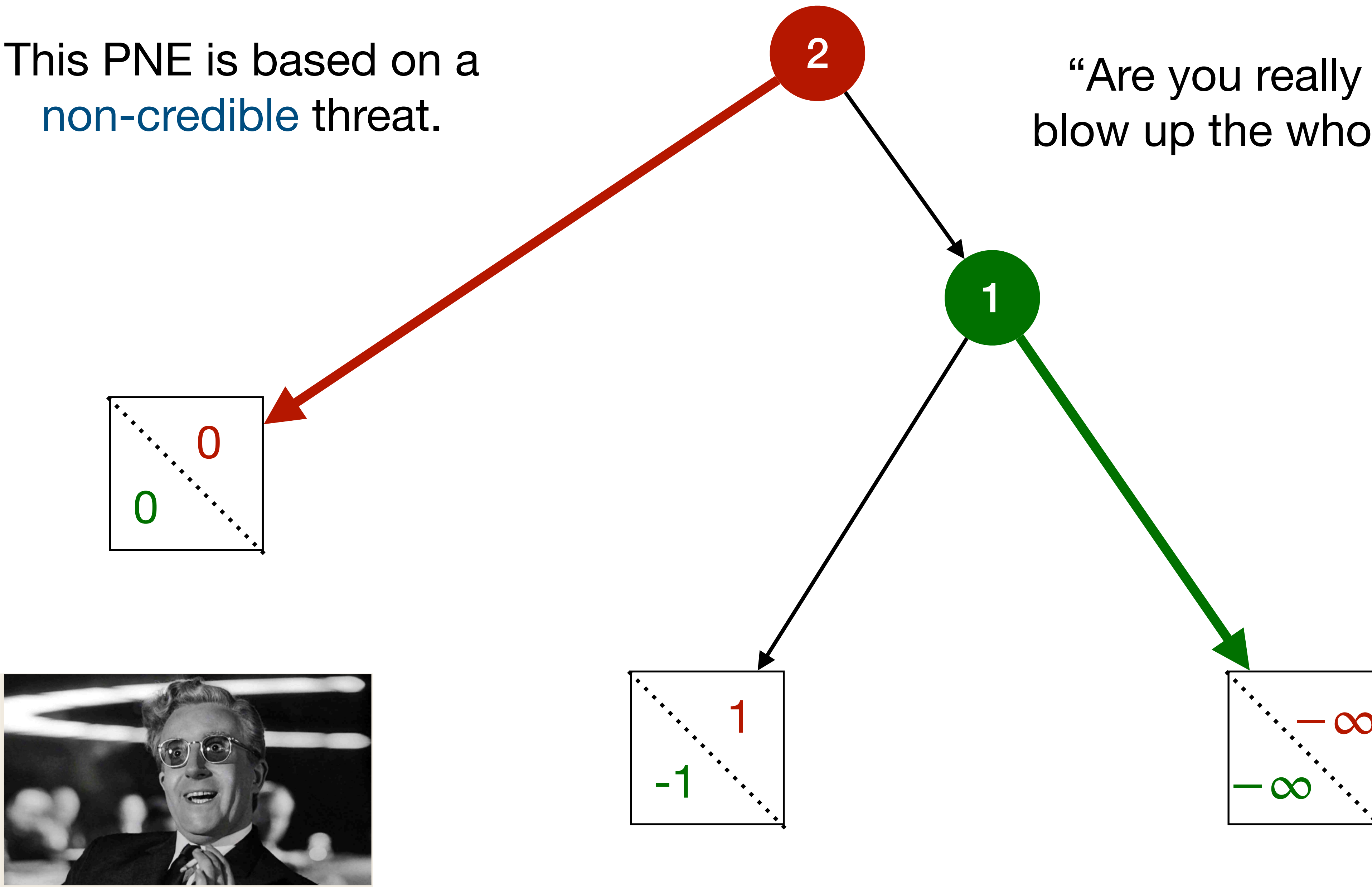
“Are you really going to blow up the whole world?”



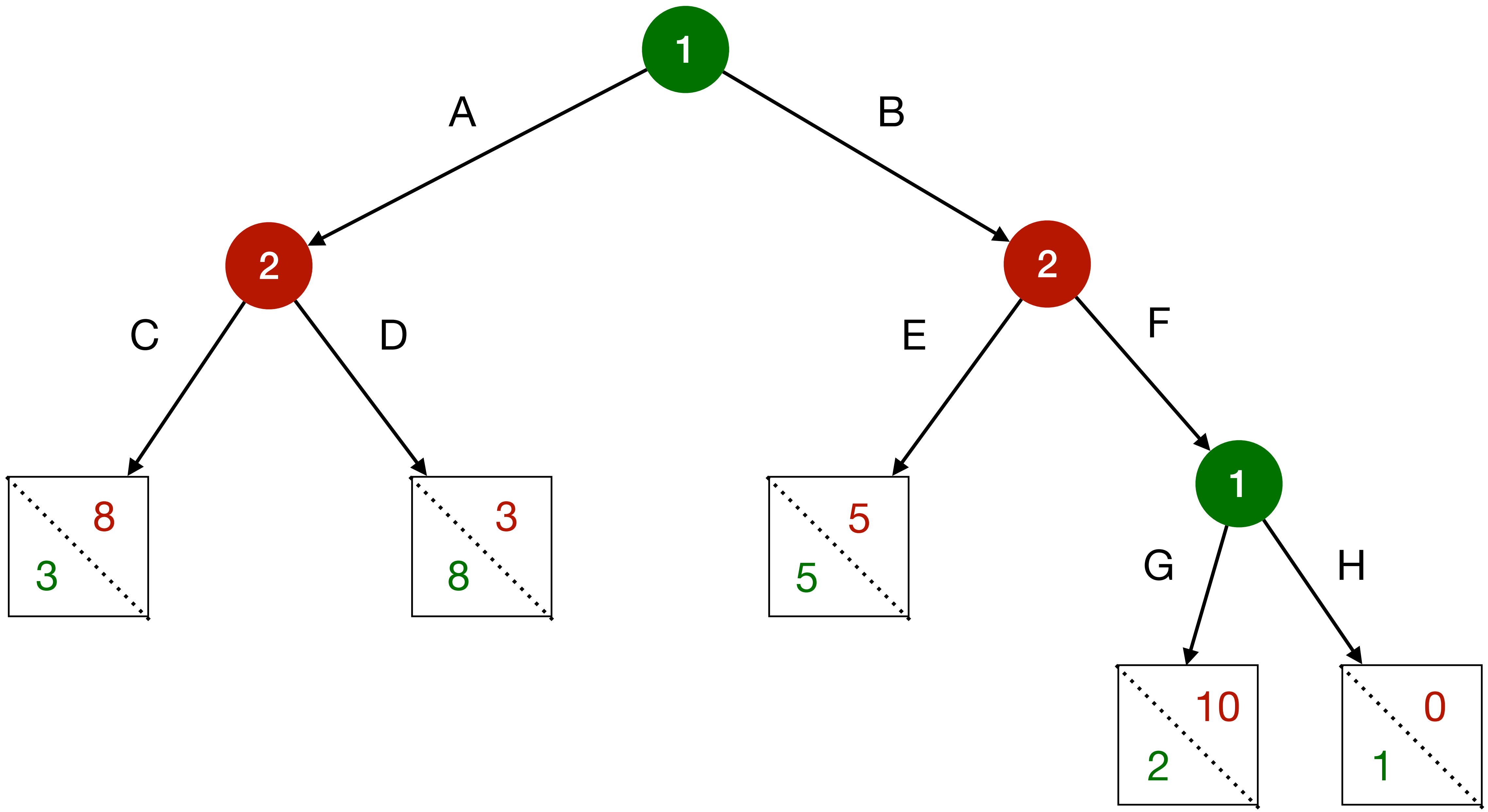

This is also PNE

This PNE is based on a non-credible threat.

“Are you really going to blow up the whole world?”

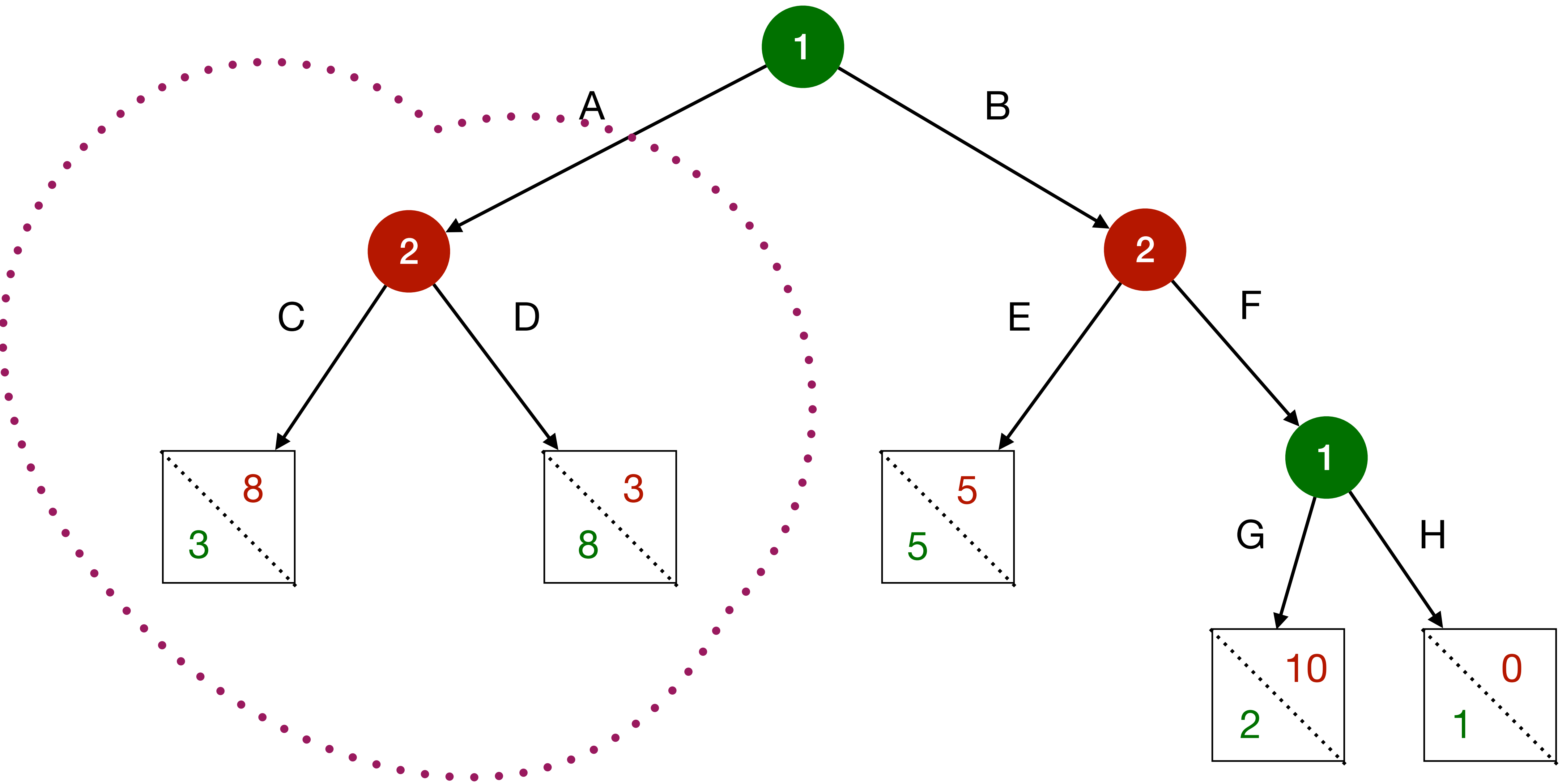


Subgames



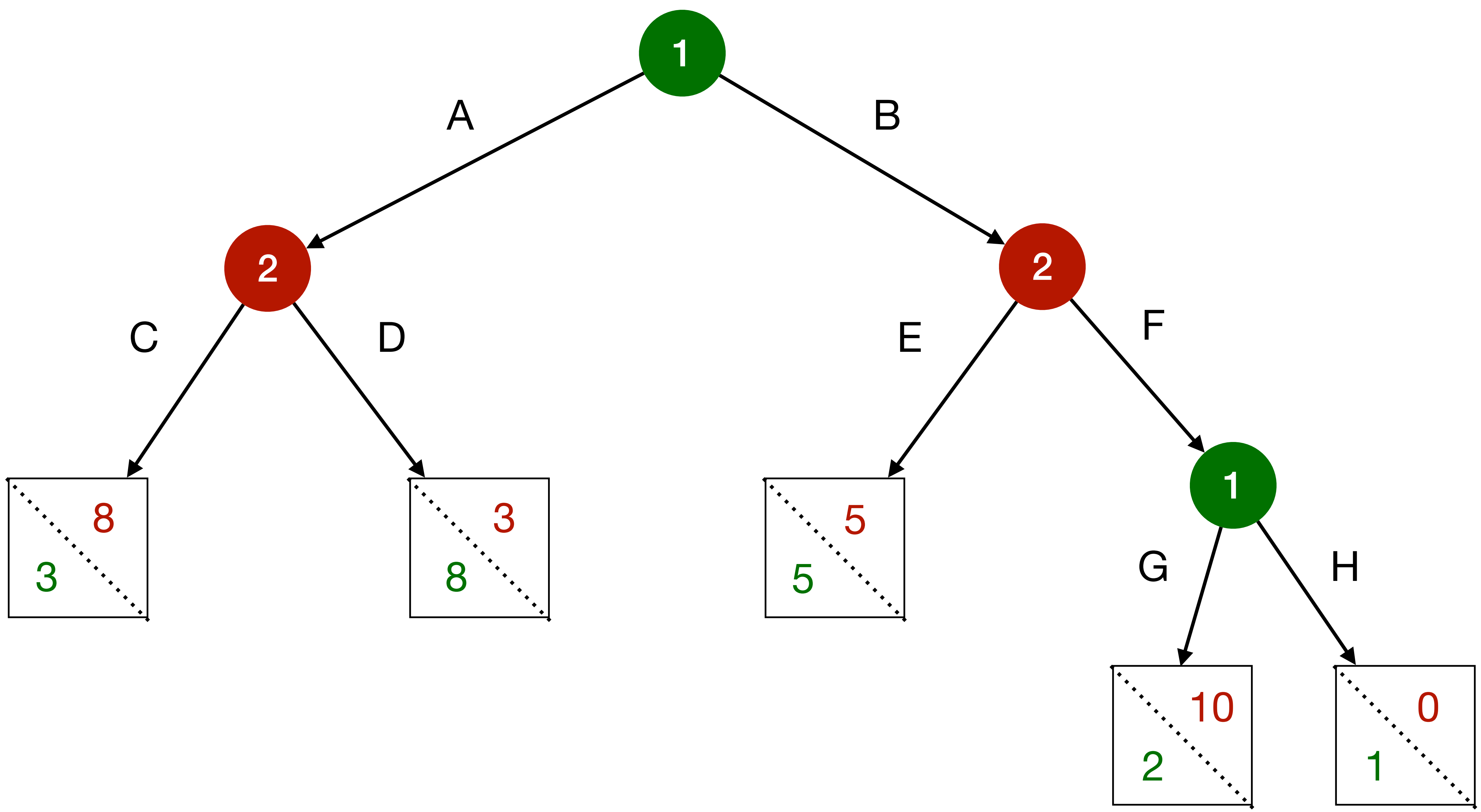
Subgame: A game rooted at node h restricted to the subtree under h.

Subgames



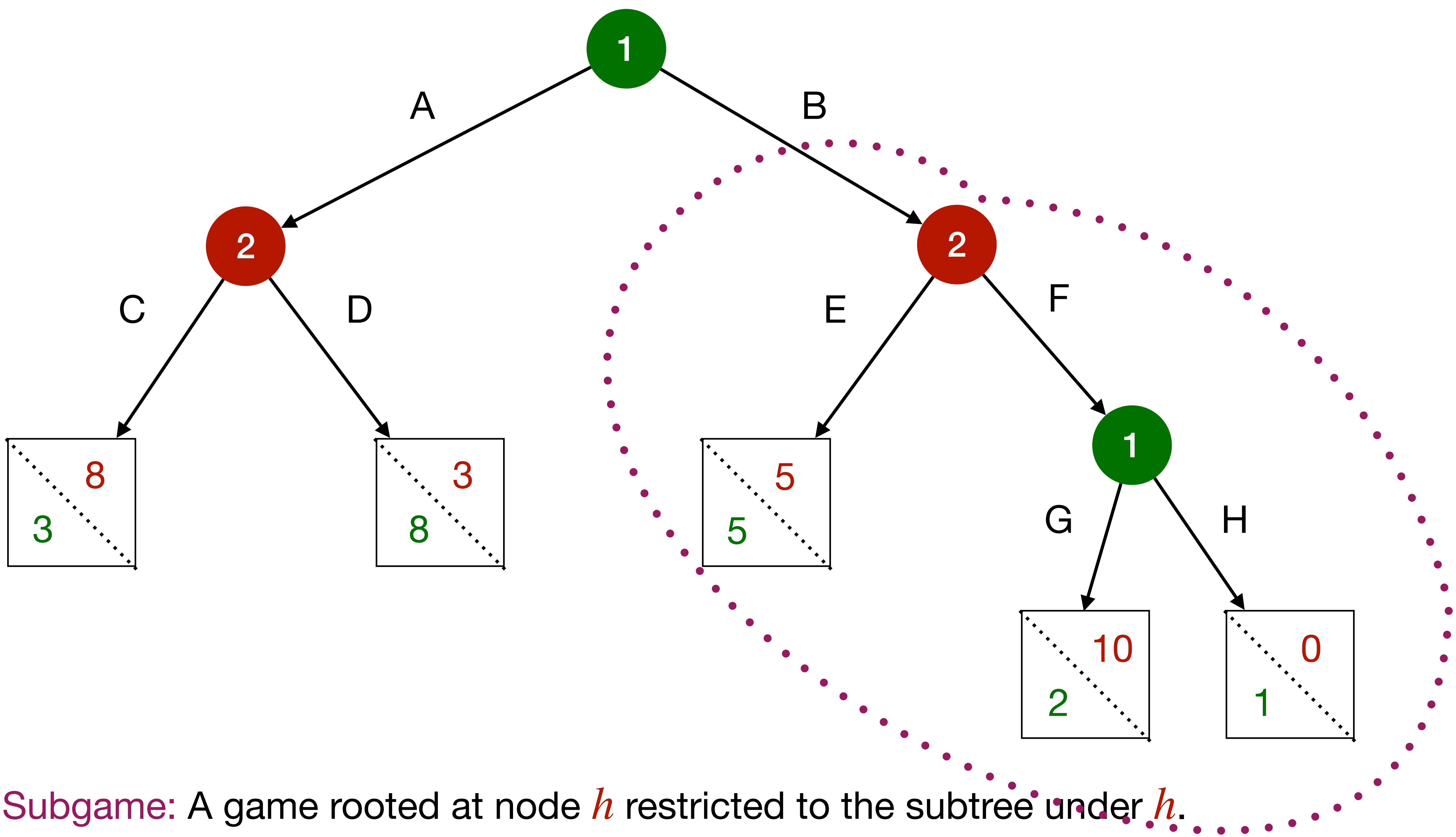
Subgame: A game rooted at node h restricted to the subtree under h.

Subgames



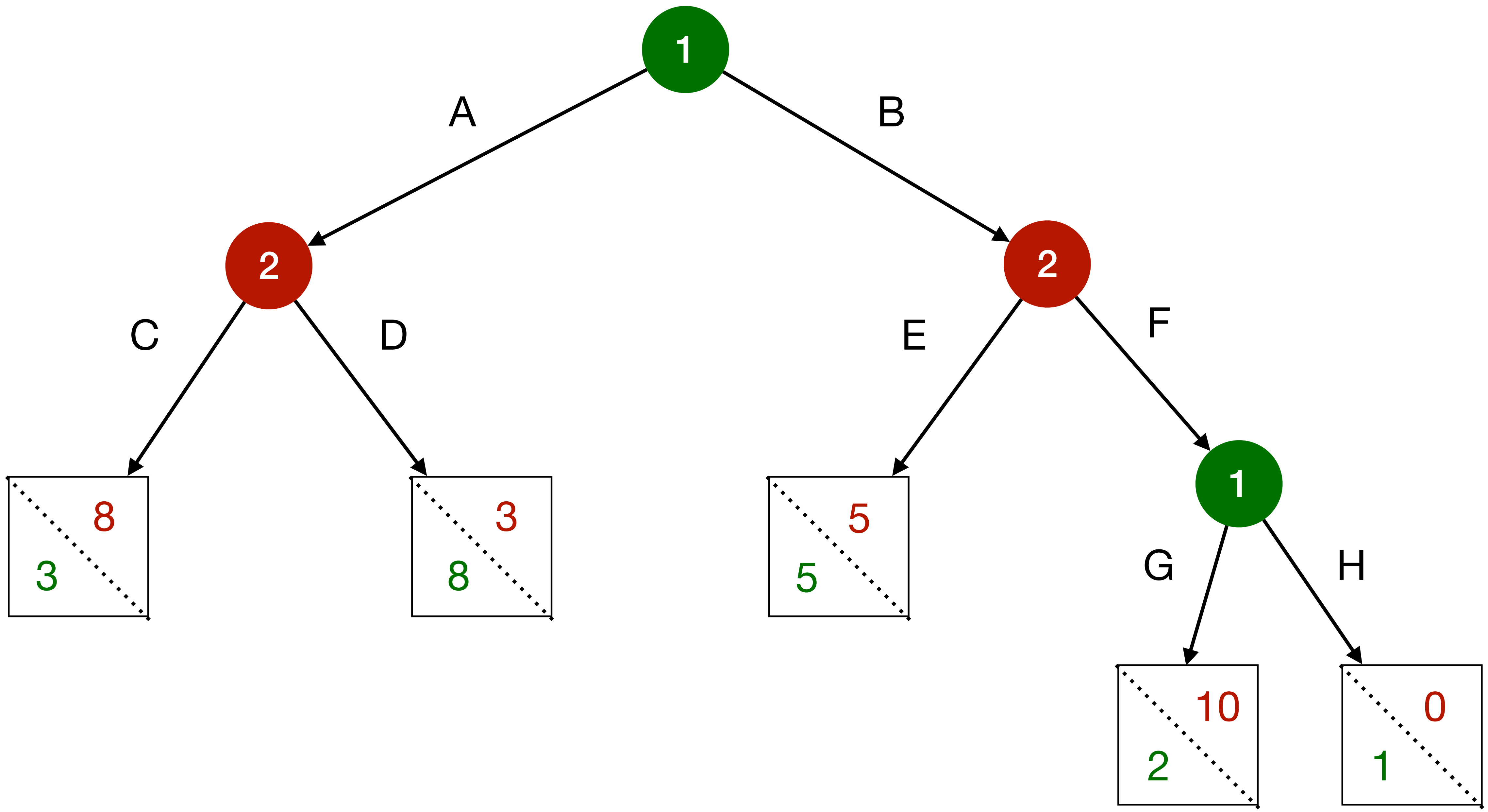
Subgame: A game rooted at node h restricted to the subtree under h.

Subgames



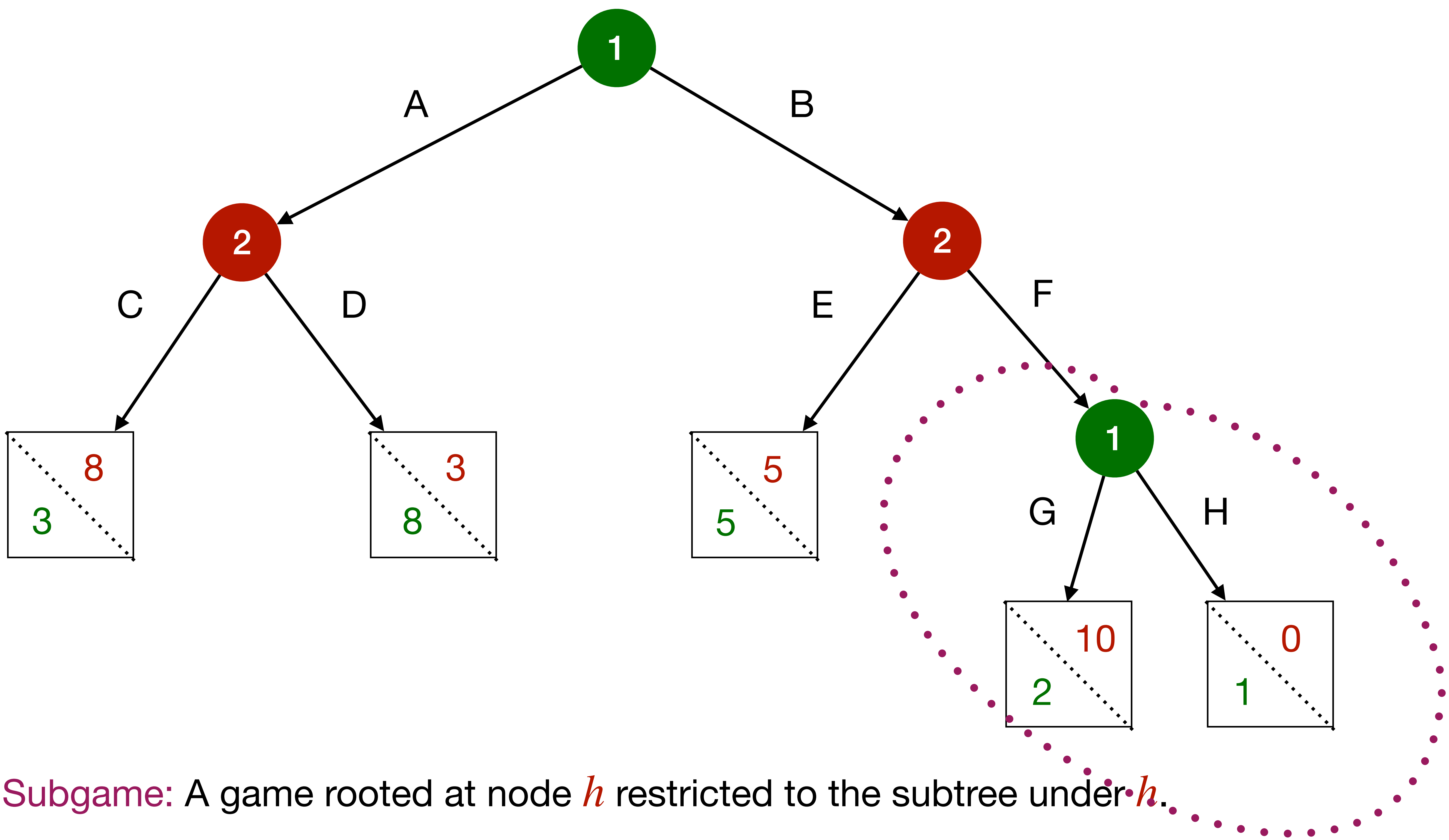
Subgame: A game rooted at node h restricted to the subtree under h.

Subgames



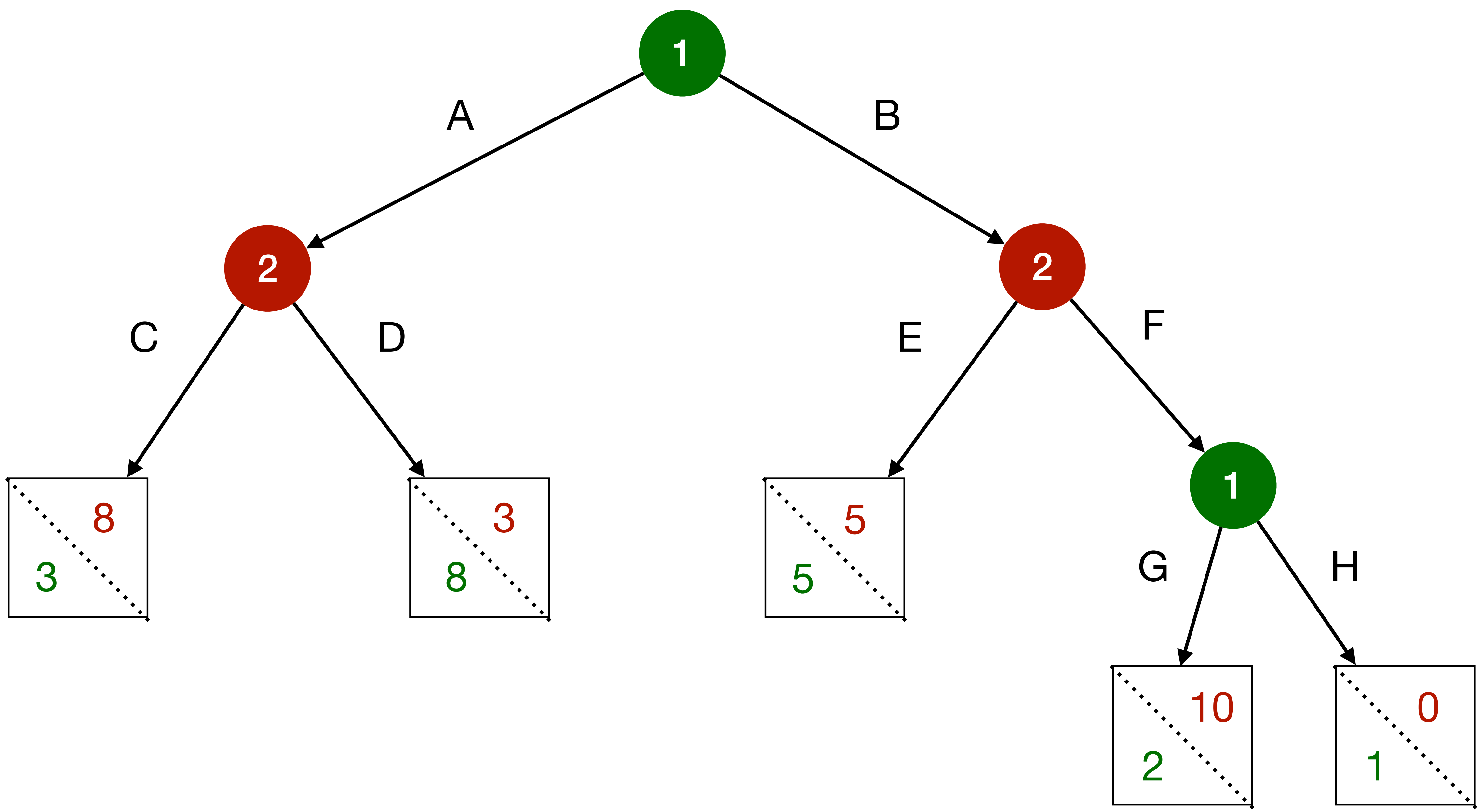
Subgame: A game rooted at node h restricted to the subtree under h.

Subgames



Subgame: A game rooted at node h restricted to the subtree under h.

Subgames



Subgame: A game rooted at node h restricted to the subtree under h.

Solution Concept #5:
Subgame Perfect Equilibrium

Solution Concept #5: Subgame Perfect Equilibrium

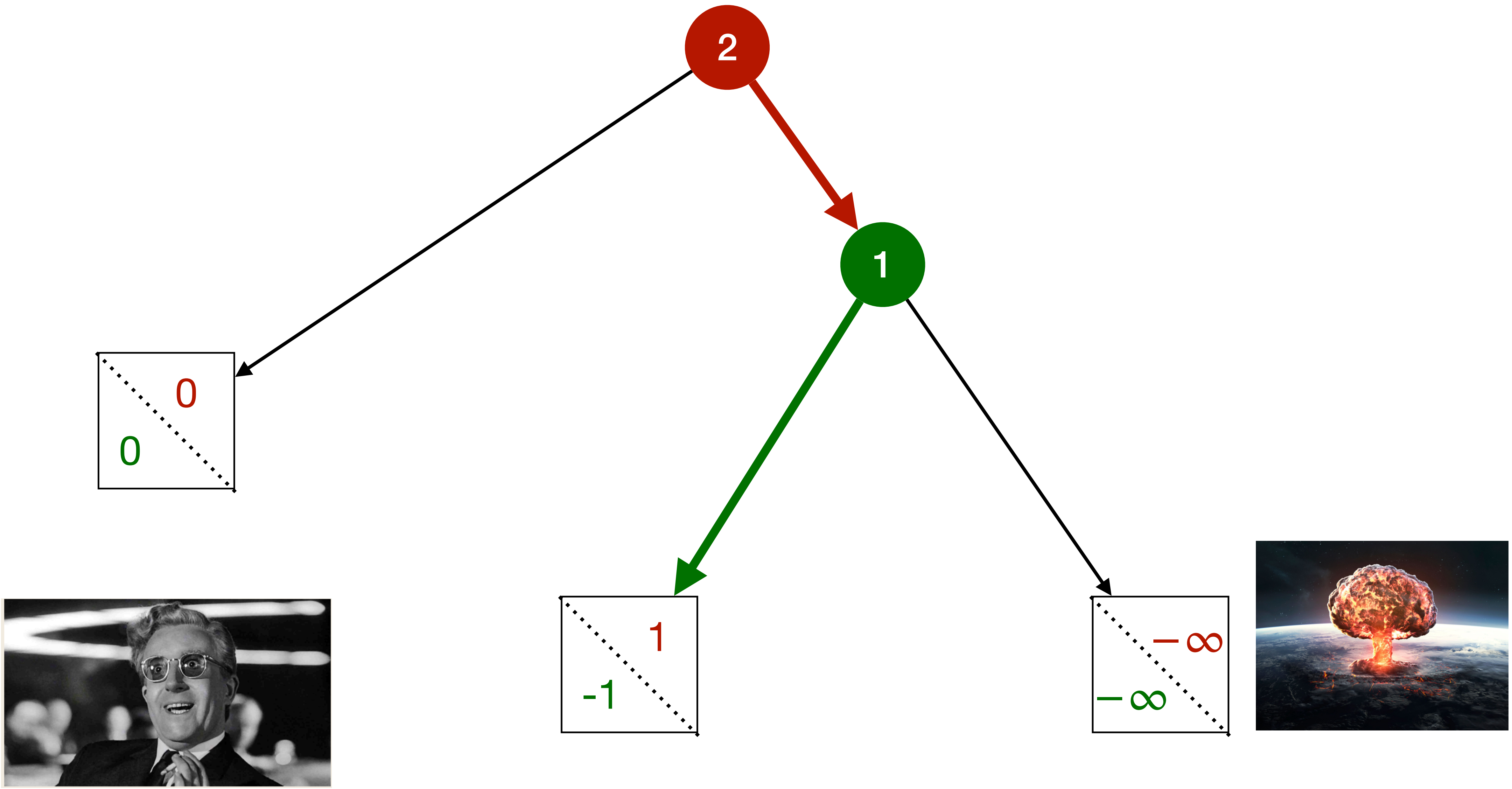
Subgame Perfect Equilibrium: A (pure) strategy profile $s = (s_1, \ldots, s_n)$ such that, if we take the restriction $\hat{s}$ of s to any subgame $\hat{G}$ of G, $\hat{s}$ is pure Nash equilibrium of $\hat{G}$.

Solution Concept #5: Subgame Perfect Equilibrium

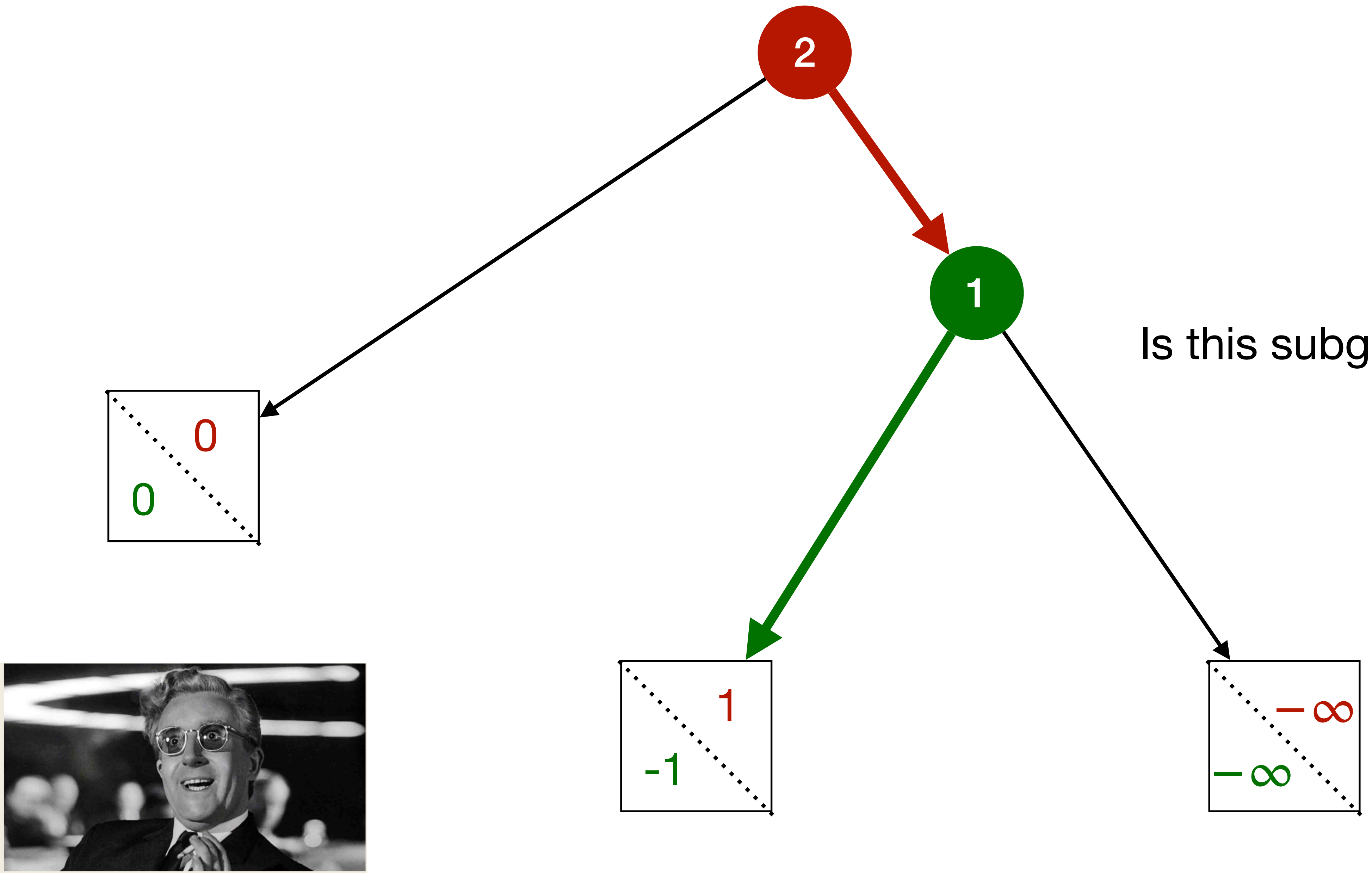
Subgame Perfect Equilibrium: A (pure) strategy profile $s = (s_1, \ldots, s_n)$ such that, if we take the restriction $\hat{s}$ of s to any subgame $\hat{G}$ of G, $\hat{s}$ is pure Nash equilibrium of $\hat{G}$.

This concept eliminates non-credible threats.

This is a PNE

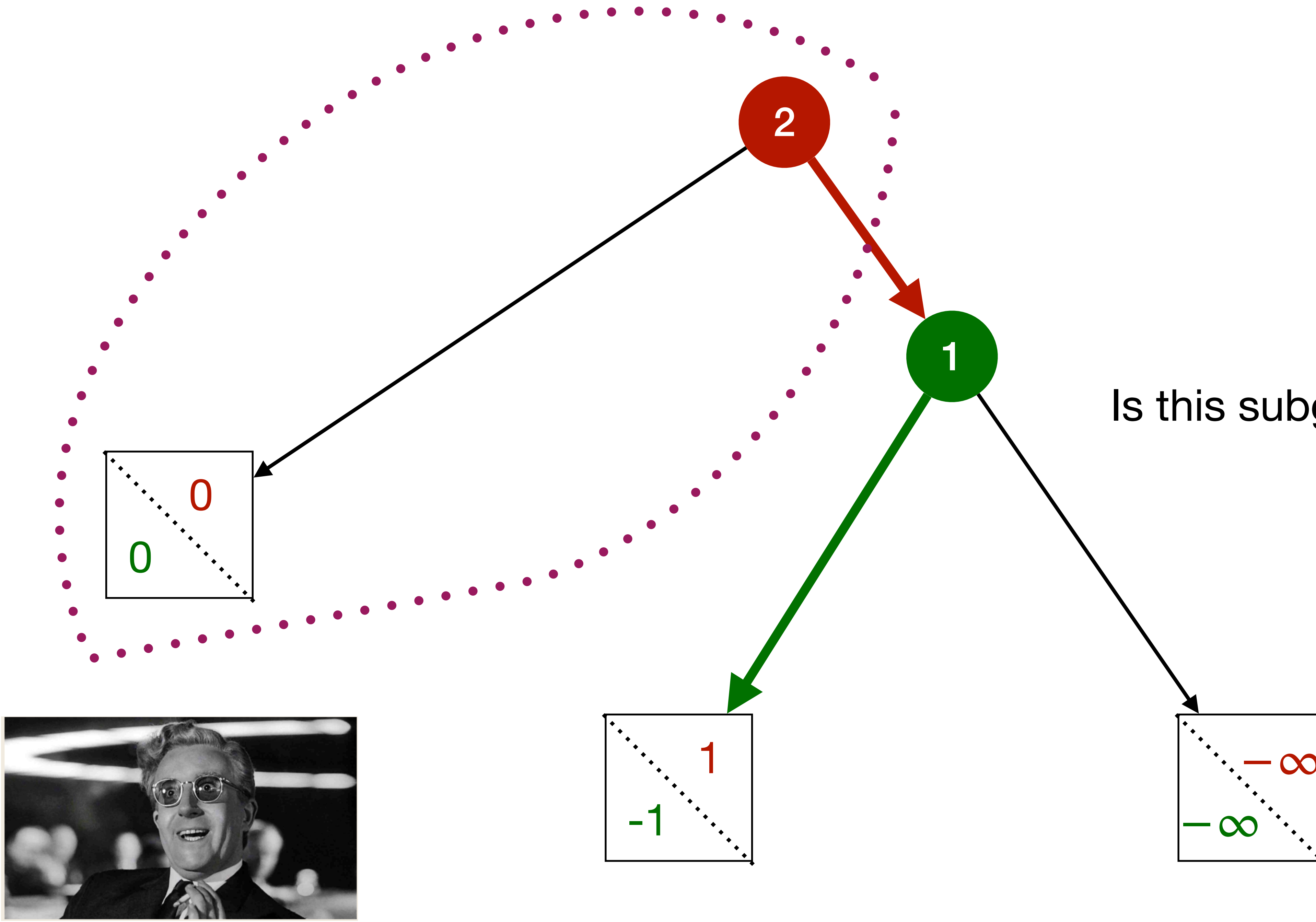


This is a PNE



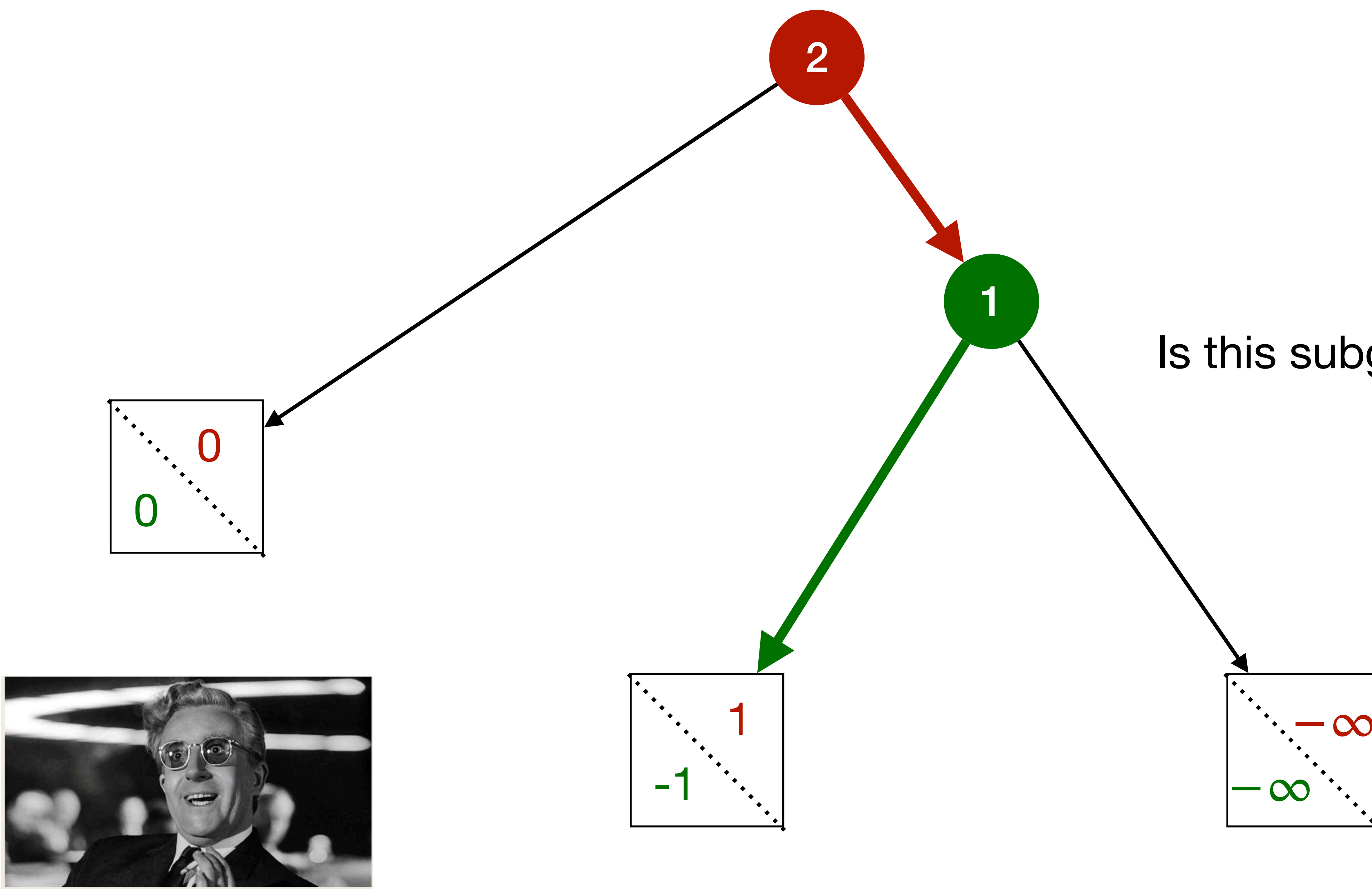
Is this subgame perfect?

This is a PNE



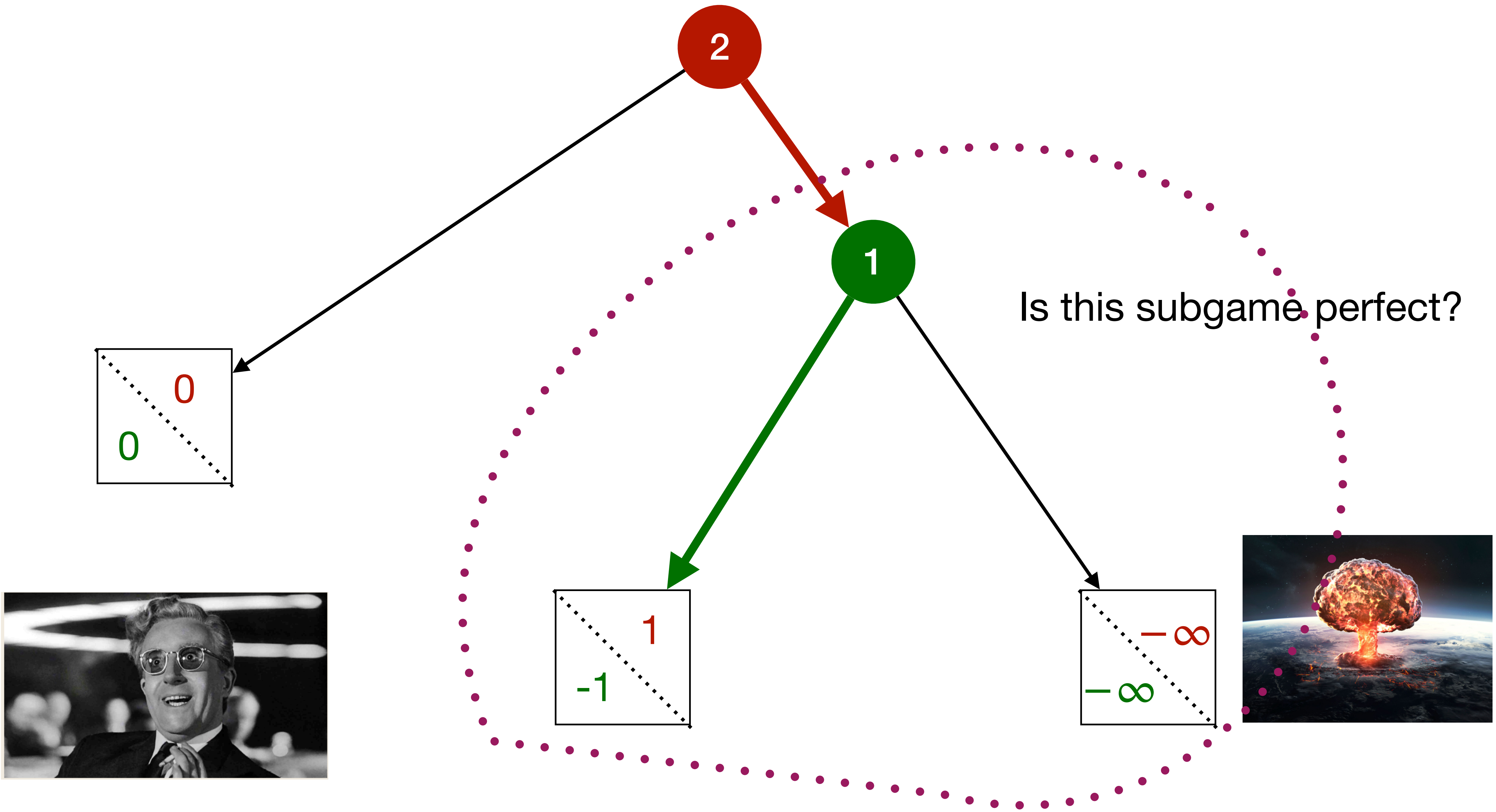
Is this subgame perfect?

This is a PNE

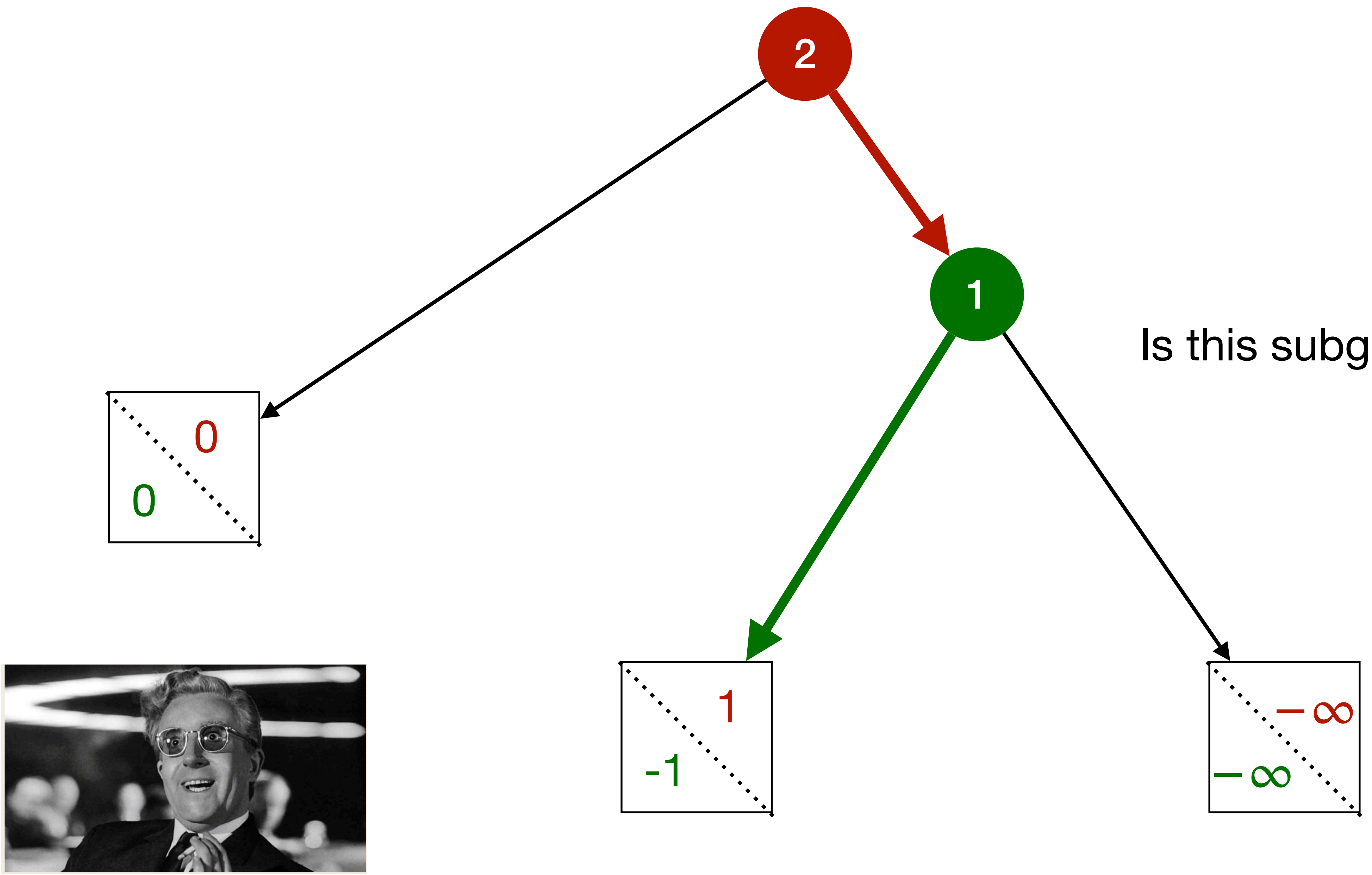


Is this subgame perfect?

This is a PNE

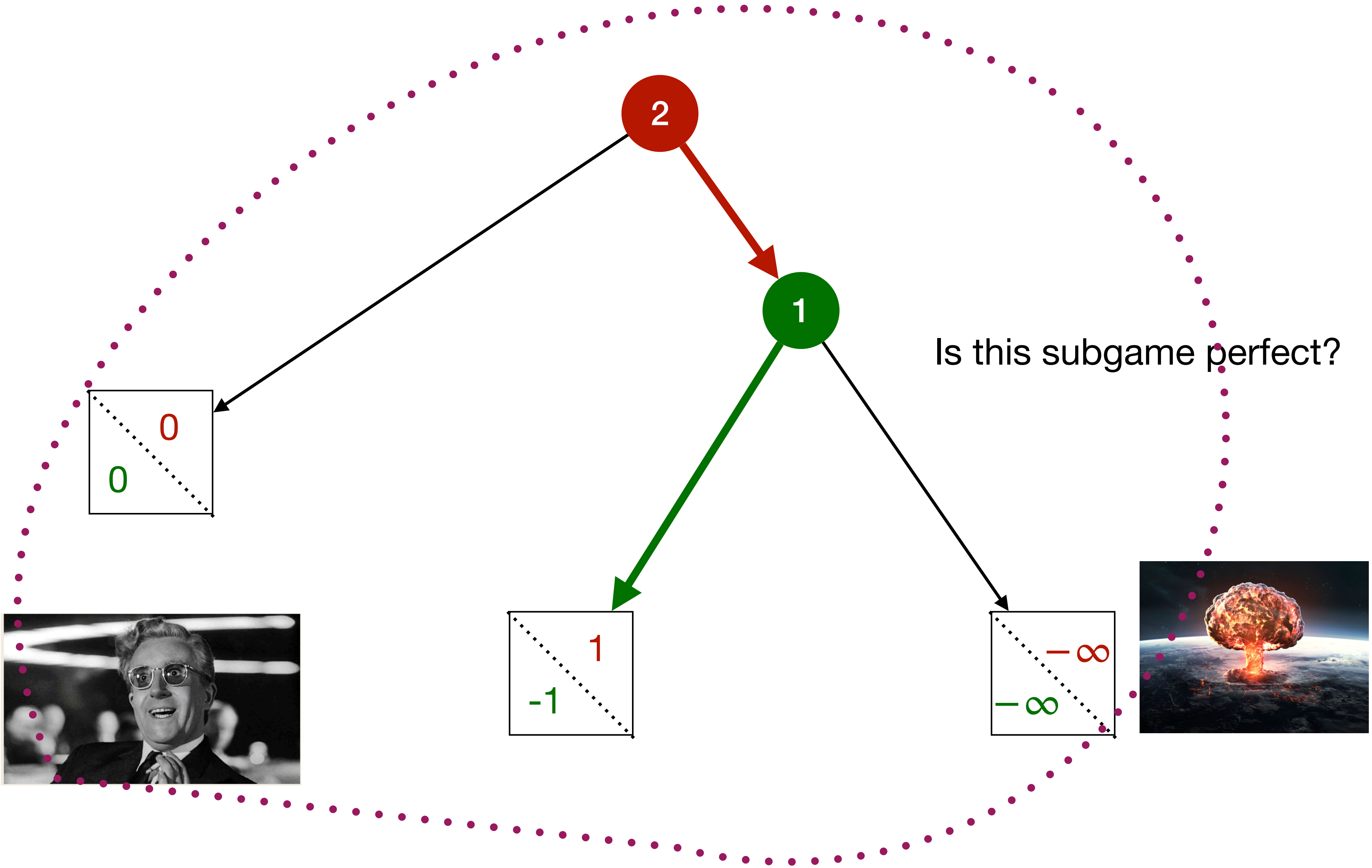


This is a PNE

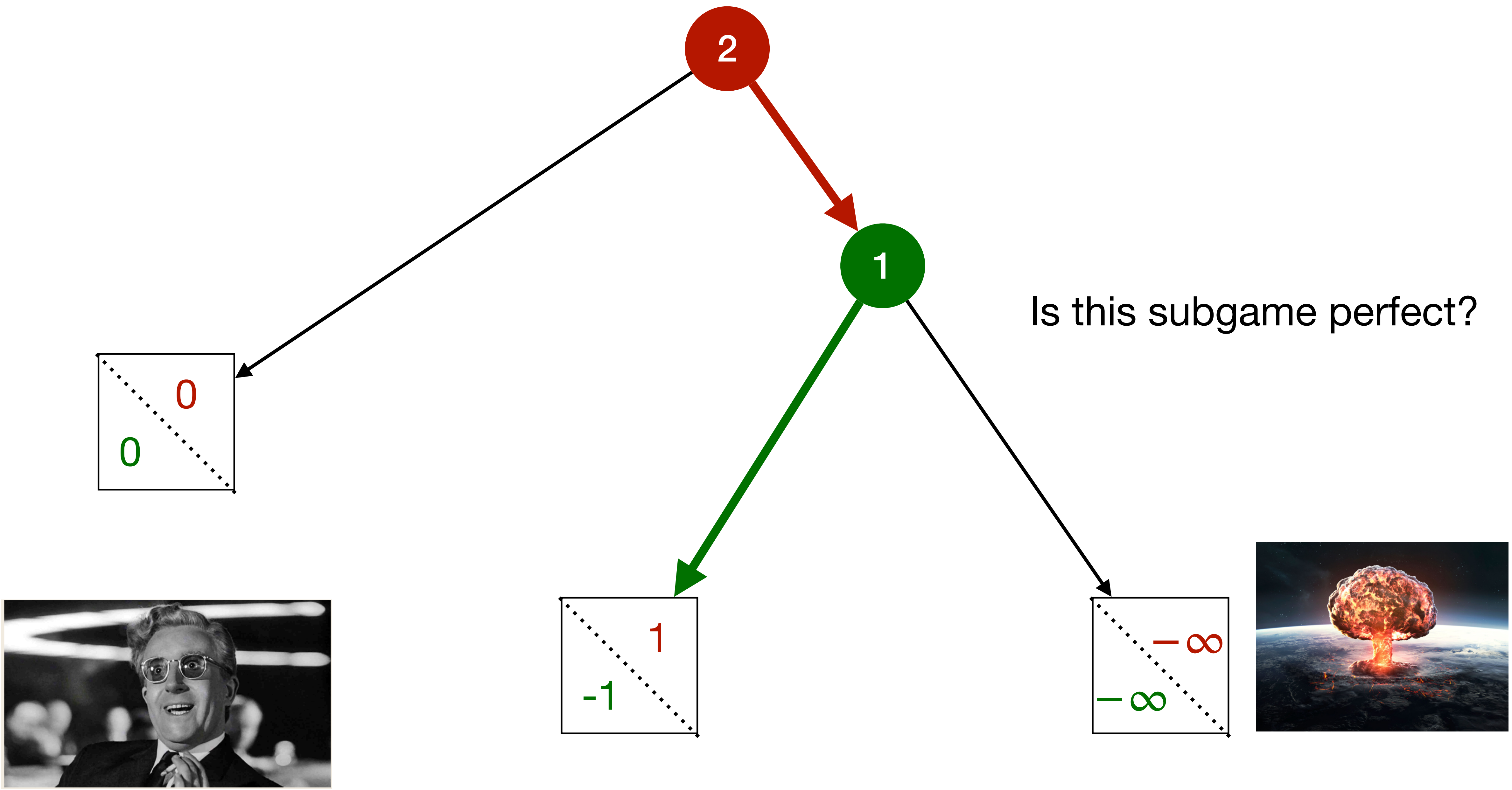


Is this subgame perfect?

This is a PNE



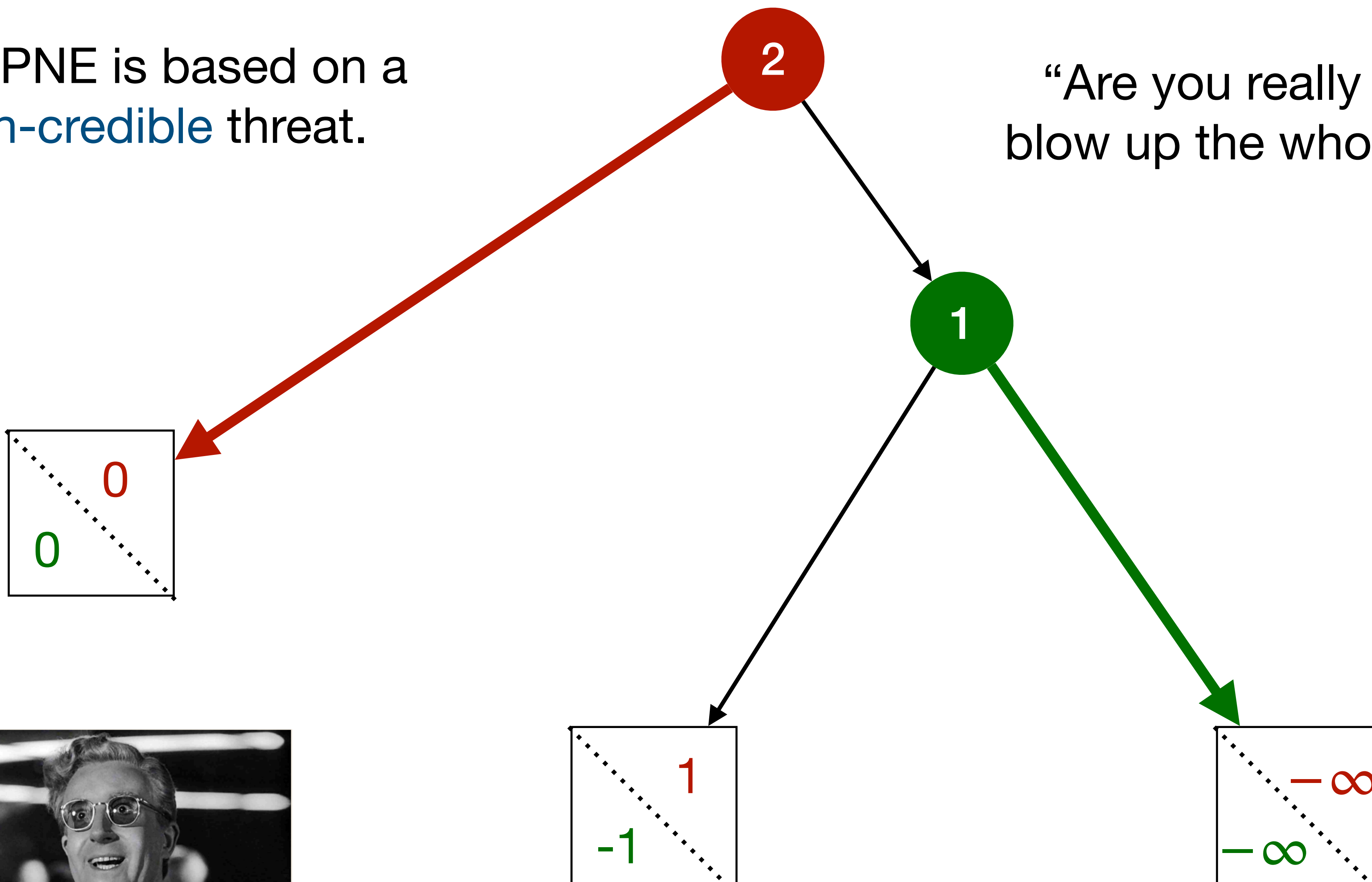
This is a PNE

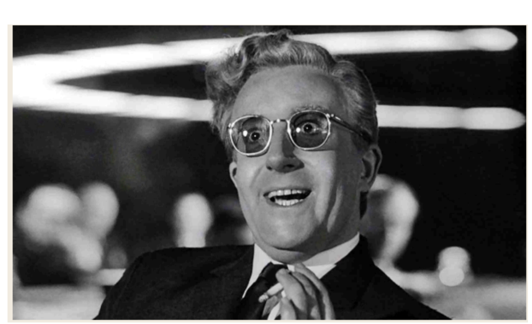


This is also PNE

This PNE is based on a non-credible threat.

“Are you really going to blow up the whole world?”

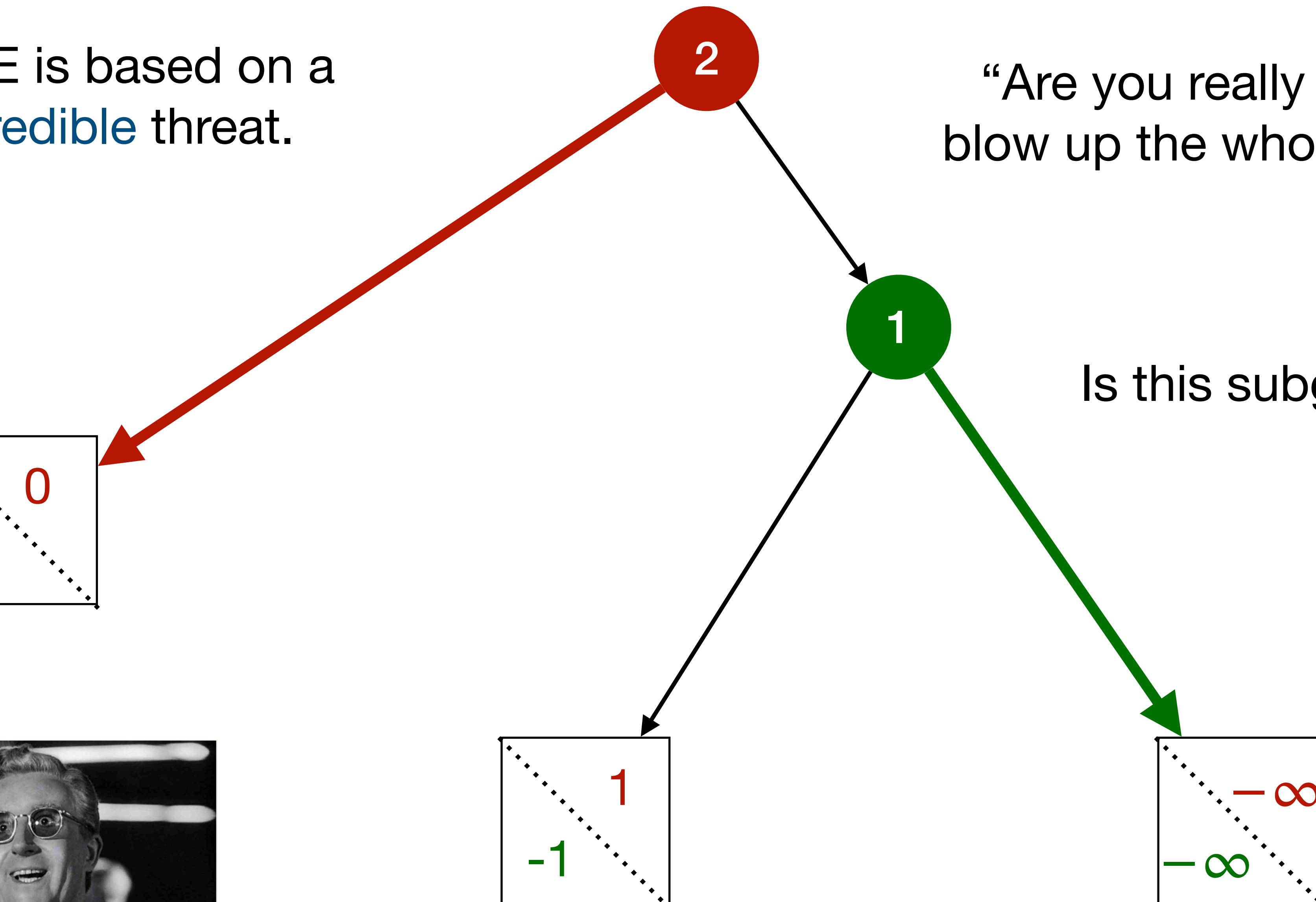




This is also PNE

This PNE is based on a non-credible threat.

"Are you really going to blow up the whole world?"

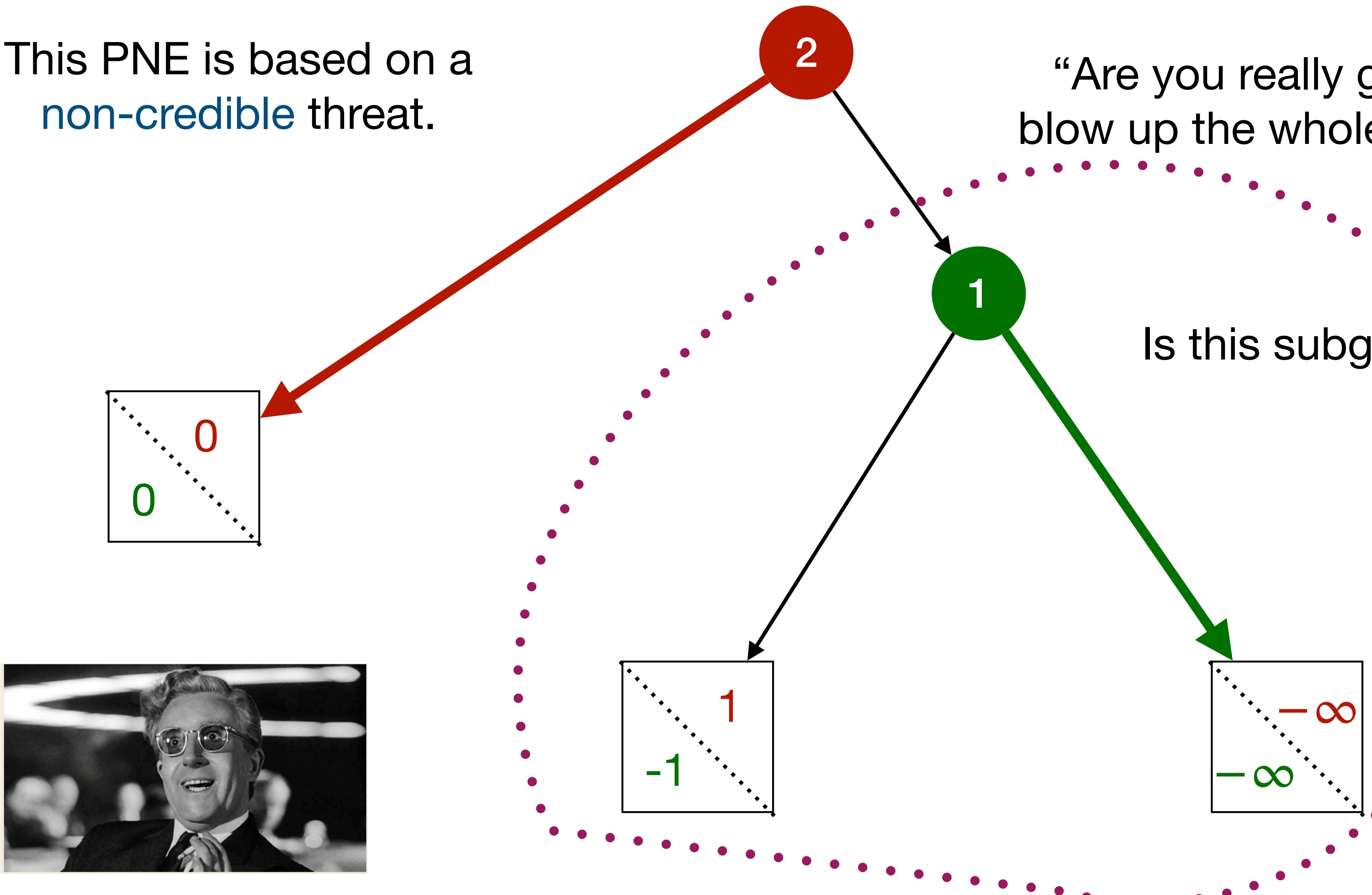


Is this subgame perfect?

This is also PNE

This PNE is based on a non-credible threat.

“Are you really going to blow up the whole world?”



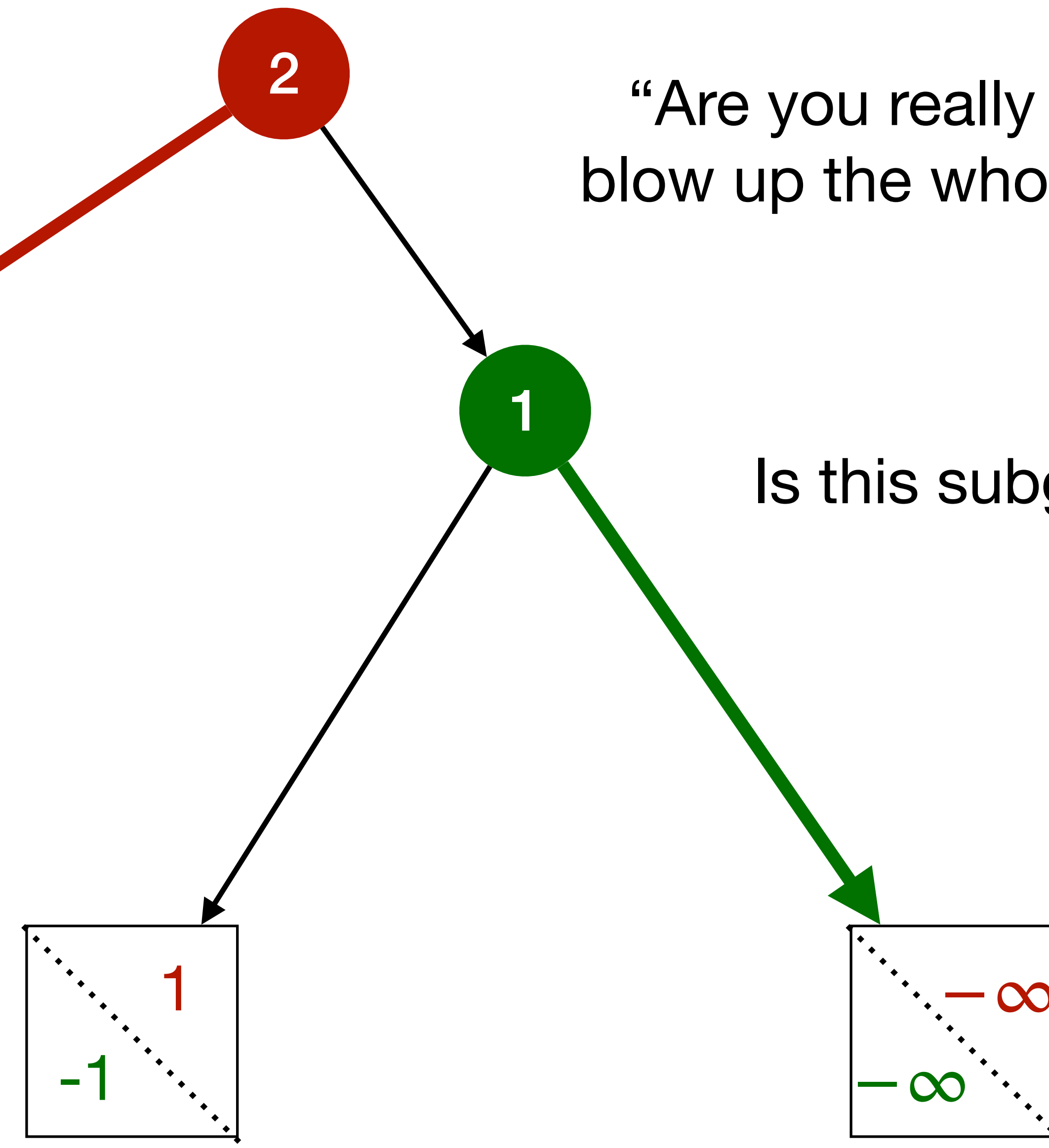
Is this subgame perfect?

This is also PNE

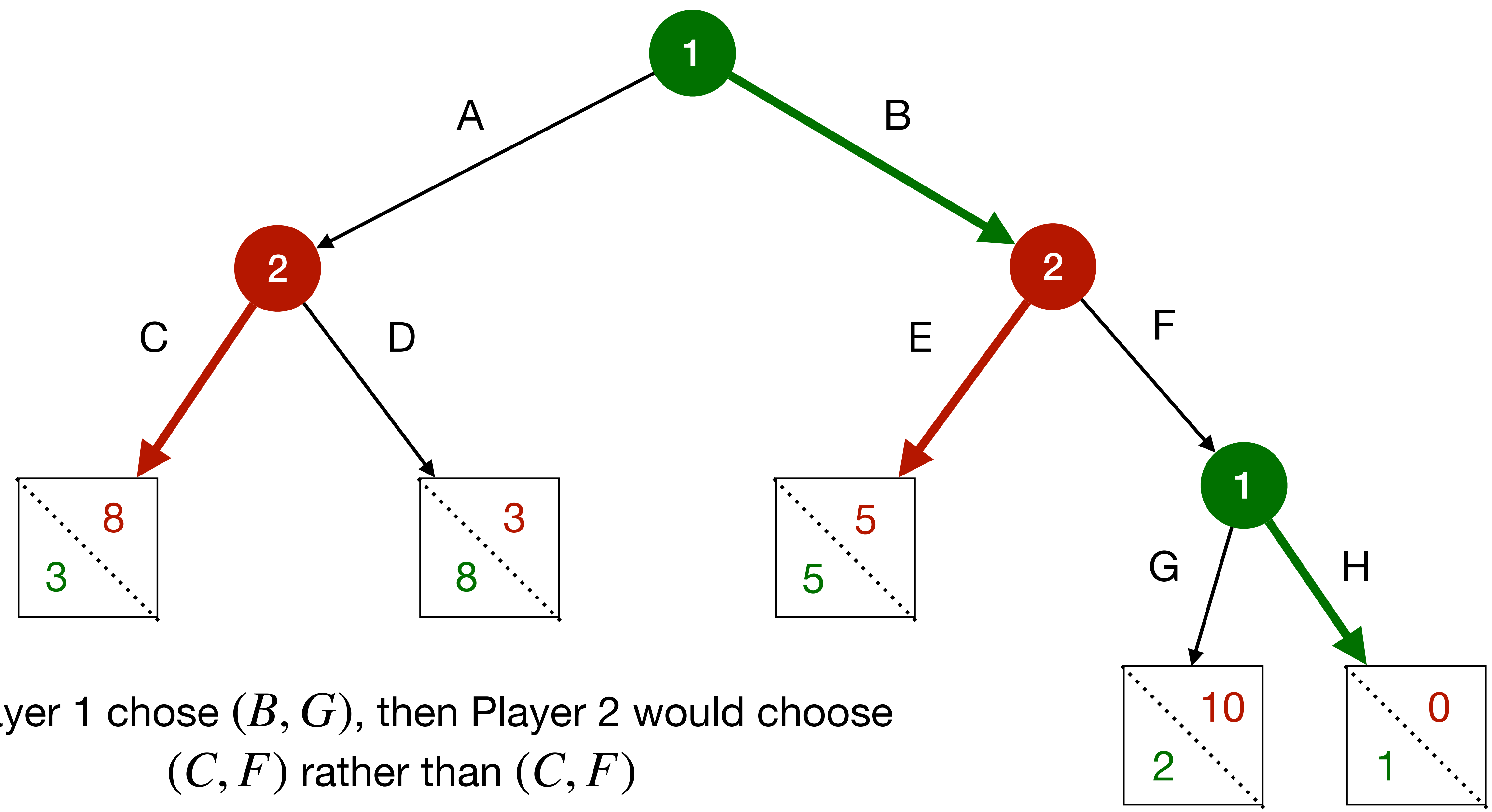
This PNE is based on a non-credible threat.

“Are you really going to blow up the whole world?”

Is this subgame perfect?



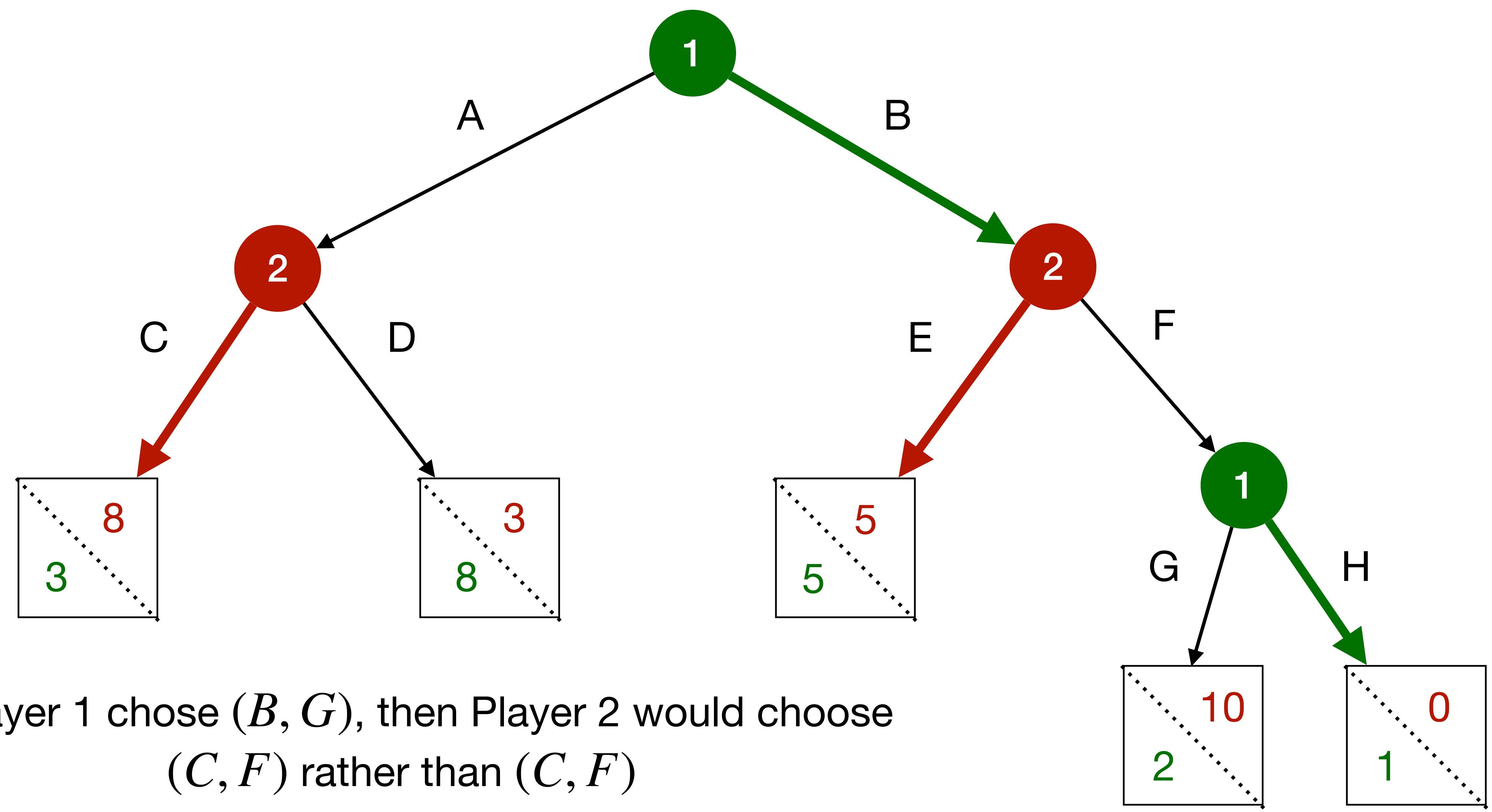
Equilibrium 3: (B, H), (C, E)



If Player 1 chose (B, G), then Player 2 would choose (C, F) rather than (C, F)

The only reason why Player 2 chooses (C, E) is because Player 1 is *threatening* with a worse action.

Equilibrium 3: (B, H), (C, E)

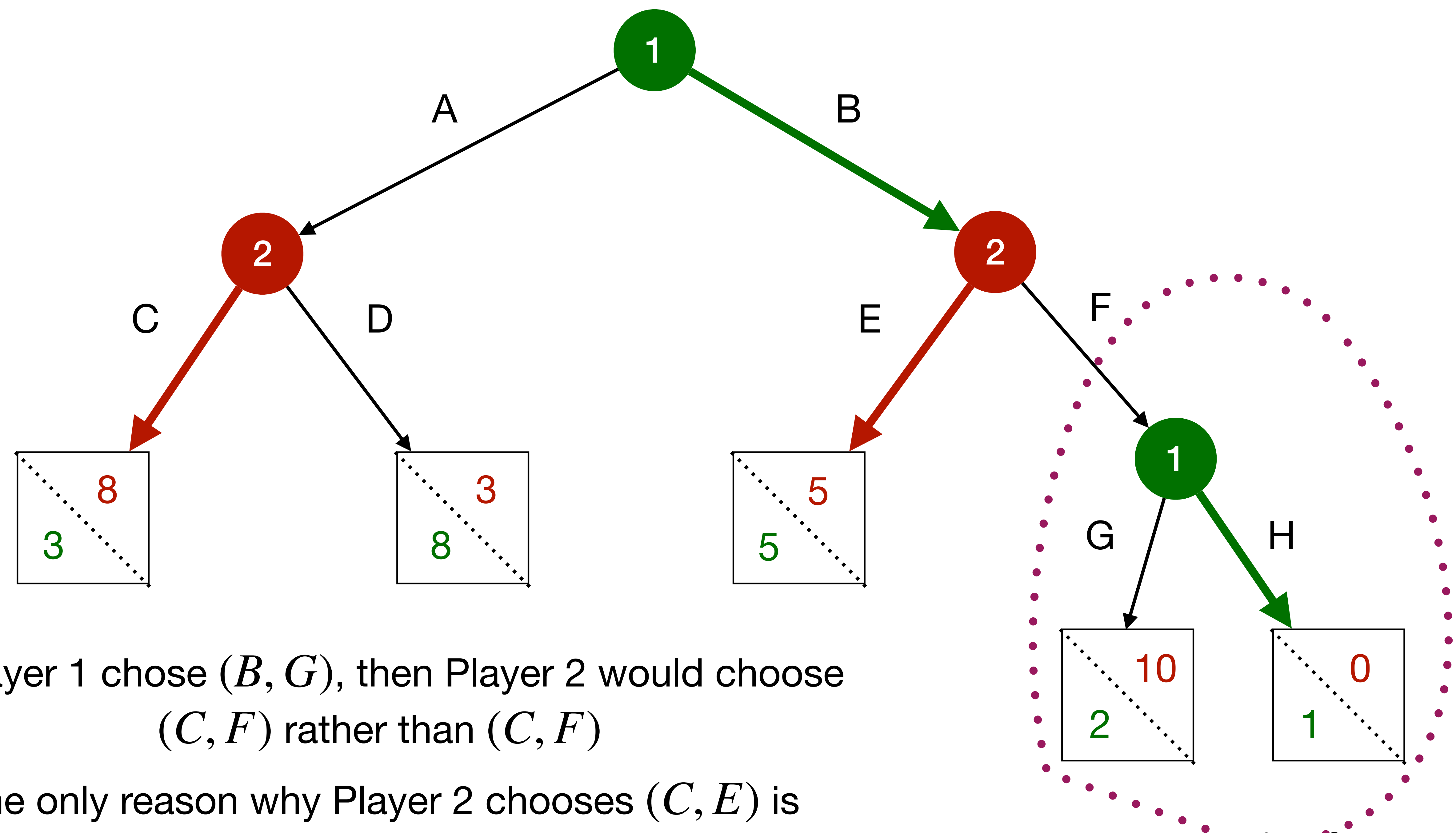


If Player 1 chose (B, G), then Player 2 would choose (C, F) rather than (C, F)

The only reason why Player 2 chooses (C, E) is because Player 1 is *threatening* with a worse action.

Is this subgame perfect?

Equilibrium 3: (B, H), (C, E)

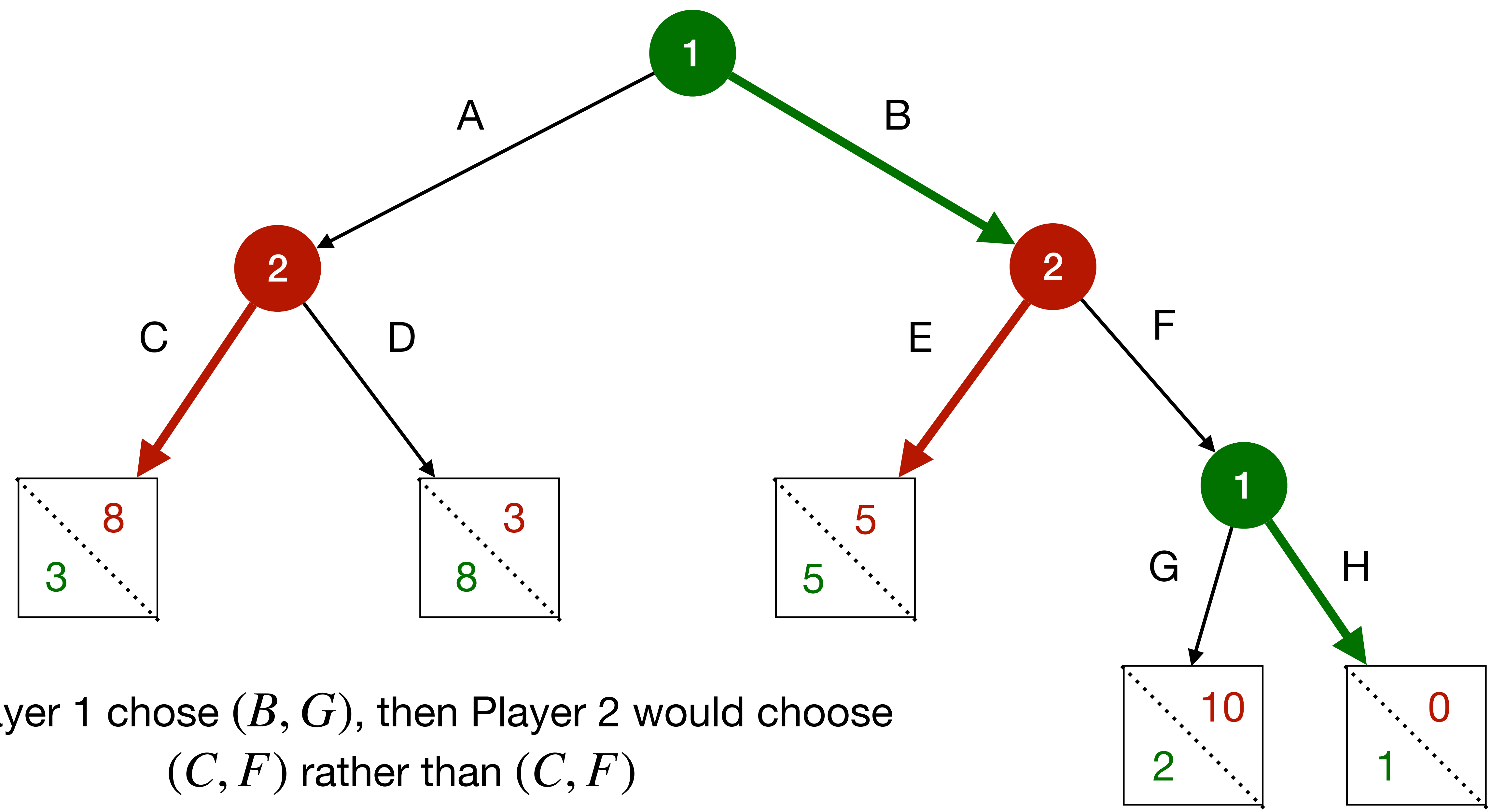


If Player 1 chose (B, G), then Player 2 would choose (C, F) rather than (C, F)

The only reason why Player 2 chooses (C, E) is because Player 1 is *threatening* with a worse action.

Is this subgame perfect?

Equilibrium 3: (B, H), (C, E)



If Player 1 chose (B, G), then Player 2 would choose (C, F) rather than (C, F)

The only reason why Player 2 chooses (C, E) is because Player 1 is *threatening* with a worse action.

Is this subgame perfect?

Homework

	(C,E)	(C,F)	(D,E)	(D,F)
(A,G)	3, 8	3, 8	8, 3	8, 3
(A,H)	3, 8	3, 8	8, 3	8, 3
(B,G)	5, 5	2, 10	5, 5	2, 10
(B,H)	5, 5	1, 0	5, 5	1, 0

Verify that out of those three, only (A,G), (C,F) is subgame perfect.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

Equilibrium in Extensive Form Games

<u>Theorem (Zermelo 1913):</u> Every perfect information extensive form game always has a pure Nash equilibrium.

<u>Theorem (Kuhn 1953):</u> Every perfect information extensive form game always has a subgame perfect pure Nash equilibrium.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

Theorem (Kuhn 1953): Every perfect information extensive form game always has a subgame perfect pure Nash equilibrium.

Proof: by backwards induction on the depth of the tree of the subgame.

The proof (sketch)
of Kuhn's Theorem

The proof (sketch) of Kuhn's Theorem

Base Case, depth 0: We are at a leaf. There are no actions to take, so it is trivially true. Each agent gets the utility $u_i(w)$ prescribed by the leaf w.

The proof (sketch) of Kuhn's Theorem

Base Case, depth 0: We are at a leaf. There are no actions to take, so it is trivially true. Each agent gets the utility $u_i(w)$ prescribed by the leaf w.

Inductive Step, depth $k+1$: Let G_w be the subgame and let A_w be the set of actions available for the player who corresponds to the root of this subgame. Let $w_a = \sigma(w, a)$ be the root of the subtree which is obtained when the player chooses strategy $a \in A_w$, and let G_{w_a} be the corresponding subgame. By construction, the depth of G_{w_a} is k.

The proof (sketch) of Kuhn's Theorem

Base Case, depth 0: We are at a leaf. There are no actions to take, so it is trivially true. Each agent gets the utility $u_i(w)$ prescribed by the leaf w.

Inductive Step, depth $k+1$: Let G_w be the subgame and let A_w be the set of actions available for the player who corresponds to the root of this subgame. Let $w_a = \sigma(w, a)$ be the root of the subtree which is obtained when the player chooses strategy $a \in A_w$, and let G_{w_a} be the corresponding subgame. By construction, the depth of G_{w_a} is k.

By the induction hypothesis, every such G_{w_a} has a subgame perfect PNE, denoted by $s^{w_a} = (s_1^{w_a}, \ldots, s_n^{w_a})$.

The proof (sketch) of Kuhn's Theorem

By the induction hypothesis, every such G_{w_a} has a subgame perfect PNE, denoted by $s^{w_a} = (s_1^{w_a}, \ldots, s_n^{w_a})$.

The proof (sketch) of Kuhn's Theorem

By the induction hypothesis, every such G_{w_a} has a subgame perfect PNE, denoted by $s^{w_a} = (s_1^{w_a}, \ldots, s_n^{w_a})$.

Let player i be the player that plays at node w. The player will chose the action $a^* \in A_w$ that maximises her payoff in the game G_w.

The proof (sketch) of Kuhn's Theorem

By the induction hypothesis, every such G_{w_a} has a subgame perfect PNE, denoted by $s^{w_a} = (s_1^{w_a}, \ldots, s_n^{w_a})$.

Let player i be the player that plays at node w. The player will chose the action $a^* \in A_w$ that maximises her payoff in the game G_w.

So in the strategy profile for the game G_w, we define

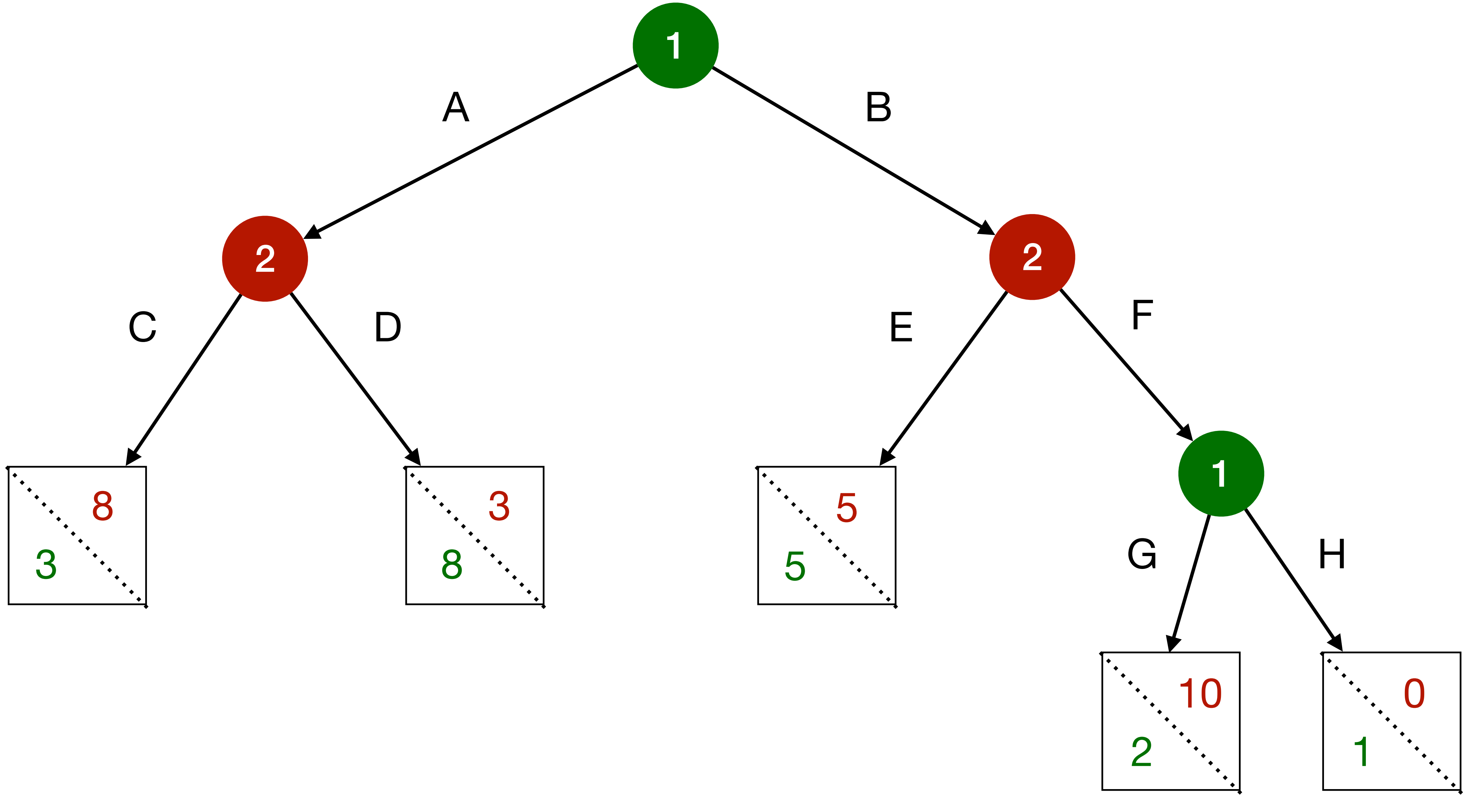
$$s_i^w = s_i^{w_a} \cup \{w \to a^*\}$$

$$s_j^w = s_j^{w_a}$$

An efficient algorithm for computing SPPNE

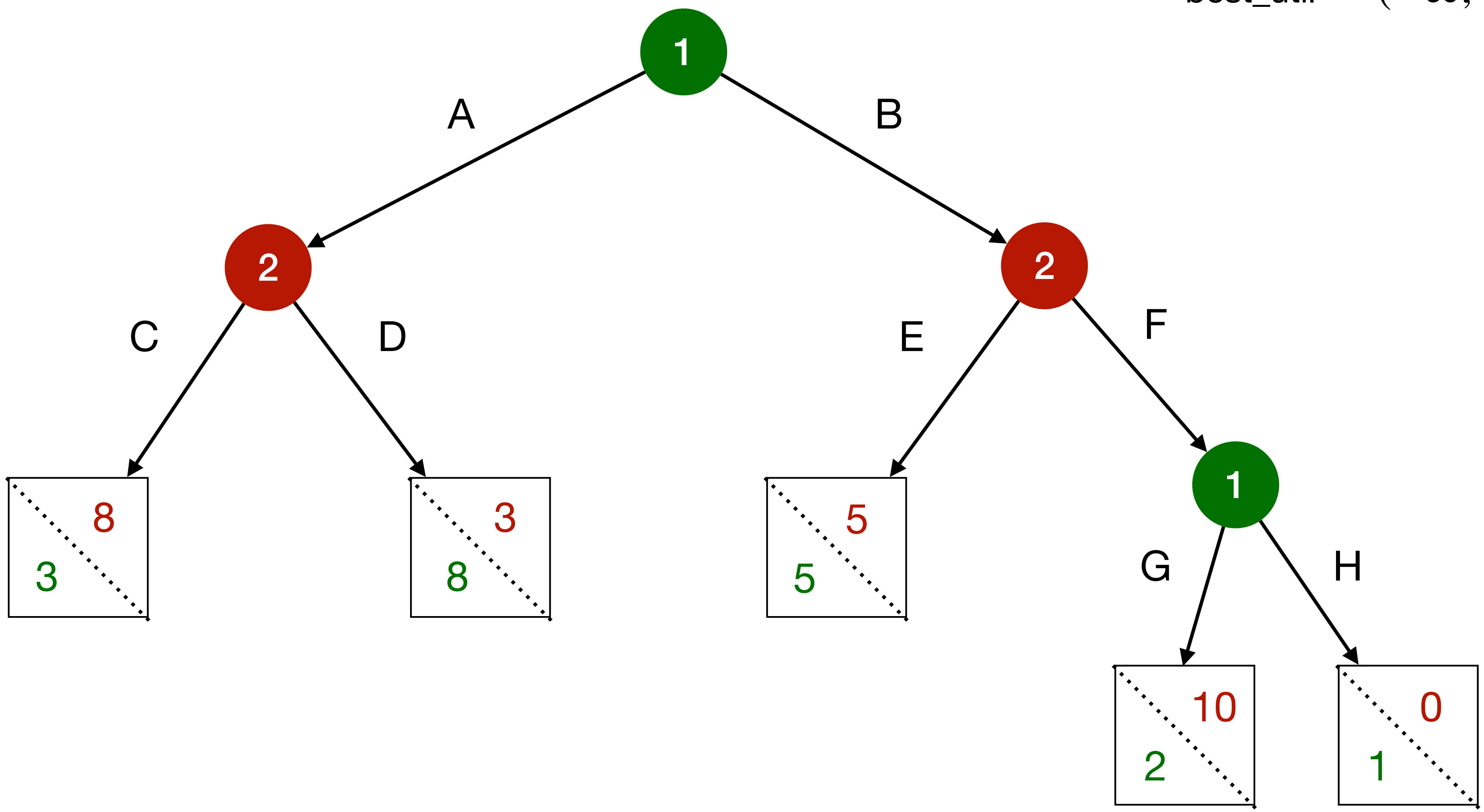
function $\textsc{BackwardInduction}$ (node h) **returns** $u(h)$
if $h \in Z$ **then**
  return $u(h)$ // h is a terminal node
$best\_util \leftarrow -\infty$
forall $a \in \chi(h)$ **do**
  $util\_at\_child \leftarrow \textsc{BackwardInduction}(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
return $best\_util$

Example



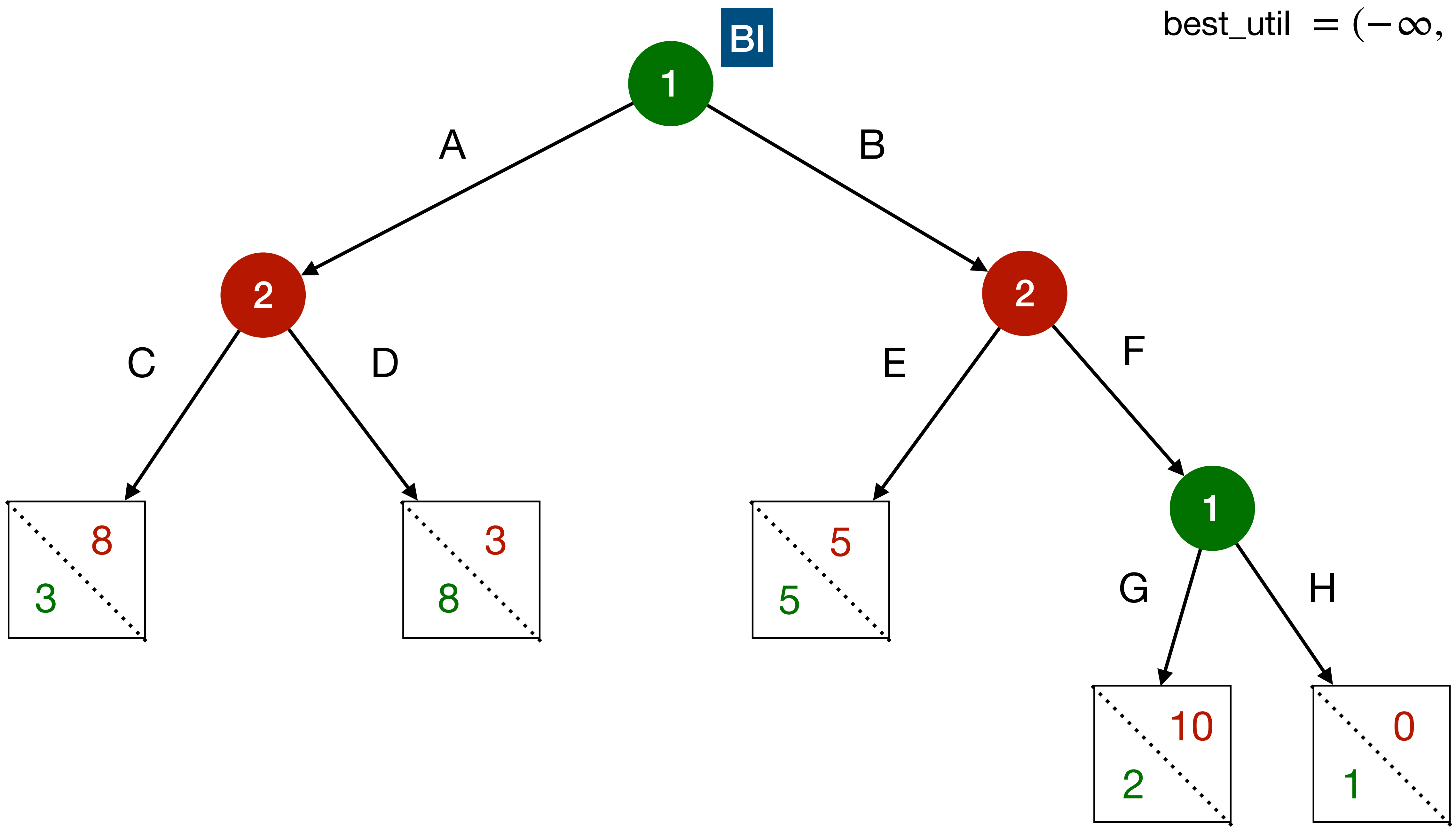
Example

best_util $= (-\infty, -\infty)$



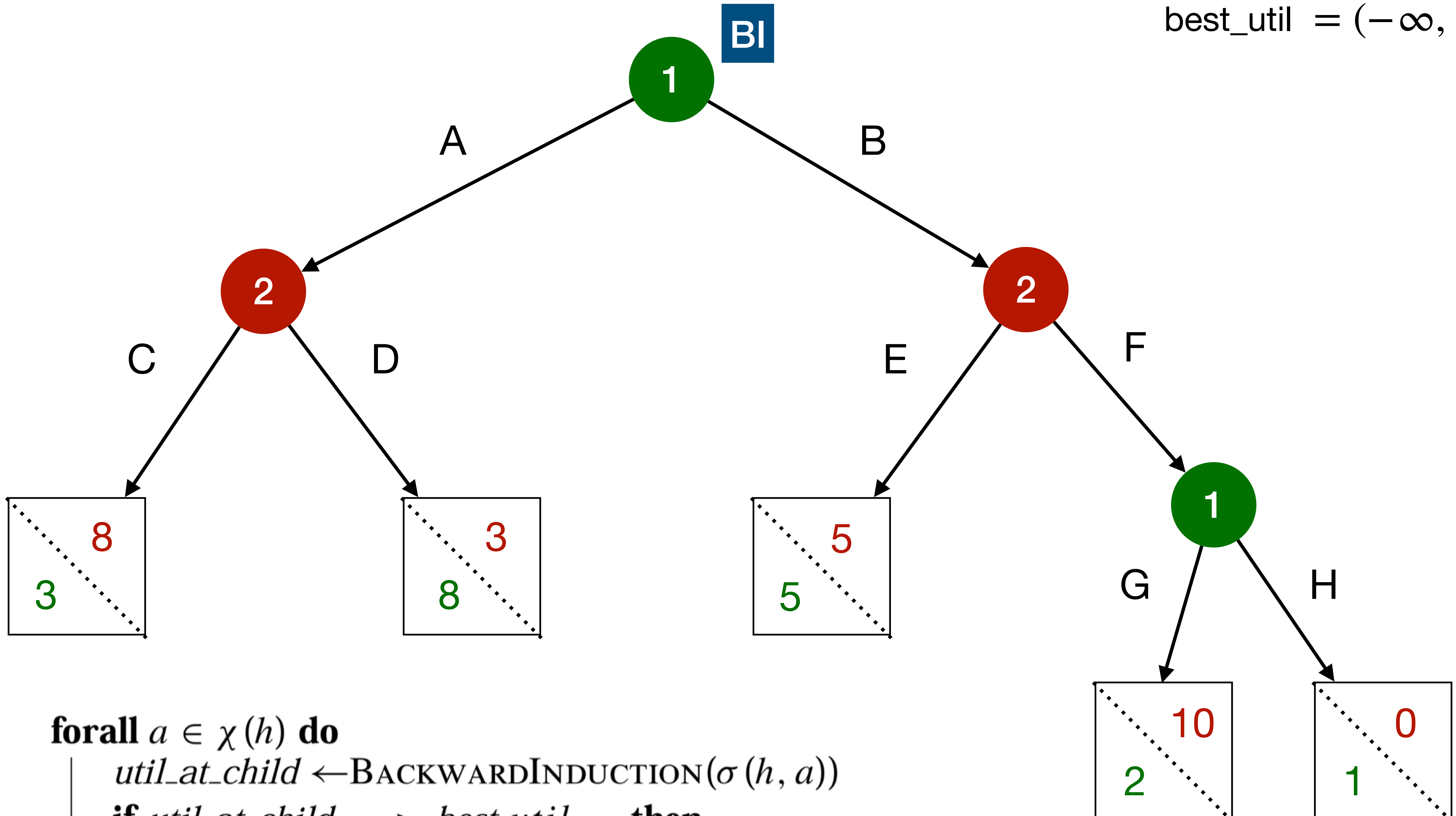
Example

$\text{best\_util} = (-\infty, -\infty)$



Example

best_util $= (-\infty, -\infty)$



forall $a \in \chi(h)$ **do**

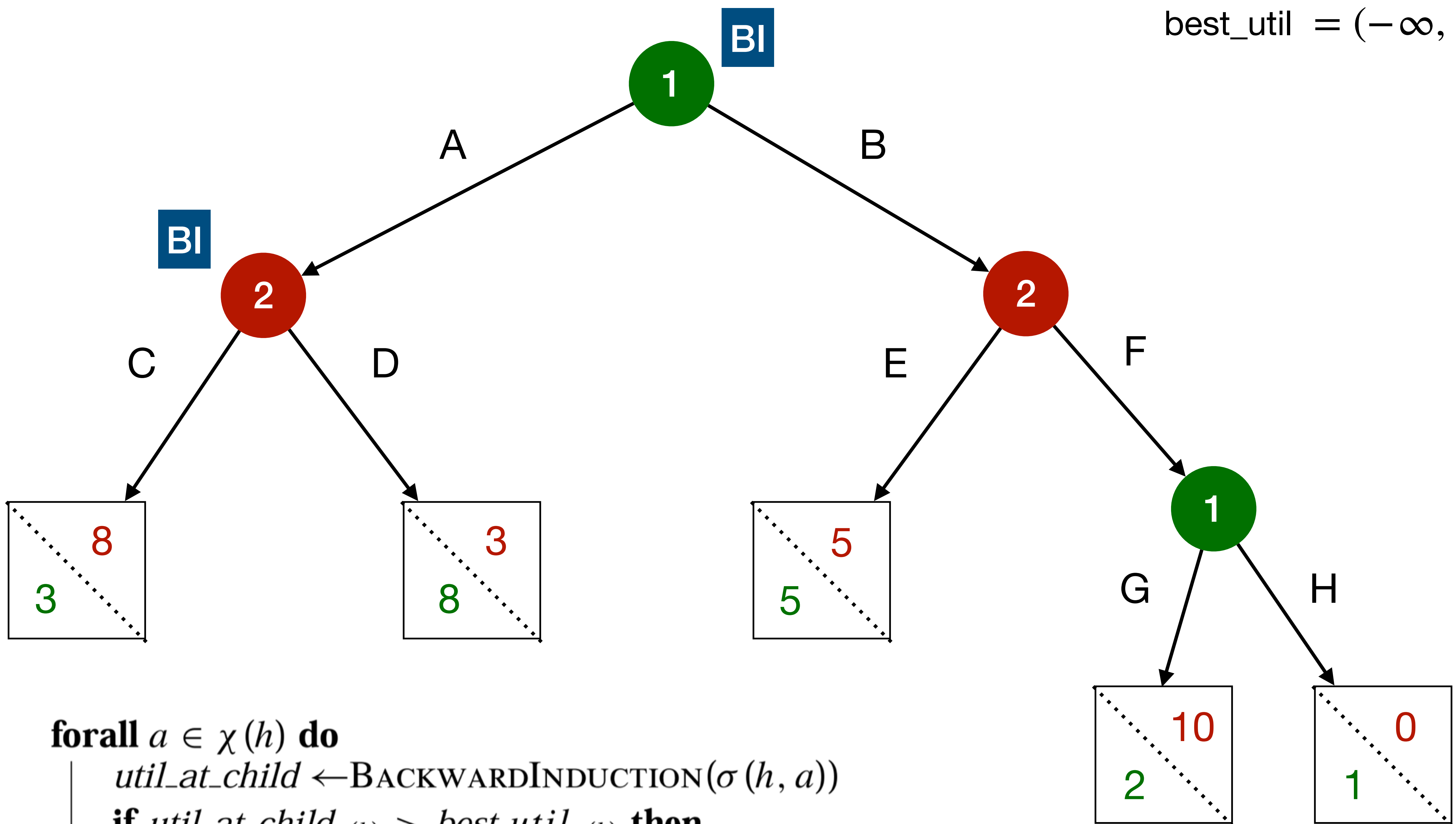
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

return *best_util*

Example

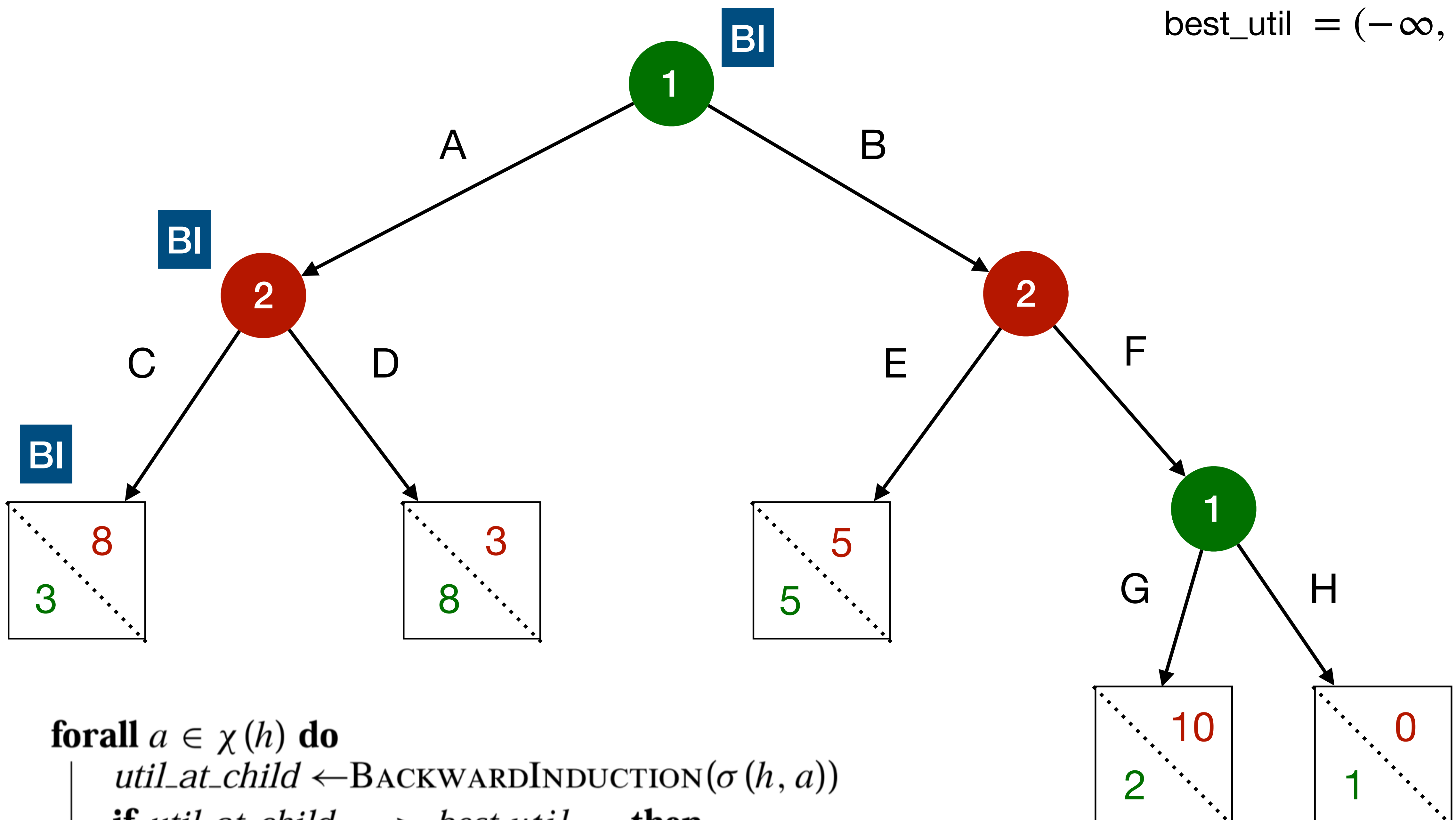


best_util $= (-\infty, -\infty)$

forall $a \in \chi(h)$ **do**
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    best_util $\leftarrow$ *util_at_child*
return *best_util*

Example

$\text{best\_util} = (-\infty, -\infty)$



forall $a \in \chi(h)$ **do**

  util_at_child $\leftarrow$ BackwardInduction$(\sigma(h, a))$

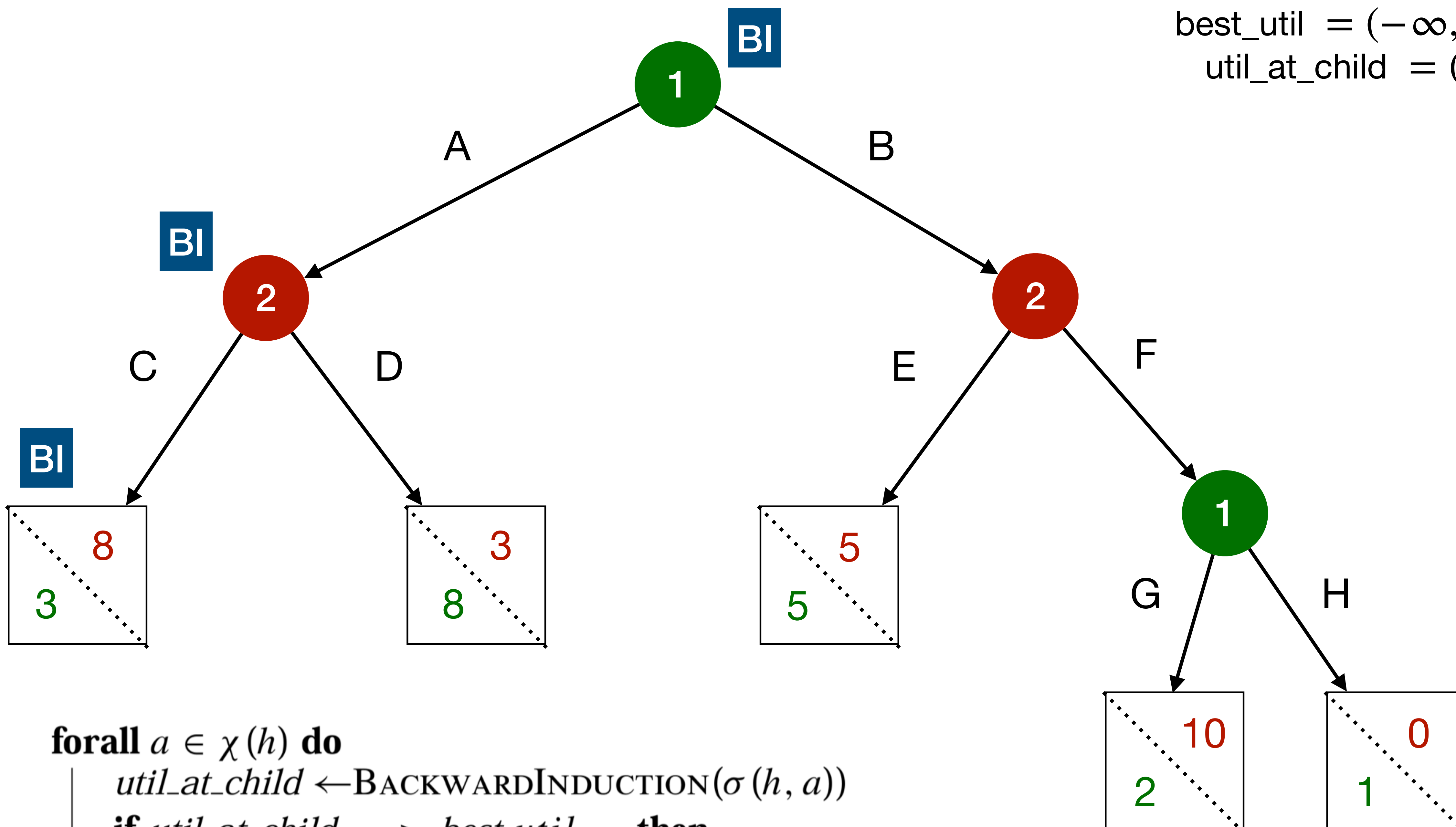
  if $\textit{util\_at\_child}_{\rho(h)} > \textit{best\_util}_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

return *best_util*

Example

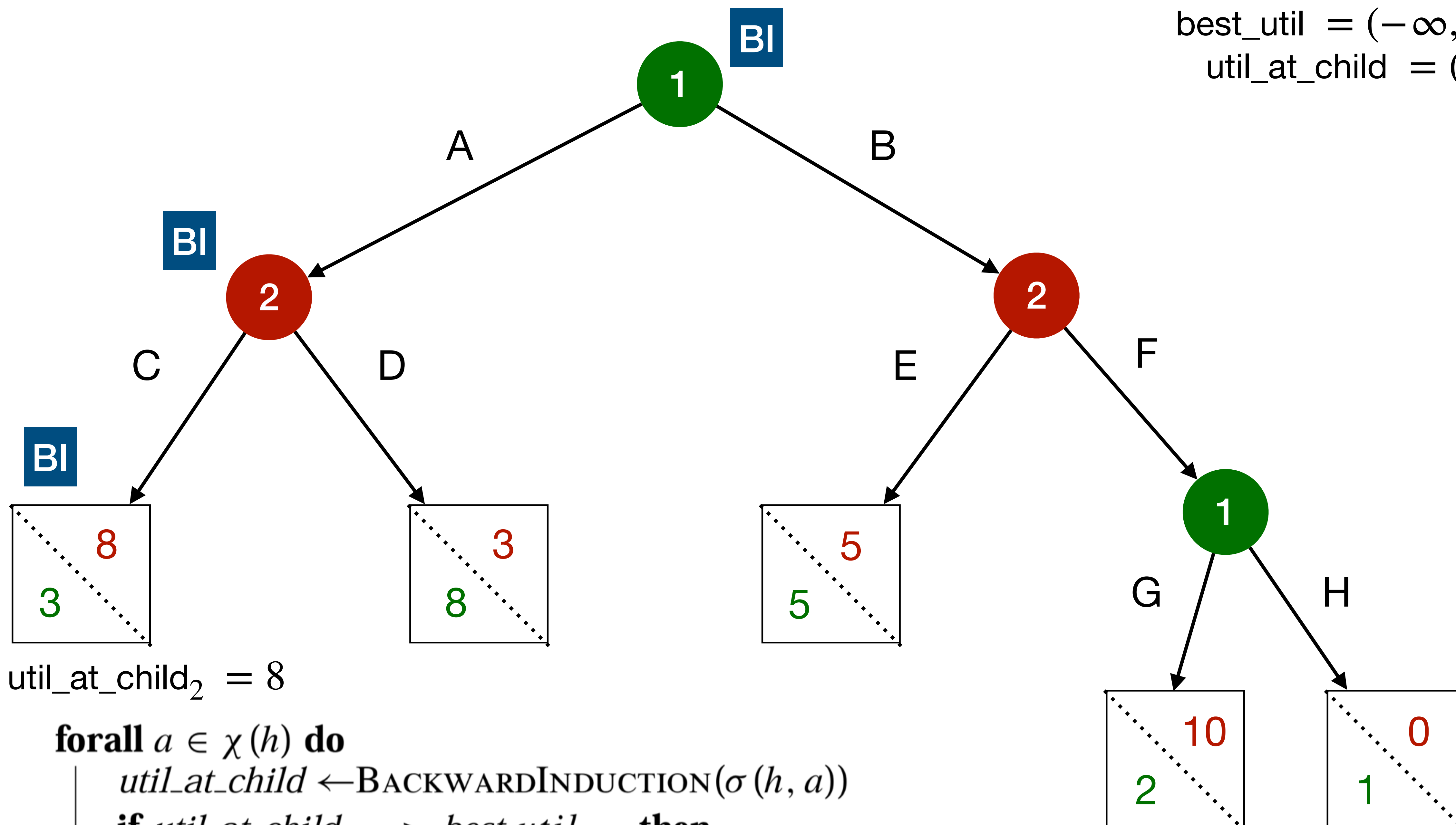
best_util $= (-\infty, -\infty)$
util_at_child $= (3,8)$



forall $a \in \chi(h)$ **do**
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    best_util $\leftarrow$ *util_at_child*
return *best_util*

Example

$$\text{best\_util} = (-\infty, -\infty)$$
$$\text{util\_at\_child} = (3,8)$$

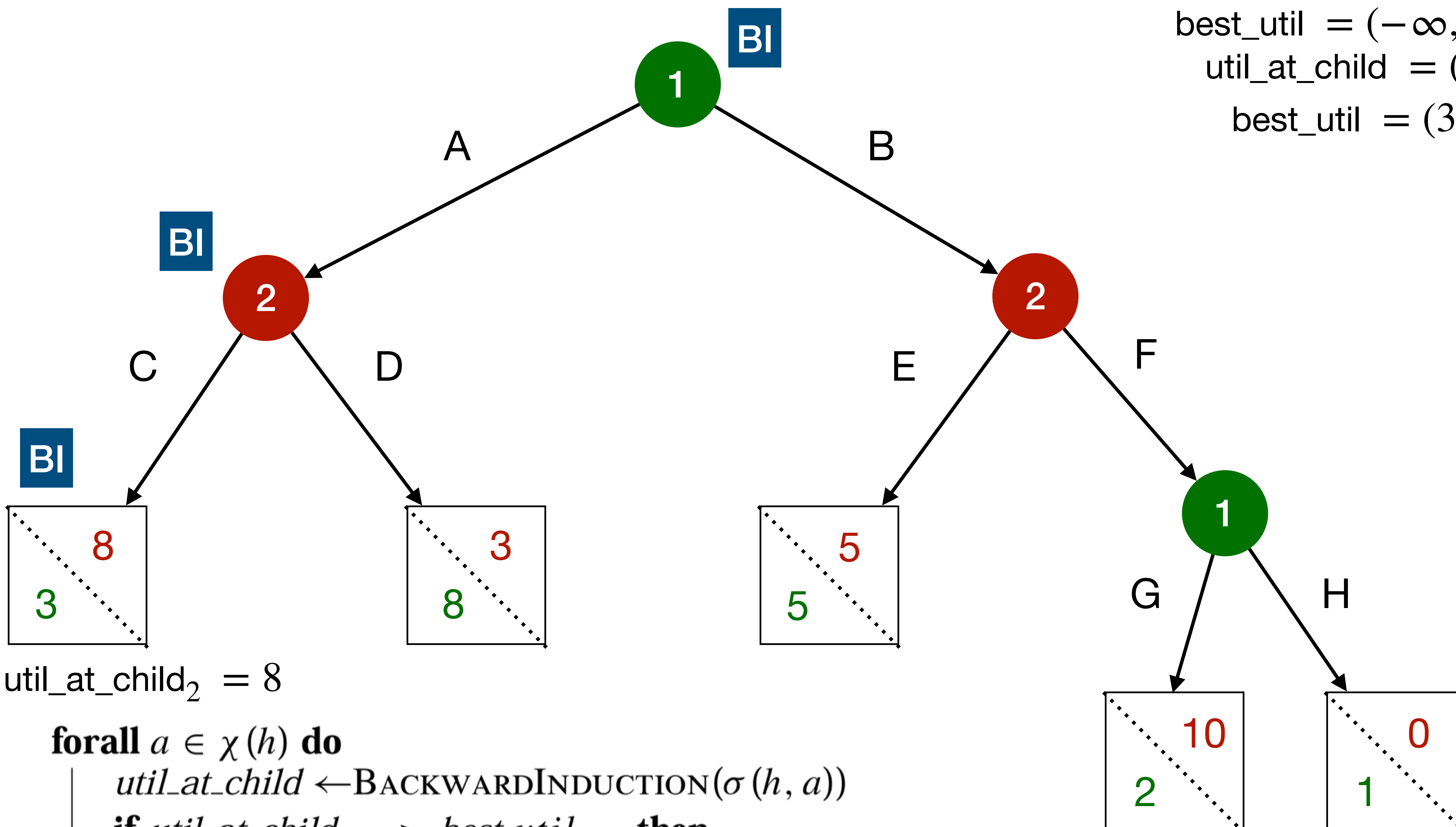


$$\text{util\_at\_child}_2 = 8$$

forall $a \in \chi(h)$ **do**
  $util\_at\_child \leftarrow \textsc{BackwardInduction}(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
return $best\_util$

Example

$$\text{best\_util} = (-\infty, -\infty)$$
$$\text{util\_at\_child} = (3,8)$$
$$\text{best\_util} = (3,8)$$



$\text{util\_at\_child}_2 = 8$

forall $a \in \chi(h)$ **do**

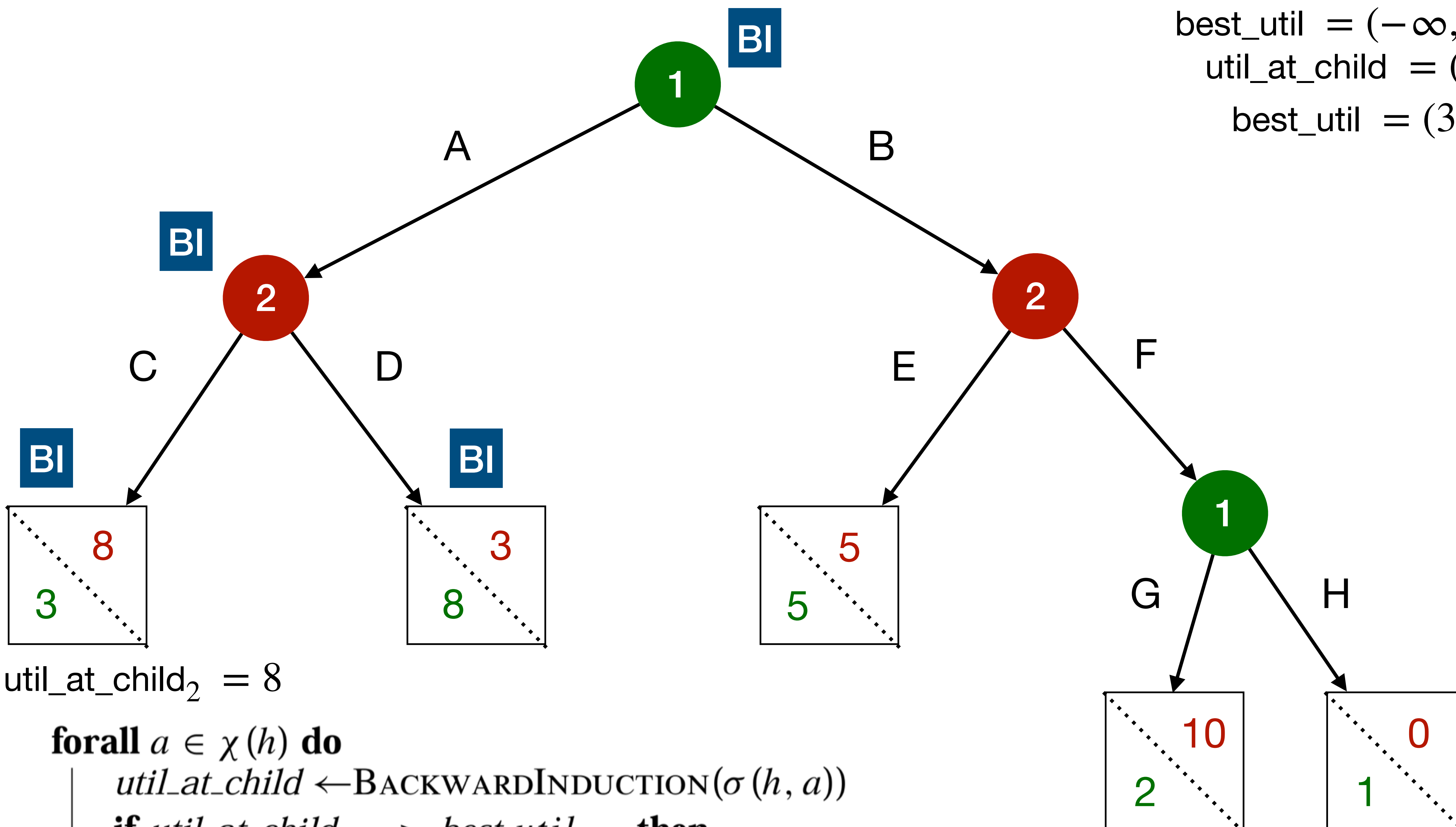
  $\mathit{util\_at\_child} \leftarrow \textsc{BackwardInduction}(\sigma(h, a))$

  if $\mathit{util\_at\_child}_{\rho(h)} > \mathit{best\_util}_{\rho(h)}$ **then**

    $\mathit{best\_util} \leftarrow \mathit{util\_at\_child}$

return $\mathit{best\_util}$

Example



best_util $= (-\infty, -\infty)$

util_at_child $= (3{,}8)$

best_util $= (3{,}8)$

util_at_child$_2$ $= 8$

forall $a \in \chi(h)$ **do**

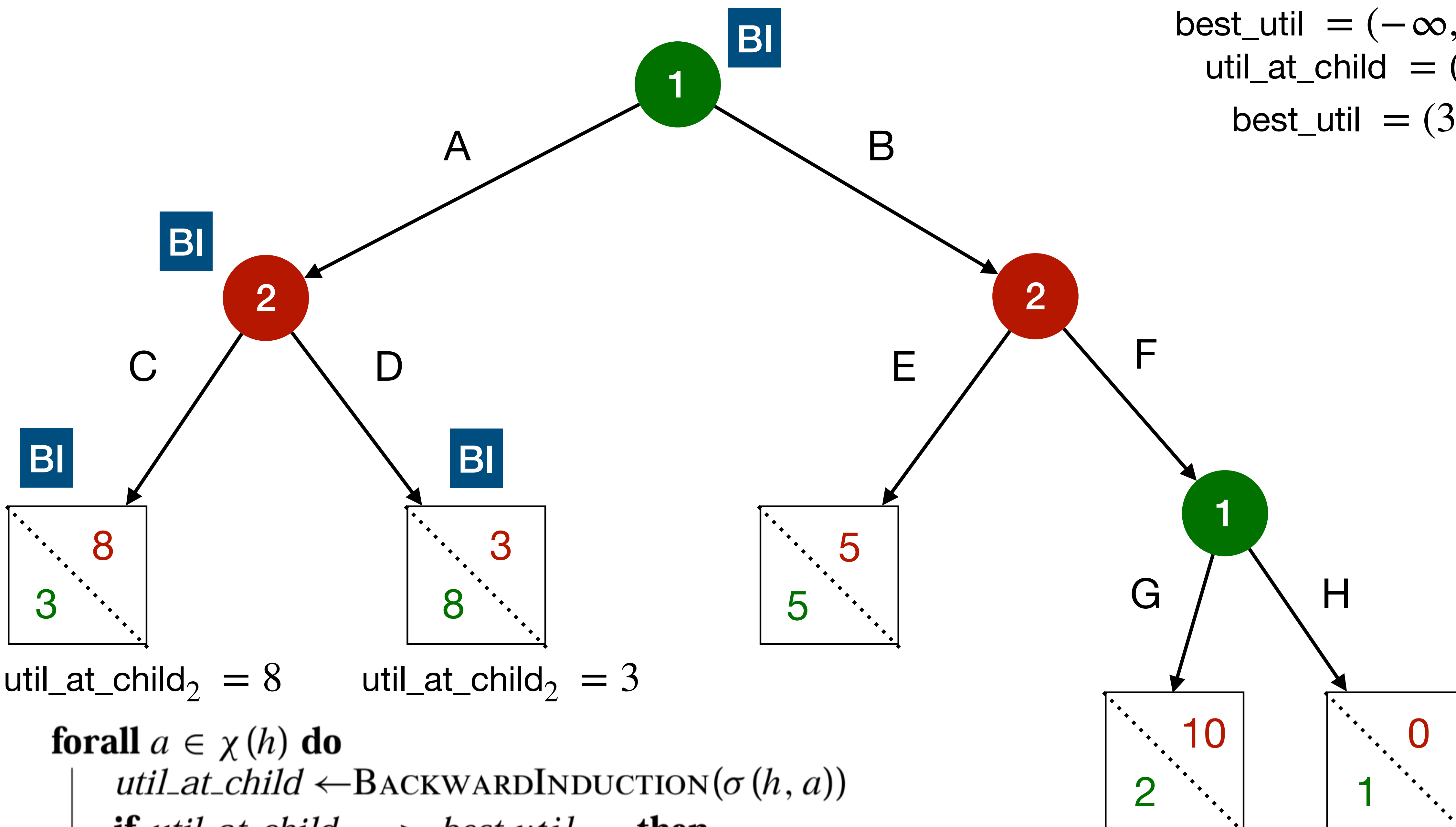
  util_at_child ← BACKWARDINDUCTION$(\sigma(h, a))$

  if *util_at_child*$_{\rho(h)} >$ *best_util*$_{\rho(h)}$ **then**

    best_util ← *util_at_child*

return *best_util*

Example



best_util $= (-\infty, -\infty)$
util_at_child $= (3,8)$
best_util $= (3,8)$

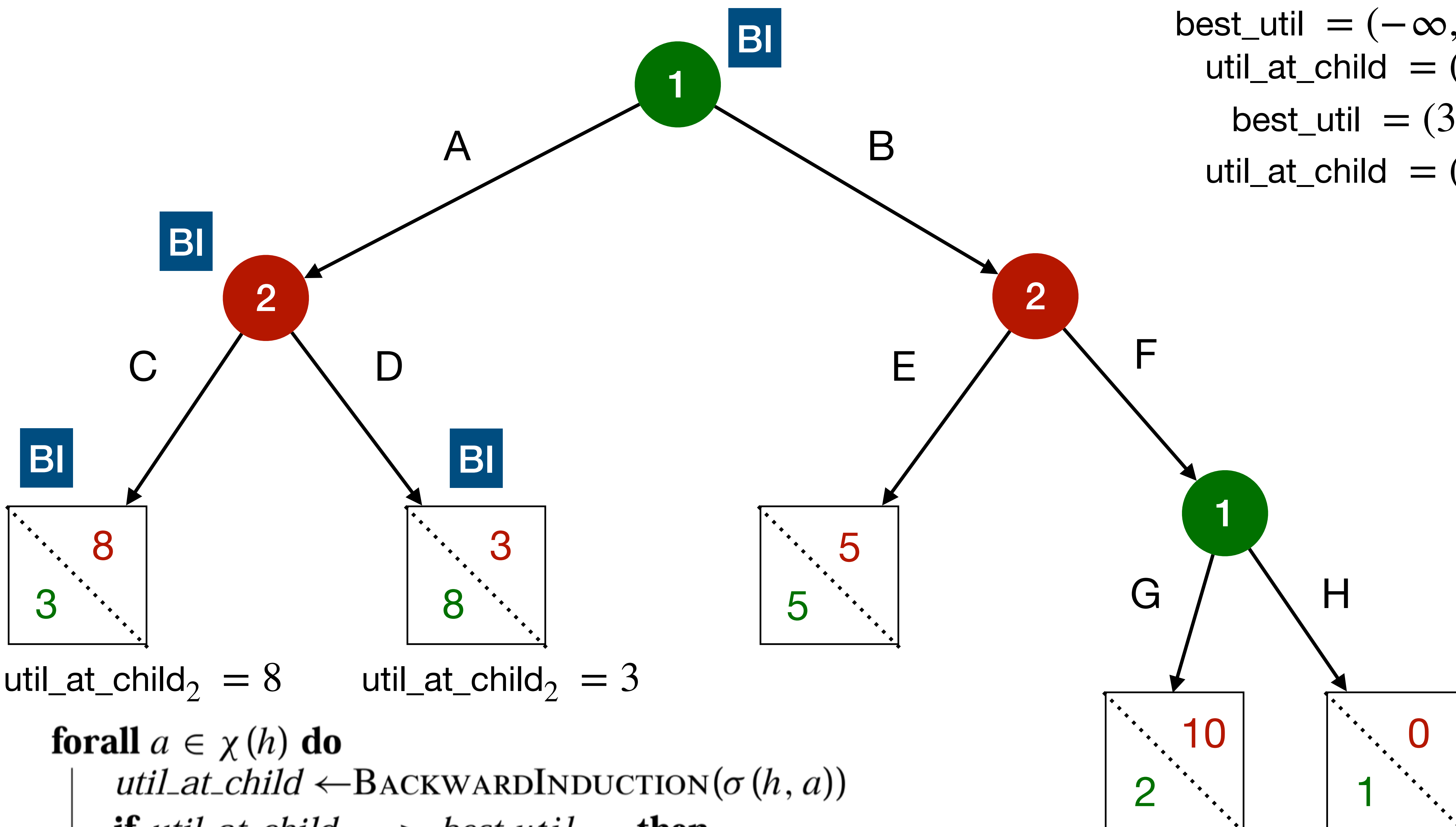
util_at_child$_2$ $= 8$

util_at_child$_2$ $= 3$

forall $a \in \chi(h)$ **do**
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    best_util $\leftarrow$ *util_at_child*
return *best_util*

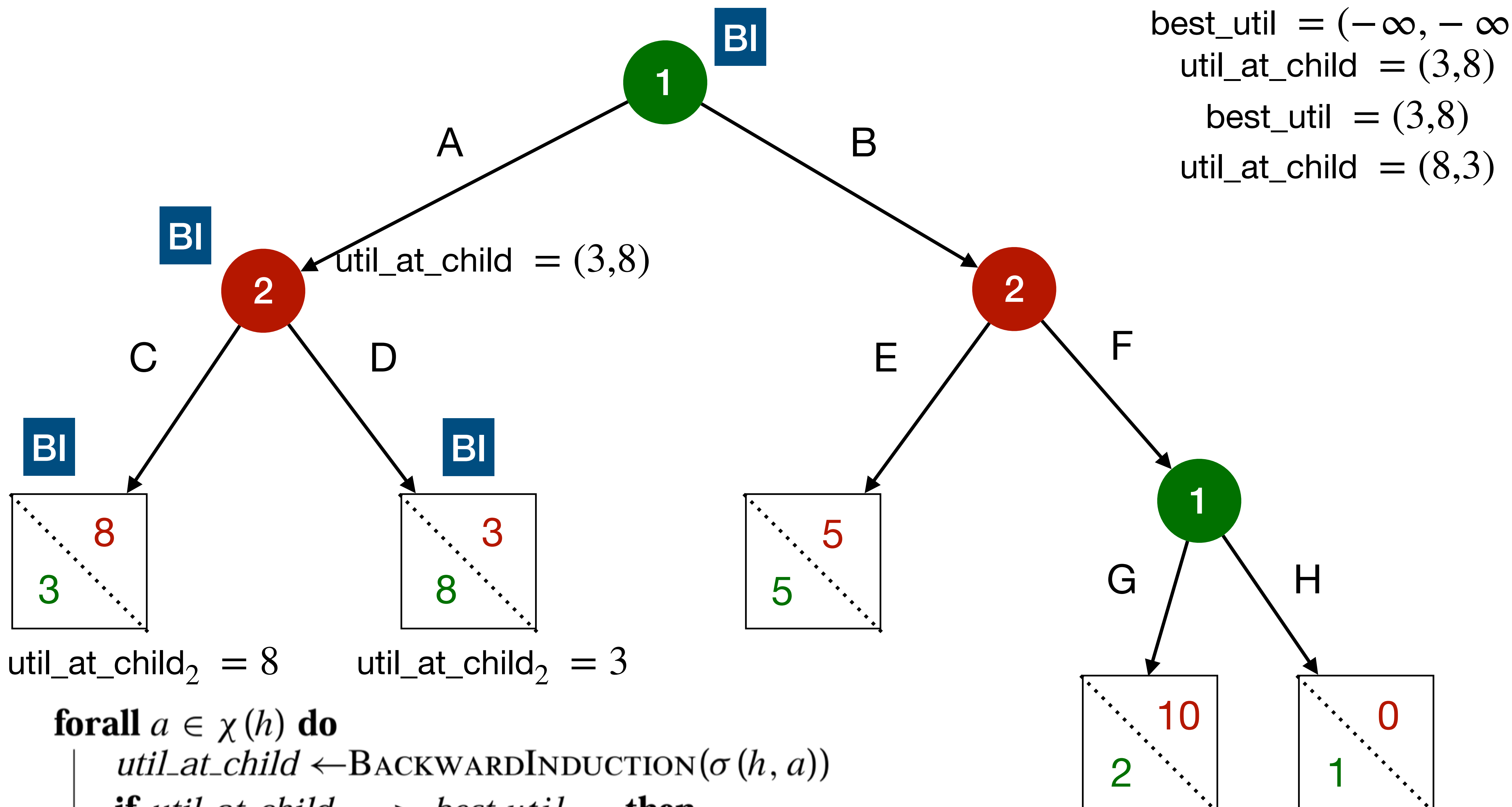
Example

$$\text{best\_util} = (-\infty, -\infty)$$
$$\text{util\_at\_child} = (3,8)$$
$$\text{best\_util} = (3,8)$$
$$\text{util\_at\_child} = (8,3)$$



forall $a \in \chi(h)$ **do**
  util_at_child $\leftarrow$ BackwardInduction$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    best_util $\leftarrow$ *util_at_child*
return *best_util*

Example



forall $a \in \chi(h)$ **do**

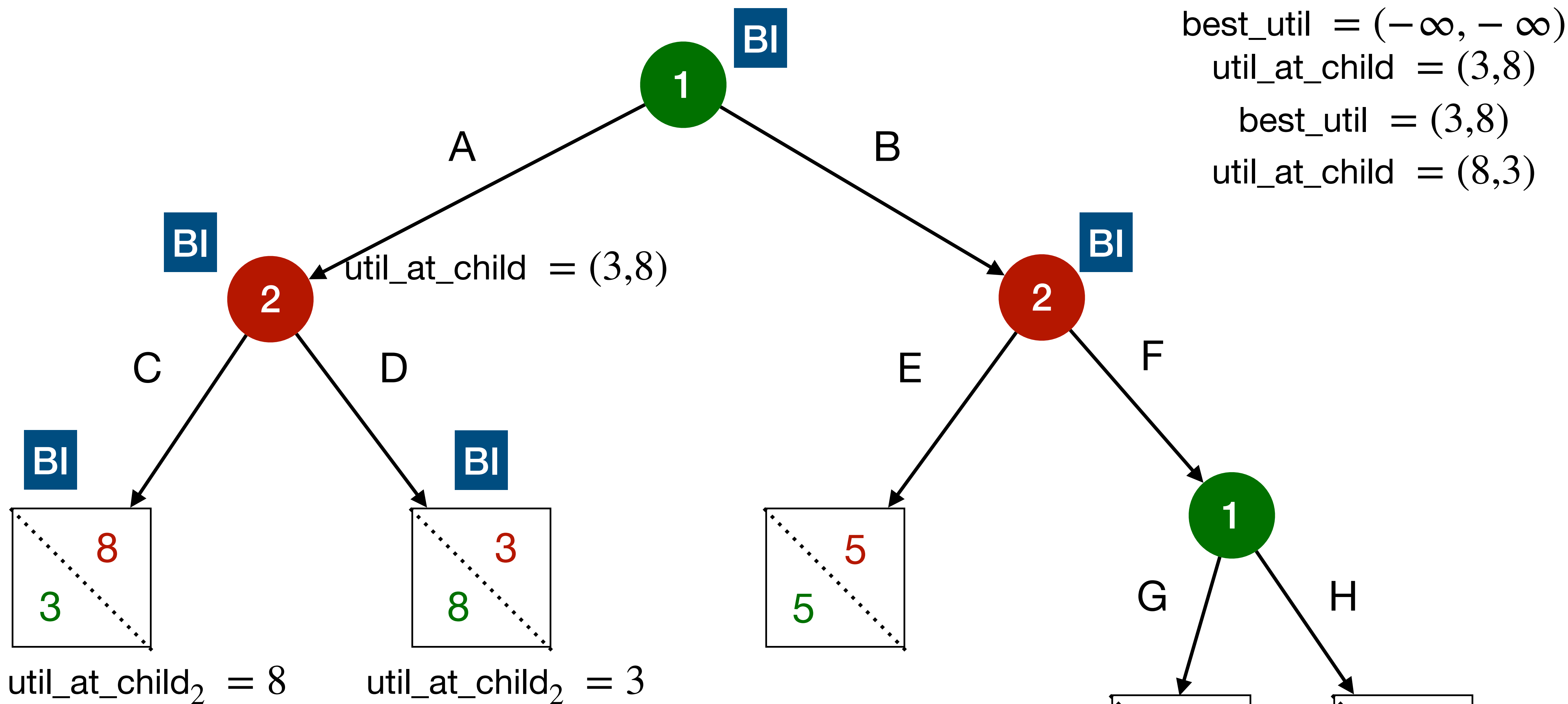
  $util\_at\_child \leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    $best\_util \leftarrow util\_at\_child$

return $best\_util$

Example



forall $a \in \chi(h)$ **do**

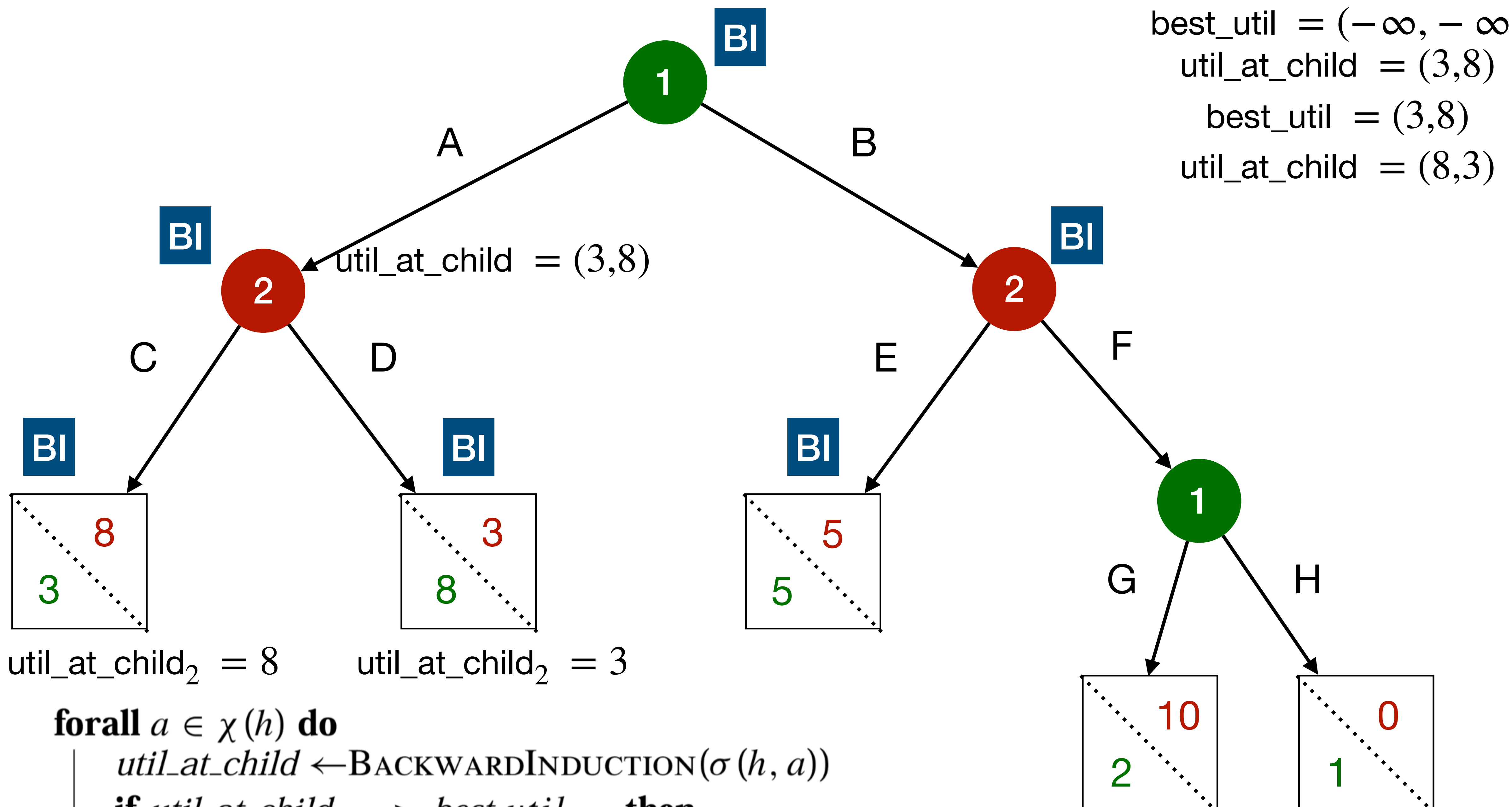
  util_at_child ← BackwardInduction$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util ← *util_at_child*

return *best_util*

Example



forall $a \in \chi(h)$ **do**

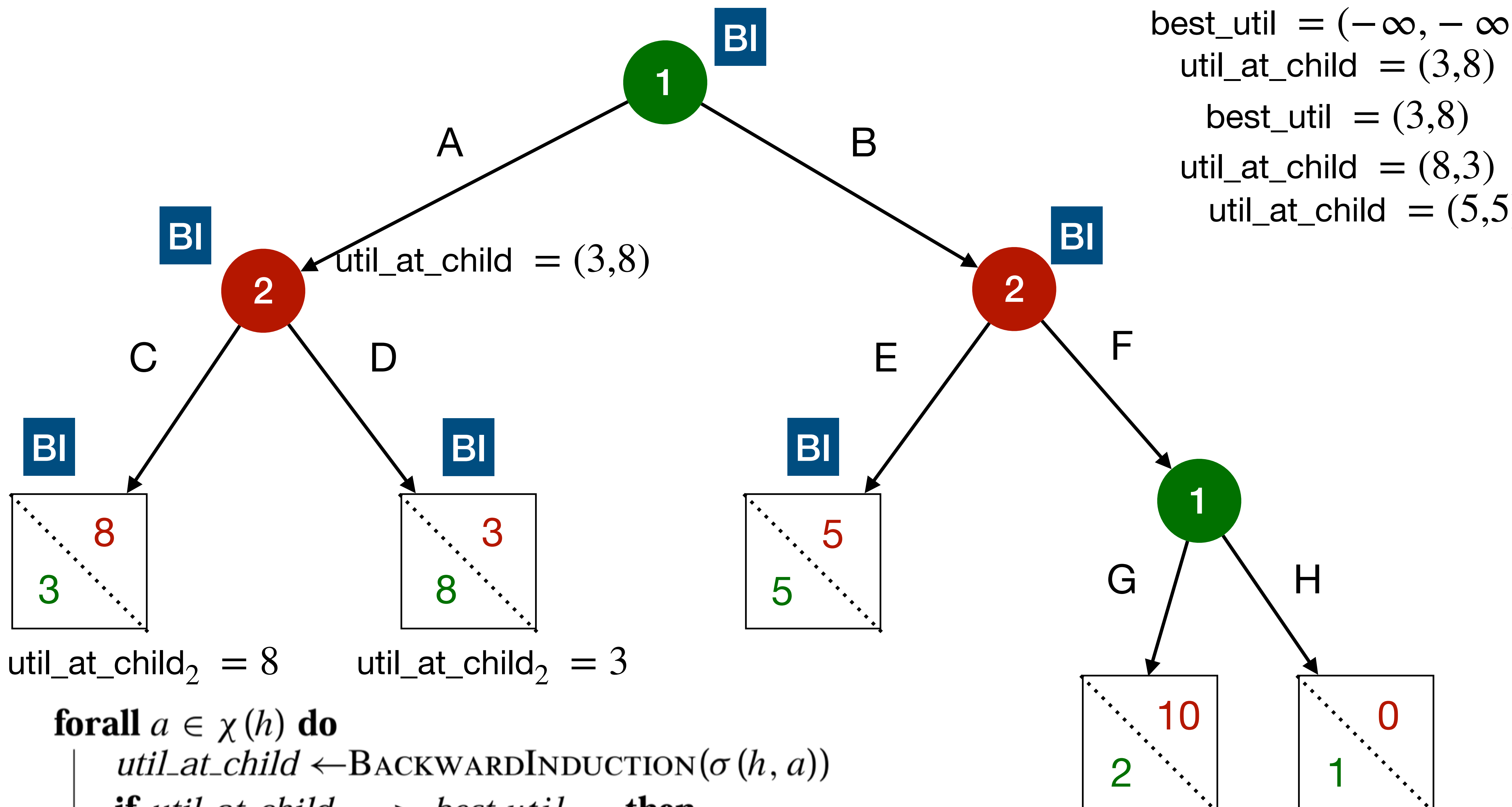
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if *util_at_child*$_{\rho(h)} >$ *best_util*$_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

return *best_util*

Example



forall $a \in \chi(h)$ **do**

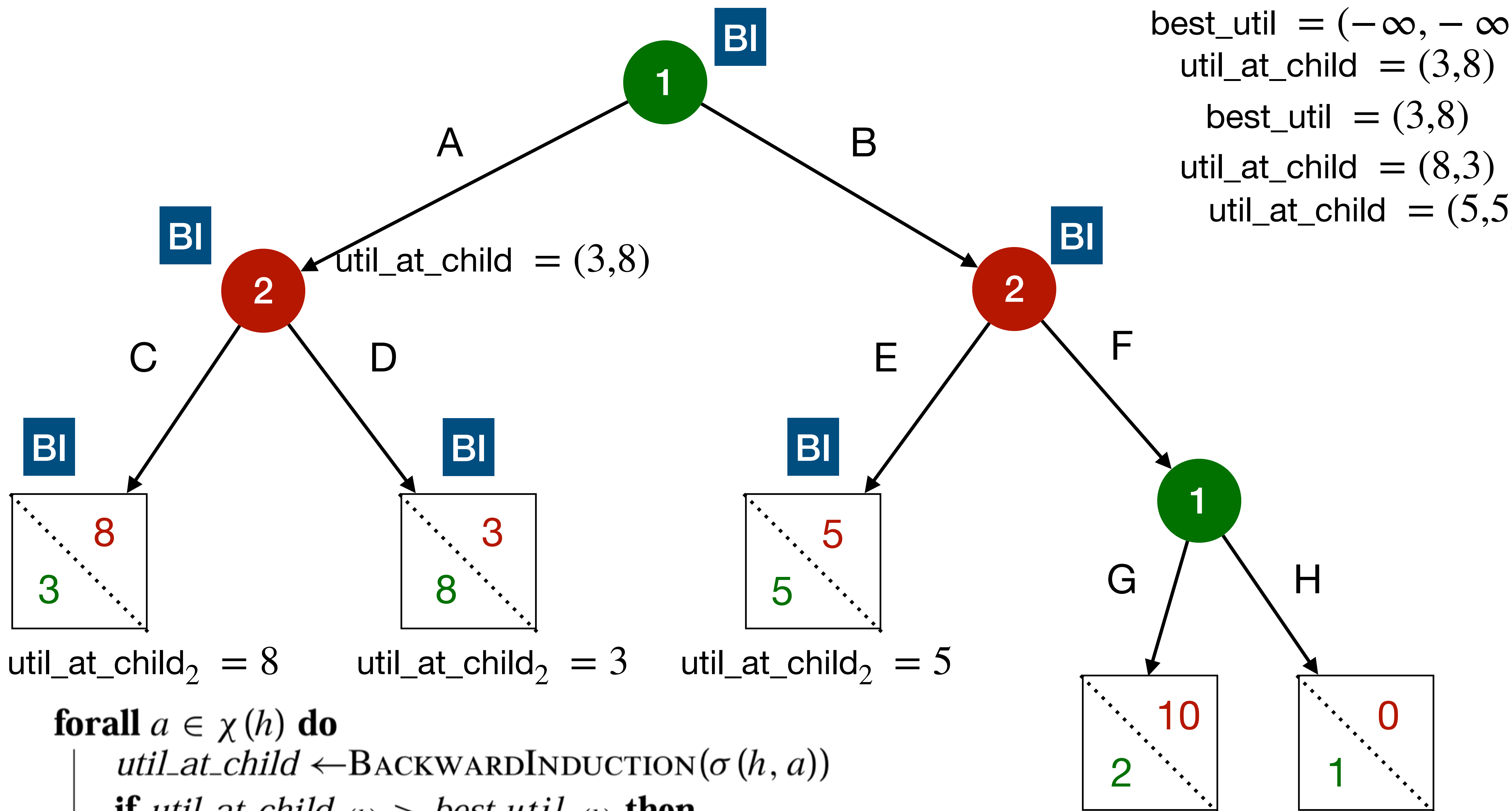
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

return *best_util*

Example



forall $a \in \chi(h)$ **do**

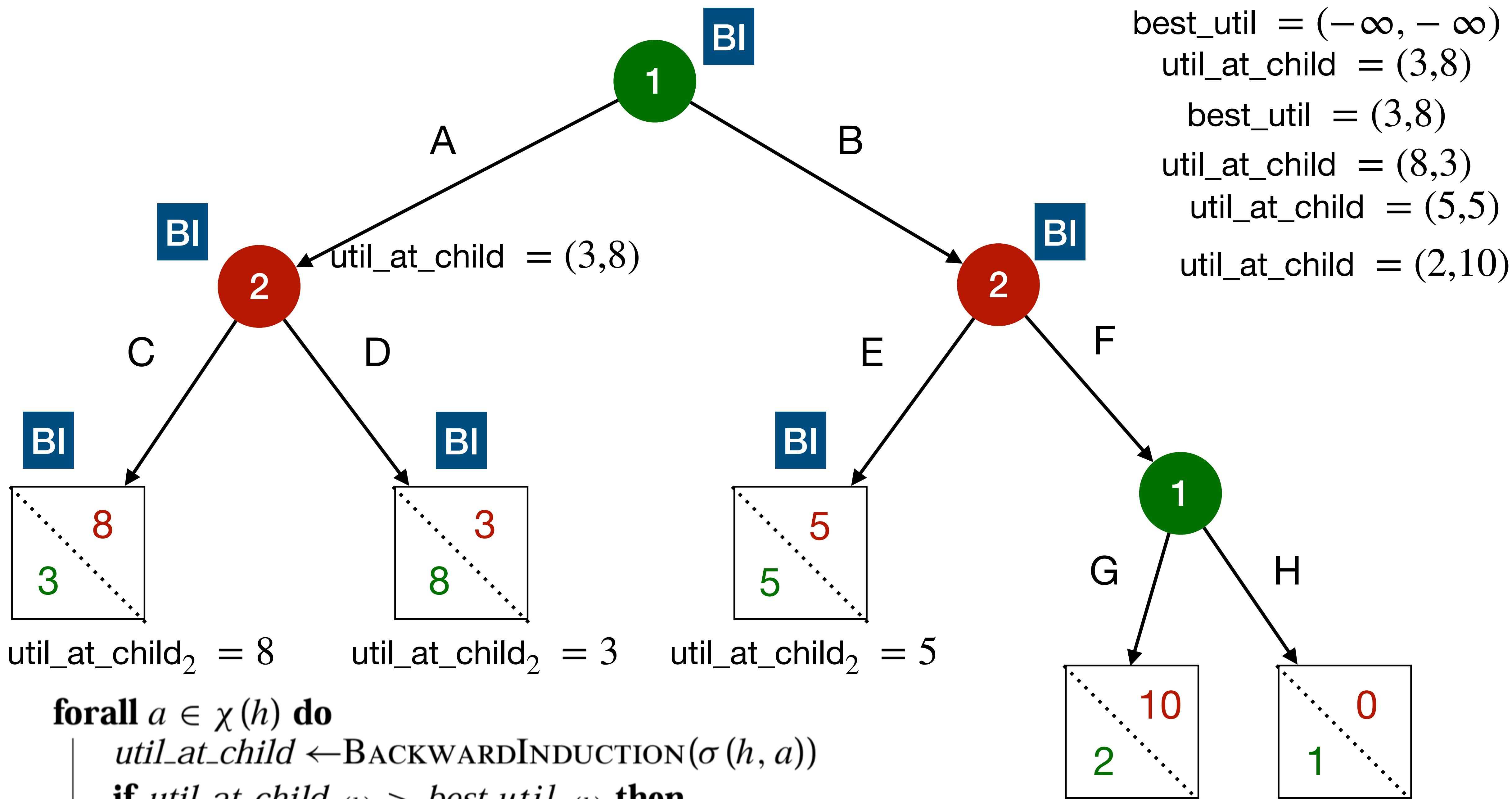
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

return *best_util*

Example



forall $a \in \chi(h)$ **do**

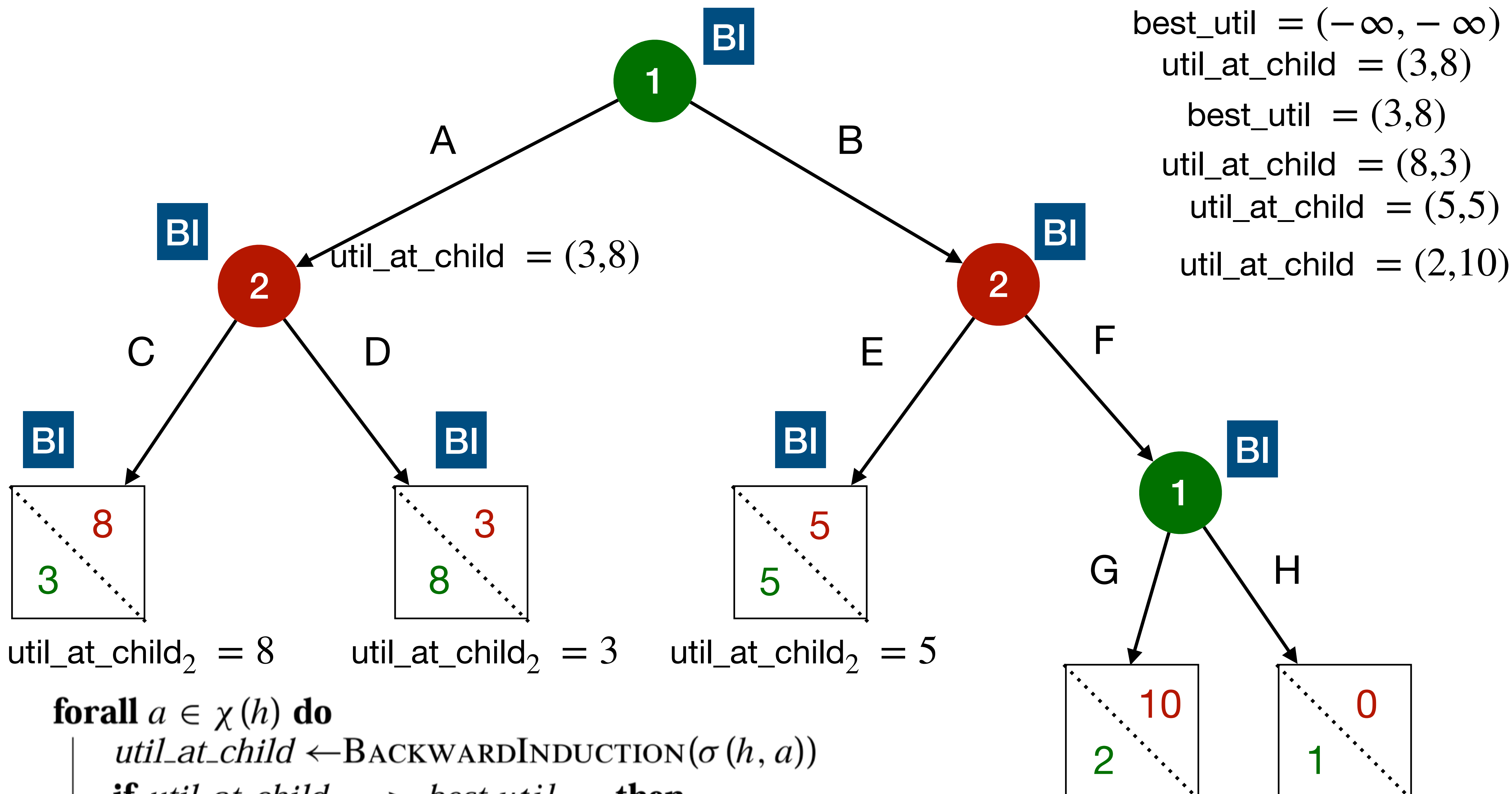
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

return *best_util*

Example



forall $a \in \chi(h)$ **do**

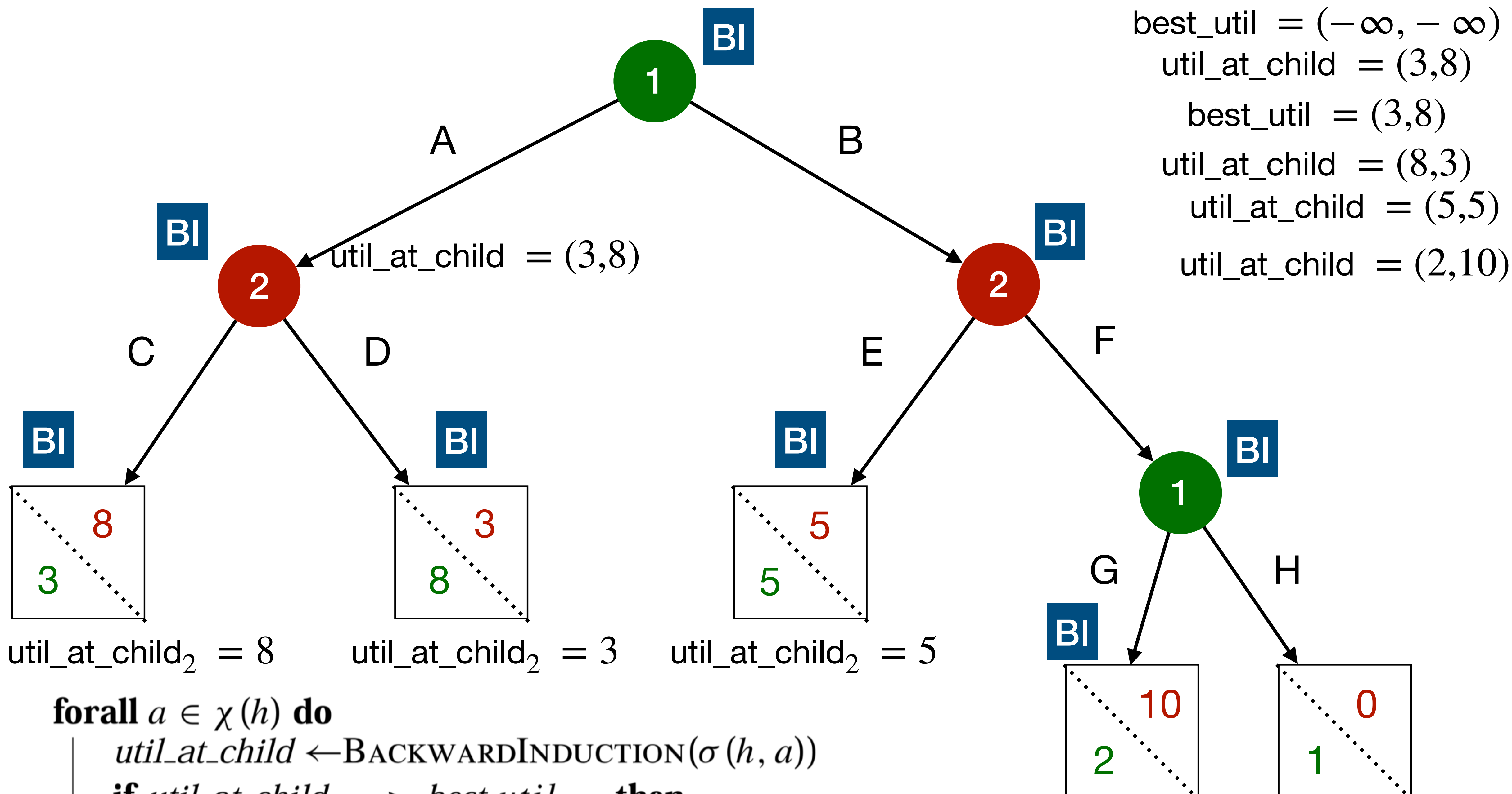
  $util\_at\_child \leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    $best\_util \leftarrow util\_at\_child$

return $best\_util$

Example



forall $a \in \chi(h)$ **do**

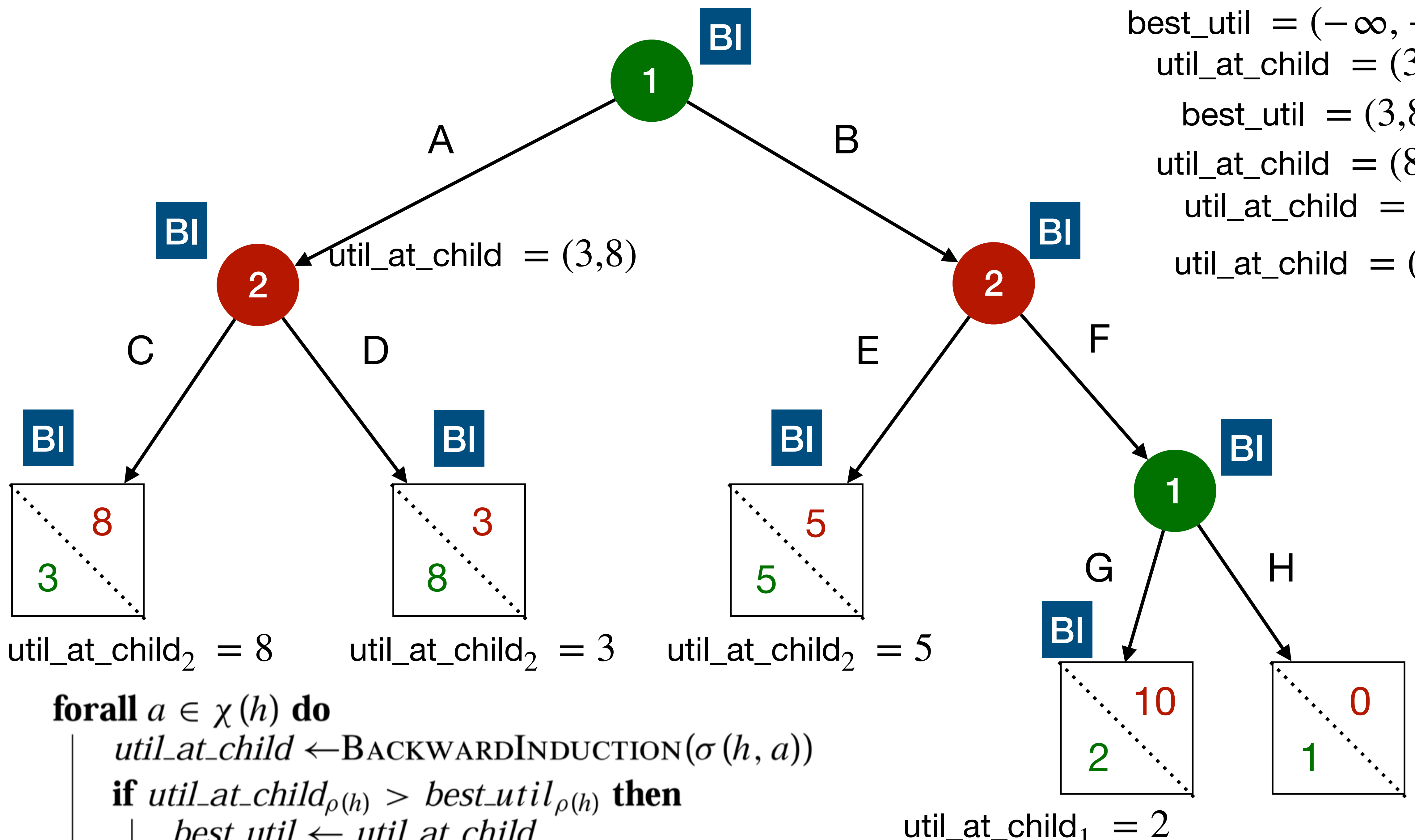
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $\textit{util\_at\_child}_{\rho(h)} > \textit{best\_util}_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

return *best_util*

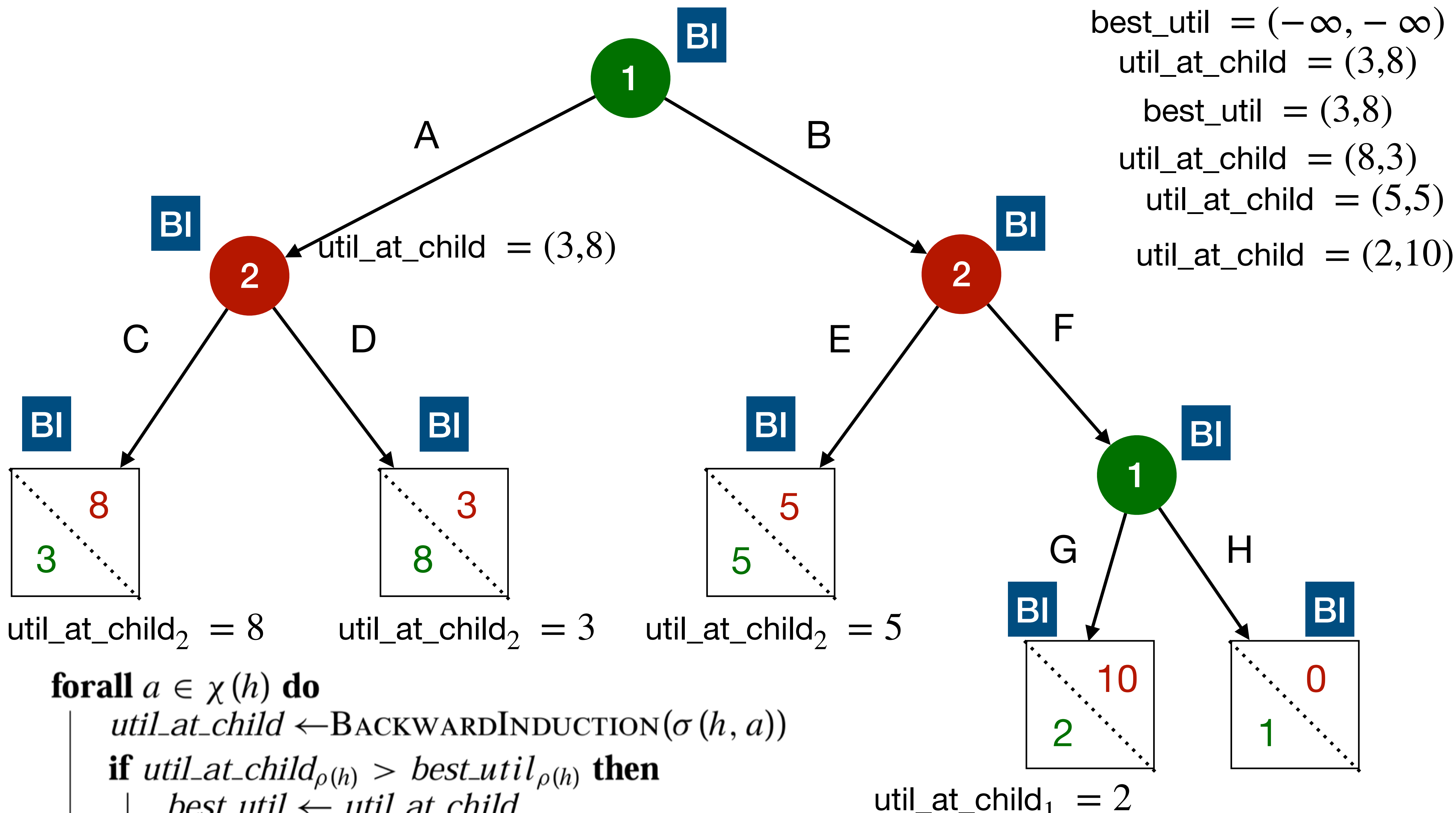
Example



best_util $= (-\infty, -\infty)$
util_at_child $= (3{,}8)$
best_util $= (3{,}8)$
util_at_child $= (8{,}3)$
util_at_child $= (5{,}5)$
util_at_child $= (2{,}10)$

forall $a \in \chi(h)$ **do**
  util_at_child ← $\textsc{BackwardInduction}(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    best_util ← *util_at_child*
return *best_util*

Example



forall $a \in \chi(h)$ **do**

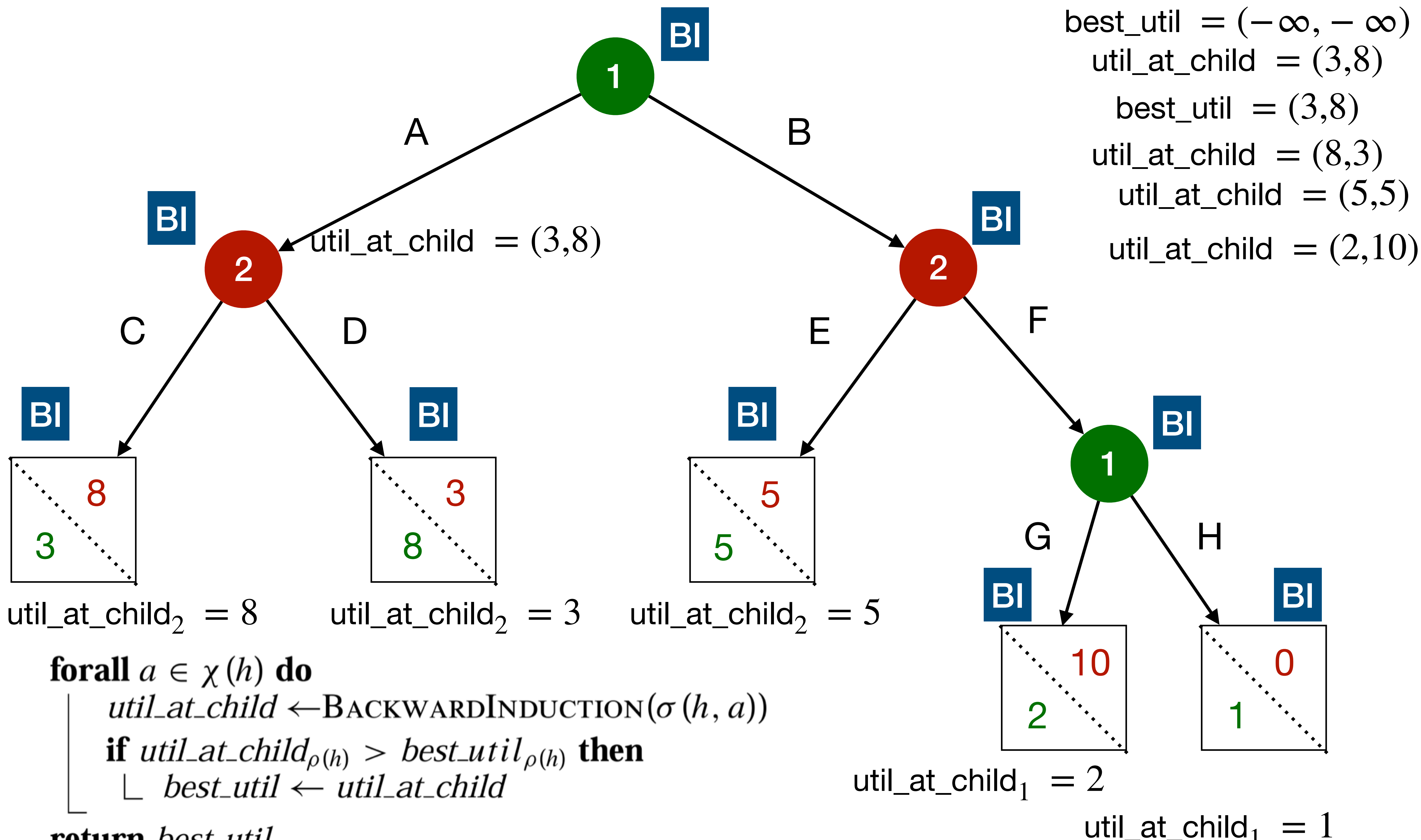
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util $\leftarrow$ *util_at_child*

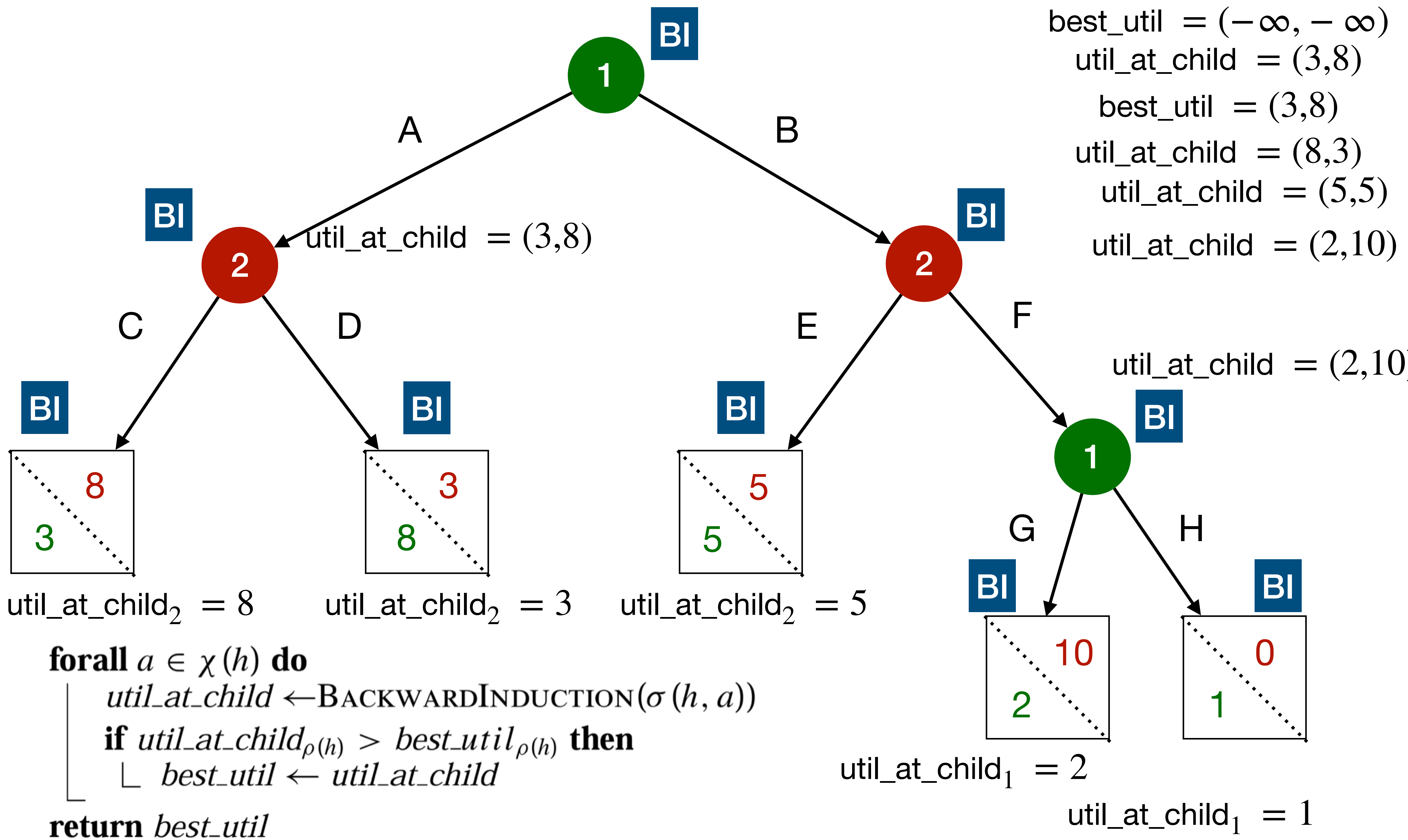
return *best_util*

Example



forall $a \in \chi(h)$ **do**
  $util\_at\_child \leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
return $best\_util$

Example



forall $a \in \chi(h)$ **do**

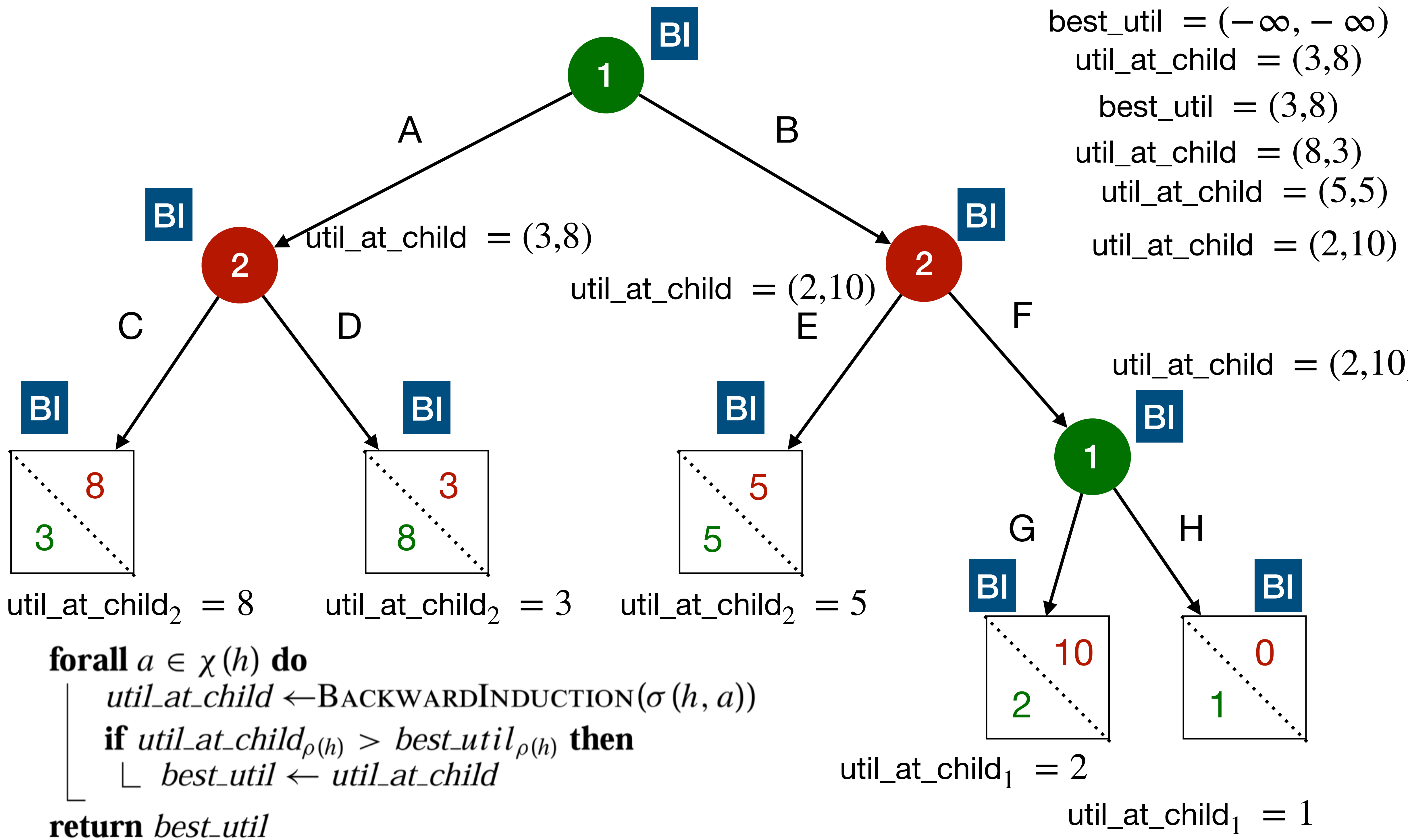
  util_at_child ←BACKWARDINDUCTION$(\sigma(h, a))$

  if *util_at_child*$_{\rho(h)}$ $>$ *best_util*$_{\rho(h)}$ **then**

    best_util ← *util_at_child*

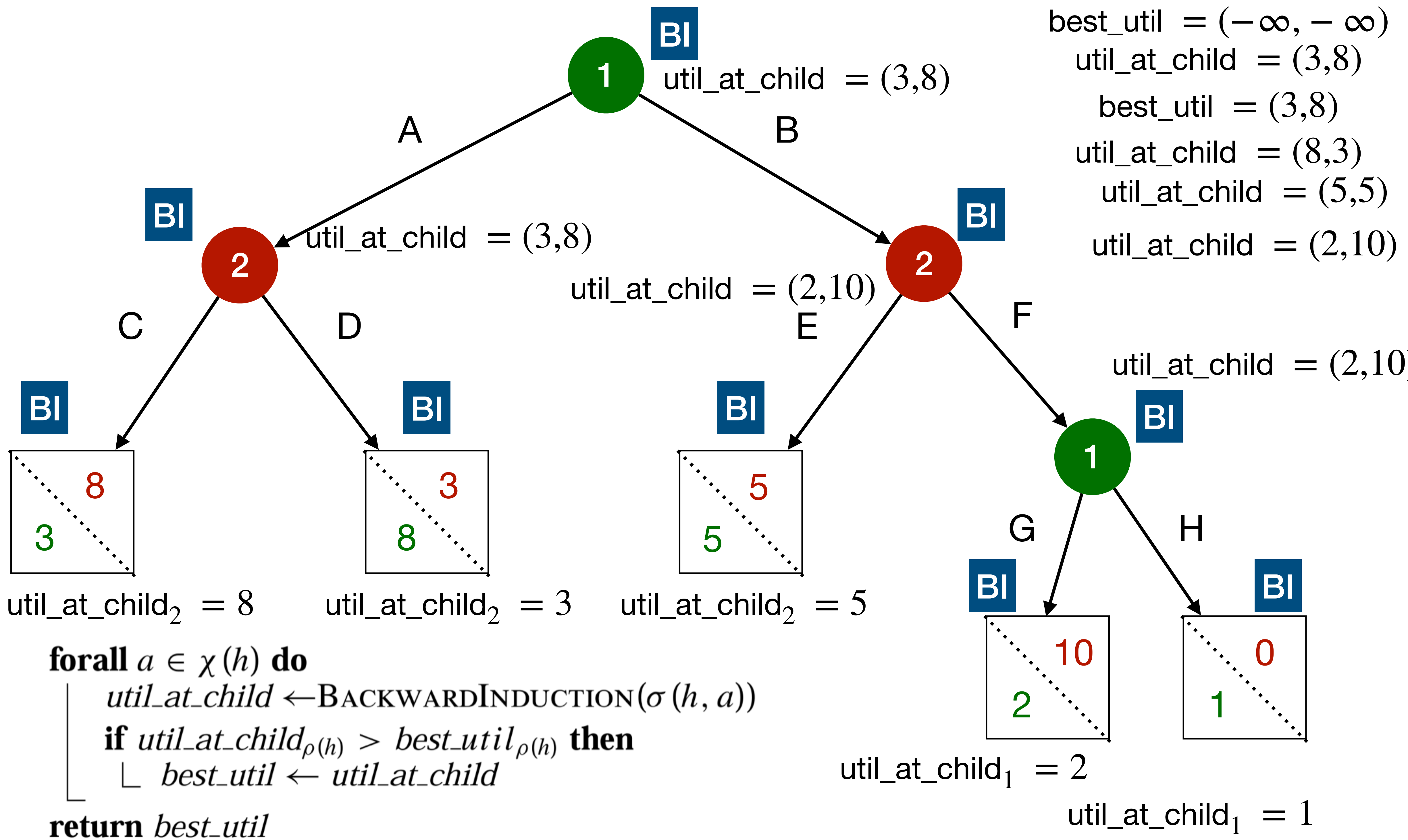
return *best_util*

Example



forall $a \in \chi(h)$ **do**
  $\quad$ *util_at_child* $\leftarrow$ BACKWARDINDUCTION$(\sigma(h,a))$
  $\quad$ **if** $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
  $\quad\quad$ *best_util* $\leftarrow$ *util_at_child*
return *best_util*

Example



forall $a \in \chi(h)$ **do**

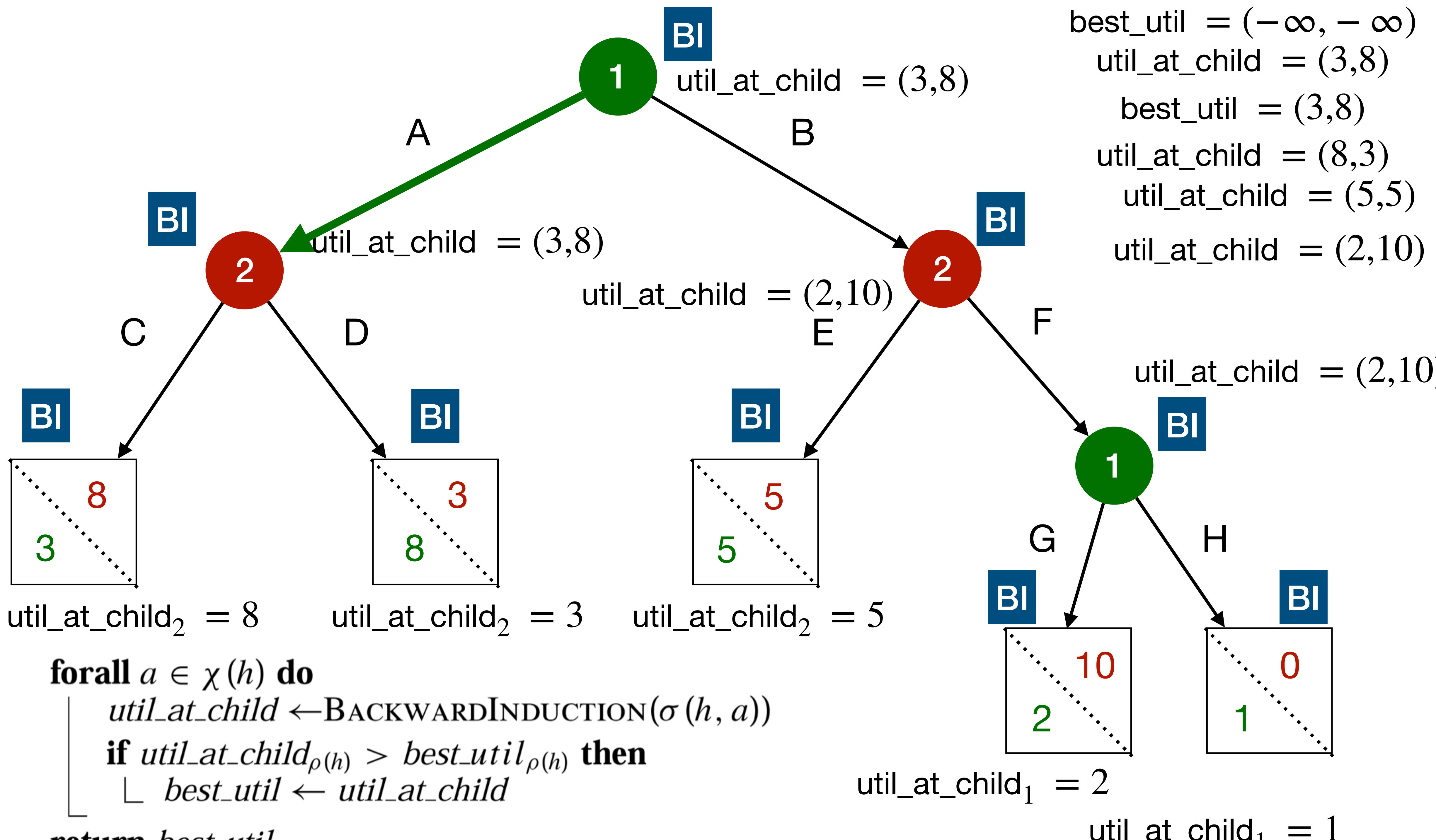
  util_at_child ← BackwardInduction($\sigma(h, a)$)

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util ← *util_at_child*

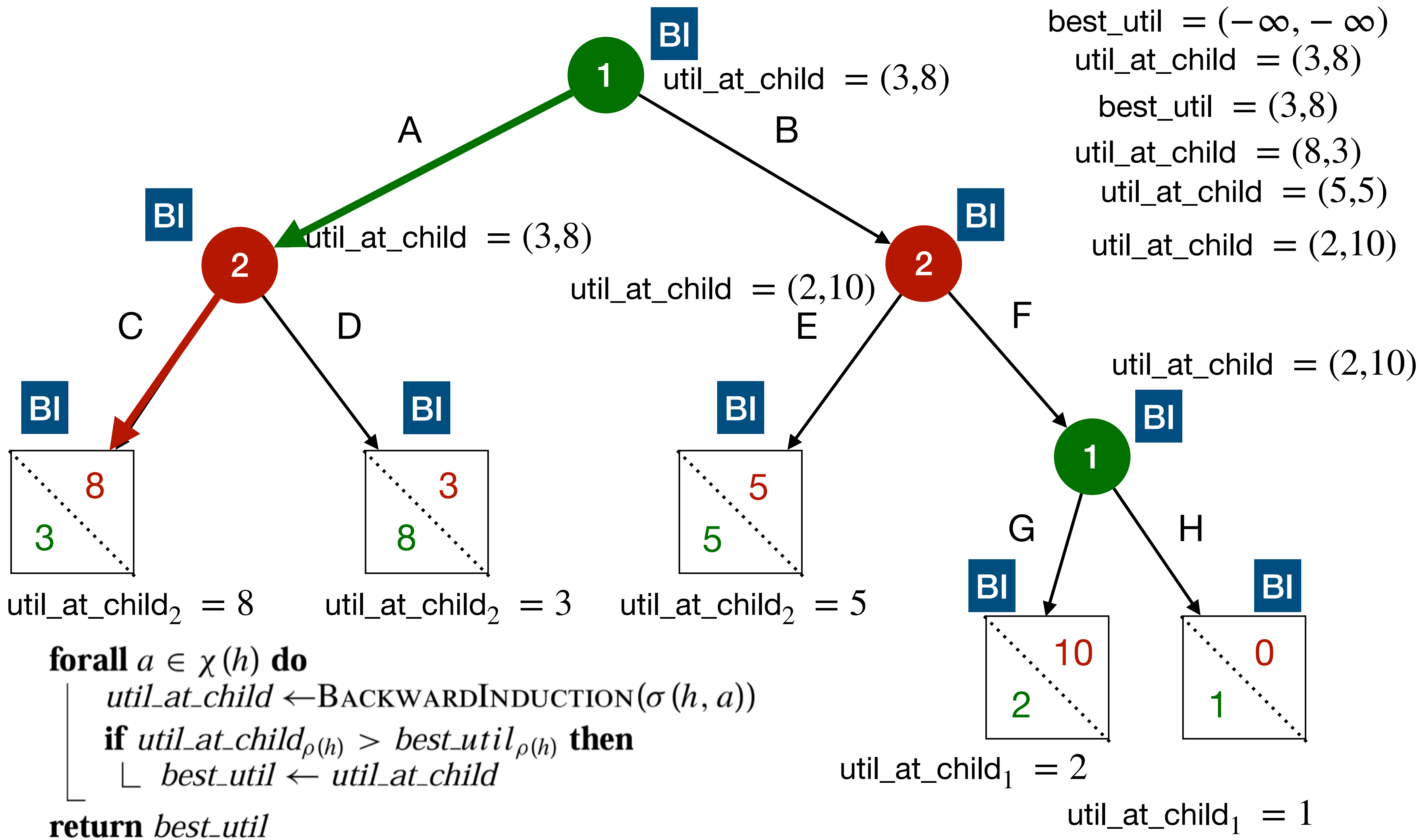
return *best_util*

Example



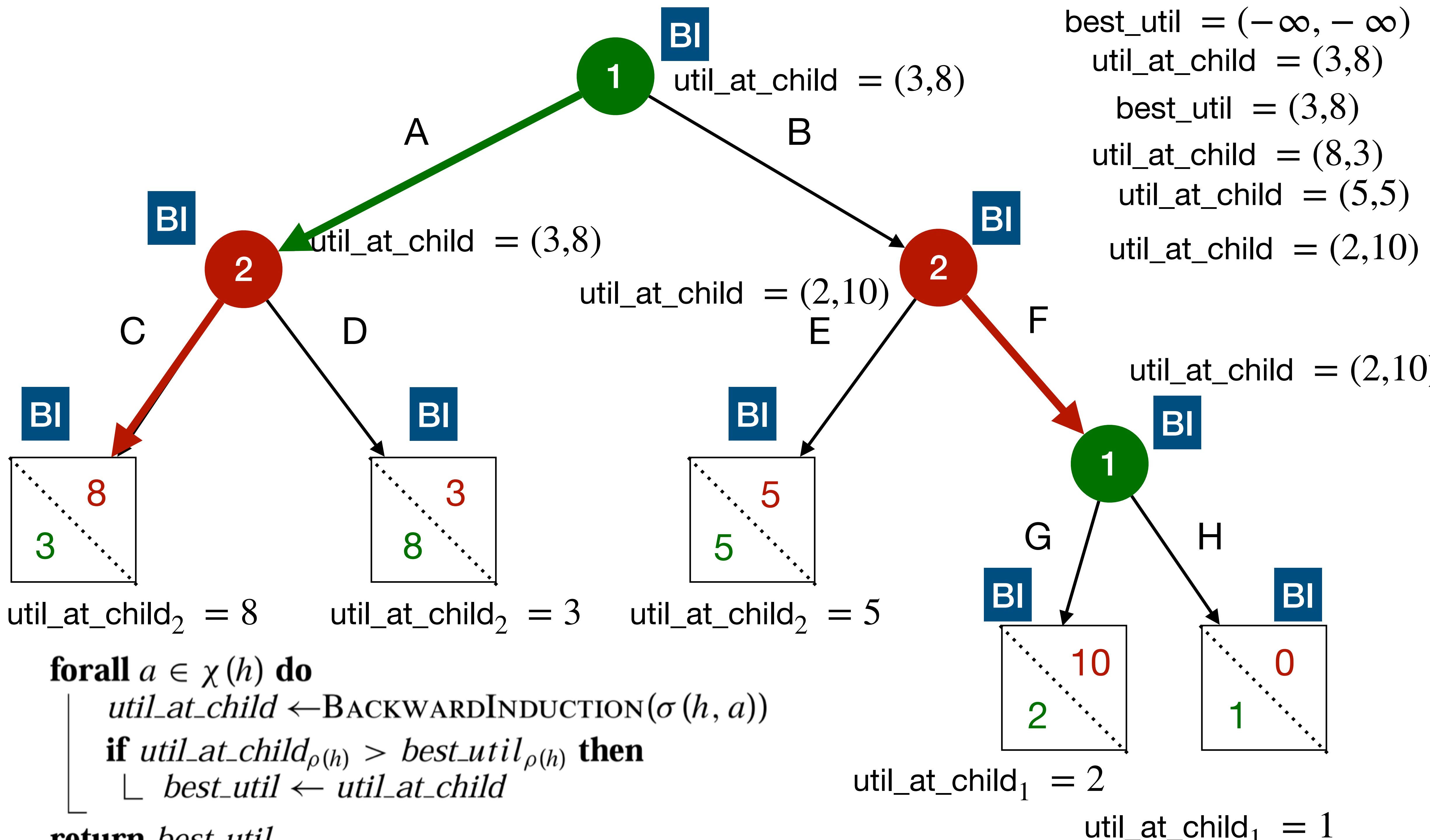
forall $a \in \chi(h)$ **do**
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    best_util $\leftarrow$ *util_at_child*
return *best_util*

Example



forall $a \in \chi(h)$ **do**
  util_at_child $\leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    best_util $\leftarrow$ *util_at_child*
return *best_util*

Example



forall $a \in \chi(h)$ **do**

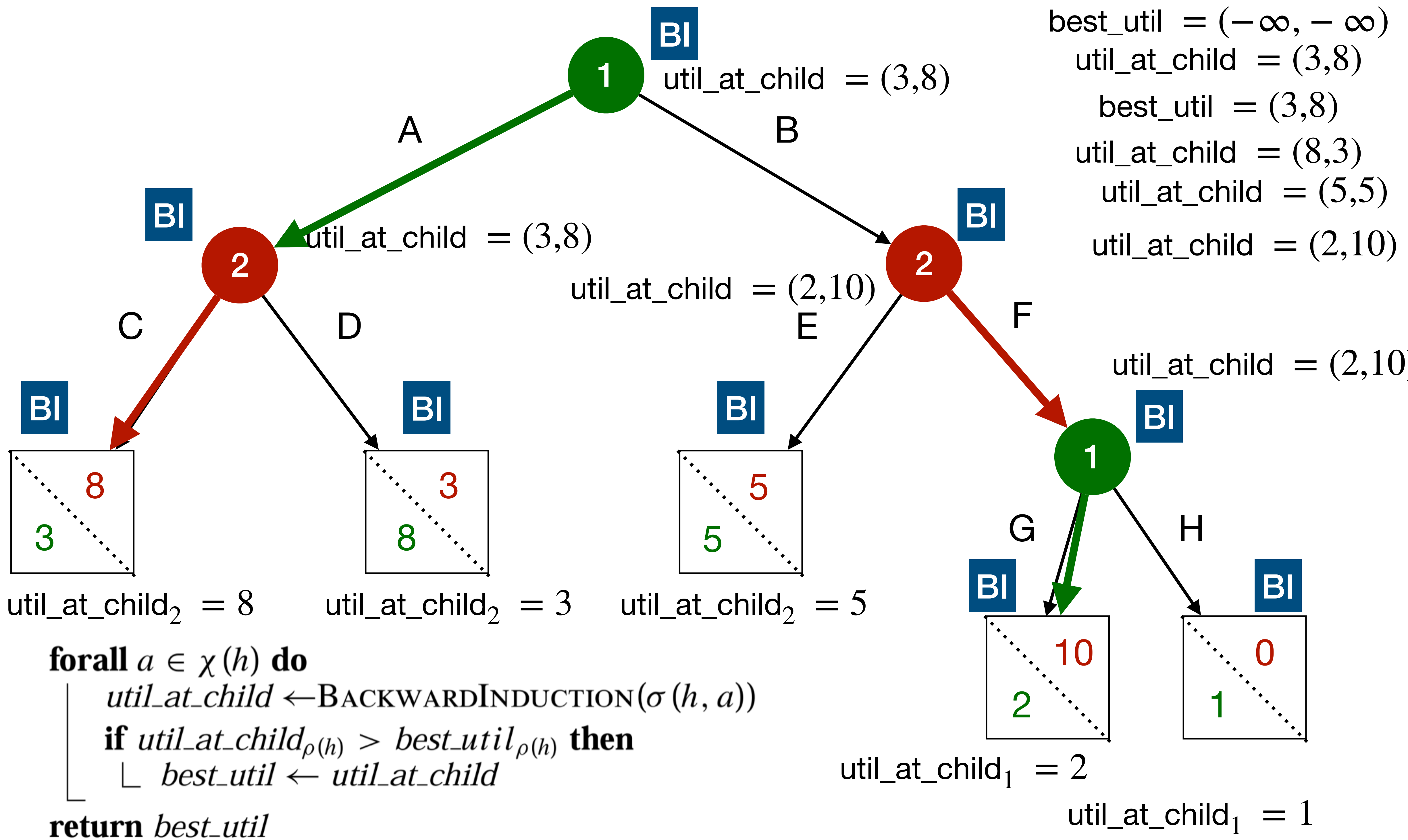
  util_at_child ←BackwardInduction($\sigma(h,a)$)

  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**

    best_util ← *util_at_child*

return *best_util*

Example



forall $a \in \chi(h)$ **do**
  $util\_at\_child \leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
return $best\_util$

So is everything fine and dandy?

So is everything fine and dandy?

It is, if we have the whole game tree as input.

So is everything fine and dandy?

It is, if we have the whole game tree as input.

But in many applications we do not really have that.

So is everything fine and dandy?

It is, if we have the whole game tree as input.

But in many applications we do not really have that.

Imagine that you want to “solve” chess, i.e., to decide if white has a winning strategy.

So is everything fine and dandy?

It is, if we have the whole game tree as input.

But in many applications we do not really have that.

Imagine that you want to “solve” chess, i.e., to decide if white has a winning strategy.

Chess is win-lose-draw game (1: white wins, -1: white loses: 0: there is a tie).

So is everything fine and dandy?

It is, if we have the whole game tree as input.

But in many applications we do not really have that.

Imagine that you want to “solve” chess, i.e., to decide if white has a winning strategy.

Chess is win-lose-draw game (1: white wins, -1: white loses: 0: there is a tie).

We could solve it using backwards induction to find if the value is 1,-1, or 0.

So is everything fine and dandy?

It is, if we have the whole game tree as input.

But in many applications we do not really have that.

Imagine that you want to “solve” chess, i.e., to decide if white has a winning strategy.

Chess is win-lose-draw game (1: white wins, -1: white loses: 0: there is a tie).

We could solve it using backwards induction to find if the value is 1,-1, or 0.

But the chess game tree has about 10^{150} nodes!

Pruning

Pruning

Maybe we do not have to consider all “branches” of the tree, i.e., all subgames.

Pruning

Maybe we do not have to consider all “branches” of the tree, i.e., all subgames.

Assume that we have a function $\text{Eval}(w)$ which (heuristically) evaluates the “score” of a node (and the subtree rooted at that node).

Pruning

Maybe we do not have to consider all “branches” of the tree, i.e., all subgames.

Assume that we have a function Eval(w) which (heuristically) evaluates the “score” of a node (and the subtree rooted at that node).

If Eval(w) we can stop the search at that node, and not explore the subtree.

(α, β)-pruning

(α, β)-pruning

We will apply the pruning idea to perfect information extensive form, zero-sum games.

(α, β)-pruning

We will apply the pruning idea to perfect information extensive form, zero-sum games.

<u>Idea: We prune if:</u>

(α, β)-pruning

We will apply the pruning idea to perfect information extensive form, zero-sum games.

<u>Idea: We prune if:</u>

- the maximiser can guarantee score $\geq \alpha$

(α, β)-pruning

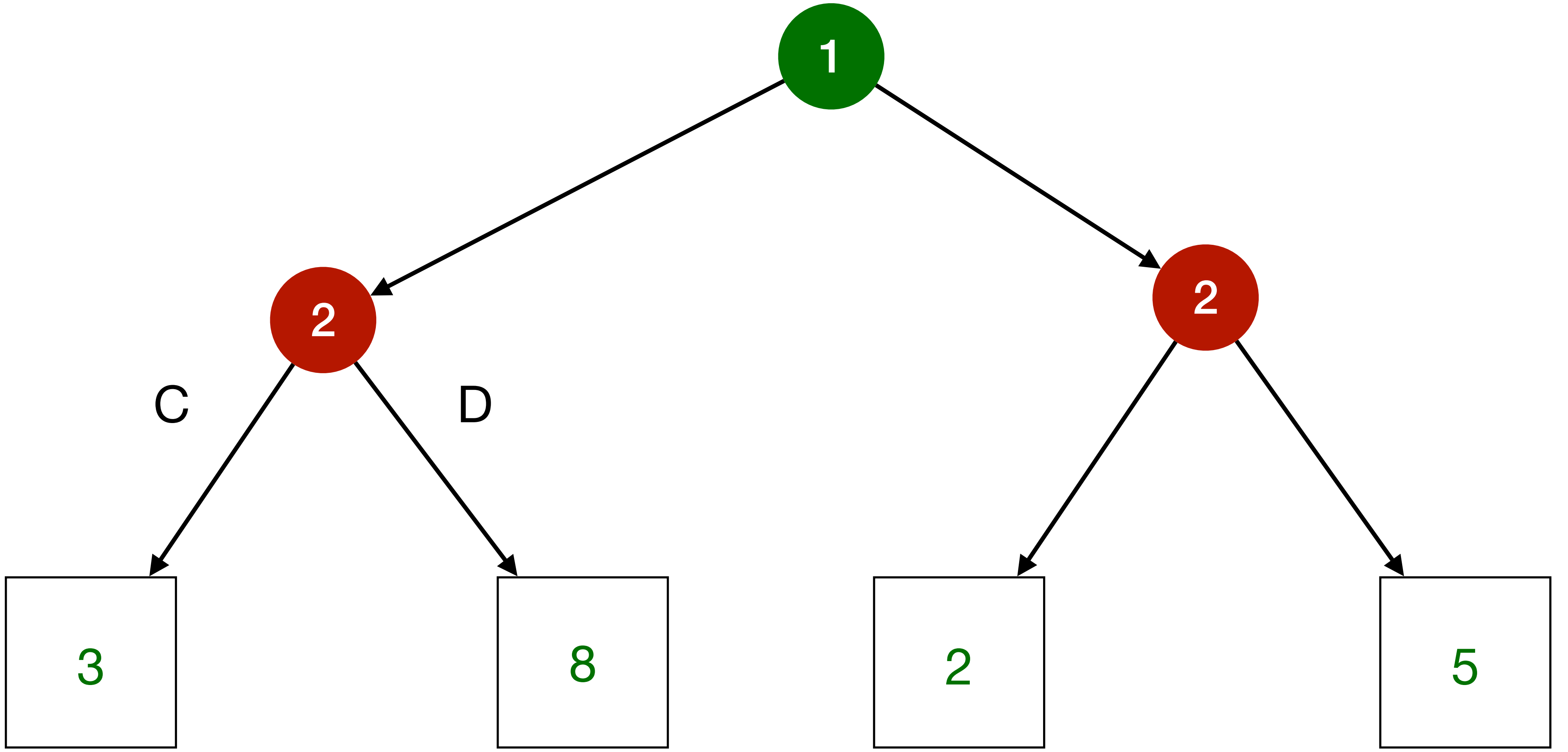
We will apply the pruning idea to perfect information extensive form, zero-sum games.

Idea: We prune if:

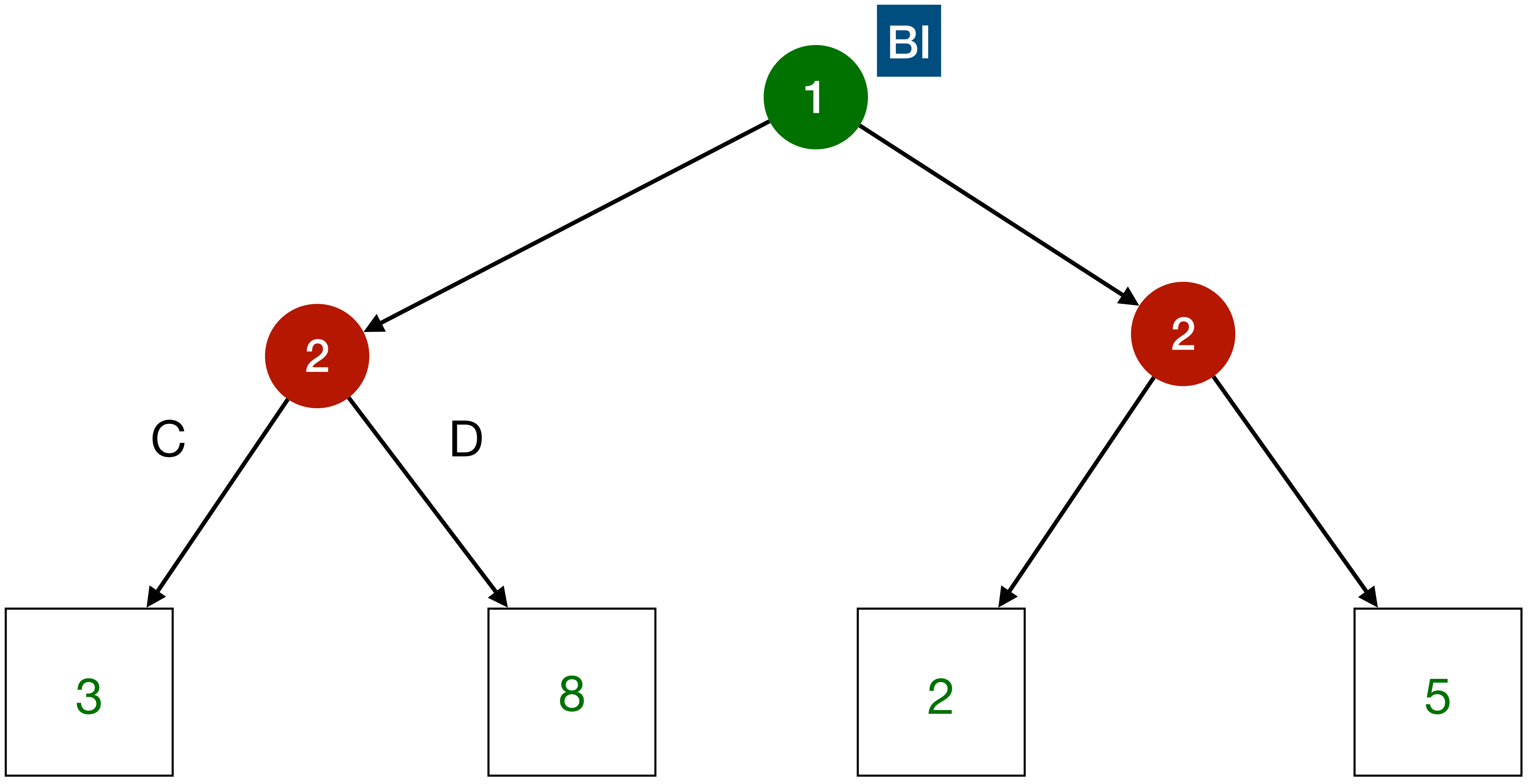
- the maximiser can guarantee score $\geq \alpha$

- the minimiser can assume score $\leq \beta$

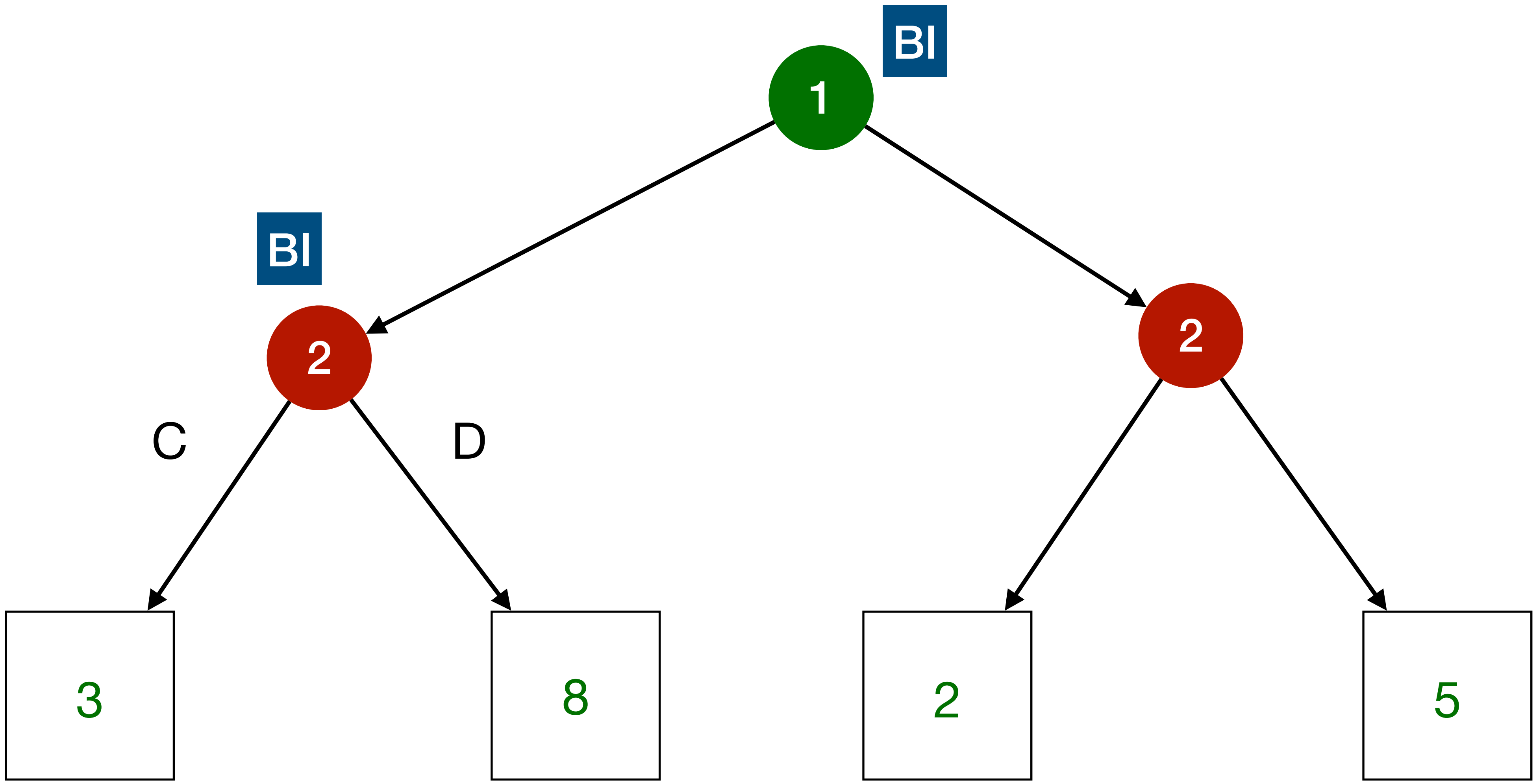
Example



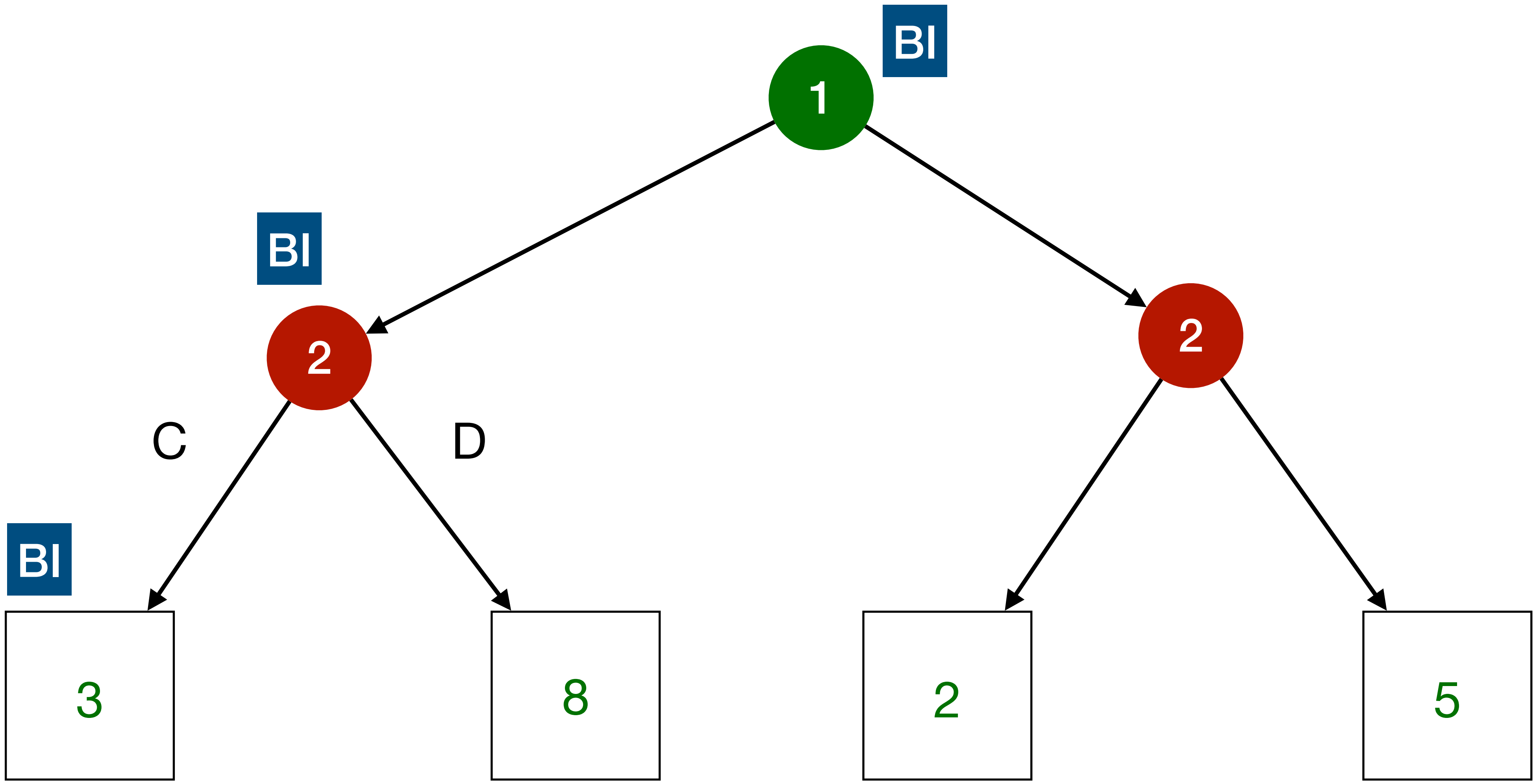
Example



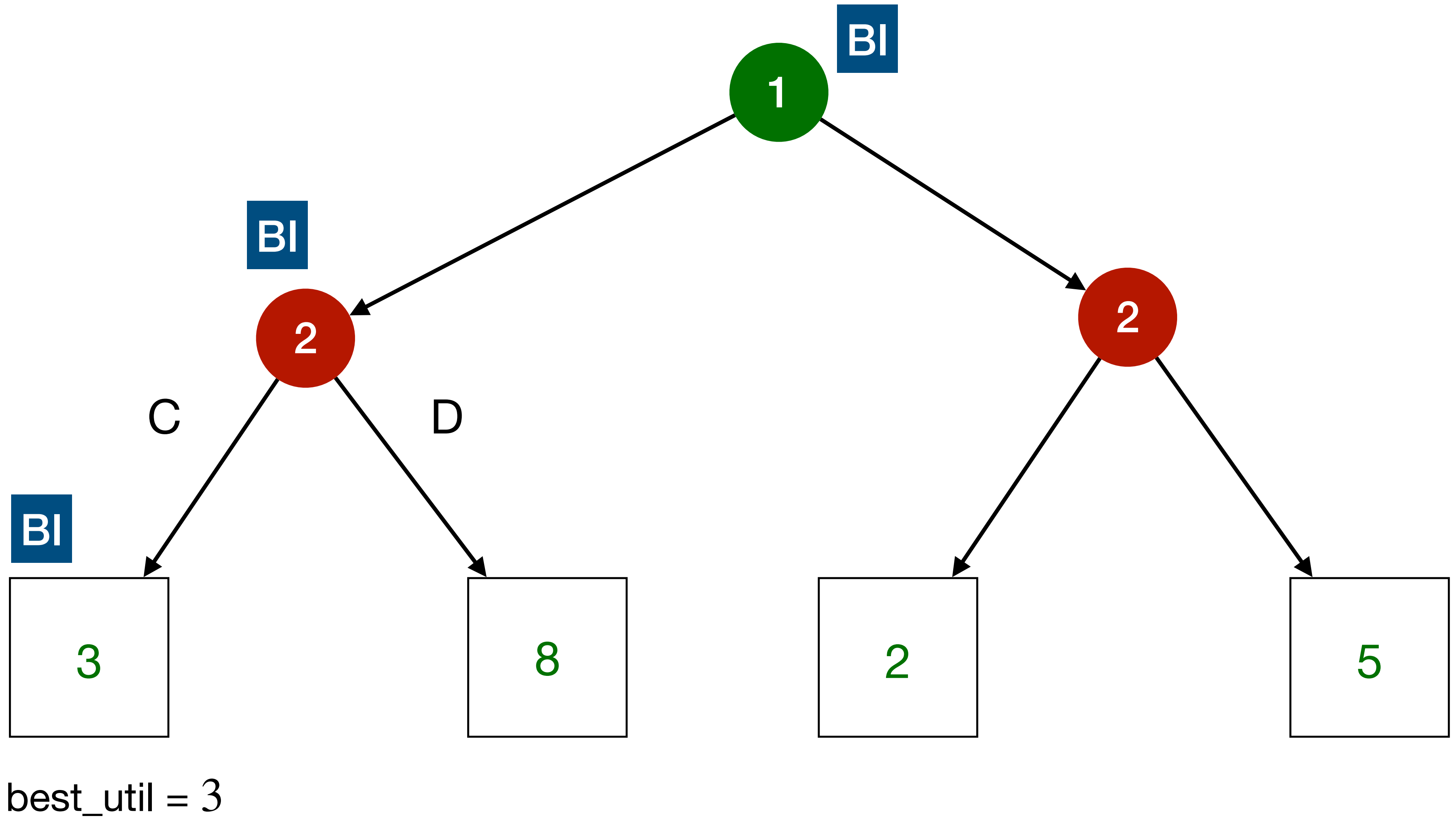
Example



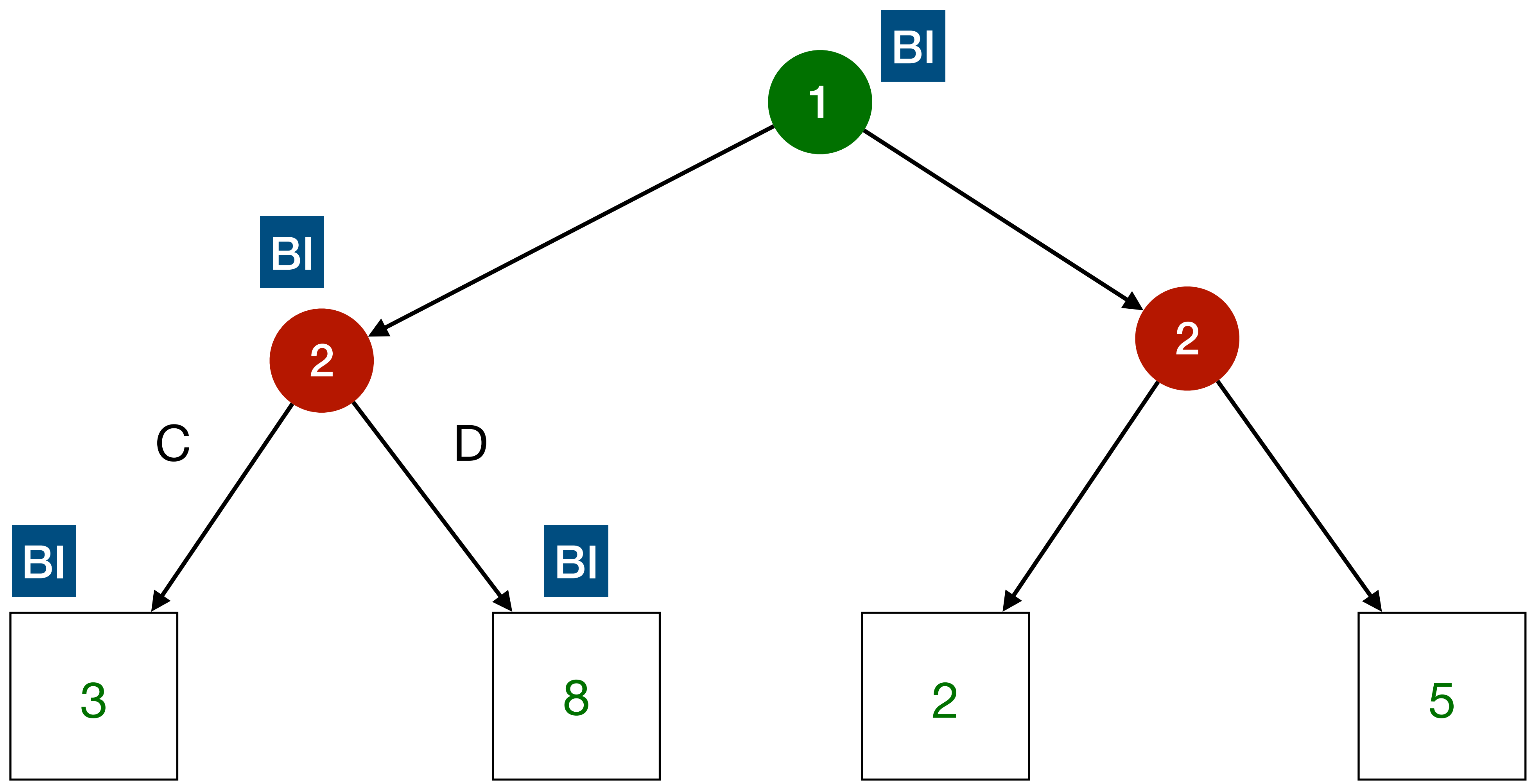
Example



Example

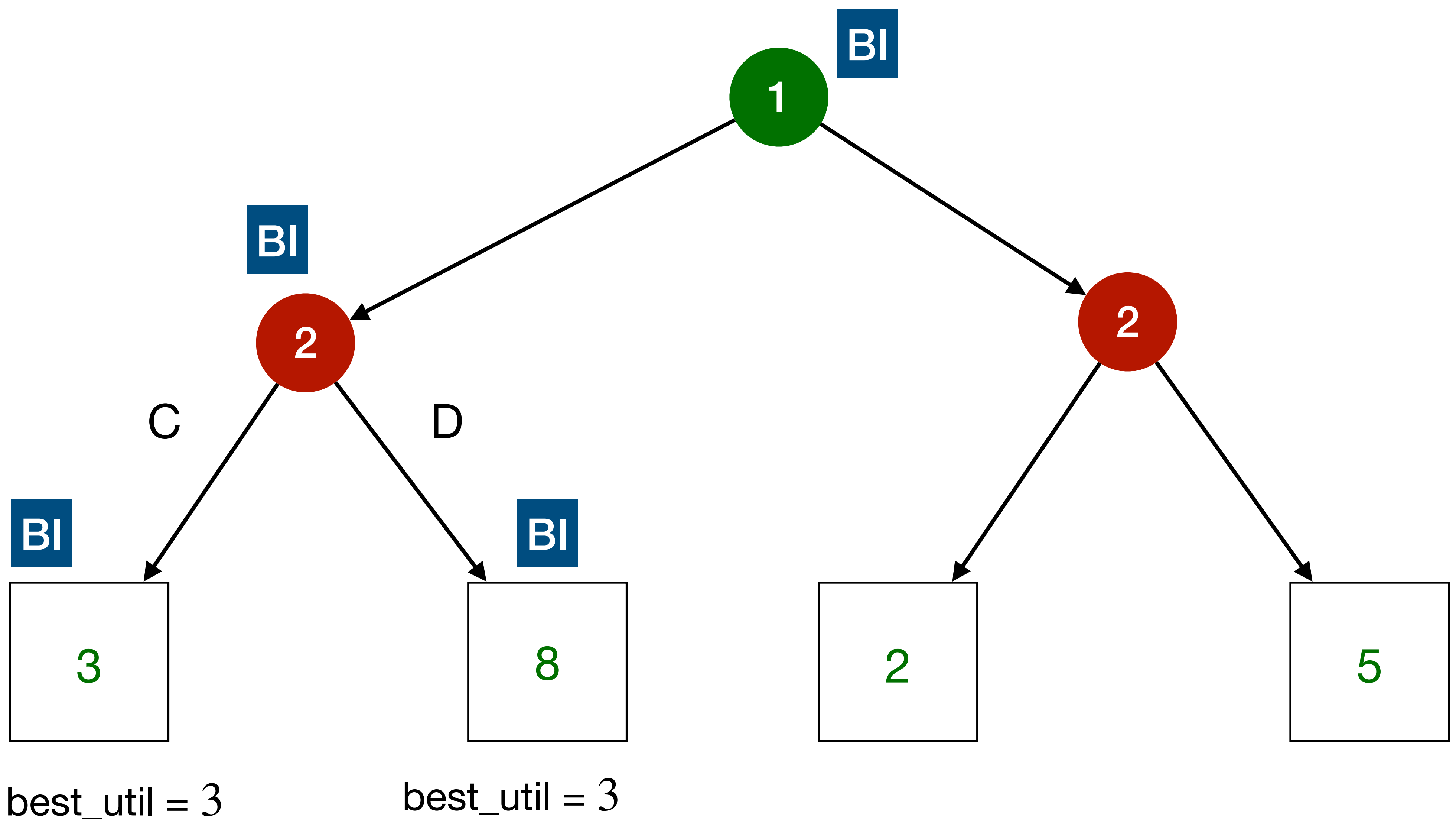


Example

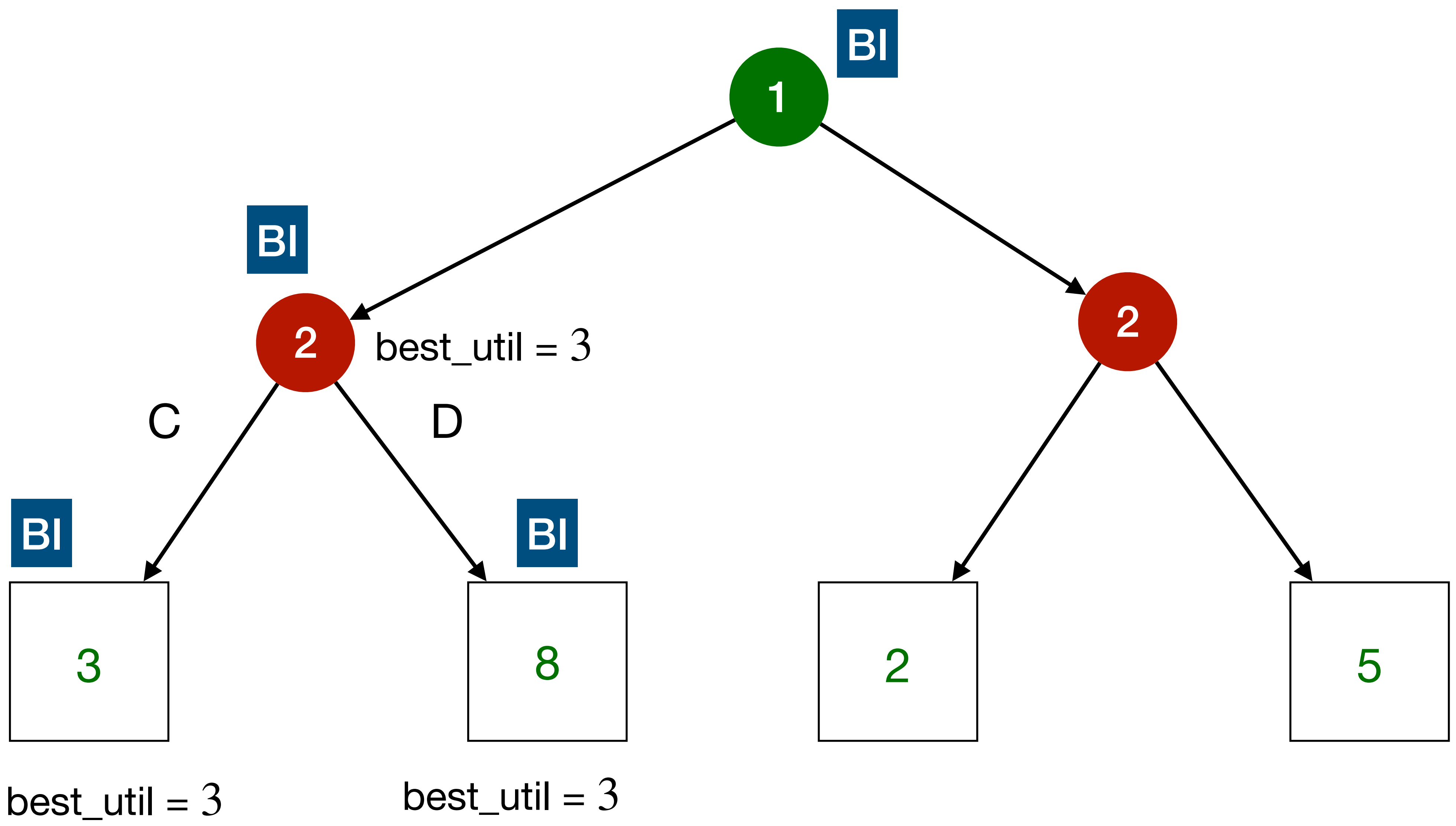


best_util = 3

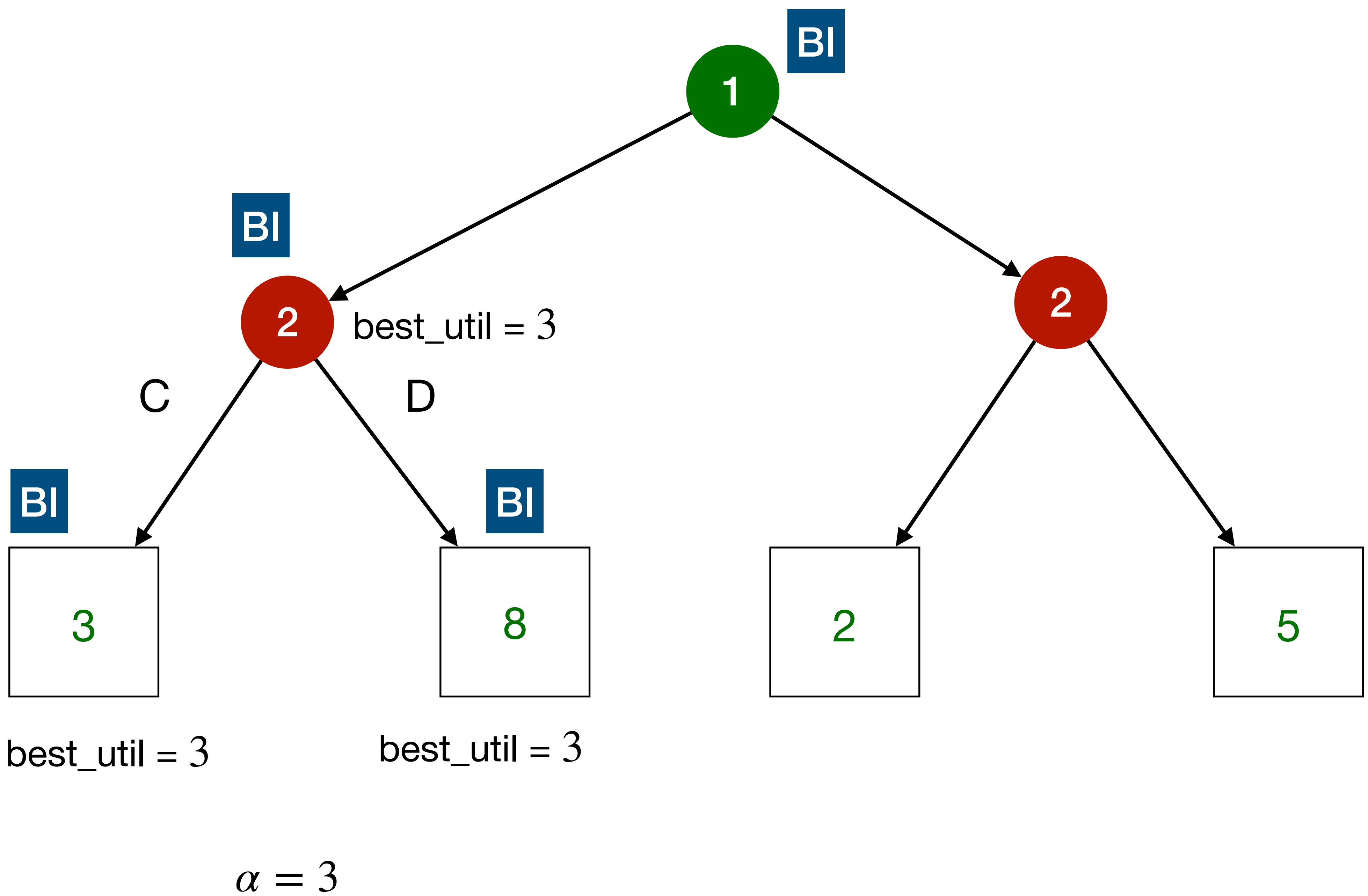
Example



Example

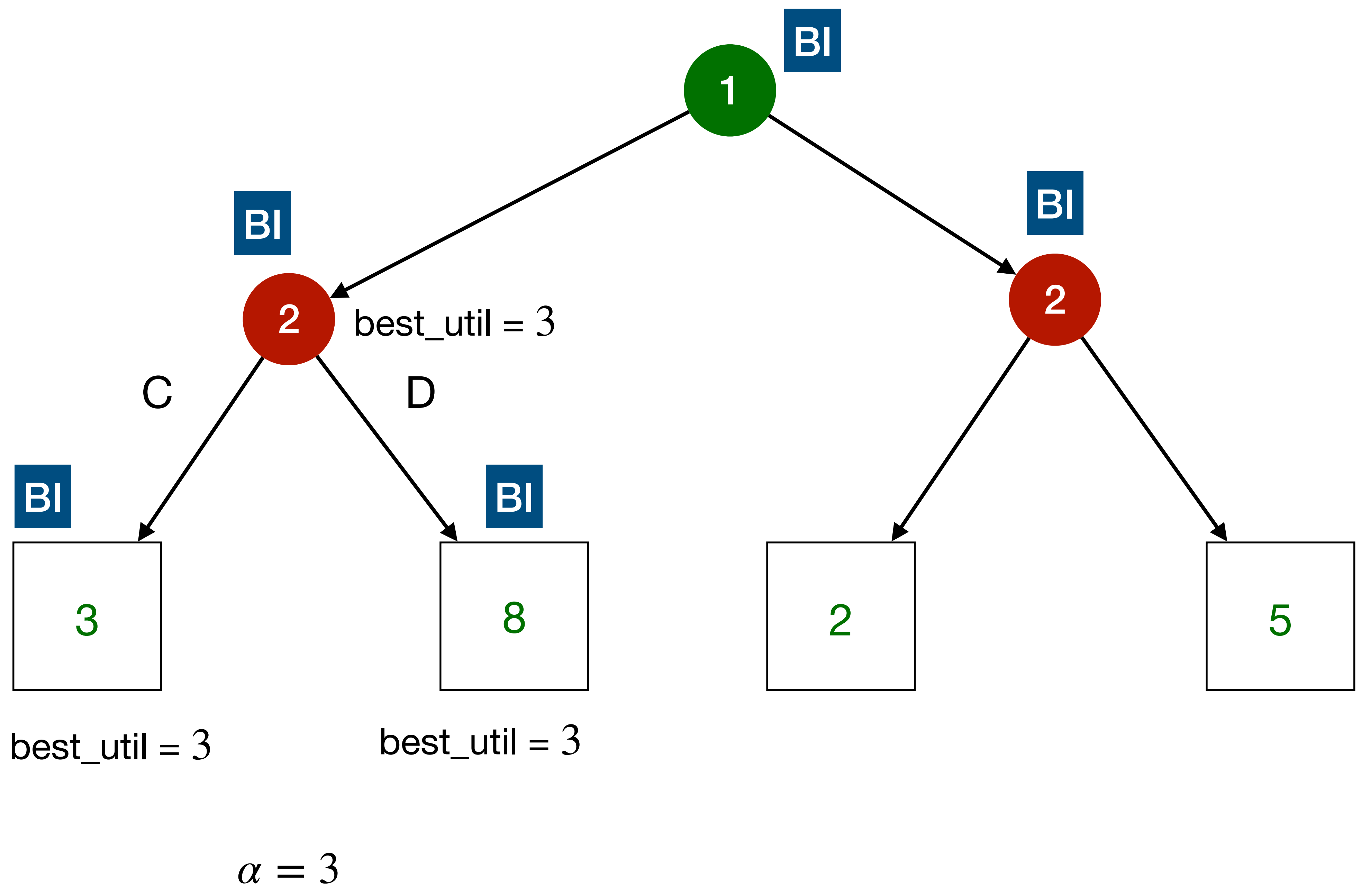


Example

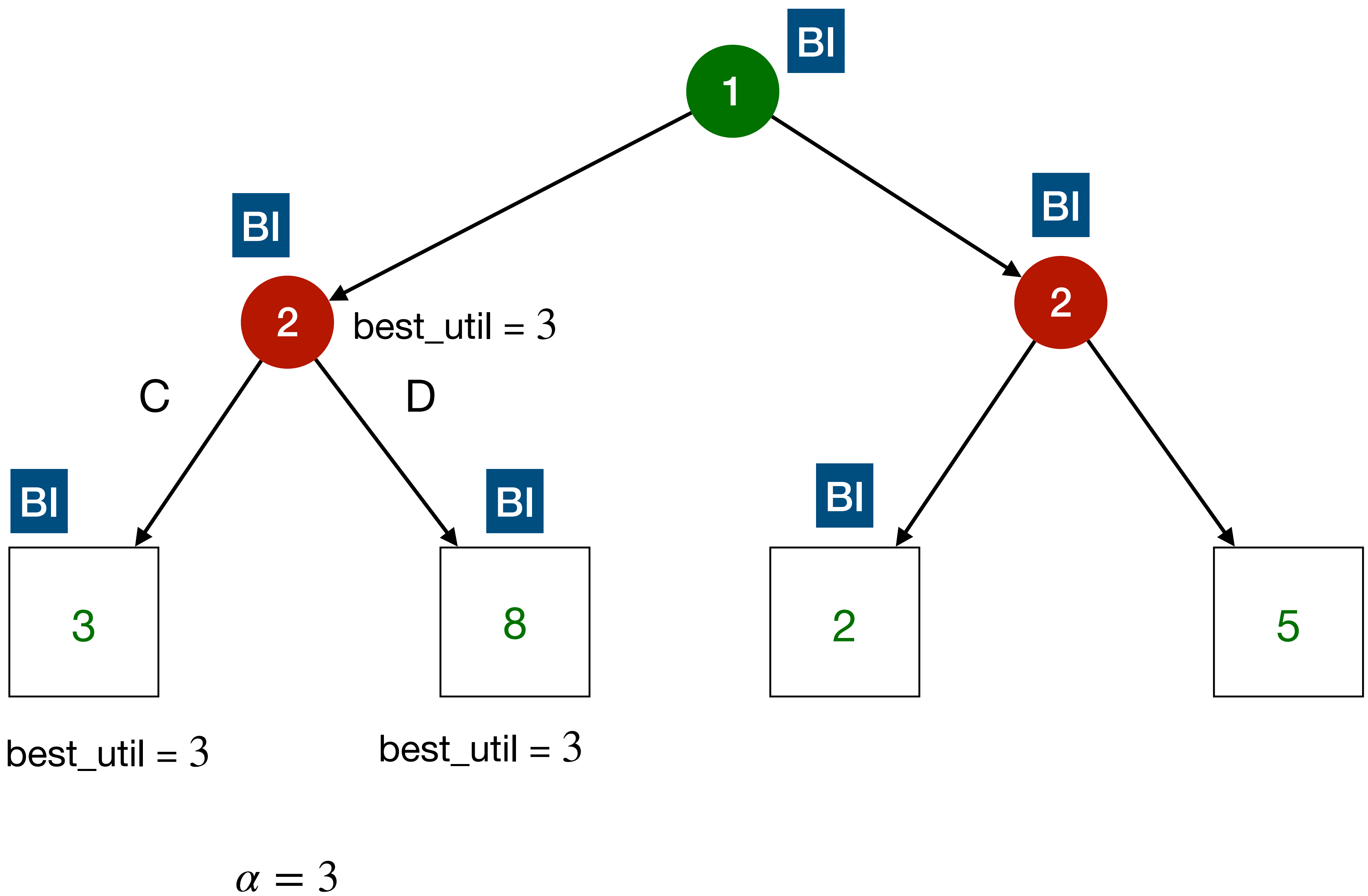


BI
1
BI
2
best_util = 3
2
C
D
BI
BI
3
8
2
5
best_util = 3
best_util = 3
$\alpha = 3$

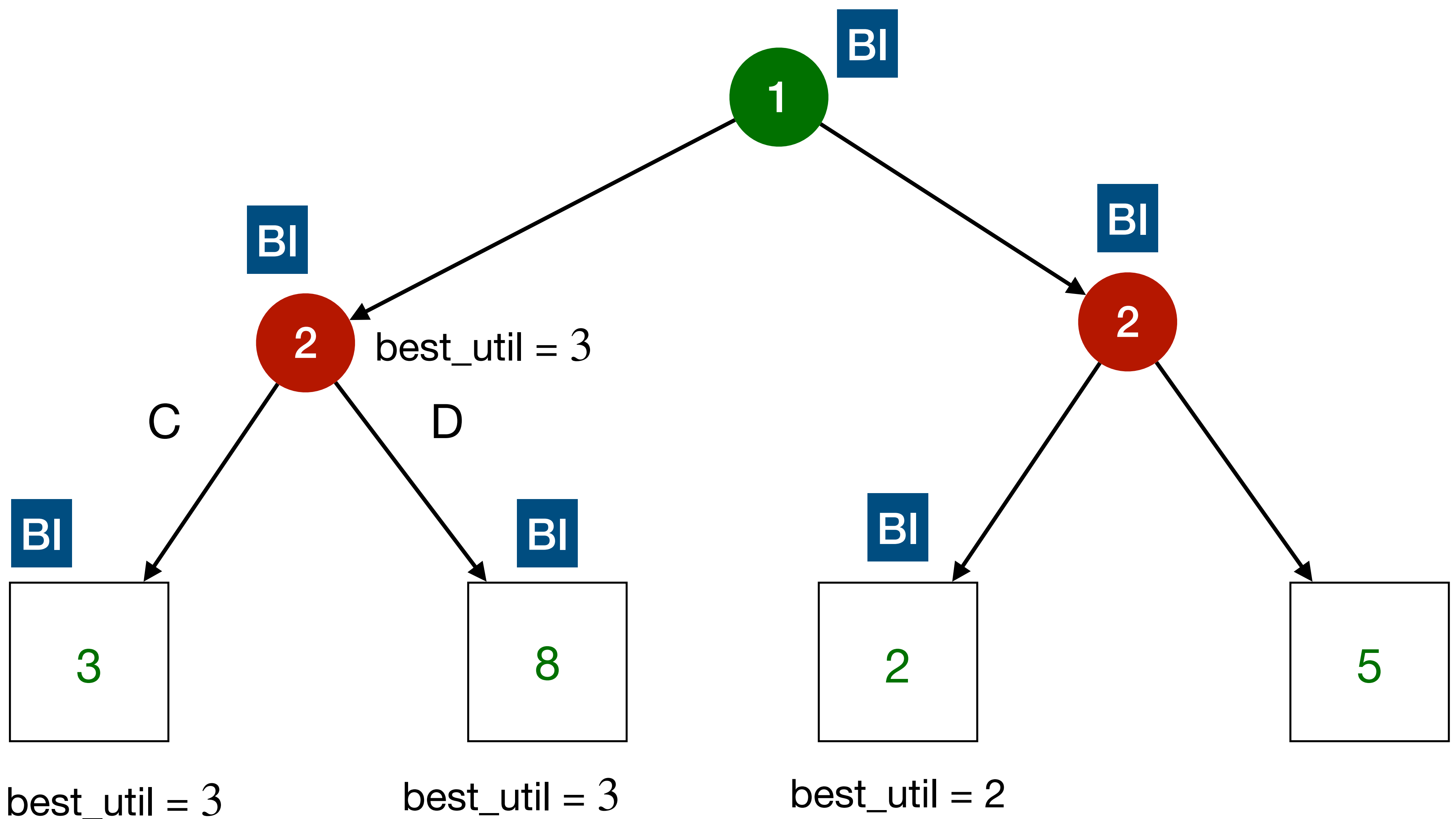
Example


BI
1
BI
2
best_util = 3
BI
2
C
D
BI
3
BI
8
2
5
best_util = 3
best_util = 3
$\alpha = 3$

Example

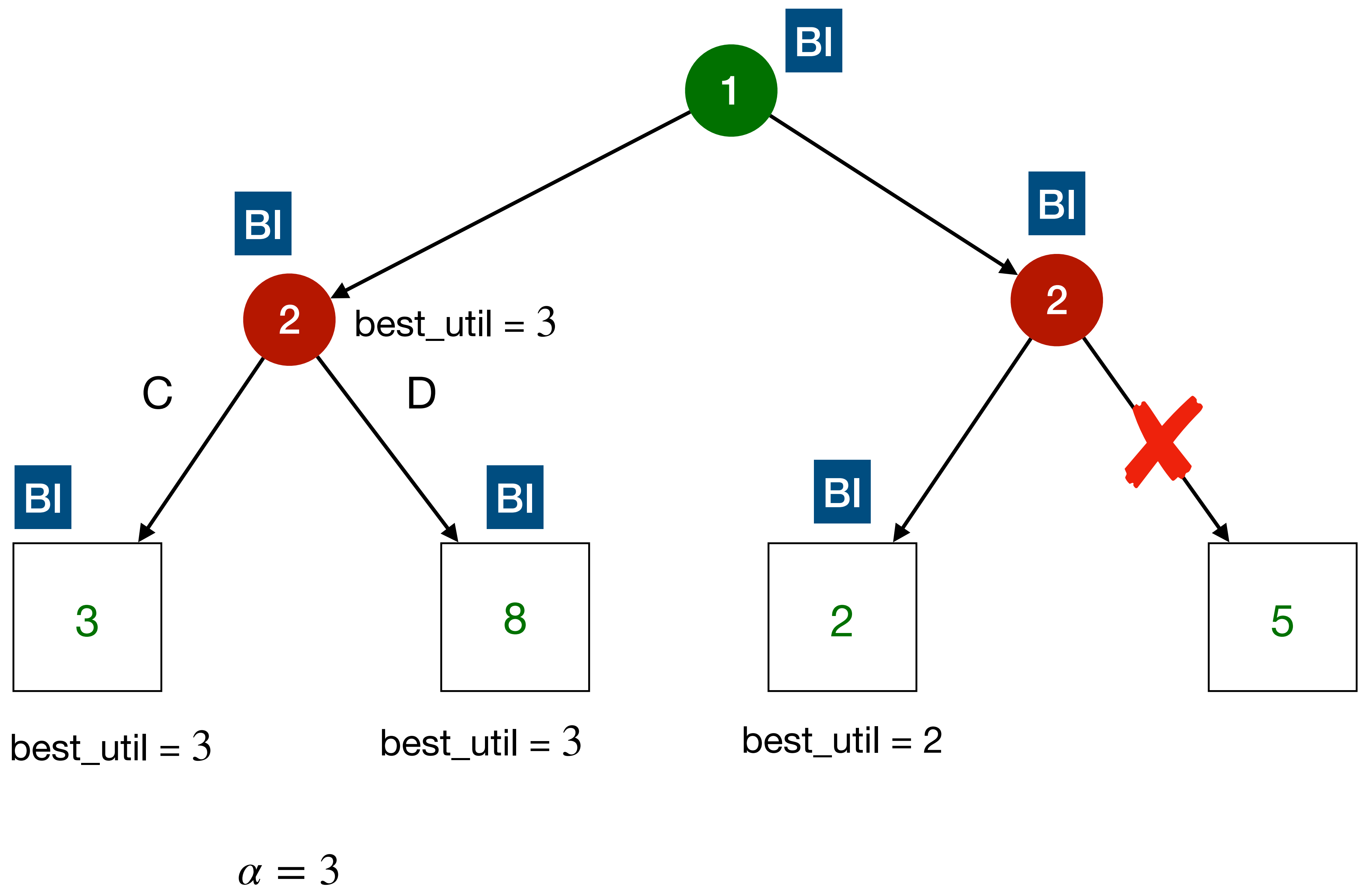

BI
1
BI
2
best_util = 3
C
D
BI
2
BI
BI
BI
3
8
2
5
best_util = 3
best_util = 3
$\alpha = 3$

Example



$\alpha = 3$

Example


BI
1
BI
2
best_util = 3
C
D
BI
2
BI
3
BI
8
BI
2
5
best_util = 3
best_util = 3
best_util = 2
$\alpha = 3$

Guess the card game

Guess the card game

Alice hides a card which is either a 5, or a 10.

Guess the card game

Alice hides a card which is either a 5, or a 10.

Bob guess which one Alice hid. If he guesses correctly, he pays Alice the number on that card divided by two in pounds, otherwise Alice pays him that amount.

Guess the card game

Alice hides a card which is either a 5, or a 10.

Bob guess which one Alice hid. If he guesses correctly, he pays Alice the number on that card divided by two in pounds, otherwise Alice pays him that amount.

We can certainly model this as a 2-player zero sum game in normal form.

Guess the card game

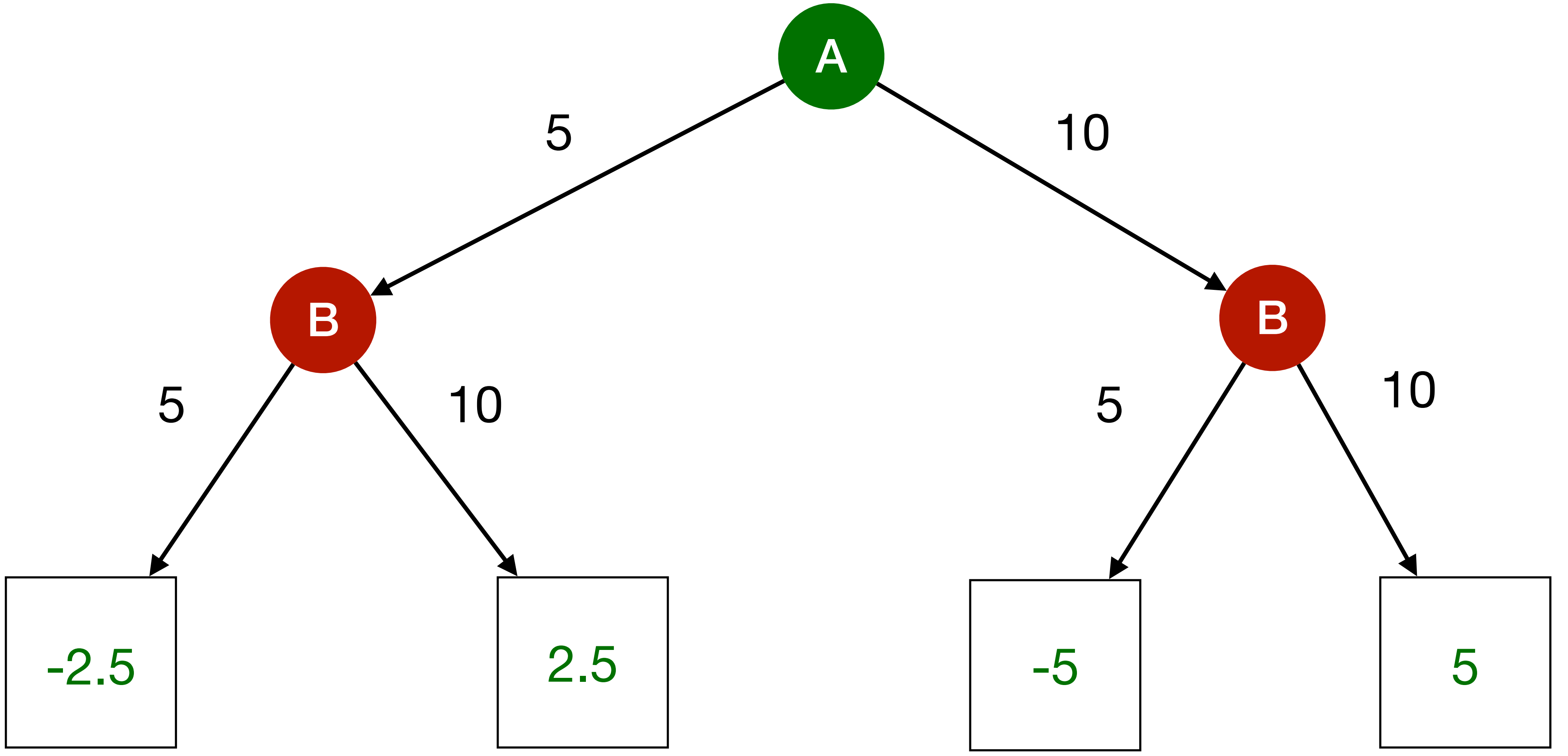
Alice hides a card which is either a 5, or a 10.

Bob guess which one Alice hid. If he guesses correctly, he pays Alice the number on that card divided by two in pounds, otherwise Alice pays him that amount.

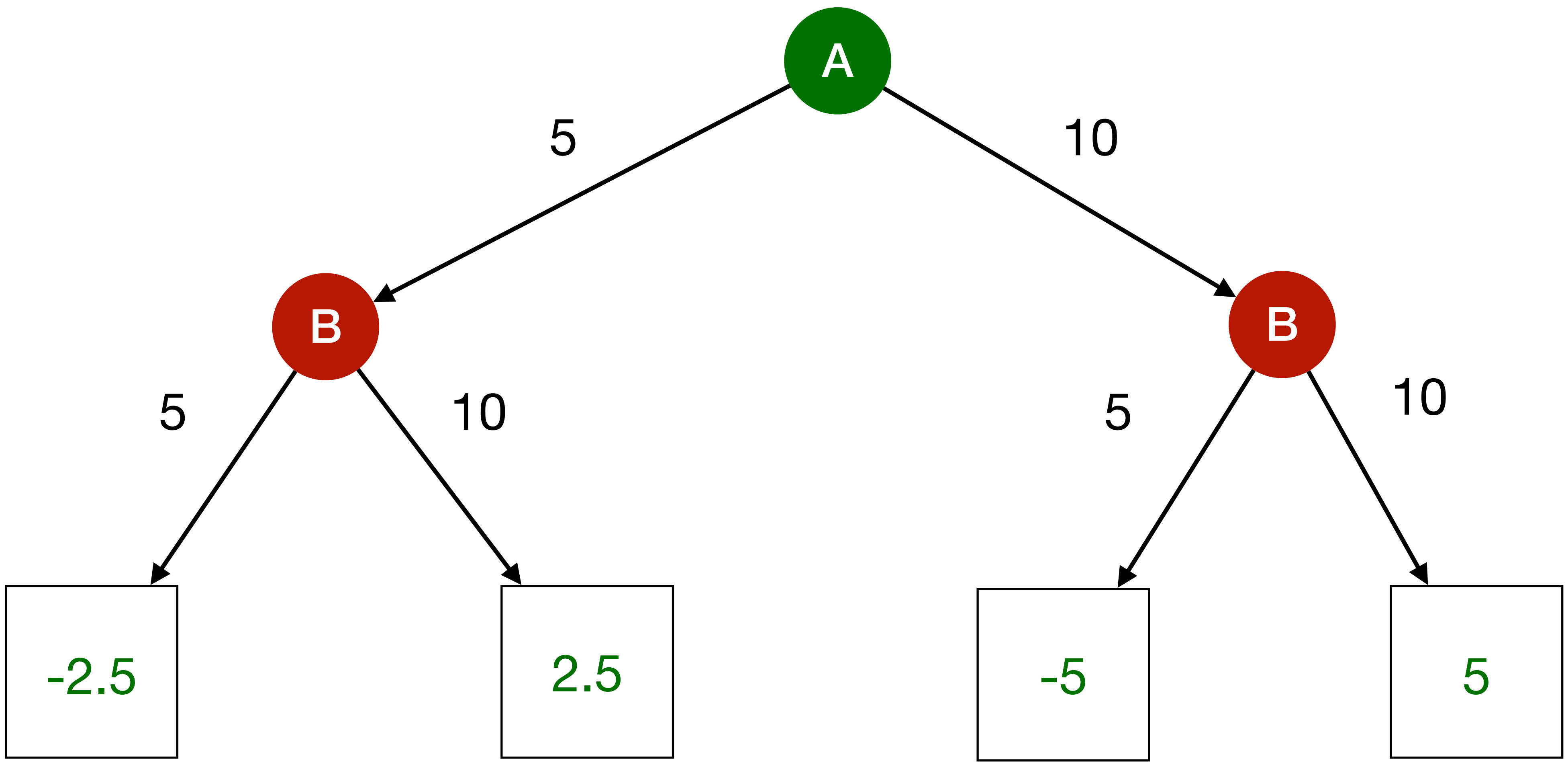
We can certainly model this as a 2-player zero sum game in normal form.

But this sounds like a sequential game: Alice hides first, and then Bob guesses. So we should be able to model it as an extensive form game.

Guess the card game

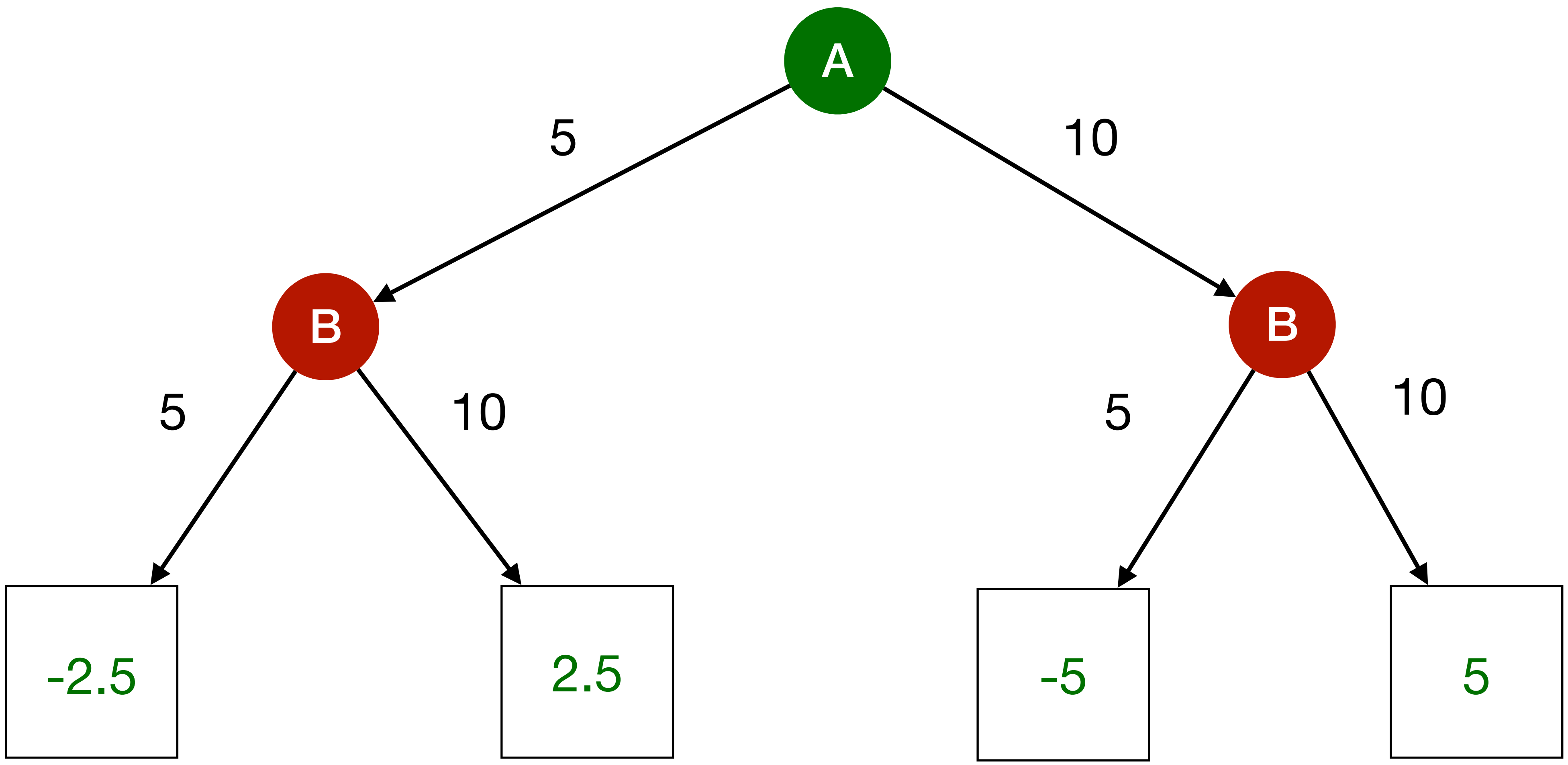


Guess the card game



Why would Bob ever guess 10 on the left, or 5 on the right?

Guess the card game



Why would Bob ever guess 10 on the left, or 5 on the right?

To accurately model the game, we have to model the fact that Bob does not know which card Alice hid!

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

1. N is a set of n players.

2. A is a set of actions.

3. H is a set of nonterminal choice nodes.

4. Z is a set of terminal nodes, $Z \cup H = \varnothing$

5. $\chi : A \to 2^A$ is the action function, mapping to each choice node a set of possible actions.

6. $\rho : H \to N$ is the player function, which determines which player takes an action at each choice node.

7. $\sigma : H \times A \to H \cup Z$ is the successor function, which maps a choice node and an action to another node (choice or terminal).

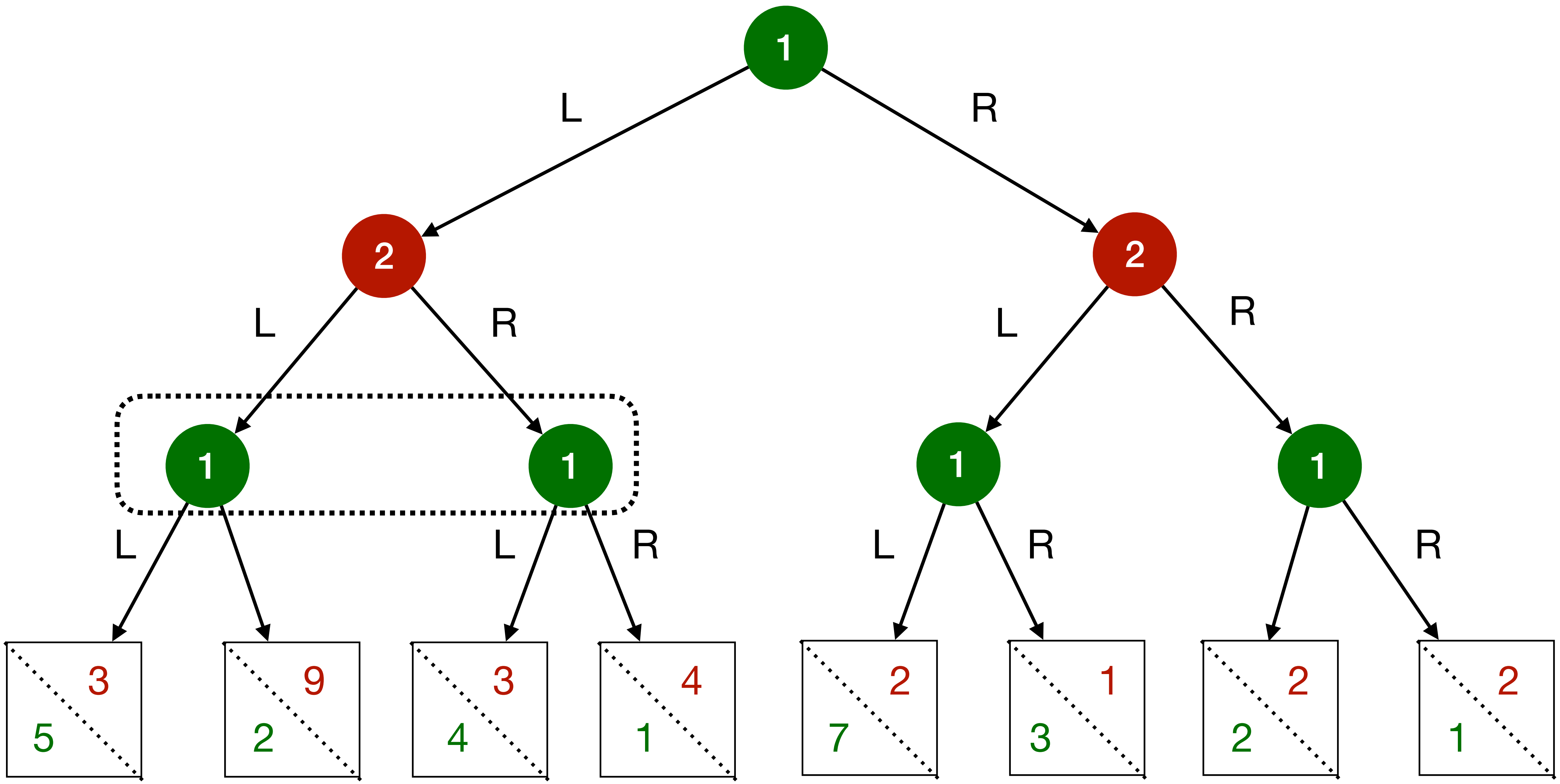
8. $u_i : Z \to \mathbb{R}$ is a utility function for each player i, mapping terminal nodes to real-valued utilities.

Extensive Form Games

Definition: A (perfect information) extensive form game in extensive form is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where

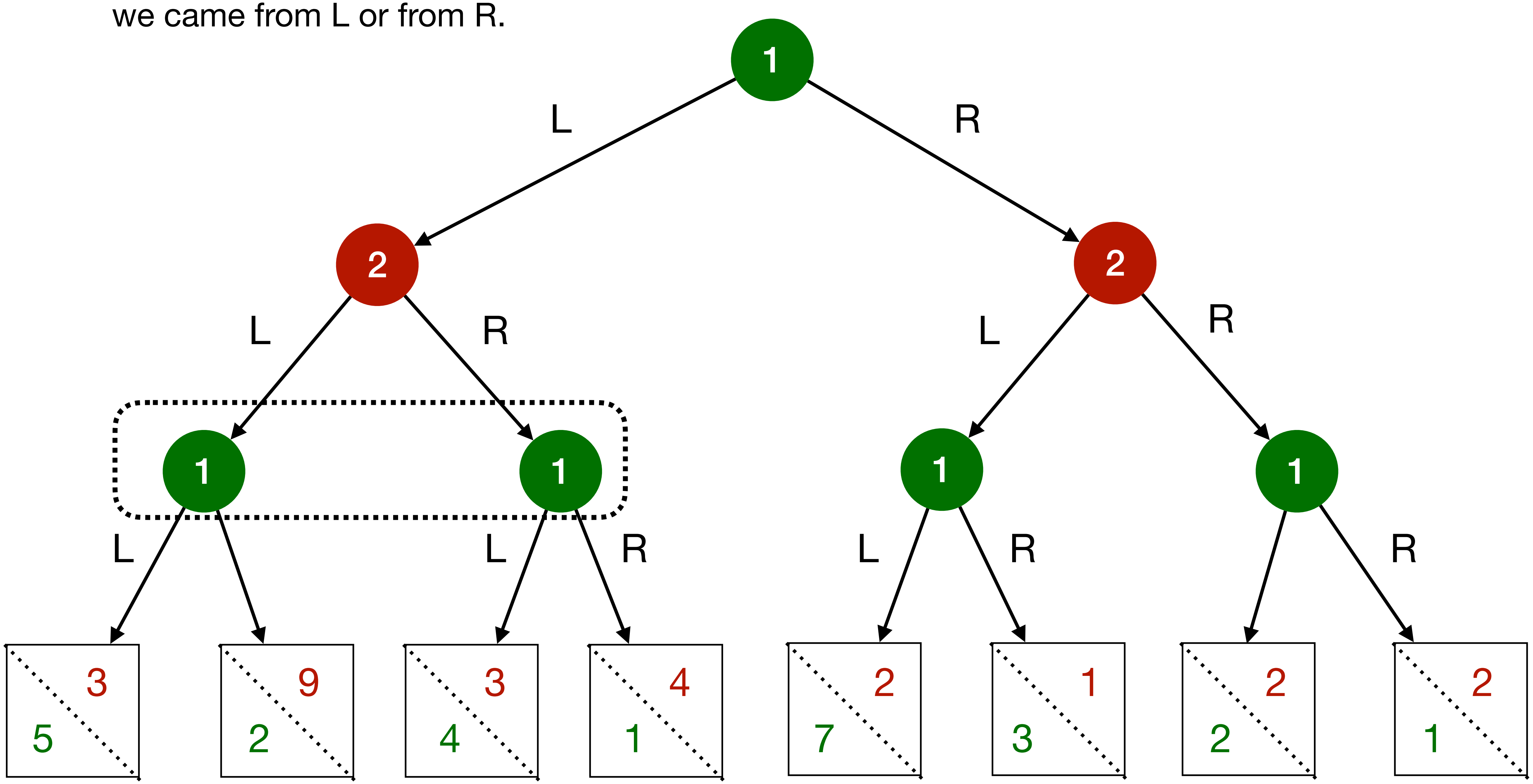
1. N is a set of n players.
2. A is a set of actions.
3. H is a set of nonterminal choice nodes.
4. Z is a set of terminal nodes, $Z \cup H = \emptyset$
5. $\chi : A \to 2^A$ is the action function, mapping to each choice node a set of possible actions.
6. $\rho : H \to N$ is the player function, which determines which player takes an action at each choice node.
7. $\sigma : H \times A \to H \cup Z$ is the successor function, which maps a choice node and an action to another node (choice or terminal).
8. $u_i : Z \to \mathbb{R}$ is a utility function for each player i, mapping terminal nodes to real-valued utilities.

Information Sets



Information Sets

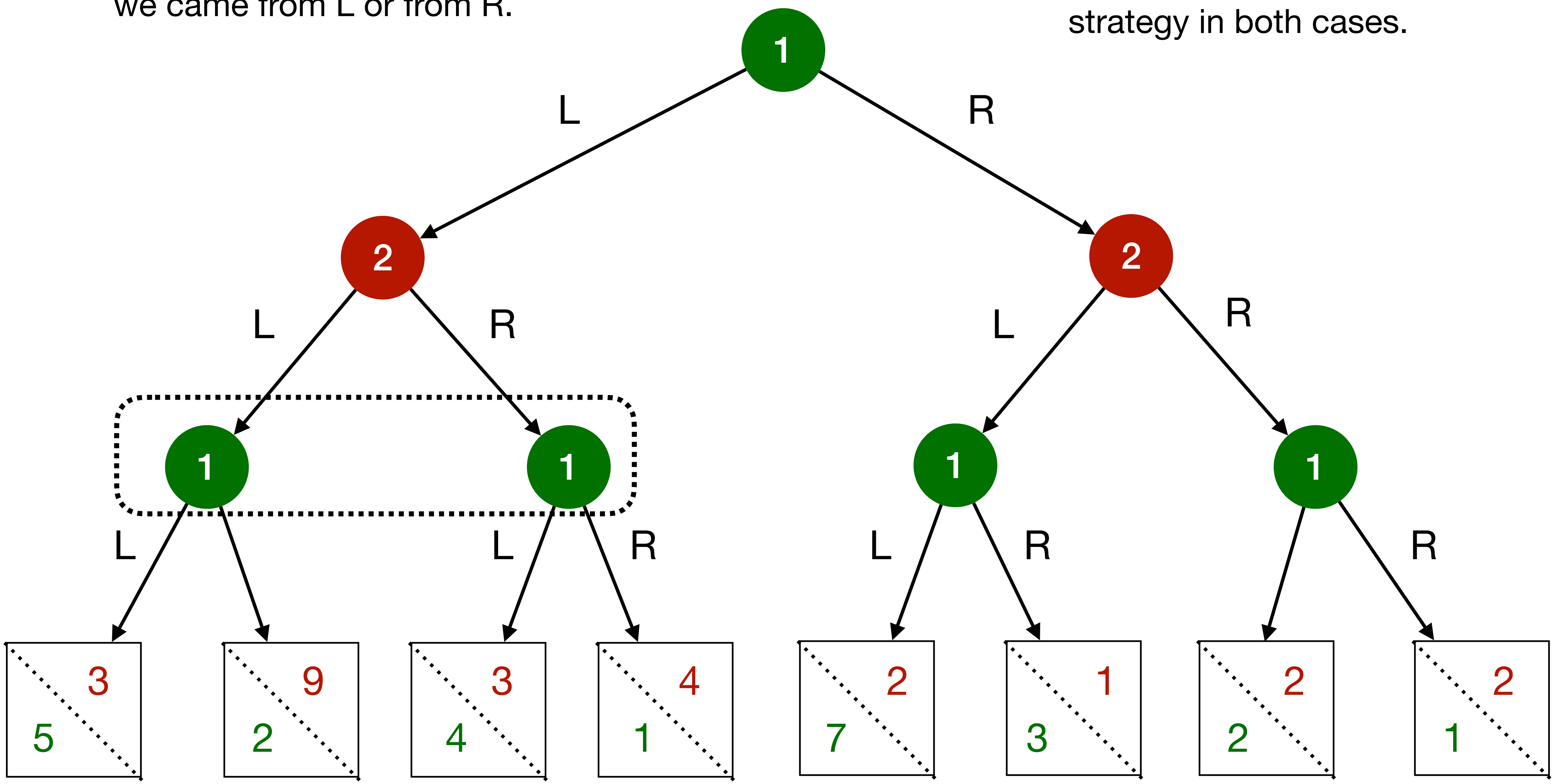
Idea: Player 1 does not know whether
we came from L or from R.



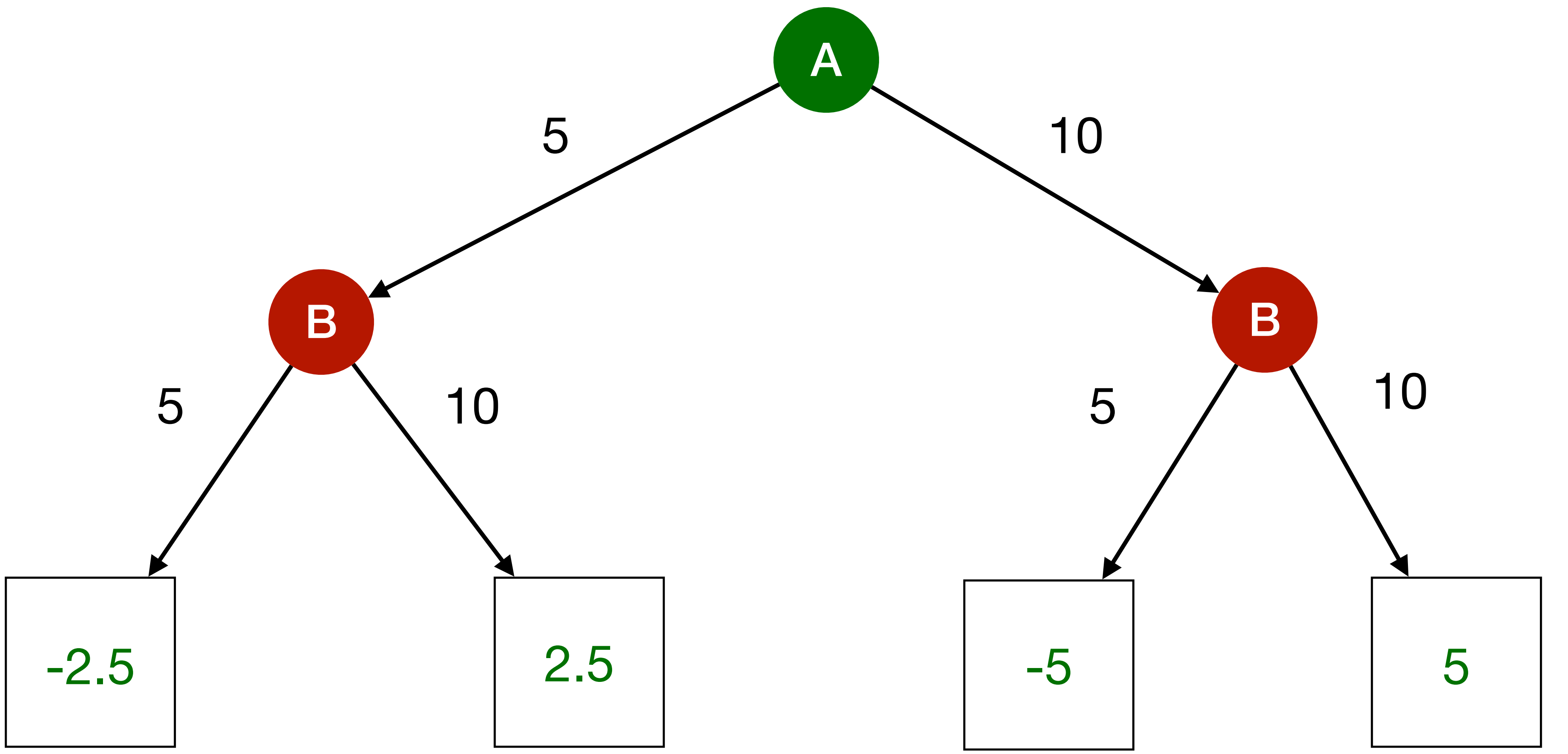
Information Sets

Idea: Player 1 does not know whether we came from L or from R.

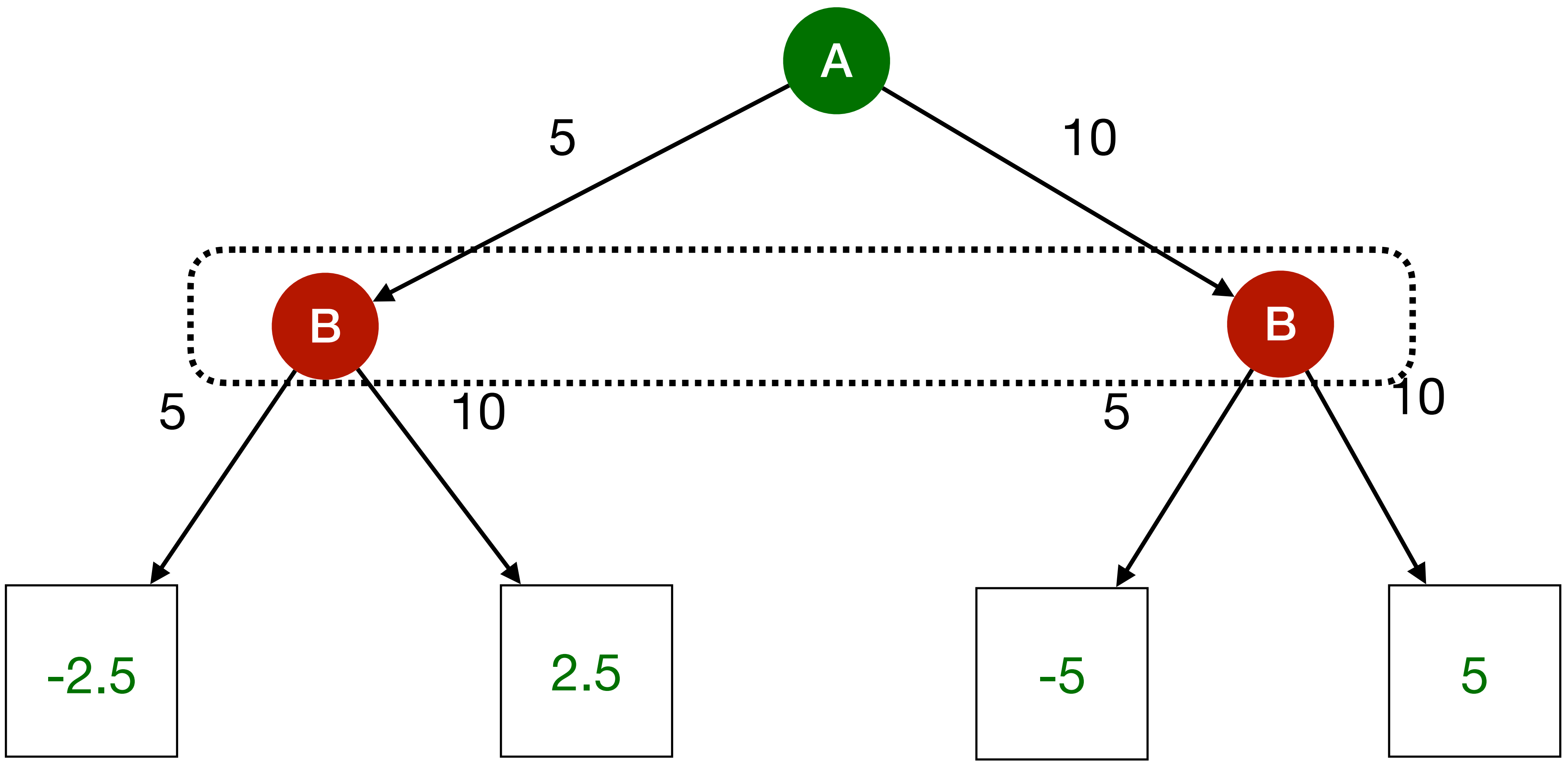
The player must choose the same strategy in both cases.



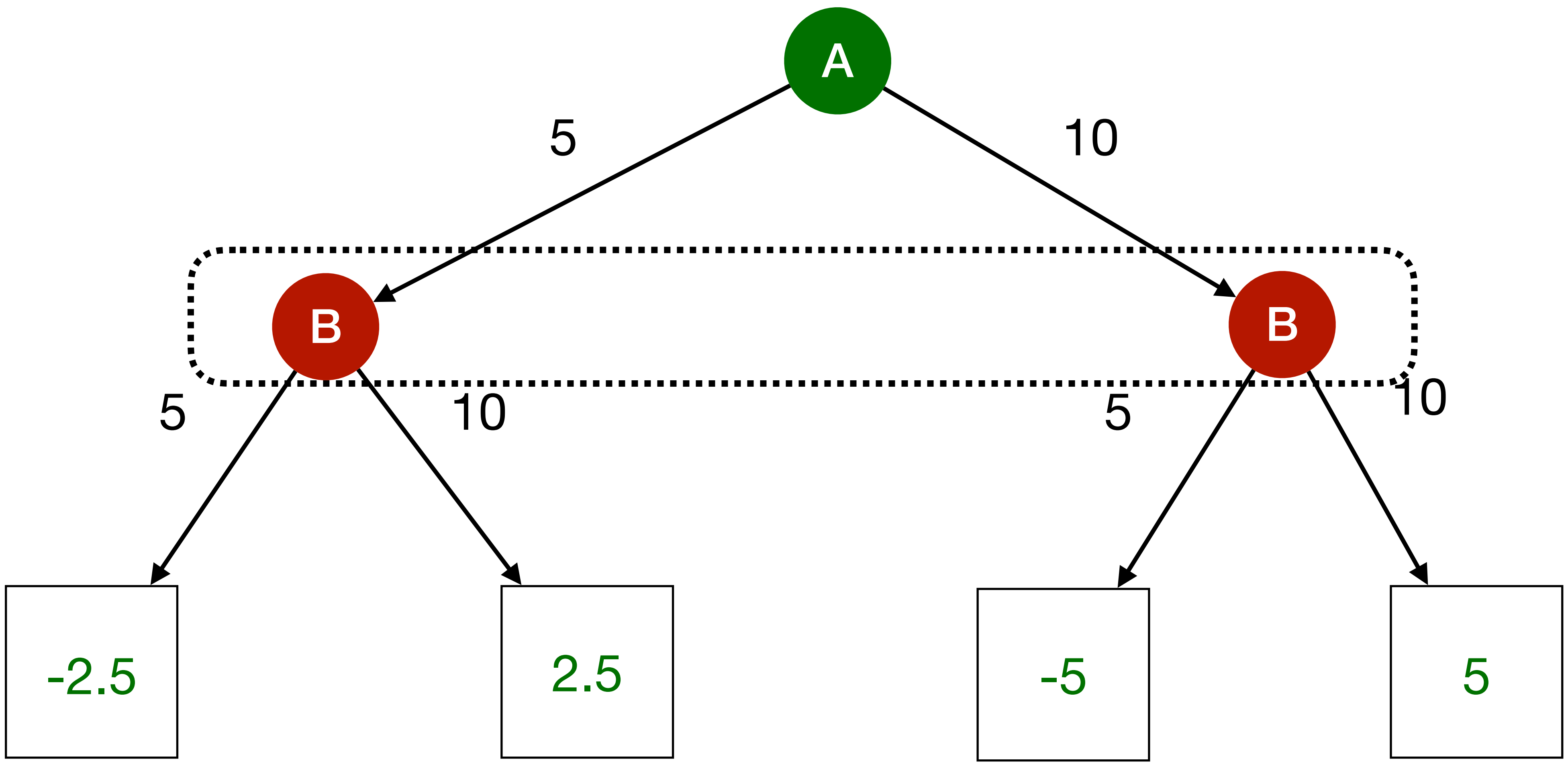
Guess the card game



Guess the card game

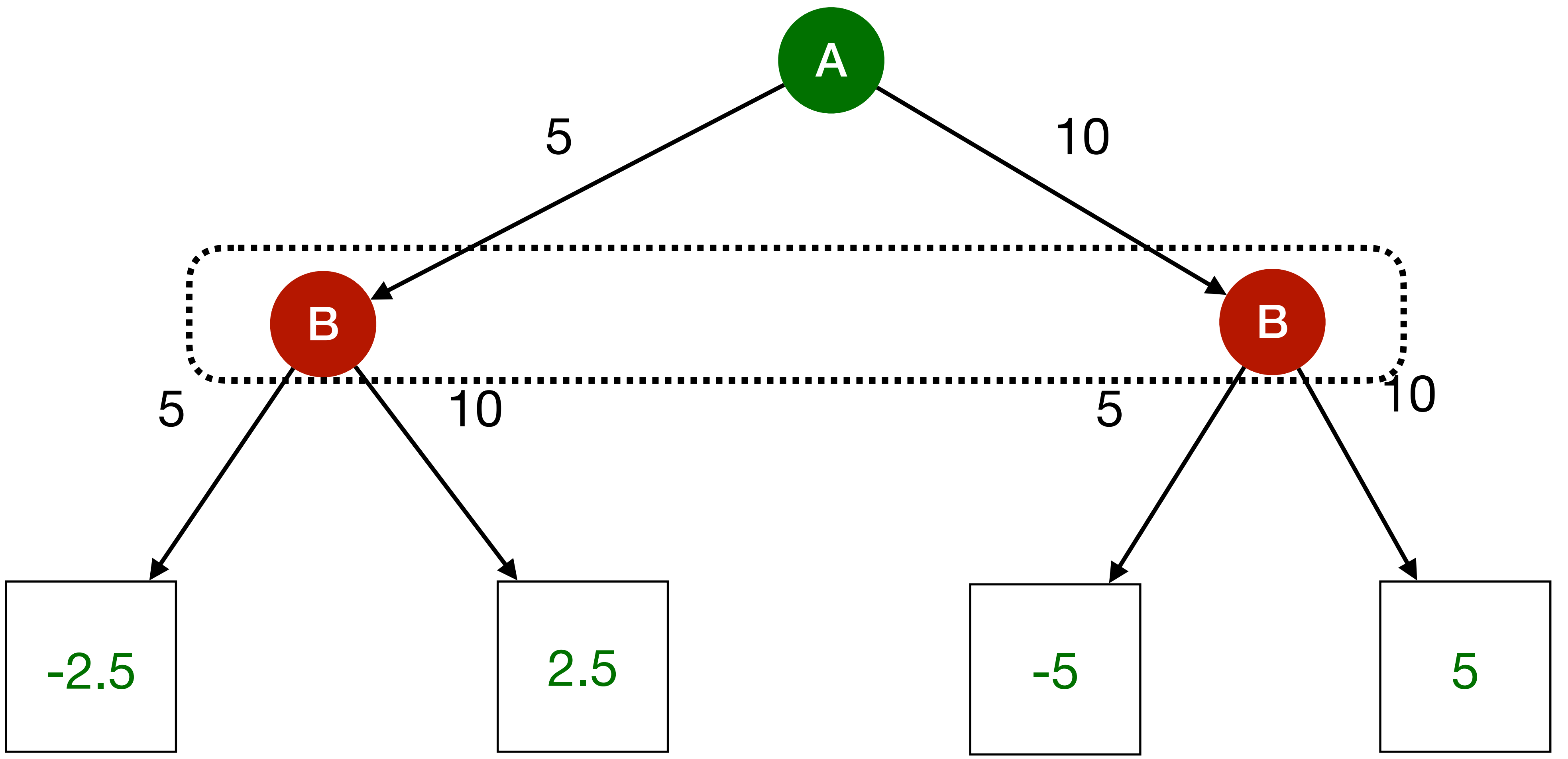


Guess the card game



Bob now has to guess either 5 or 10 in both cases.

Guess the card game



Bob now has to guess either 5 or 10 in both cases.

Or he can choose a mixed strategy!

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

This is not true for *imperfect information* extensive form games.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

This is not true for *imperfect information* extensive form games.

These games however always have mixed Nash equilibria.

Equilibrium in Extensive Form Games

Theorem (Zermelo 1913): Every perfect information extensive form game always has a pure Nash equilibrium.

This is not true for *imperfect information* extensive form games.

These games however always have mixed Nash equilibria.

Why is that?

From normal form to extensive form

From normal form to extensive form

Every imperfect information extensive form game can be transformed to an equivalent normal form game.

From normal form to extensive form

Every imperfect information extensive form game can be transformed to an equivalent normal form game.

Every normal form game can also be transformed to an imperfect information extensive form game.

From normal form to extensive form

Every imperfect information extensive form game can be transformed to an equivalent normal form game.

Every normal form game can also be transformed to an imperfect information extensive form game.

We can compute mixed Nash equilibria of imperfect information extensive form games using this connection, but

From normal form to extensive form

Every imperfect information extensive form game can be transformed to an equivalent normal form game.

Every normal form game can also be transformed to an imperfect information extensive form game.

We can compute mixed Nash equilibria of imperfect information extensive form games using this connection, but

- the transformation might take exponential time

From normal form to extensive form

Every imperfect information extensive form game can be transformed to an equivalent normal form game.

Every normal form game can also be transformed to an imperfect information extensive form game.

We can compute mixed Nash equilibria of imperfect information extensive form games using this connection, but

- the transformation might take exponential time

- we don't know how to find the mixed equilibria for the normal form game efficiently.

From normal form to extensive form

Every imperfect information extensive form game can be transformed to an equivalent normal form game.

Every normal form game can also be transformed to an imperfect information extensive form game.

We can compute mixed Nash equilibria of imperfect information extensive form games using this connection, but

- the transformation might take exponential time

- we don't know how to find the mixed equilibria for the normal form game efficiently.

We will talk more about mixed strategies in II-EFGs next time.