

CS3243 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

ADVERSARIAL SEARCH

Adapted from Nicholas Teh

CS3243 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

CONTENT SUMMARY

KEY CONCEPTS

▶ **Adversarial Search**

- ▶ Tracing tree with Alpha-Beta pruning algorithm
 - ▶ Detect nodes that are pruned/not evaluated
 - ▶ Knowing values/range of values for which the nodes will/will not be pruned
- ▶ The Minimax algorithm

CS3243 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

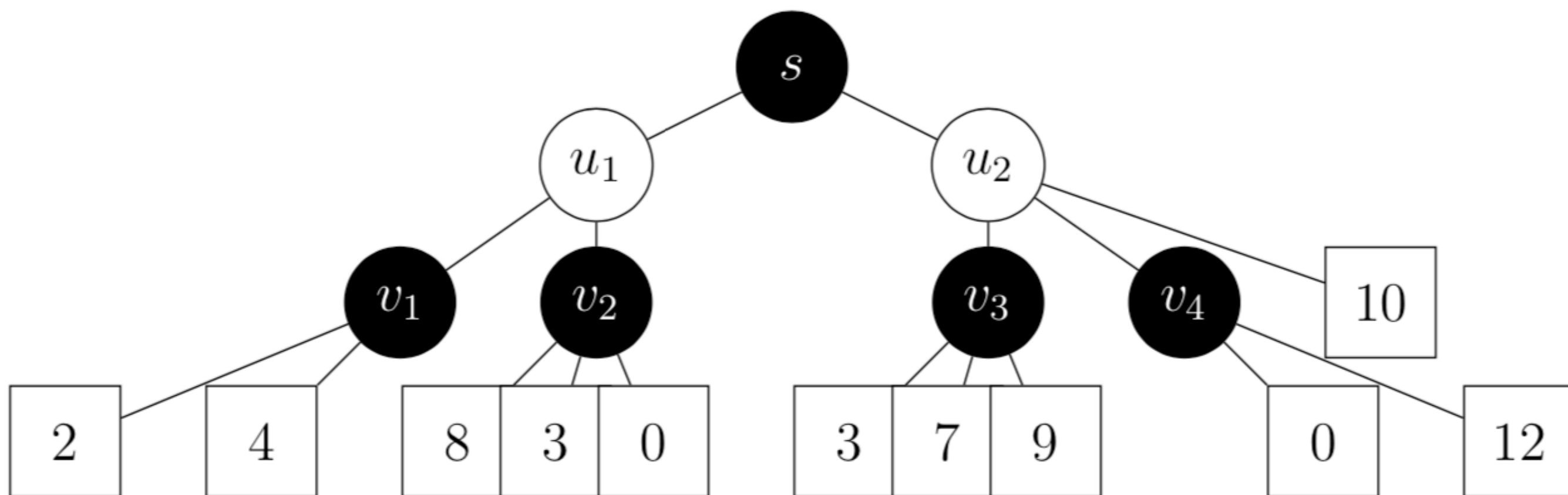
ADVERSARIAL SEARCH

MASTERING ADVERSARIAL SEARCH GRAPHS

- ▶ α : worst case (lower bound) for MAX $\alpha \geq \#$
 β : worst case (upper bound) for MIN $\beta \leq \#$
- ▶ Have α for every MAX node, start with $-\infty$
Have β for every MIN node, start with ∞
- ▶ When propagating upwards (or deep-compare),
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining
If prune, value don't copy upwards
- ▶ Compare all the way up for conflict if deep tree

TUTORIAL 5 QUESTION 1

- ▶ Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



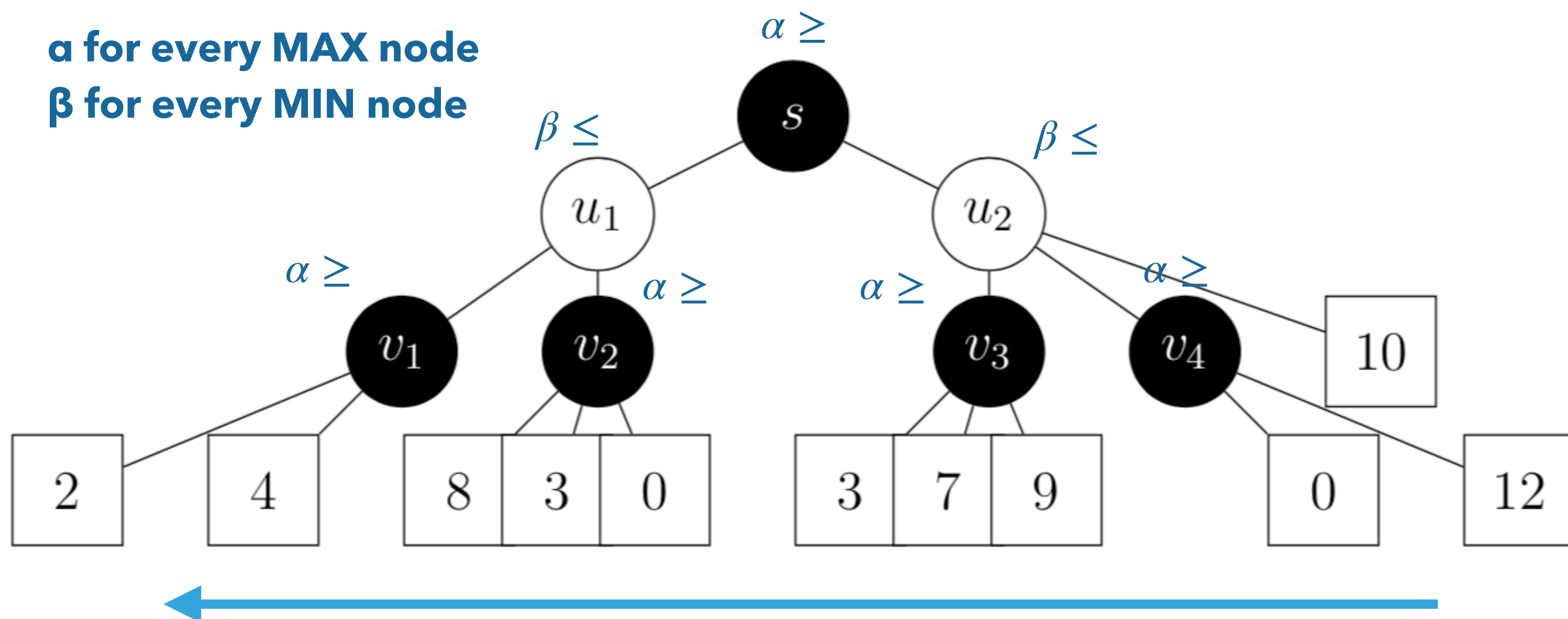
TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

α for every MAX node
 β for every MIN node

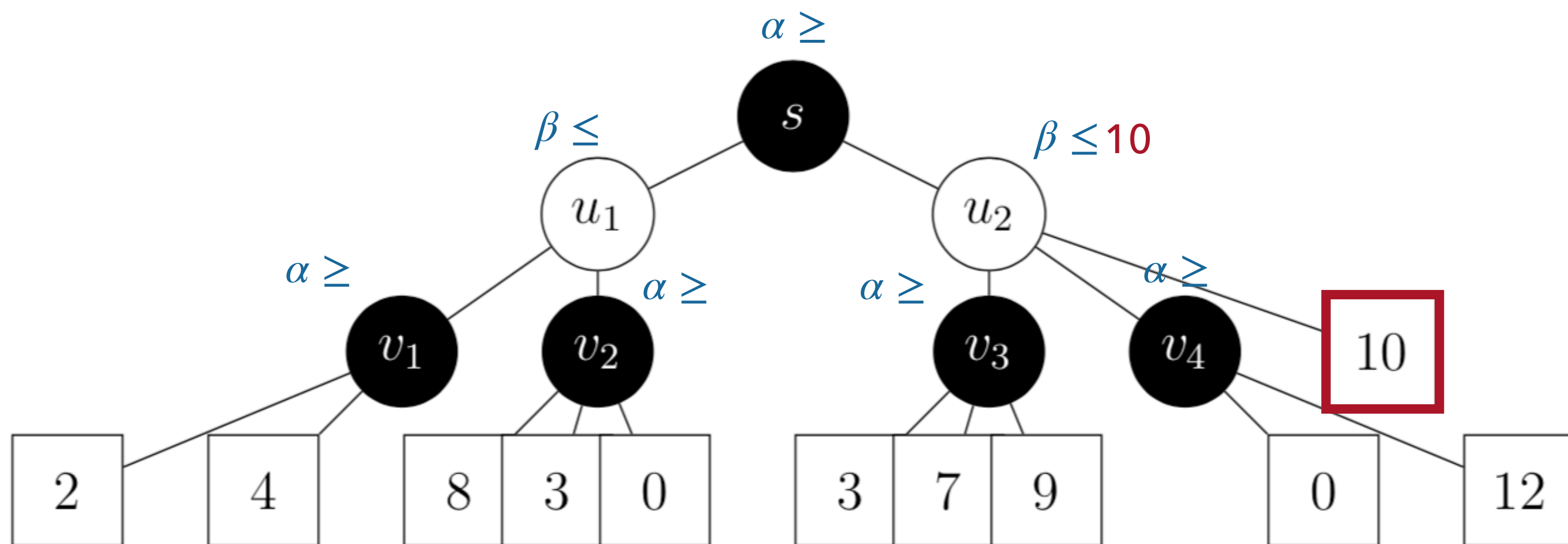


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

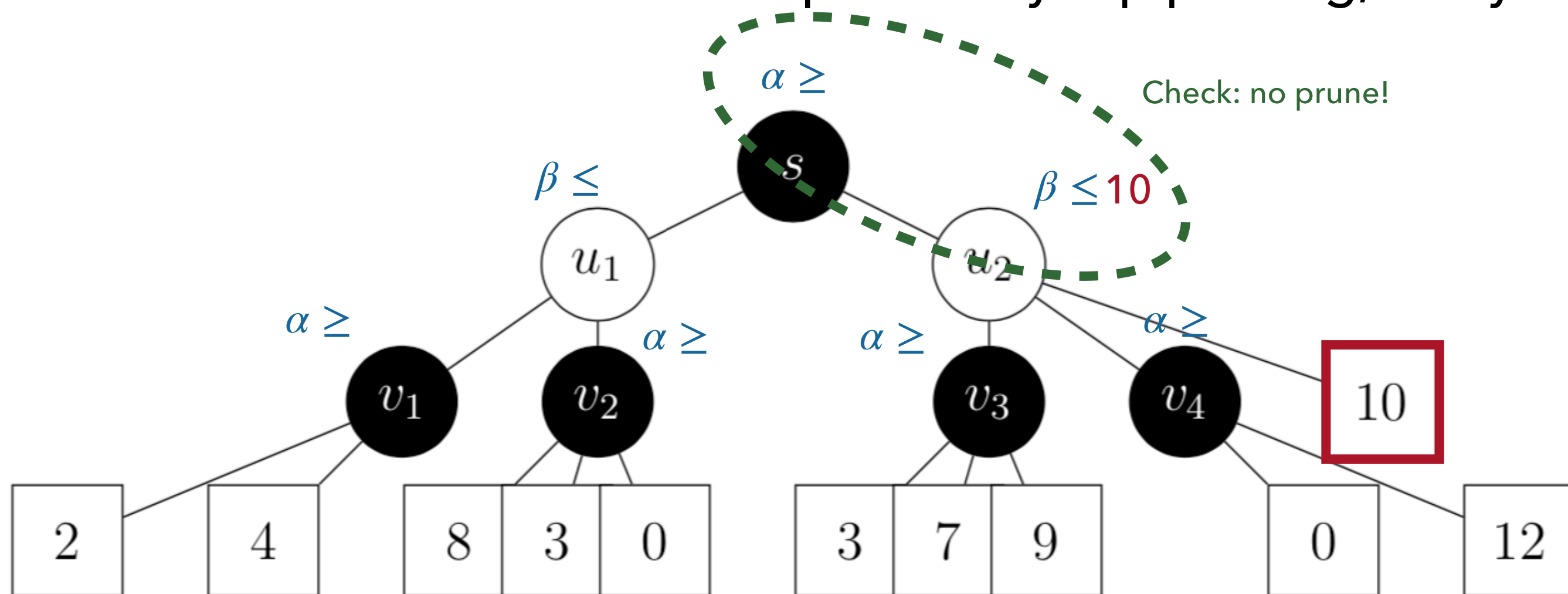


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

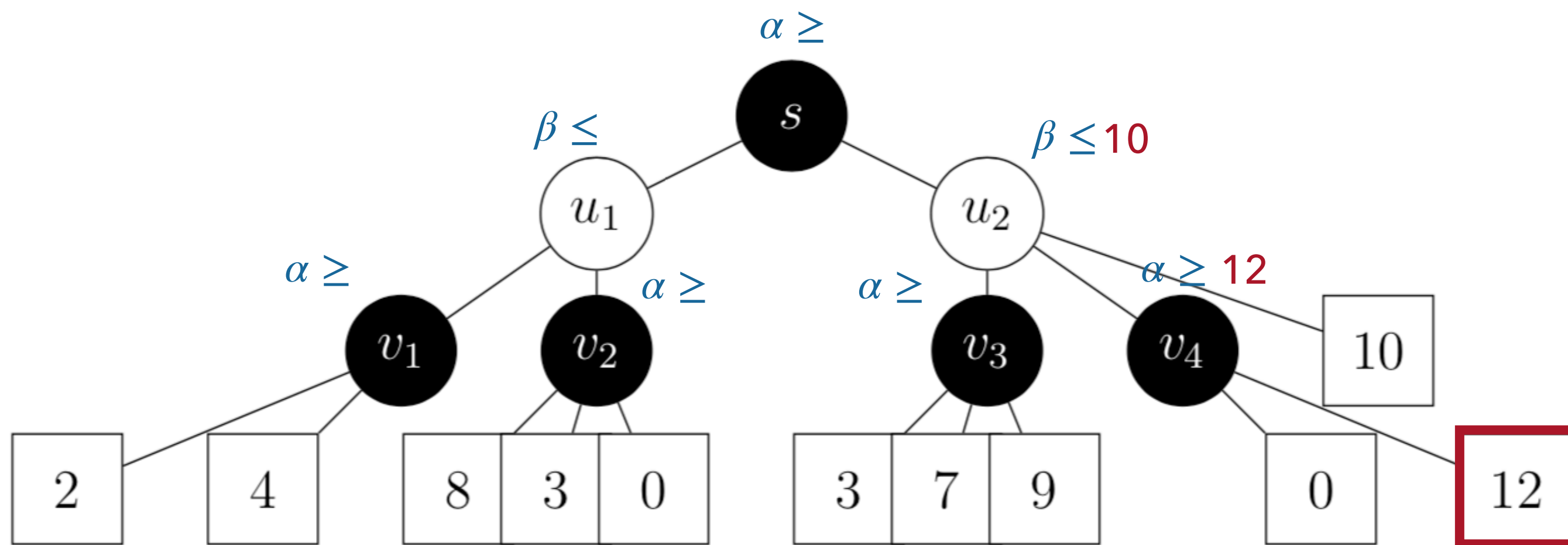


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

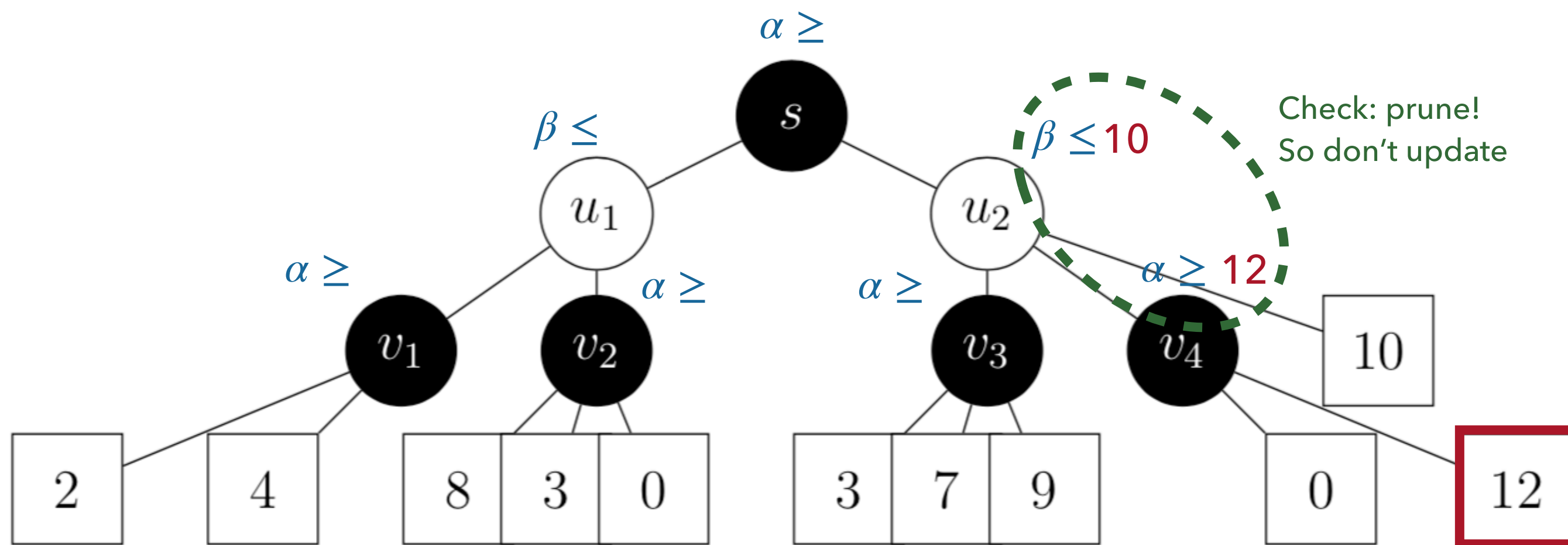
- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
 PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

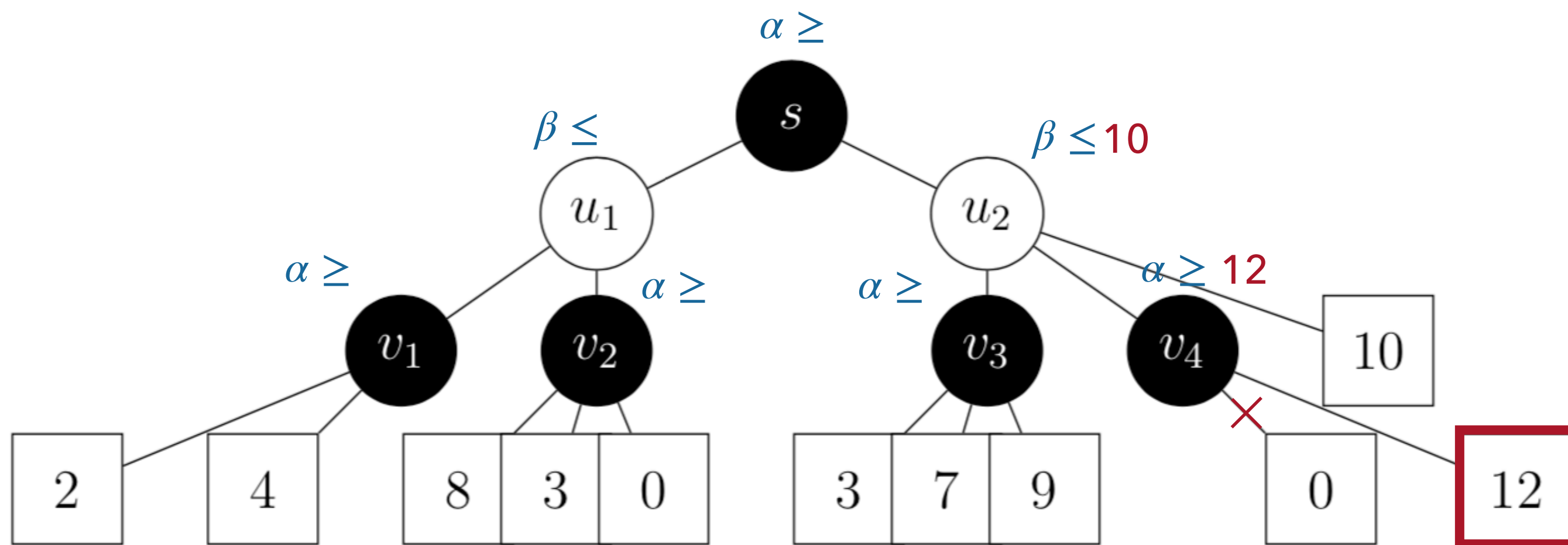
- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
 PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

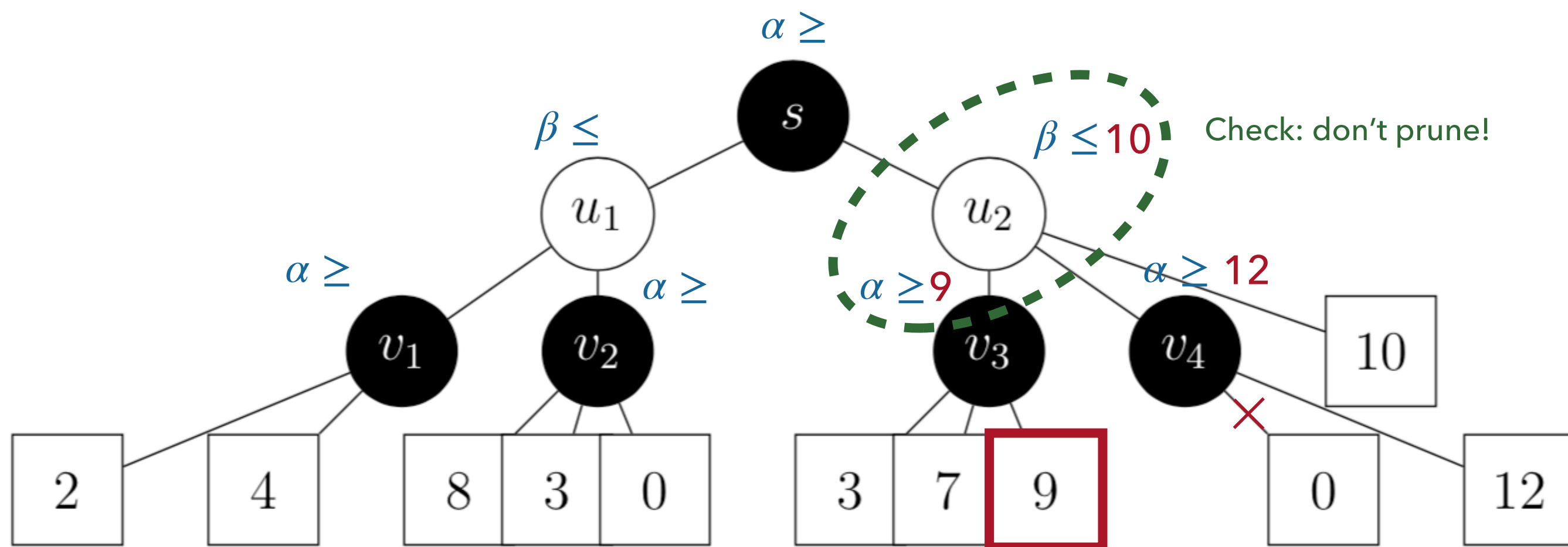
- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
 PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

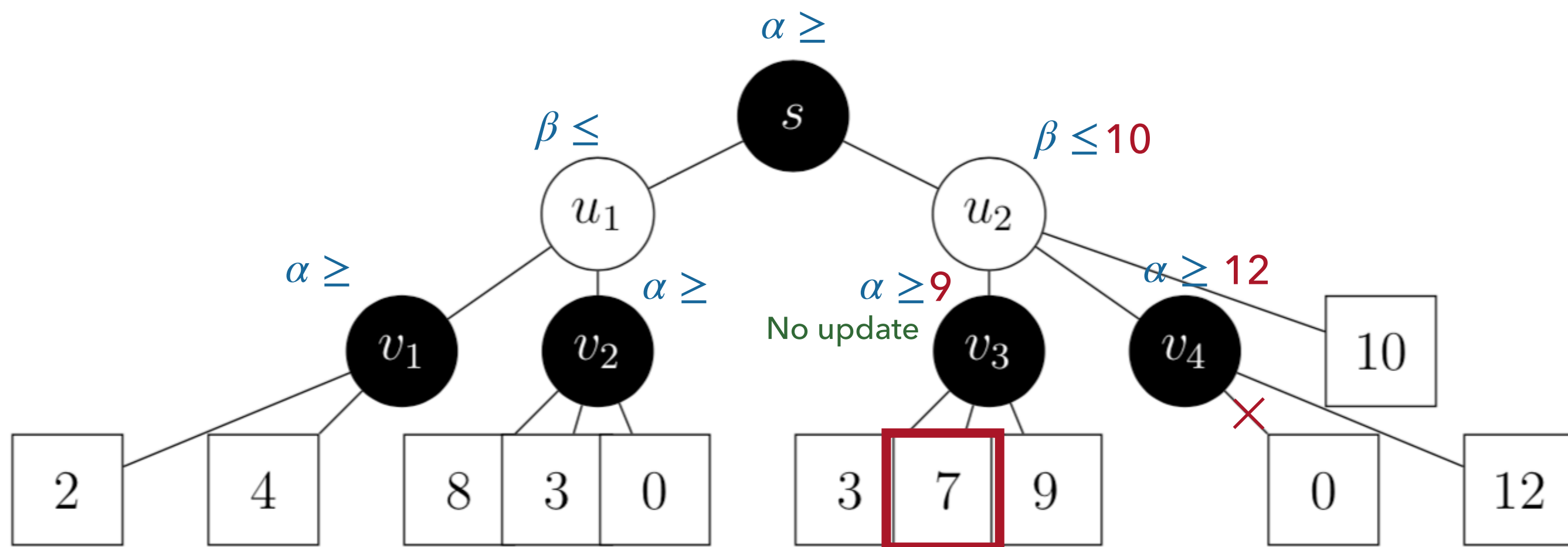
- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
 PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

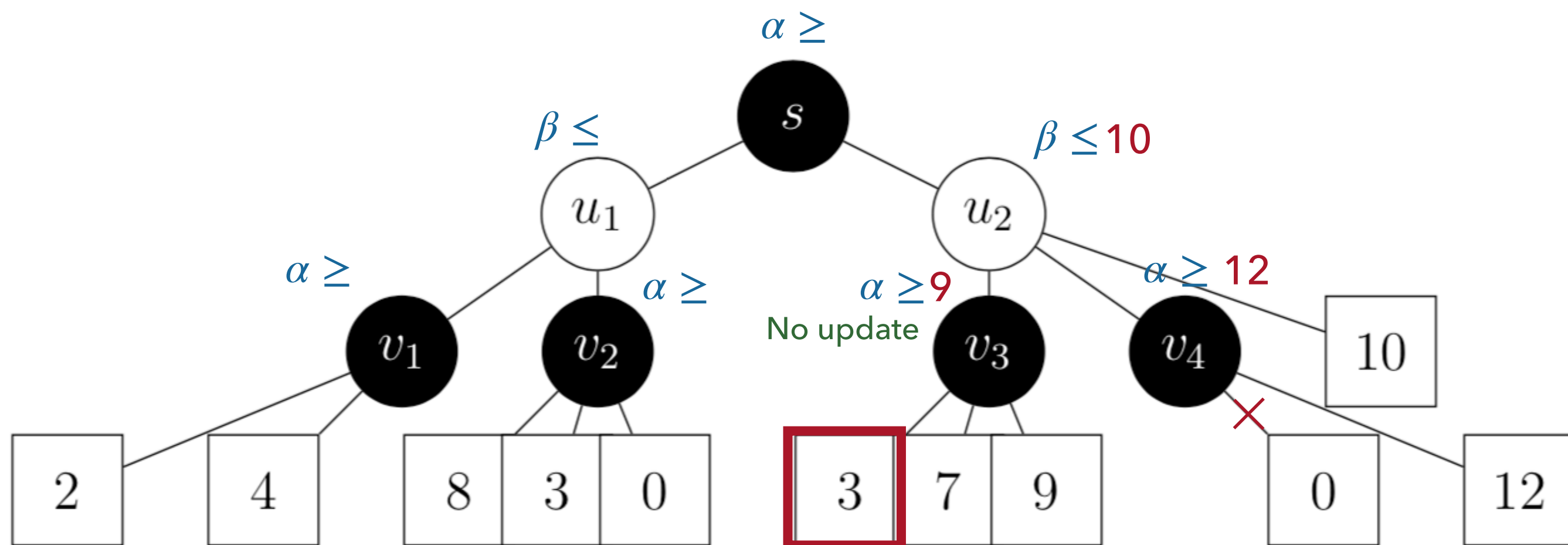
- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
 PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

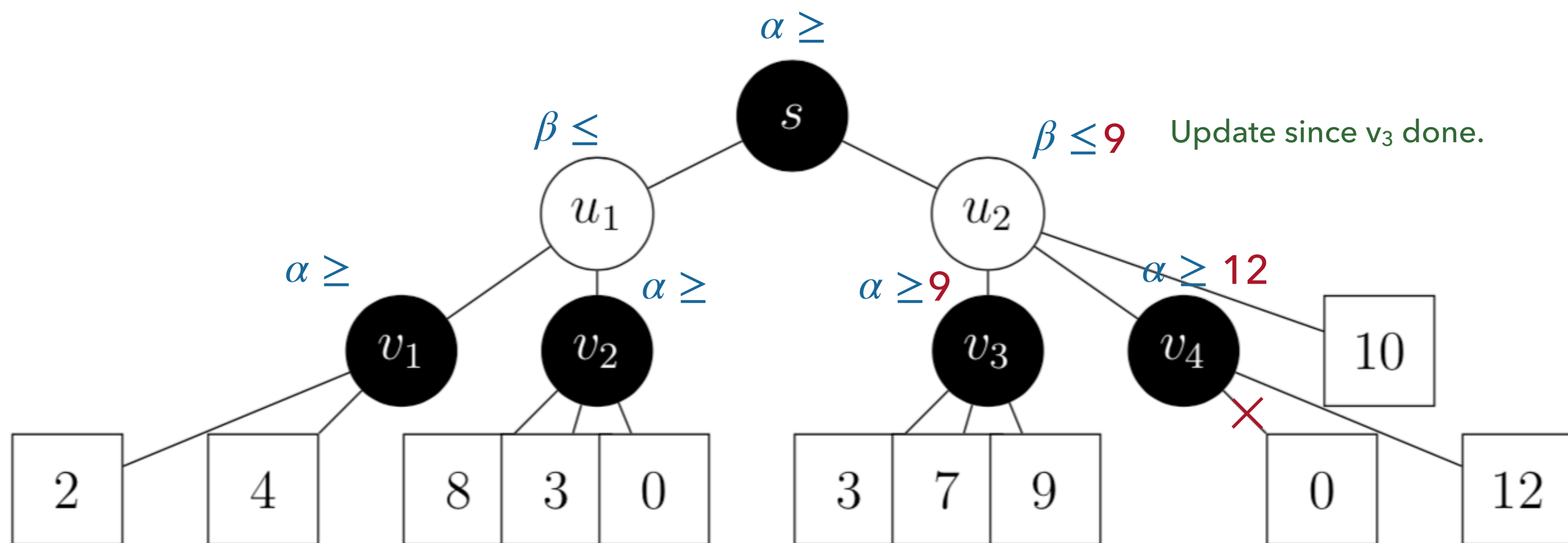


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

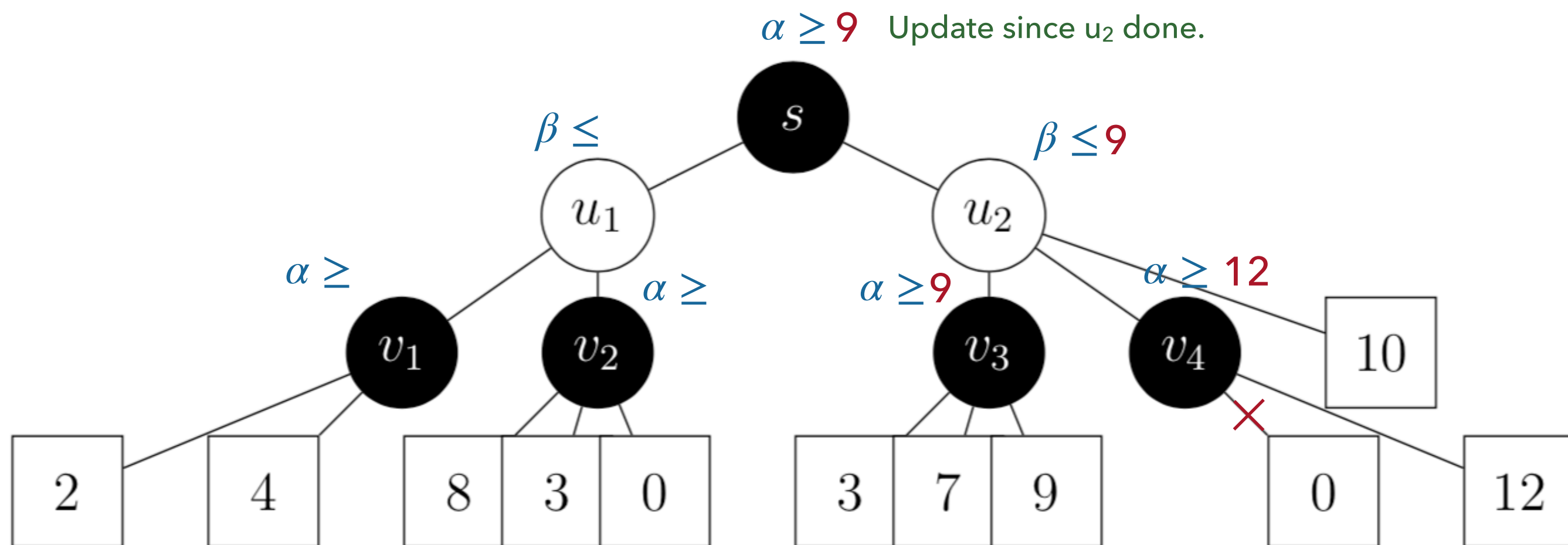


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

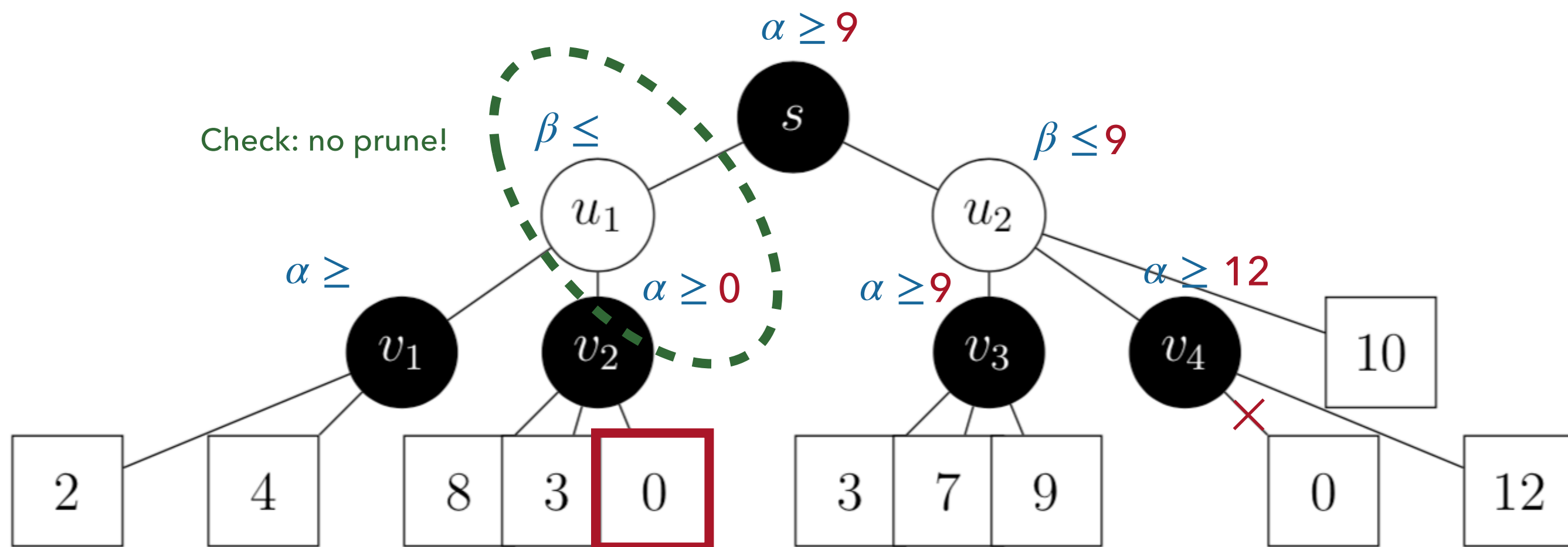


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

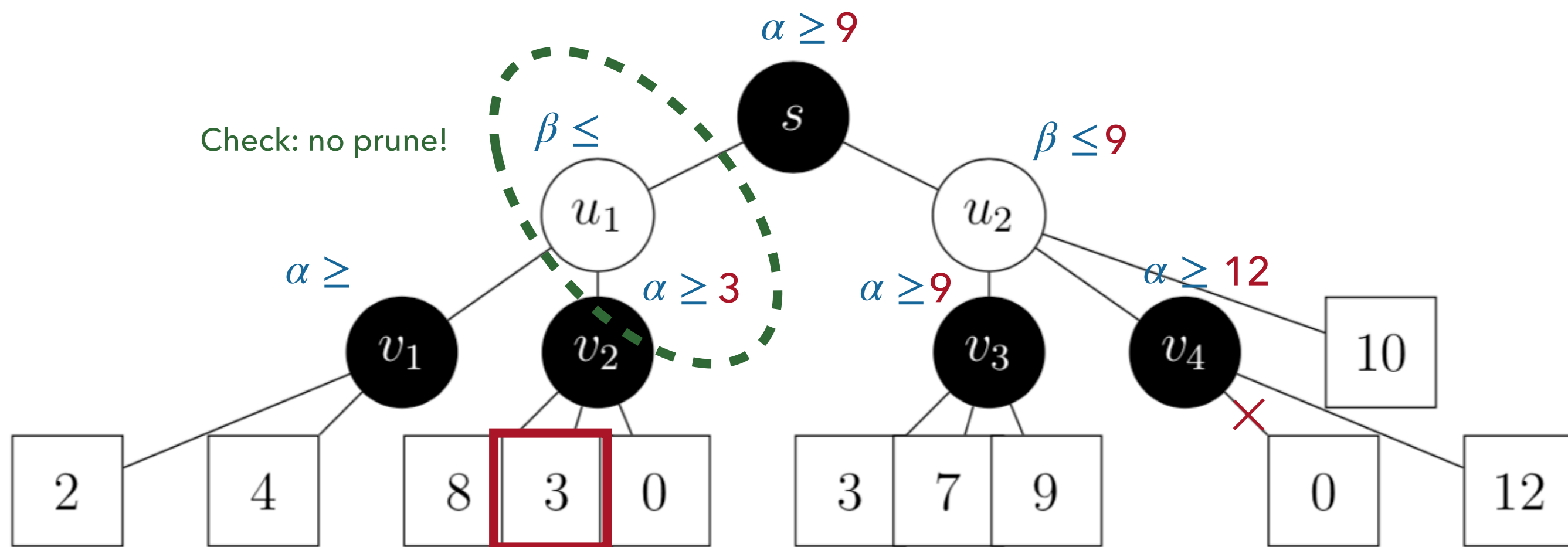


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

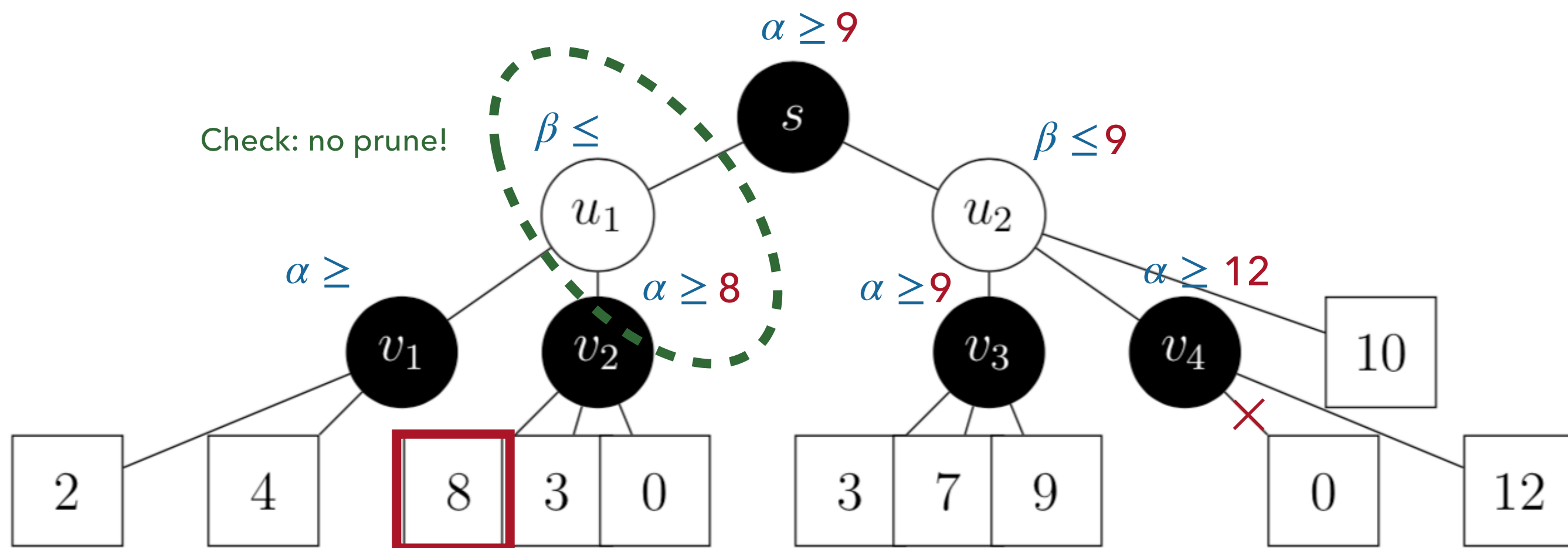


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

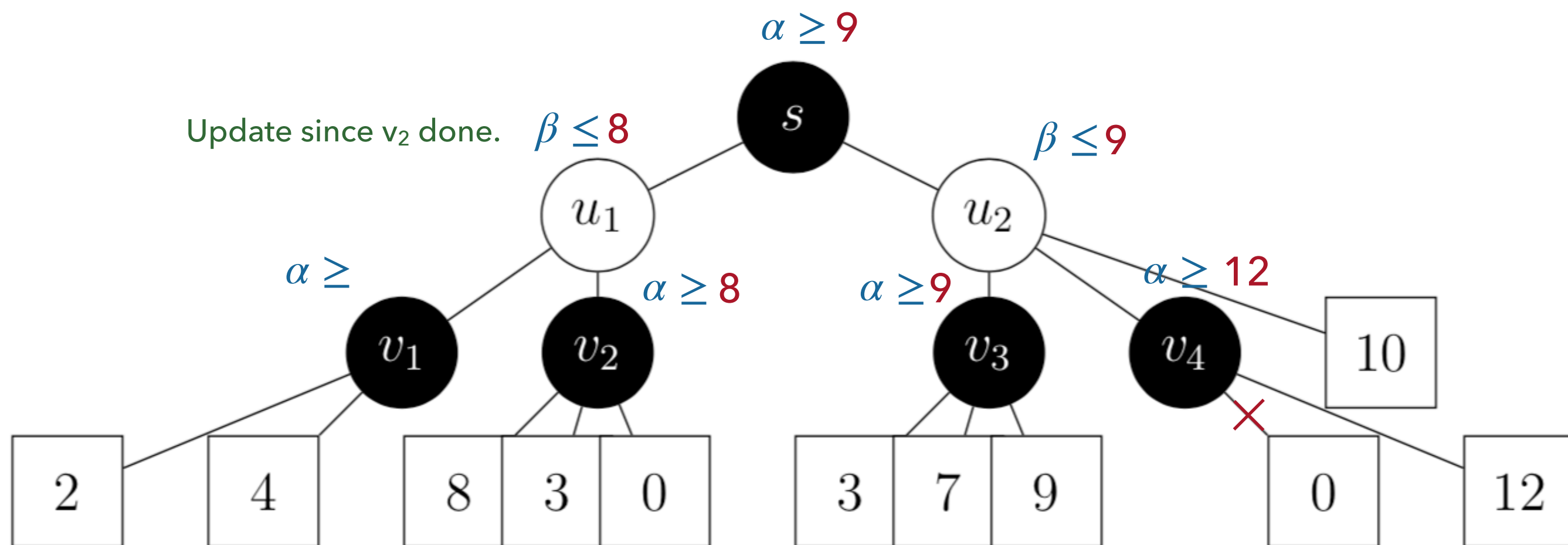
- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
 PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.

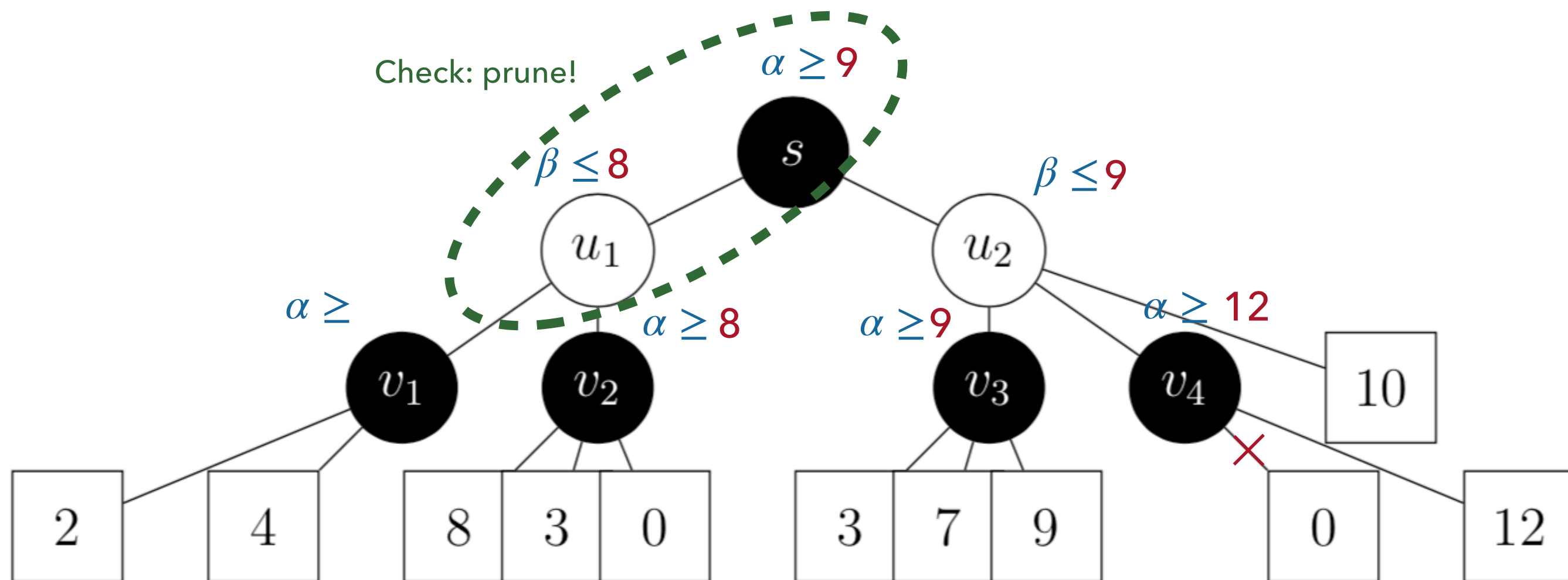


TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

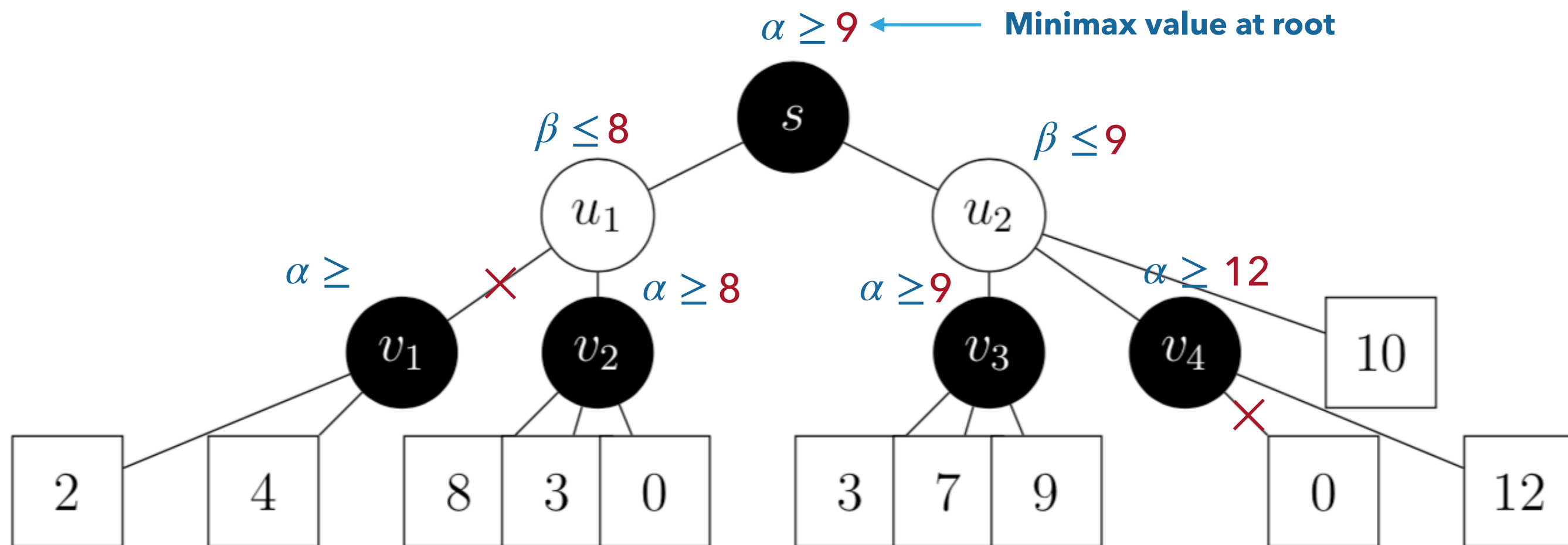
- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



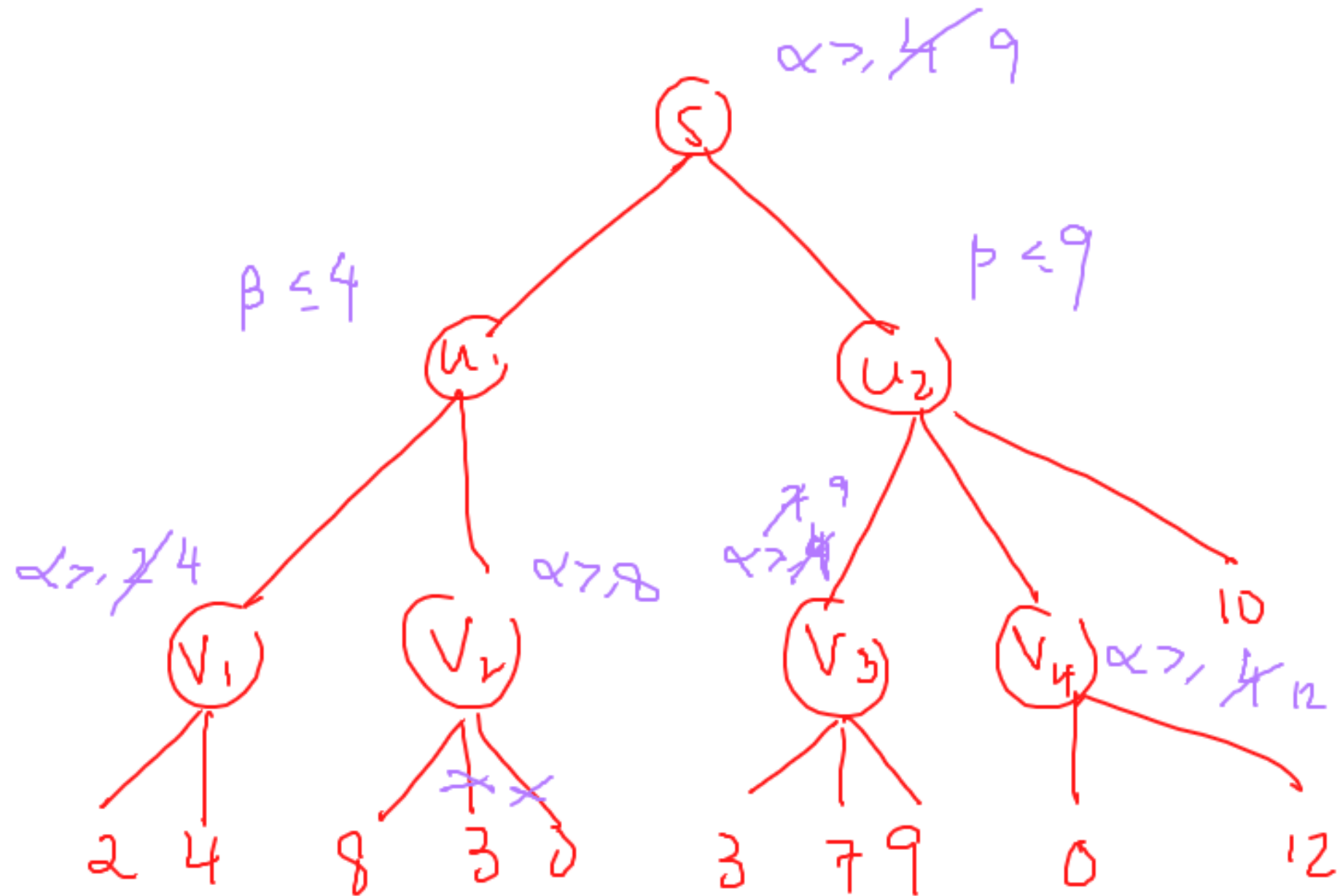
TUTORIAL 5 QUESTION 1

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
 PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; mark with an 'X' all ARCS that are pruned by α - β pruning, if any.



For justification, tell us the alpha/beta values of the immediate node will do.

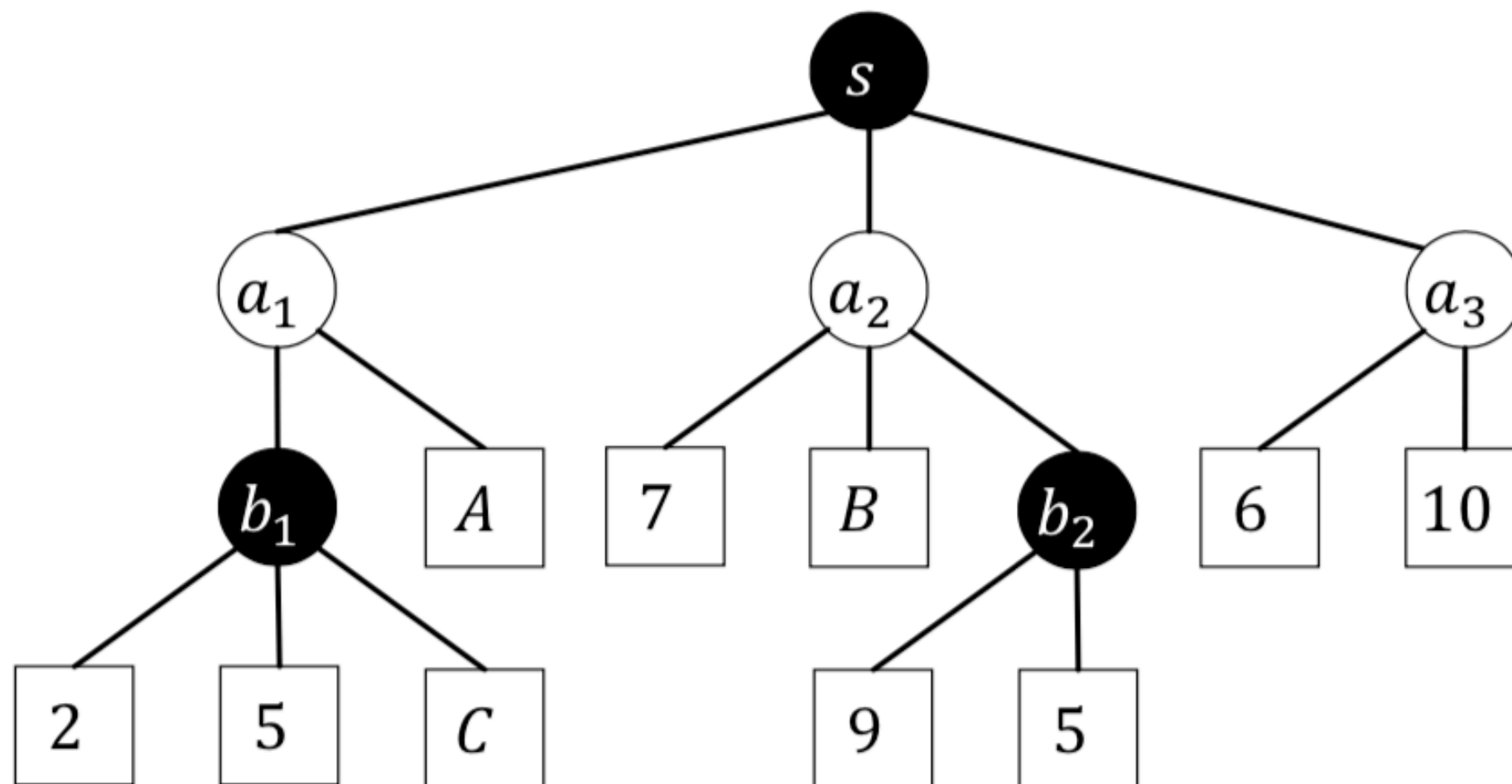


MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > _____

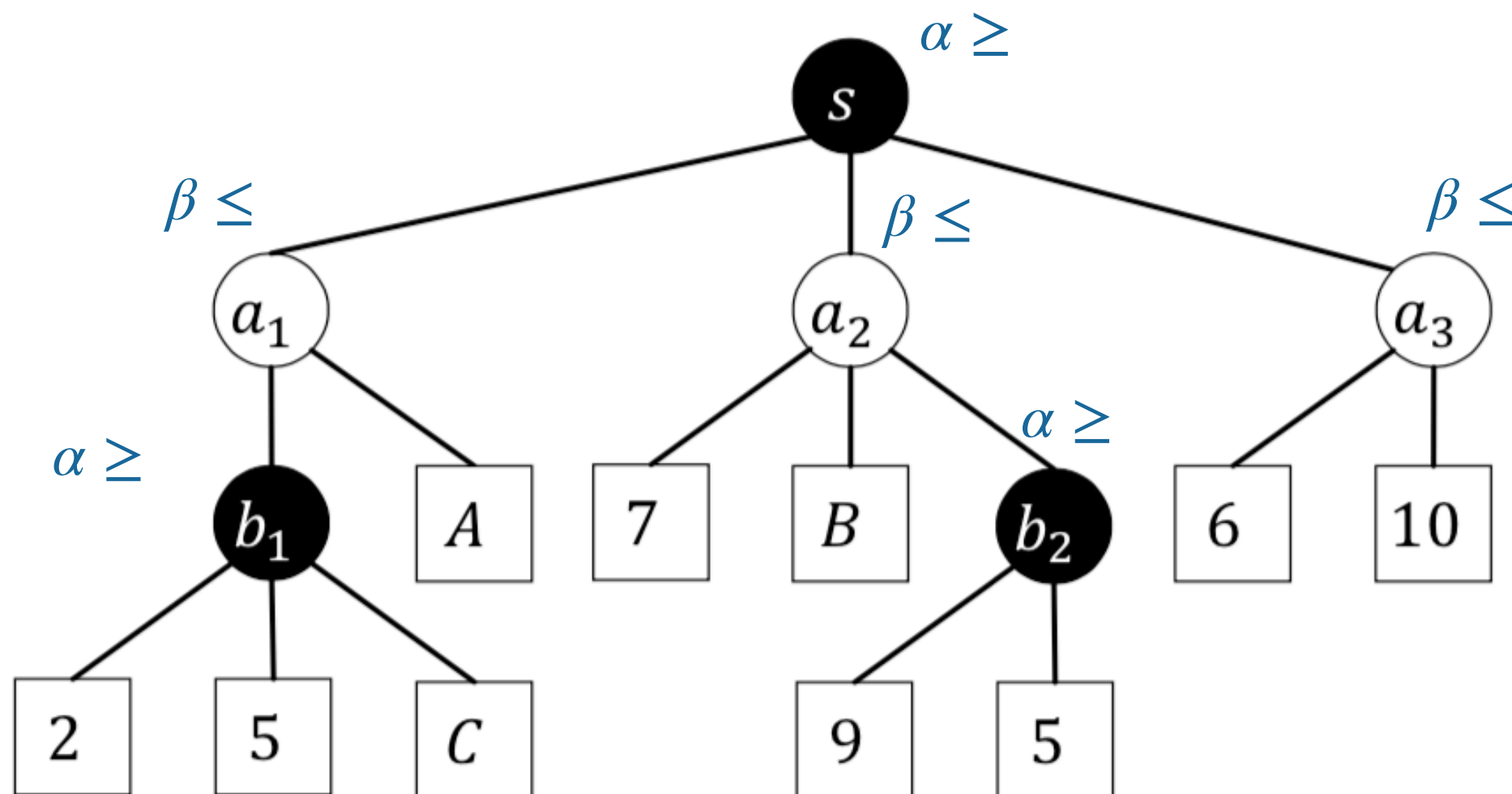
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > _____

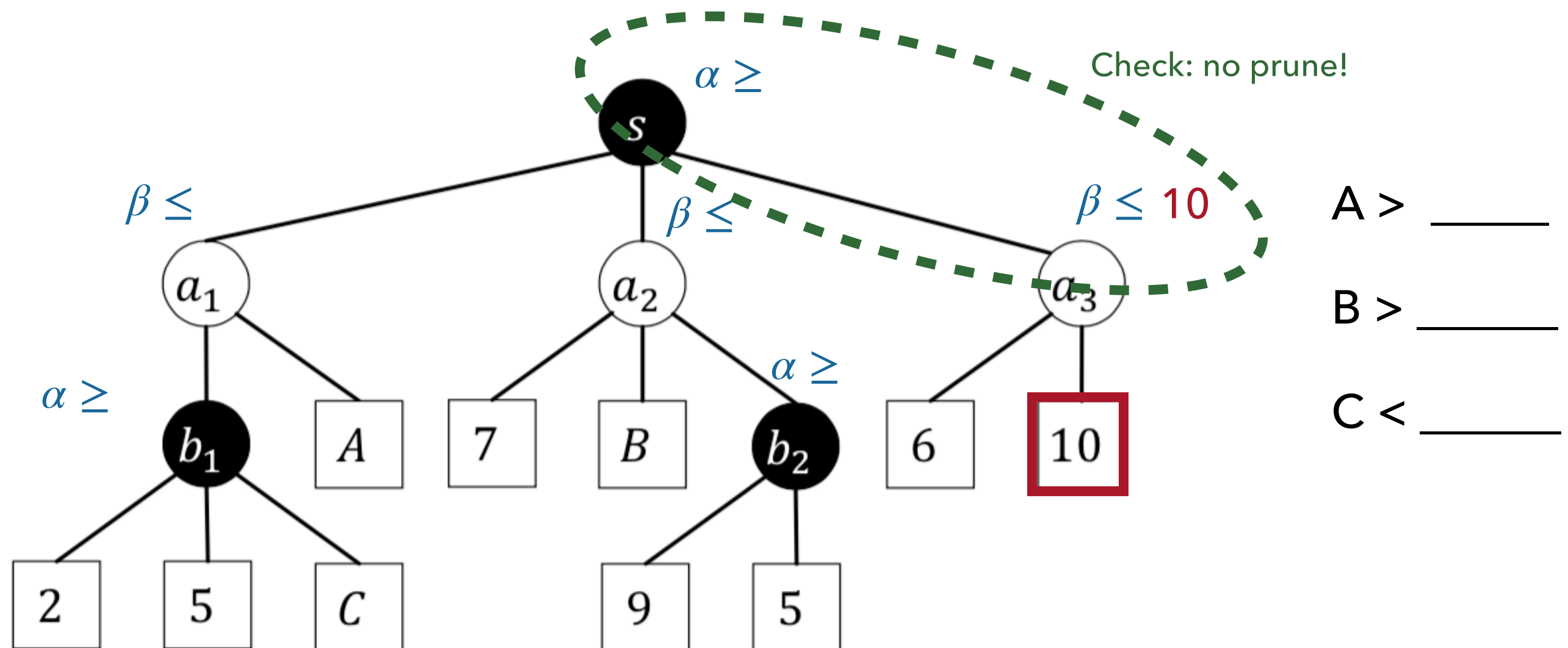
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.

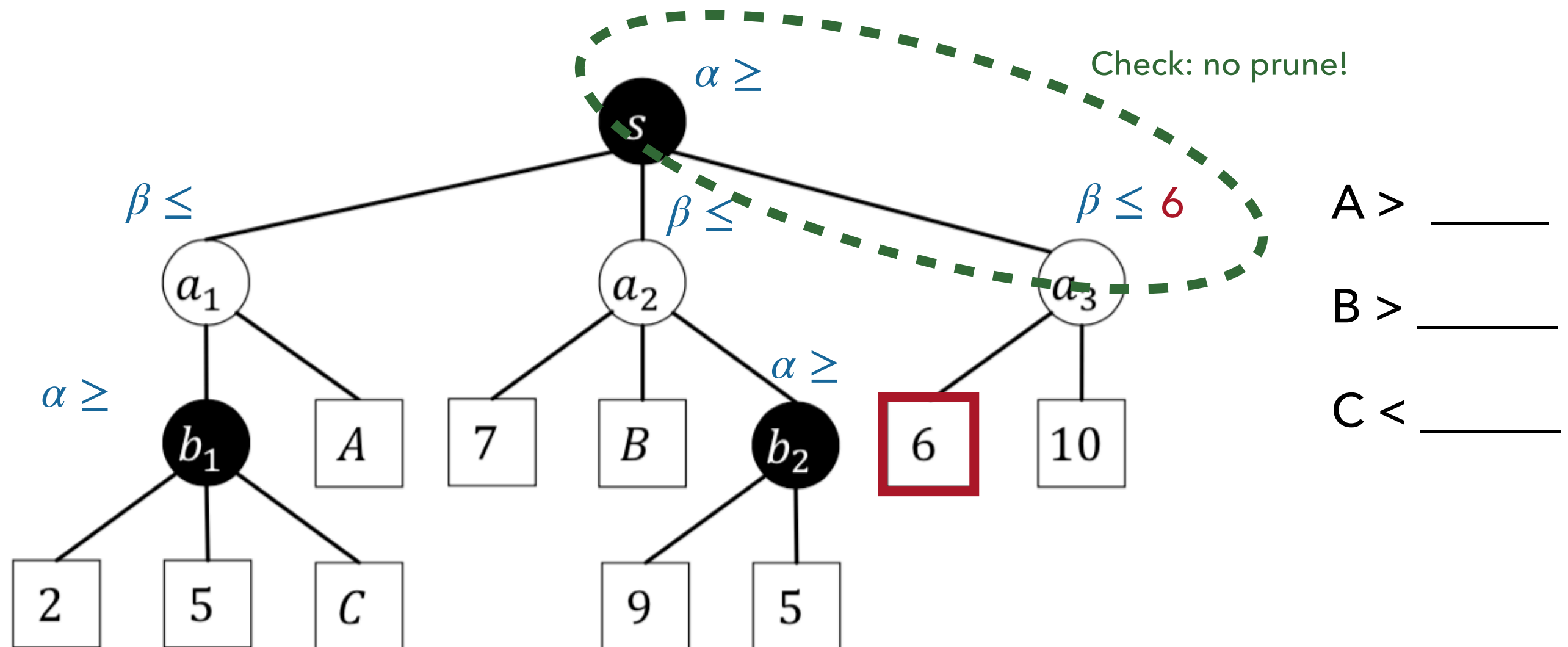


MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.

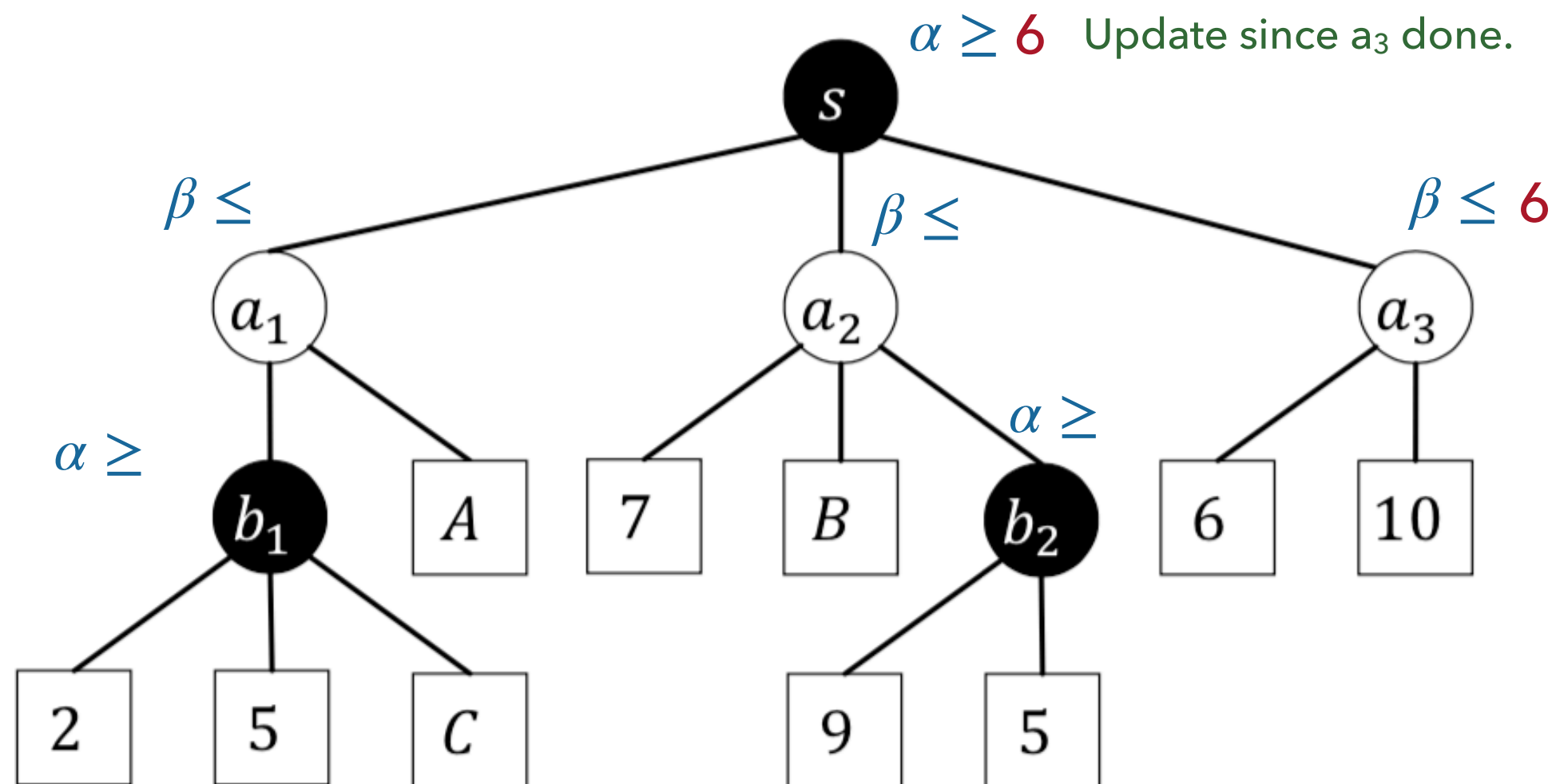


MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > _____

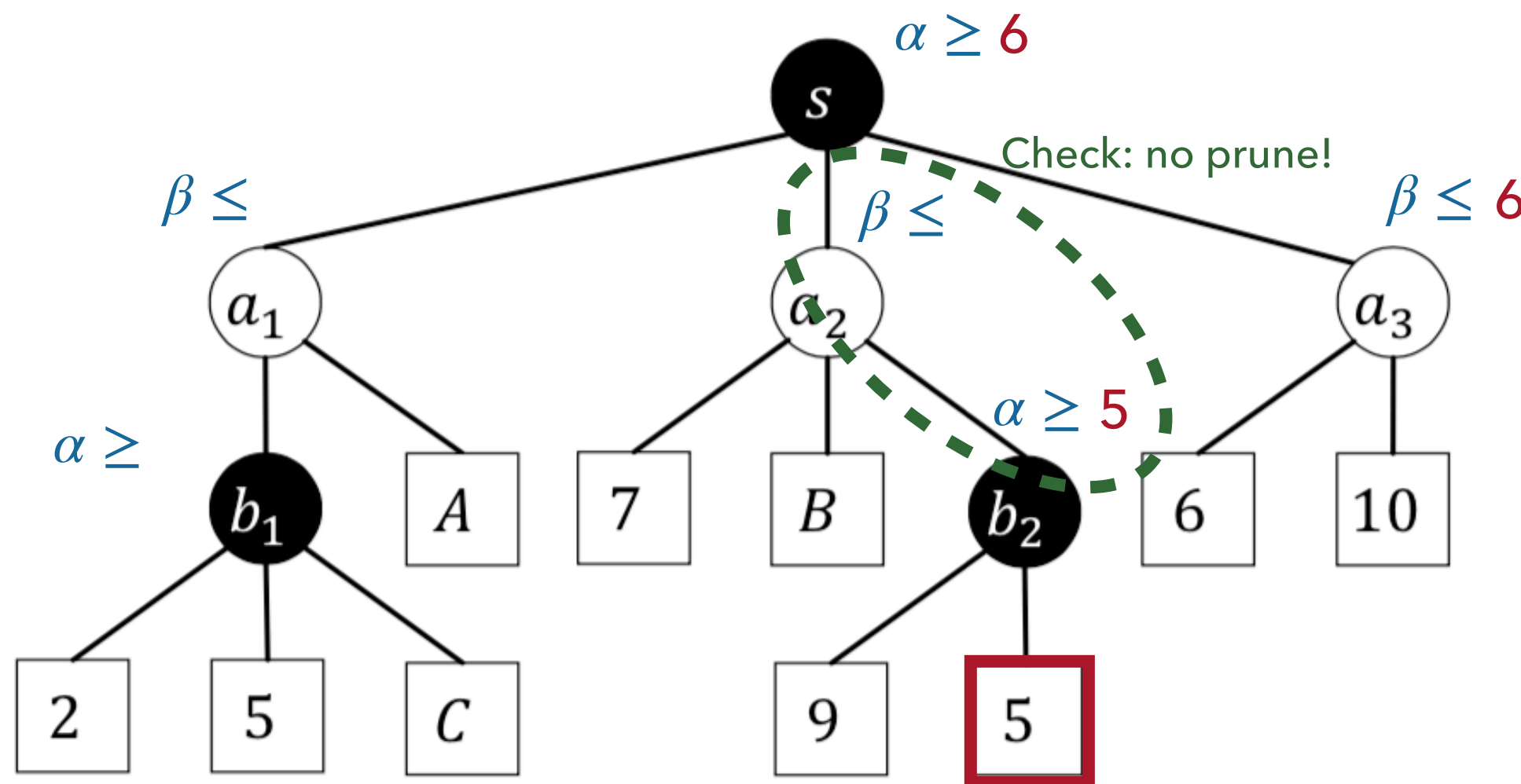
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > _____

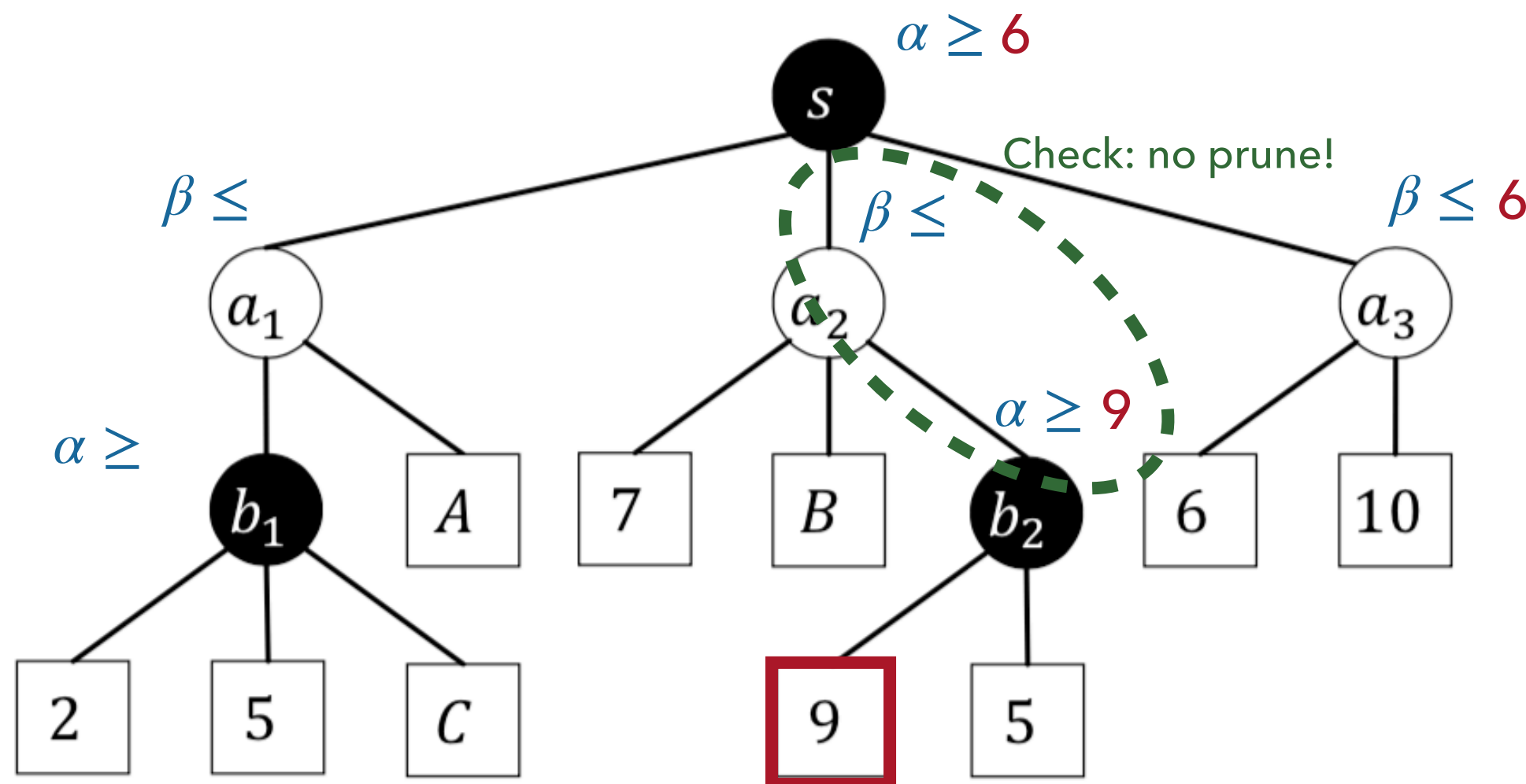
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > _____

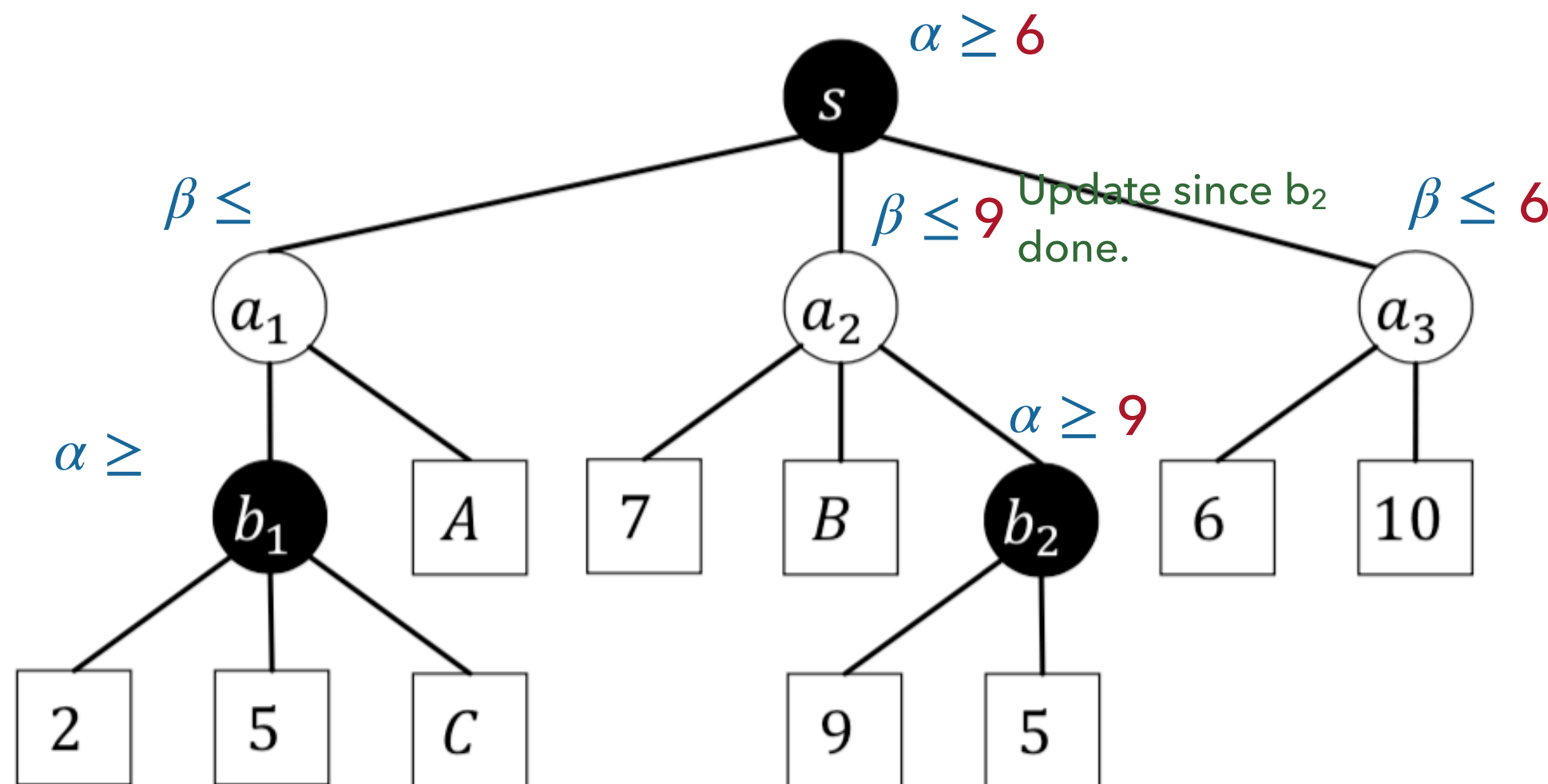
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > _____

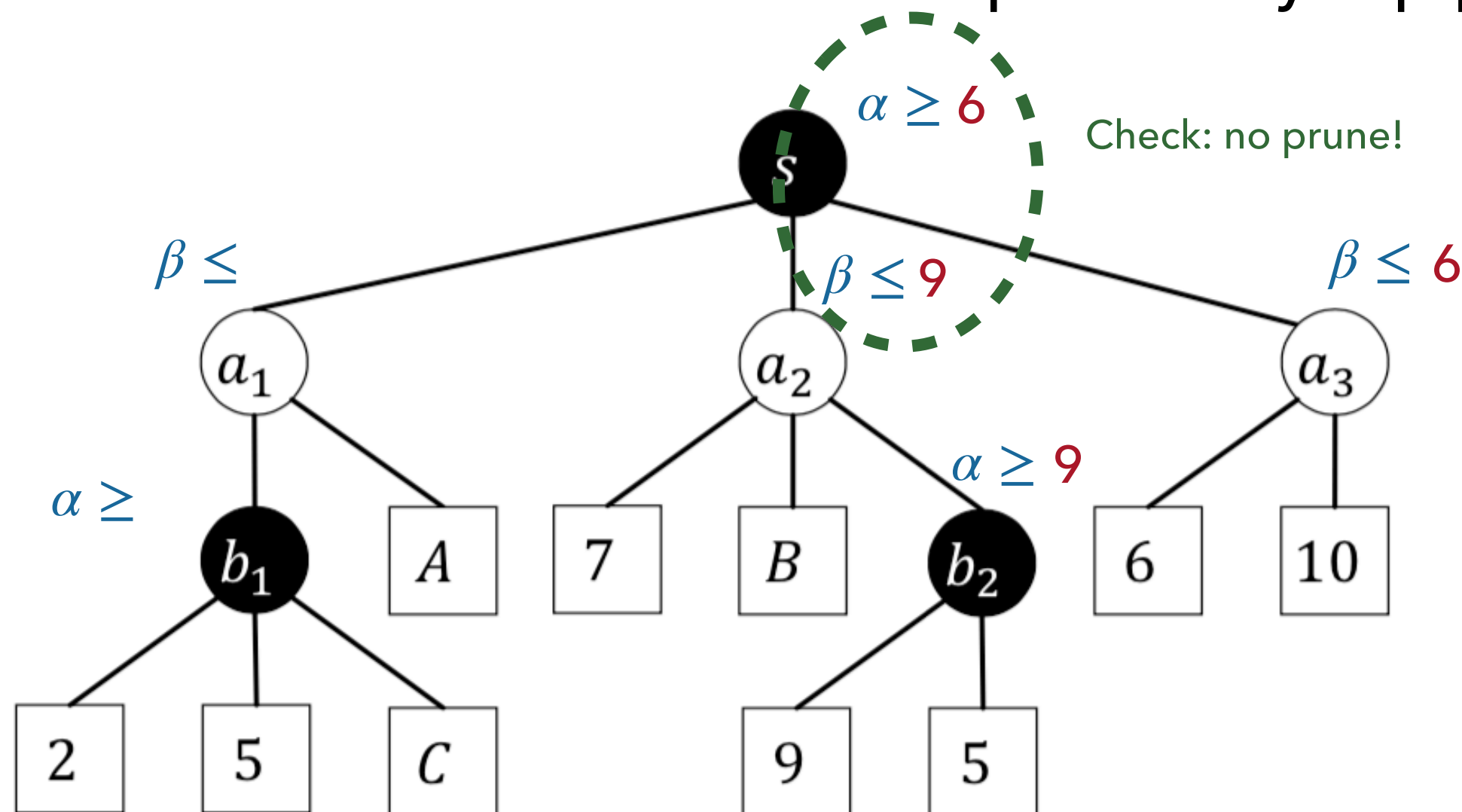
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > _____

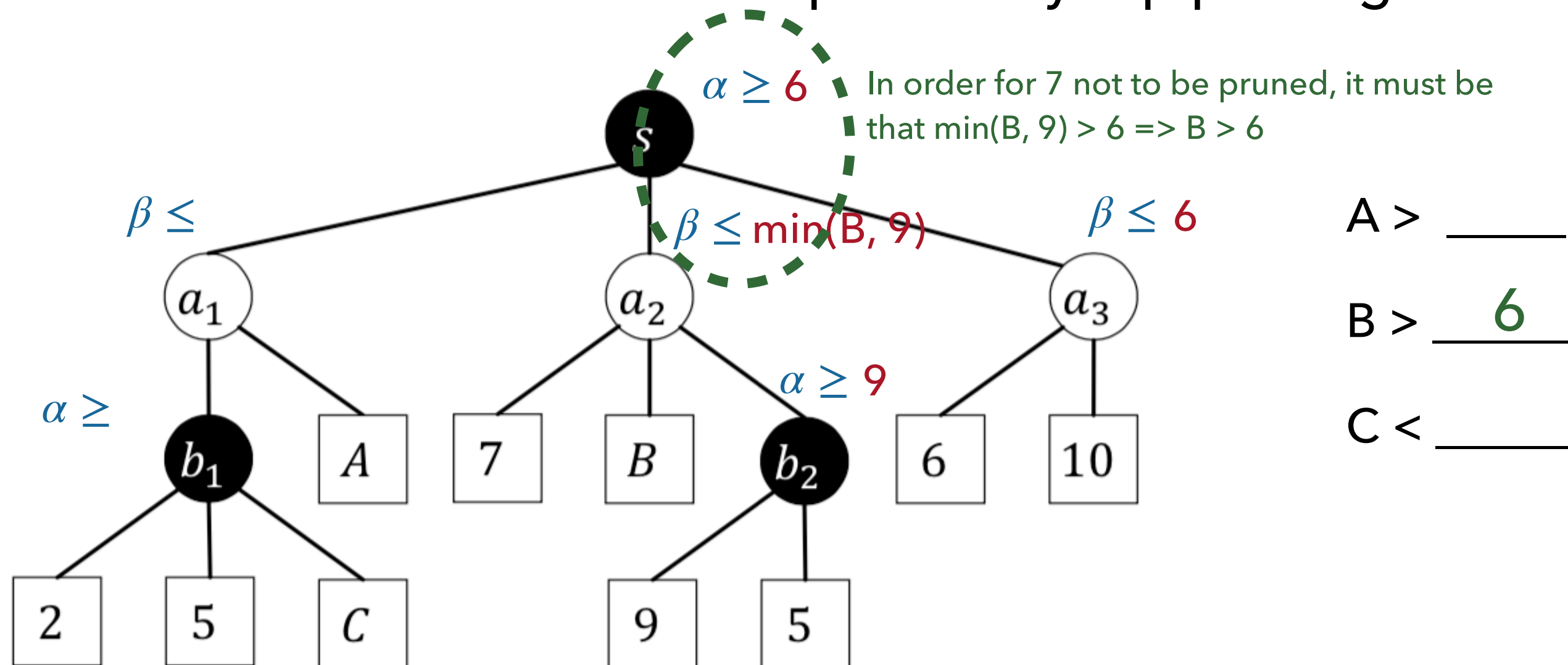
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.

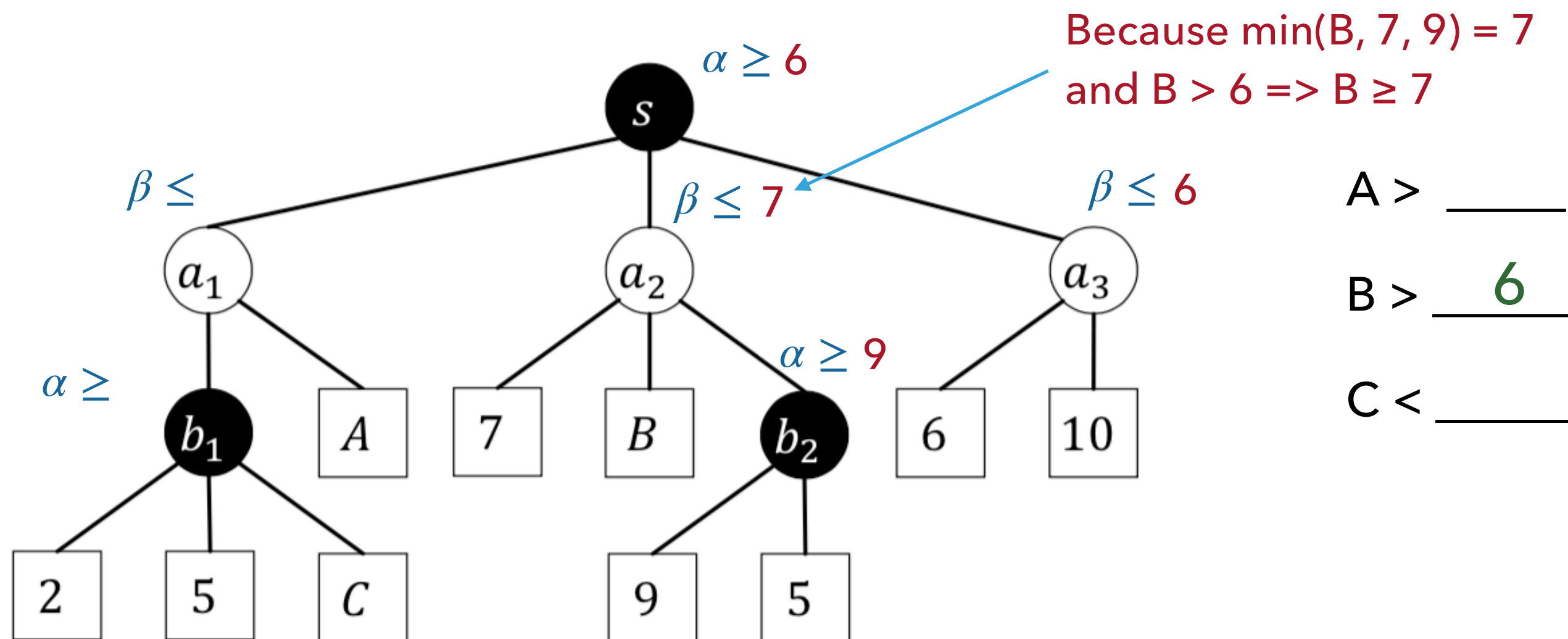


MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.

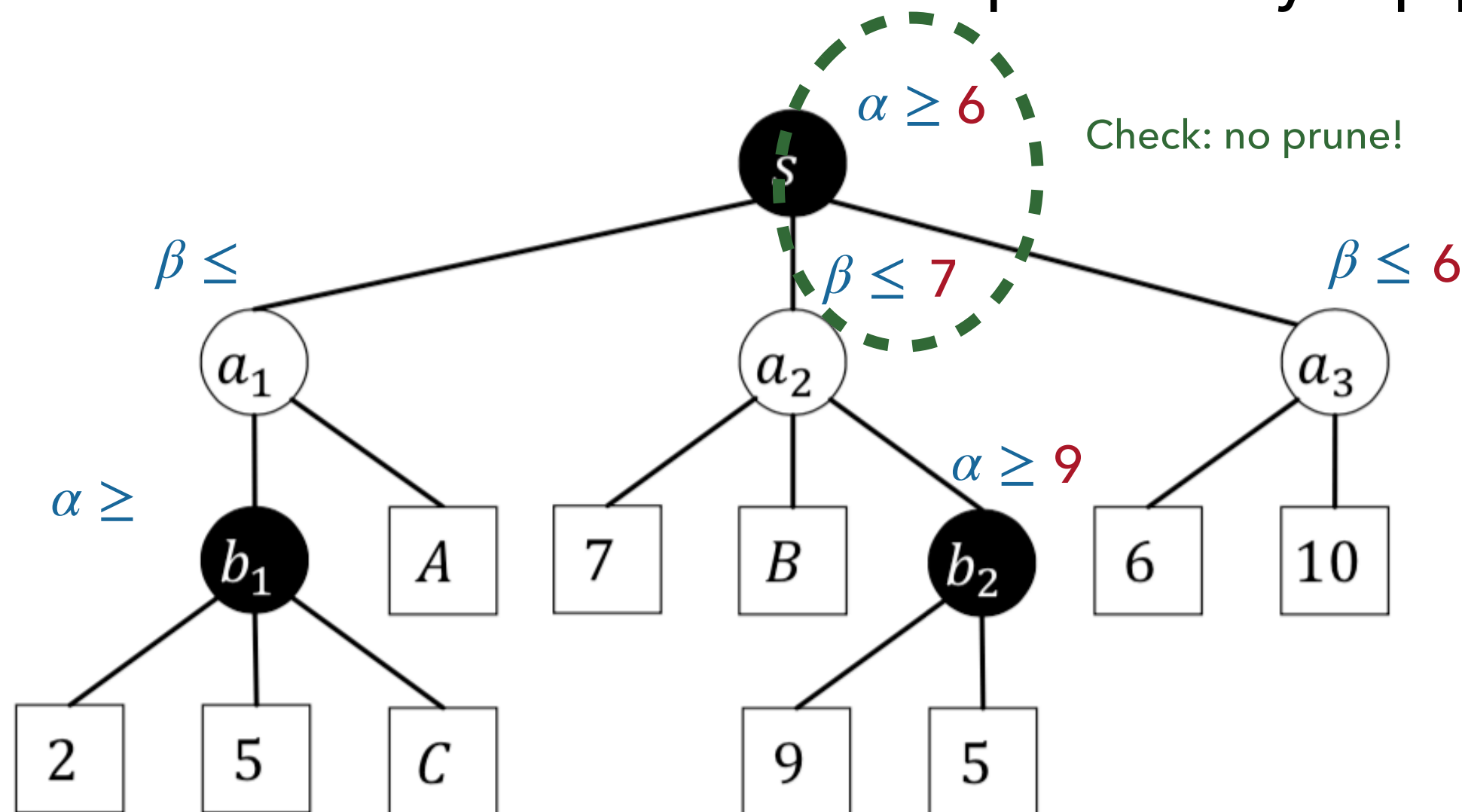


MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > 6

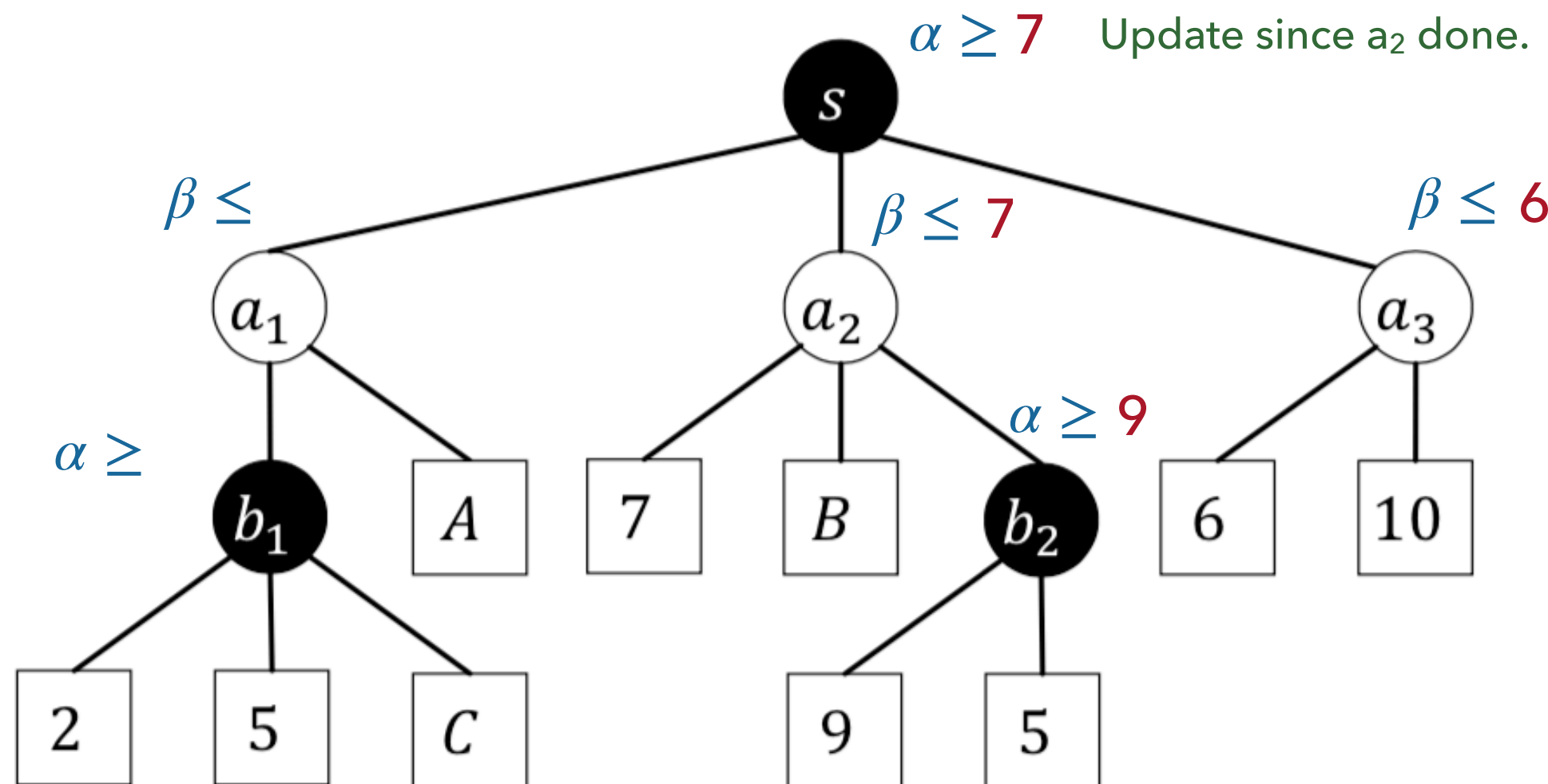
C < _____

MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.



A > _____

B > 6

C < _____

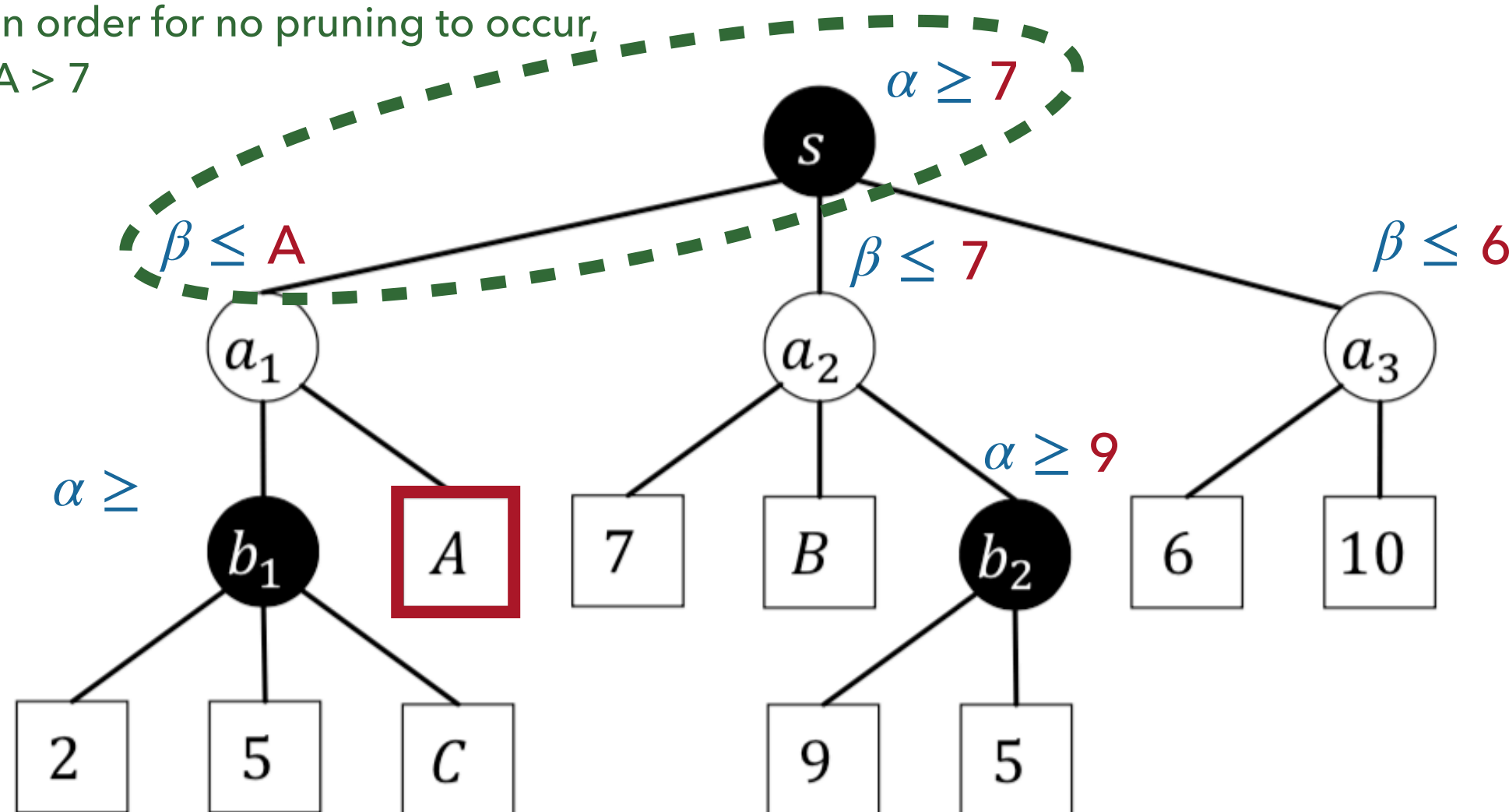
MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.

In order for no pruning to occur,
 $A > 7$



$$A > \underline{7}$$

$$B > \underline{6}$$

$$C < \underline{\quad}$$

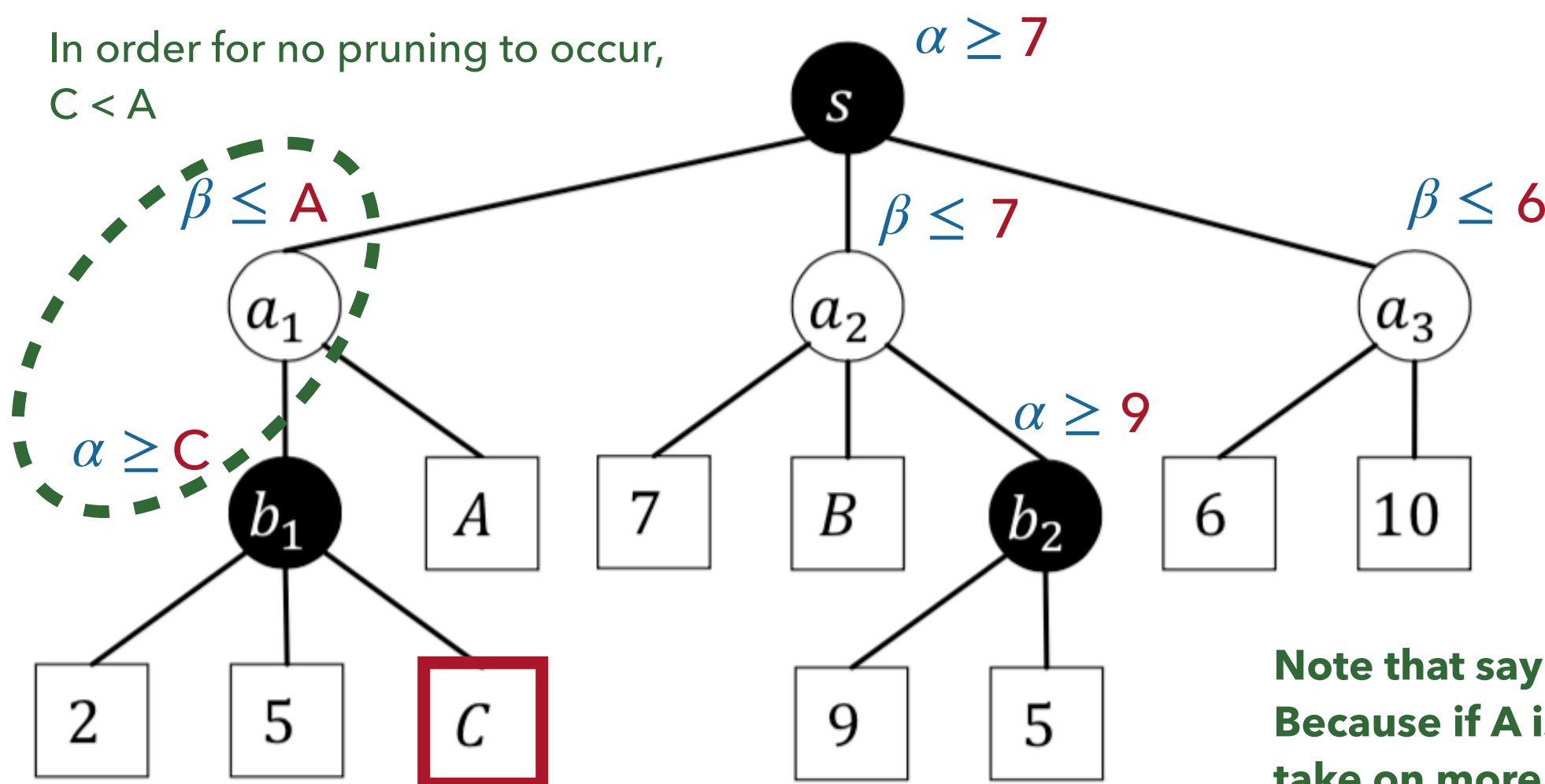
MOCK MID-TERM

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

- Assume that we iterate over nodes from **right to left**; find values to ensure no arcs are pruned by α - β pruning.

In order for no pruning to occur,
 $C < A$



$$A > \underline{7}$$

$$B > \underline{6}$$

$$C < \underline{A}$$

Note that saying $C < 8$ is insufficient. Because if A is very large, then C can take on more values than merely $C < 8$.

EXTRA QUESTION

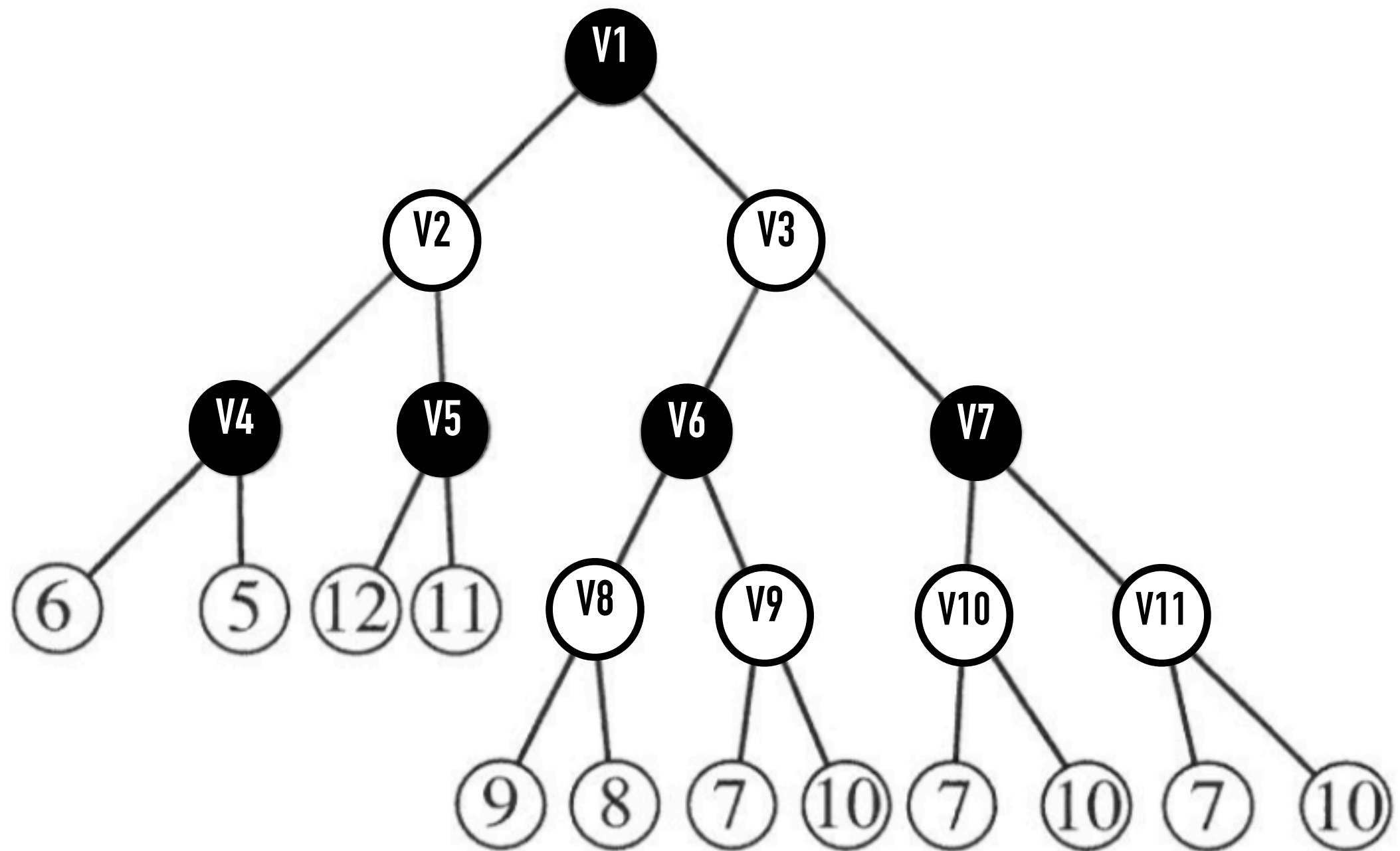
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



EXTRA QUESTION

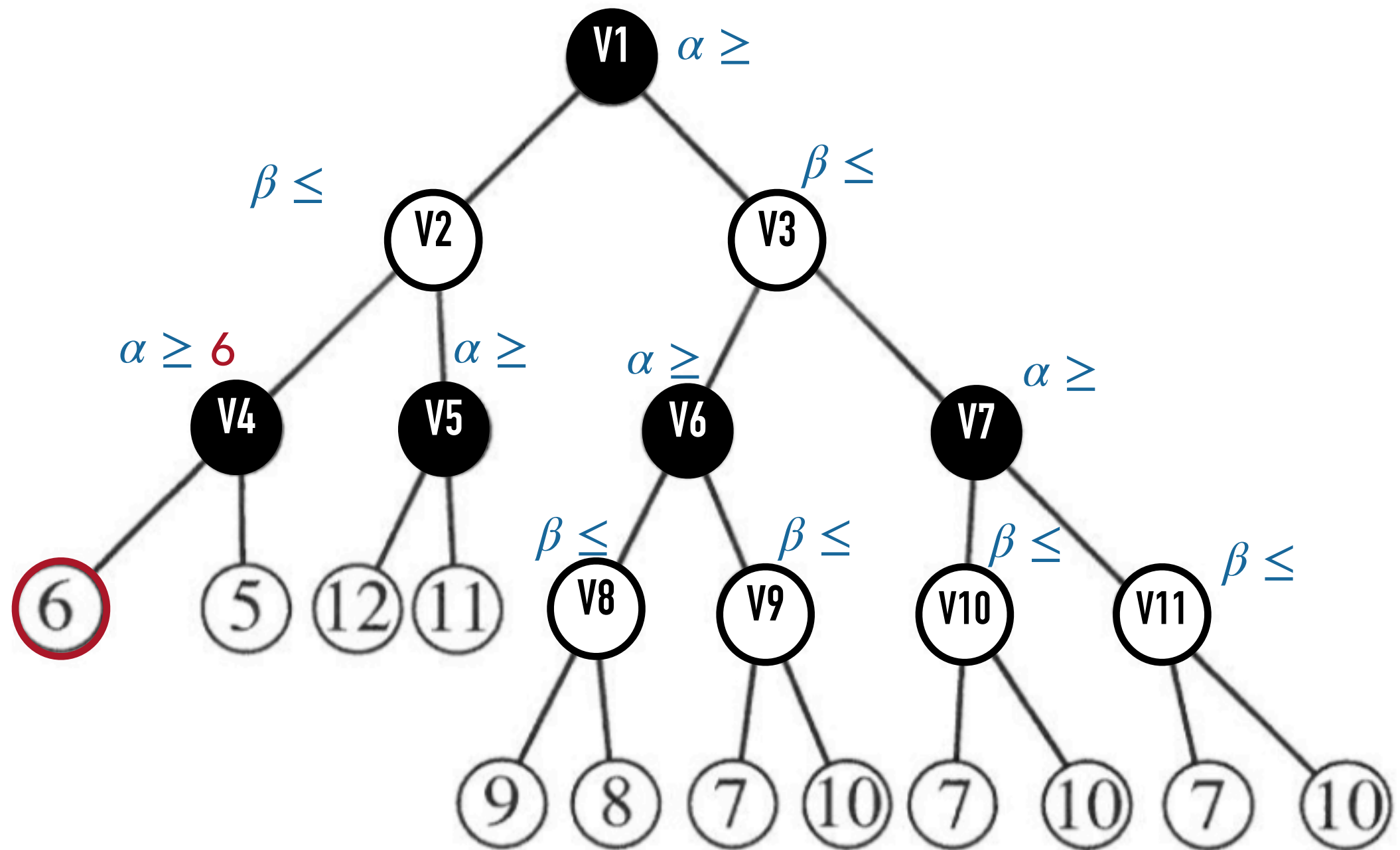
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

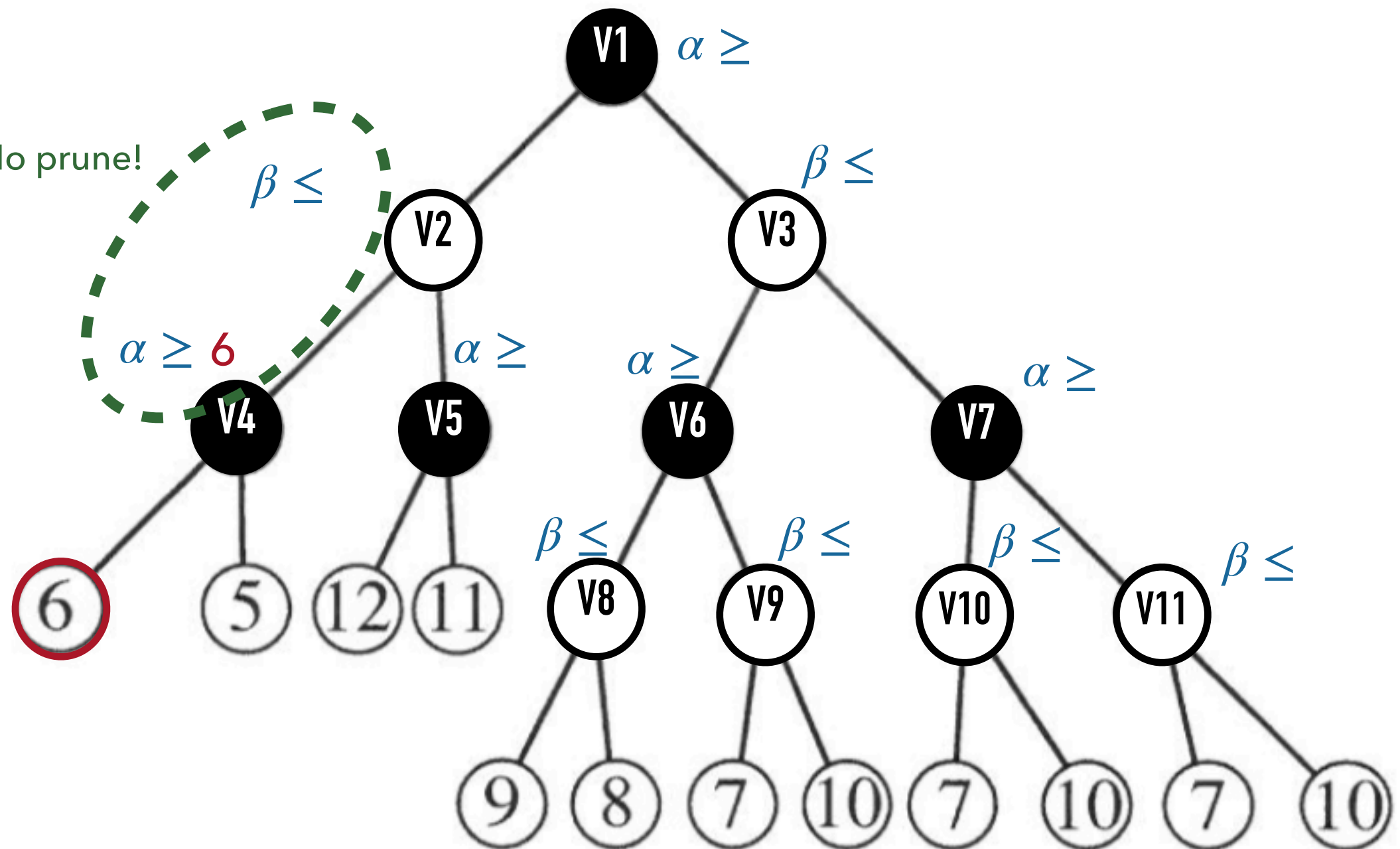
MAX

MIN

MAX

MIN

Check: No prune!



Show which arcs are pruned

EXTRA QUESTION

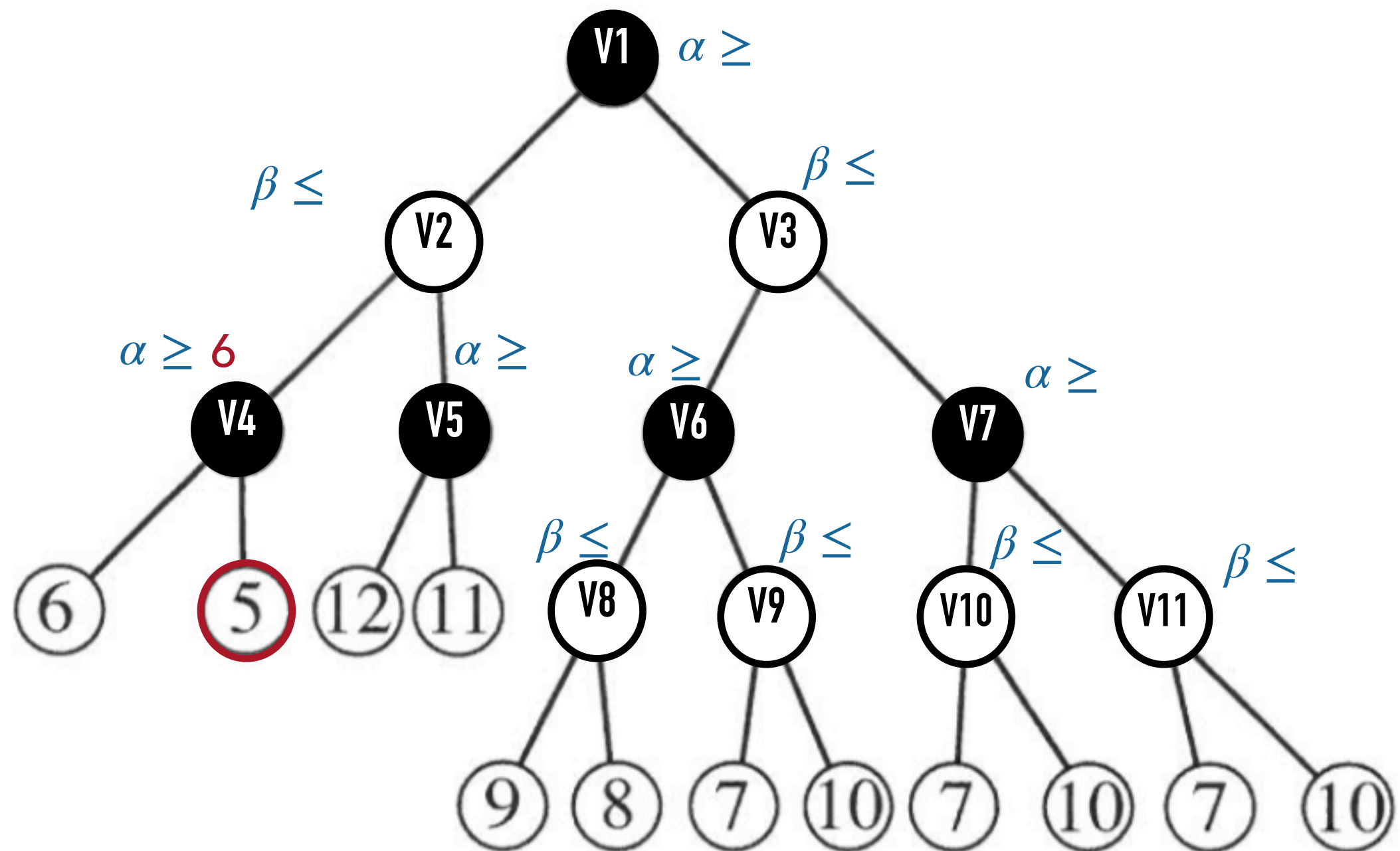
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

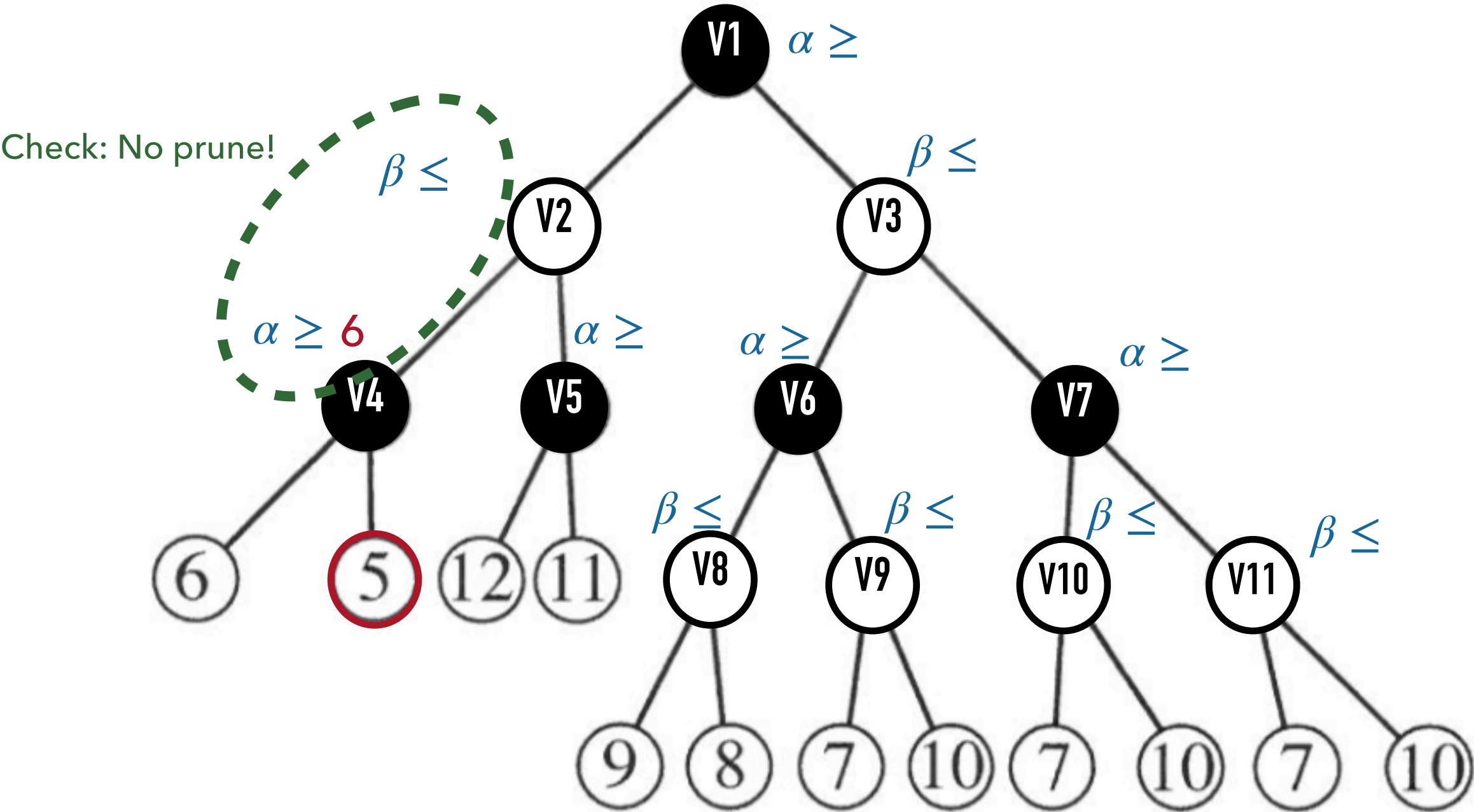
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

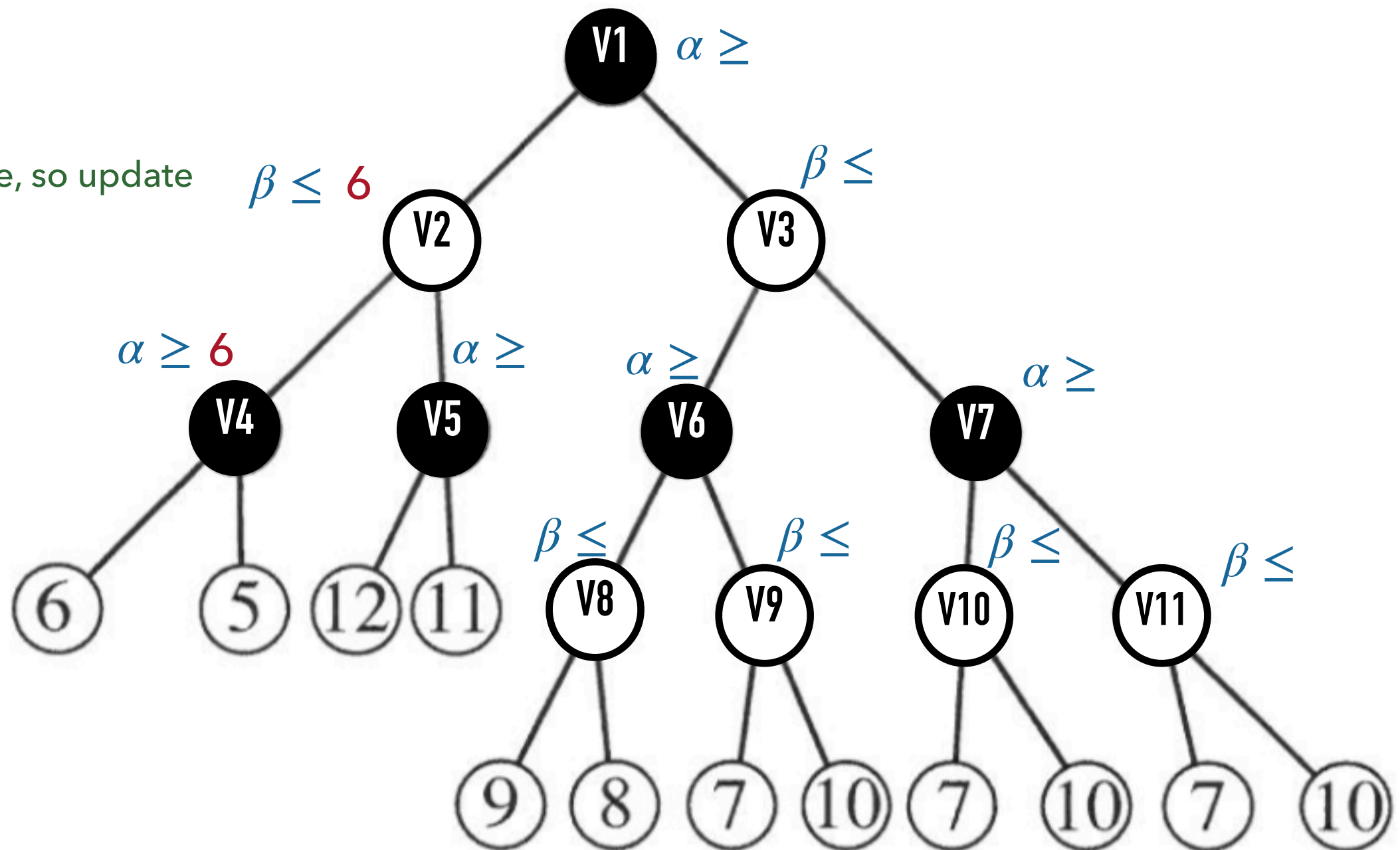
MIN



Show which arcs are pruned

PRUNING: If $[child] \beta \leq \alpha [parent]$, then prune child's remaining

MIN



Show which arcs are pruned

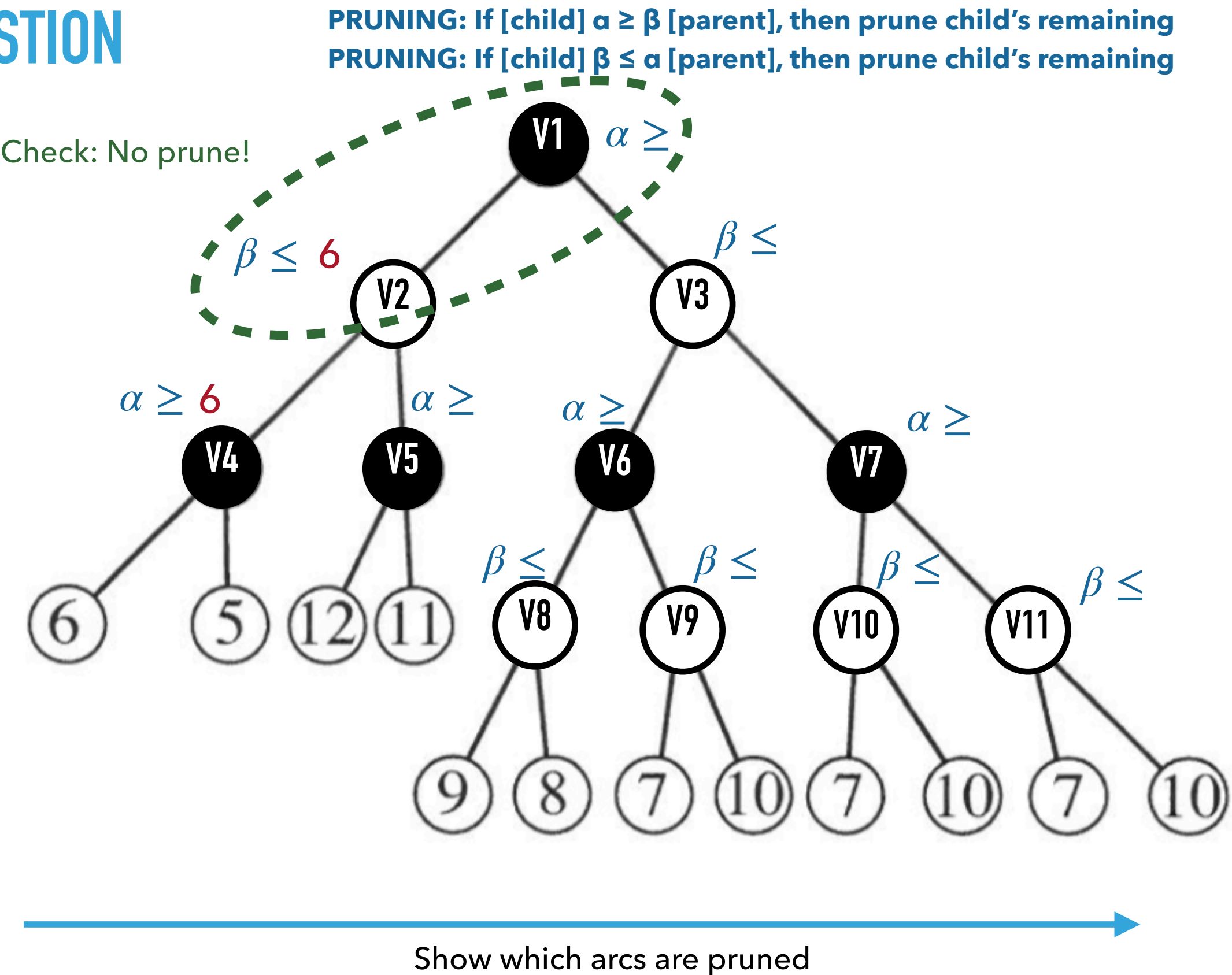
EXTRA QUESTION

MAX

MIN

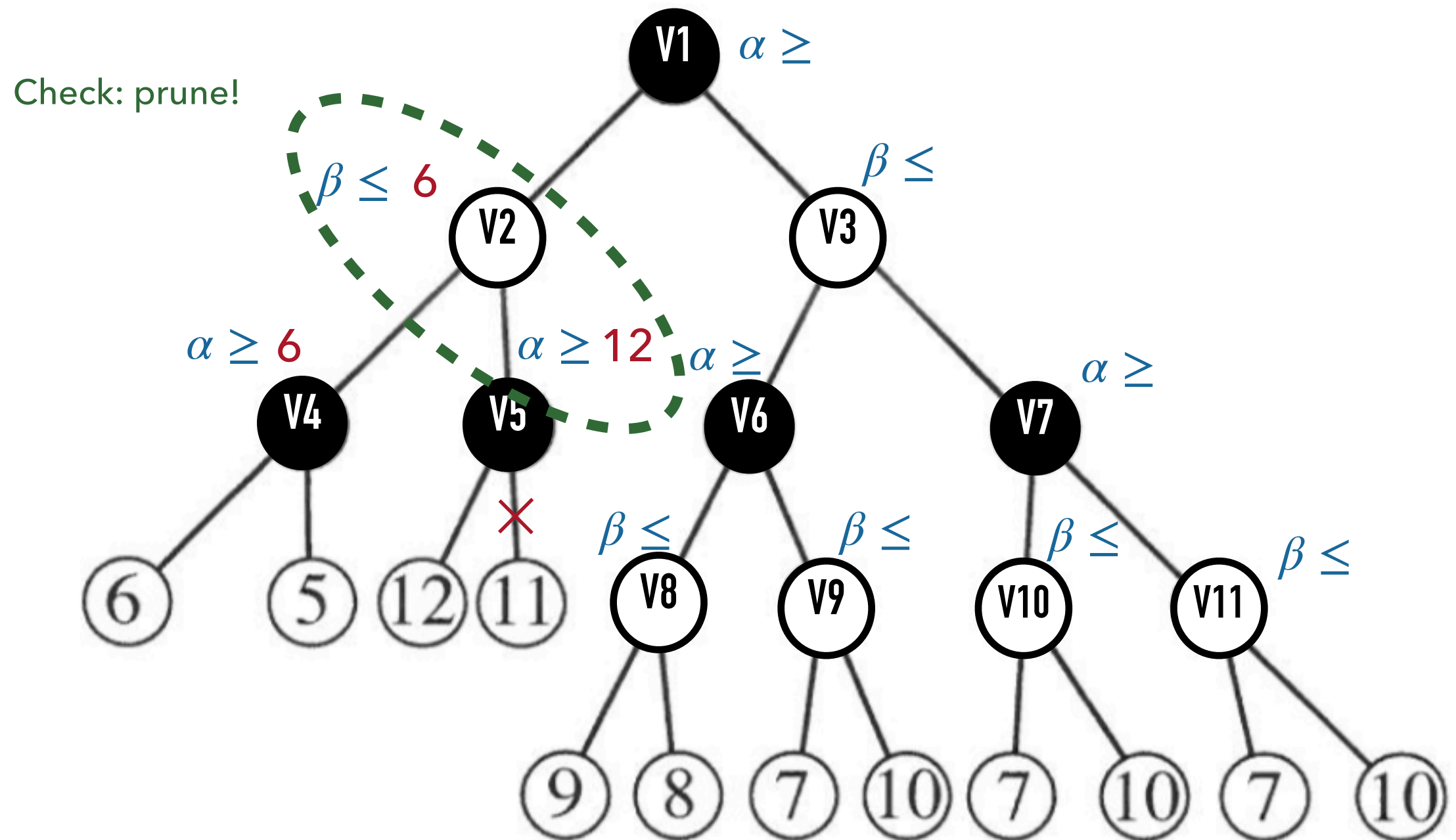
MAX

MIN



PRUNING: If $[child] \beta \leq \alpha [parent]$, then prune child's remaining

MIN



Show which arcs are pruned

EXTRA QUESTION

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

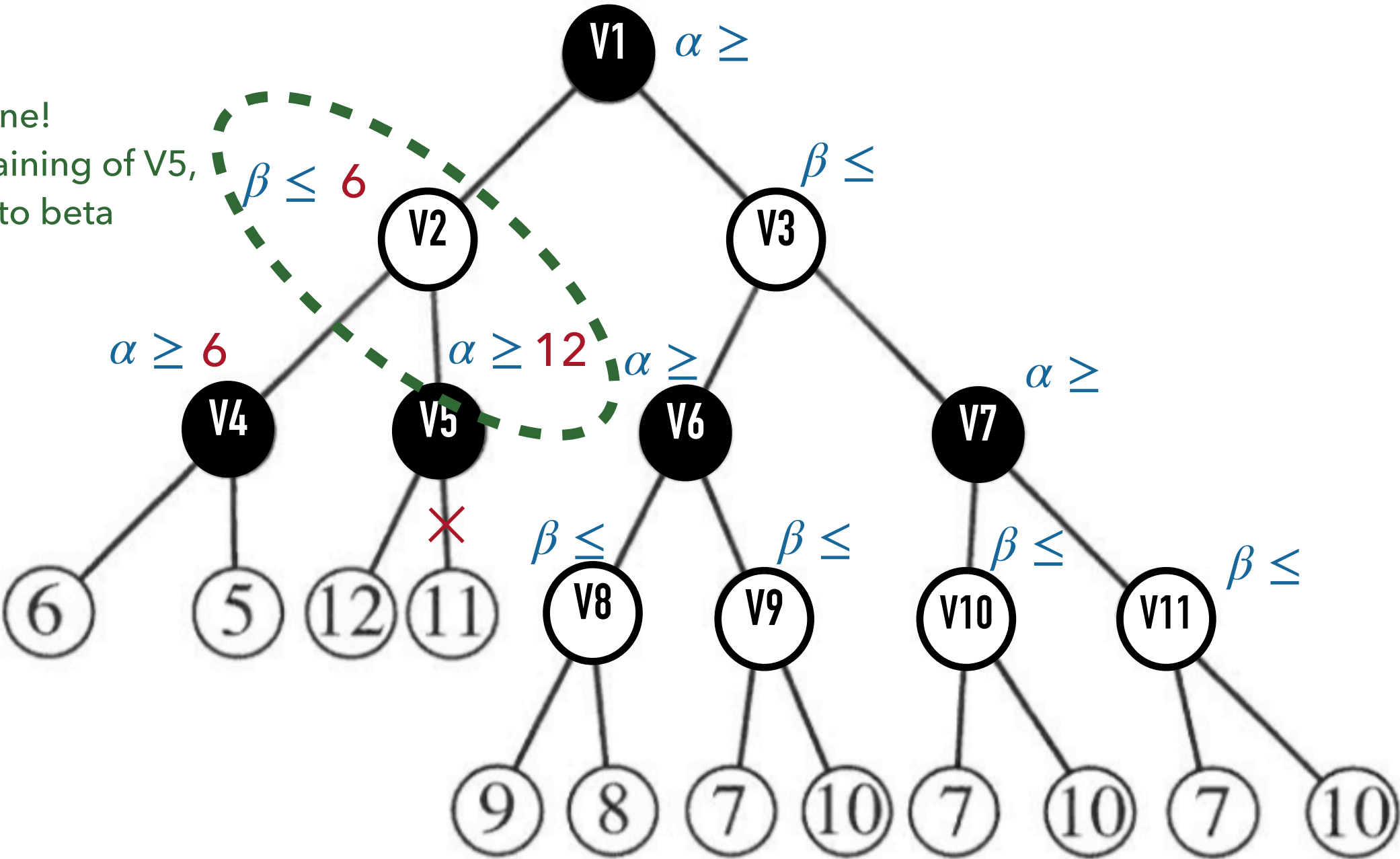
MAX

MIN

MAX

MIN

Check: prune!
prune remaining of V5,
no update to beta

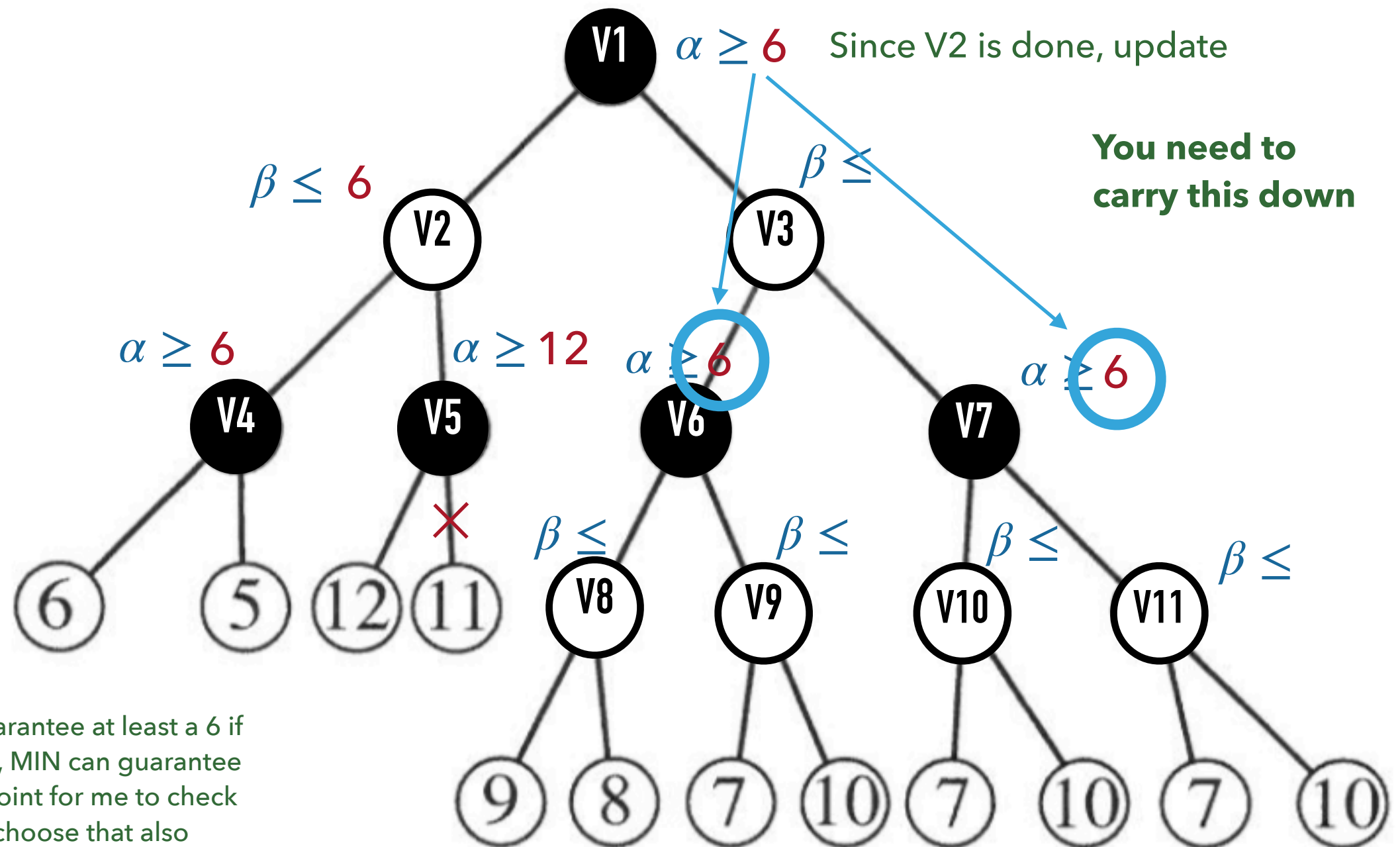


Show which arcs are pruned

PRUNING: If $[child] \beta \leq \alpha [parent]$, then prune child's remaining

MIN

MIN



Because MAX at V1 can guarantee at least a 6 if he goes LEFT. If at V8 or V9, MIN can guarantee at most 6, then there's no point for me to check that MIN node further - V6 choose that also upper bounded by 6, so V6 will not pick that MIN node. Similar argument for V7.

Show which arcs are pruned

EXTRA QUESTION

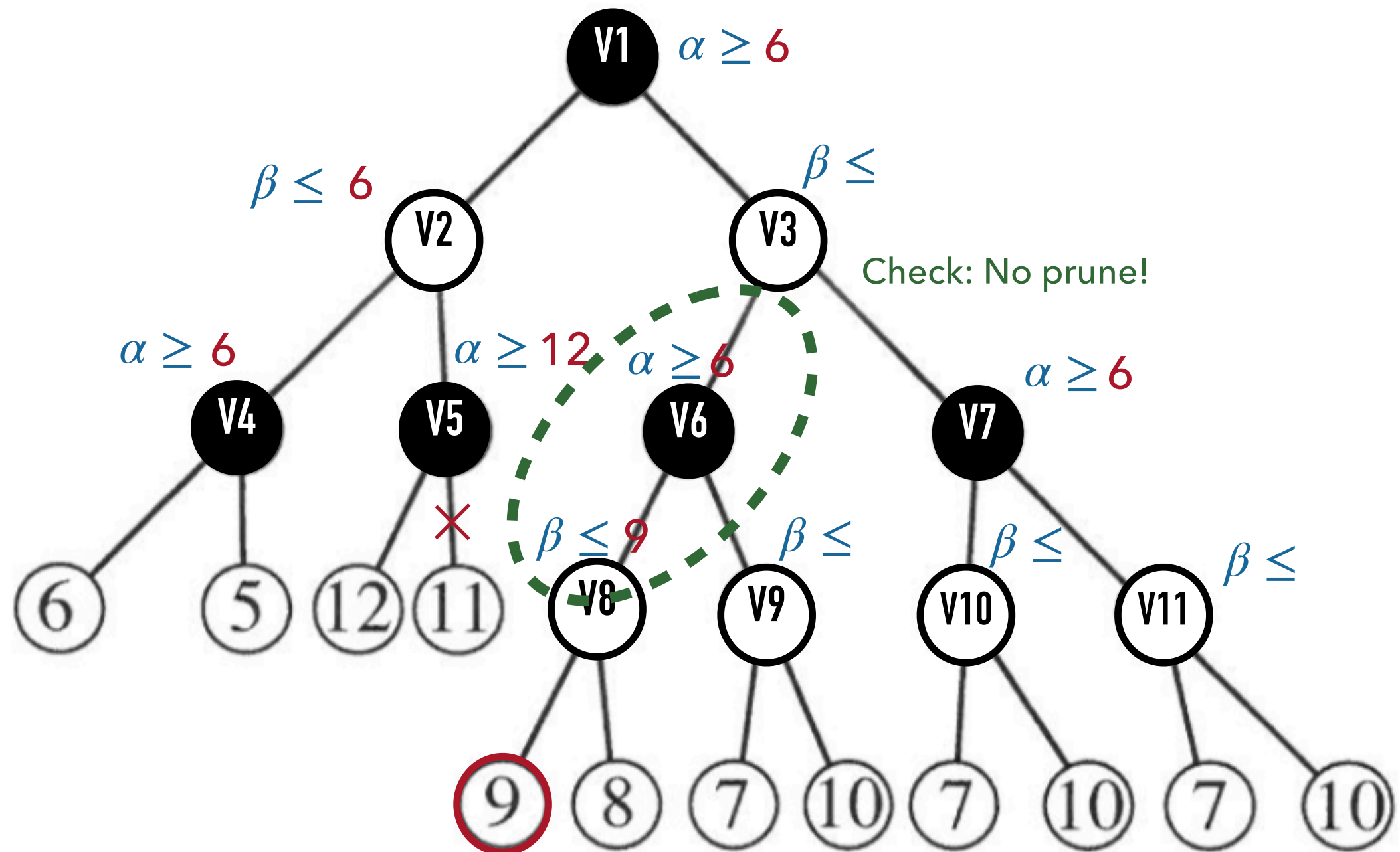
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

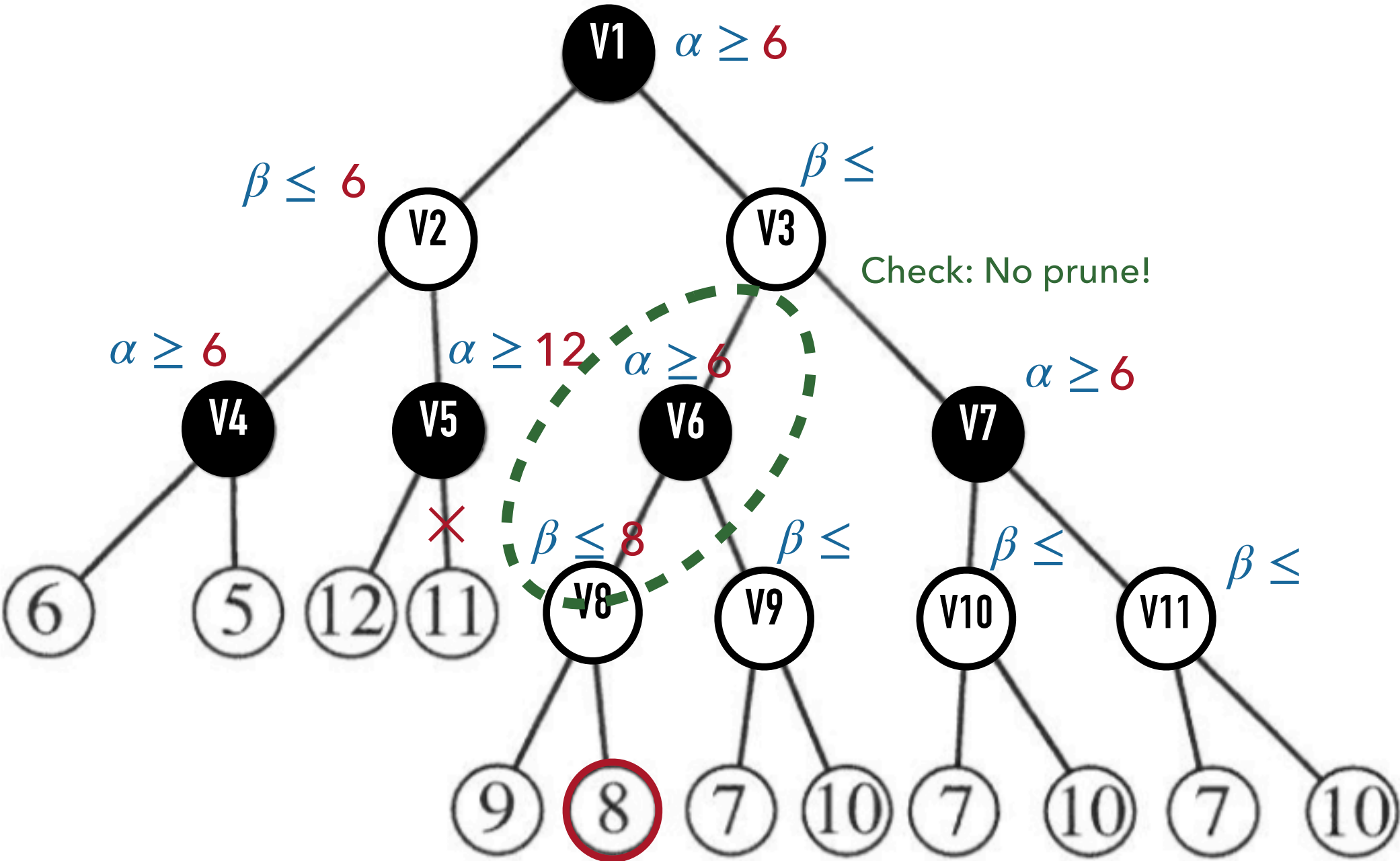
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

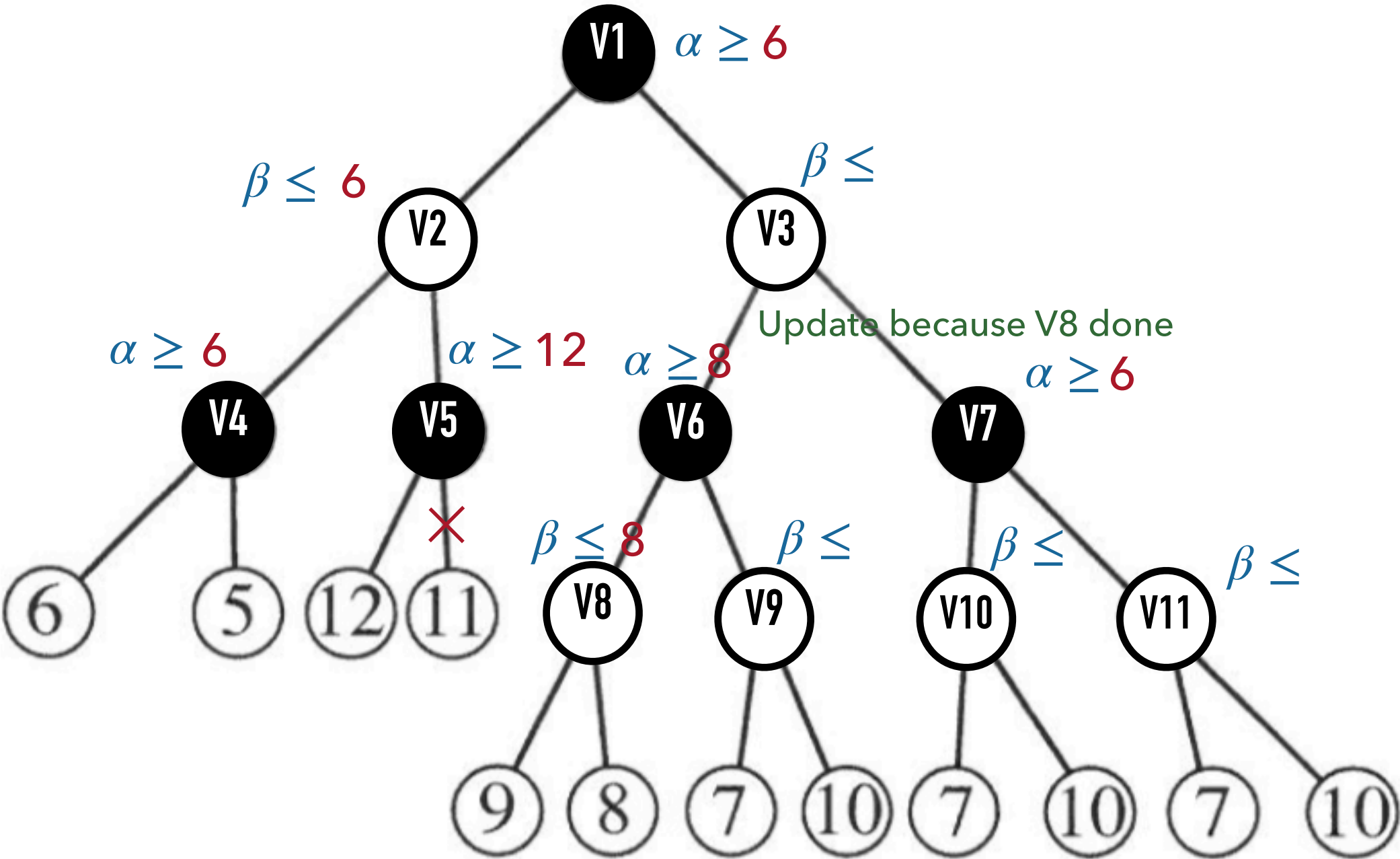
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



EXTRA QUESTION

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

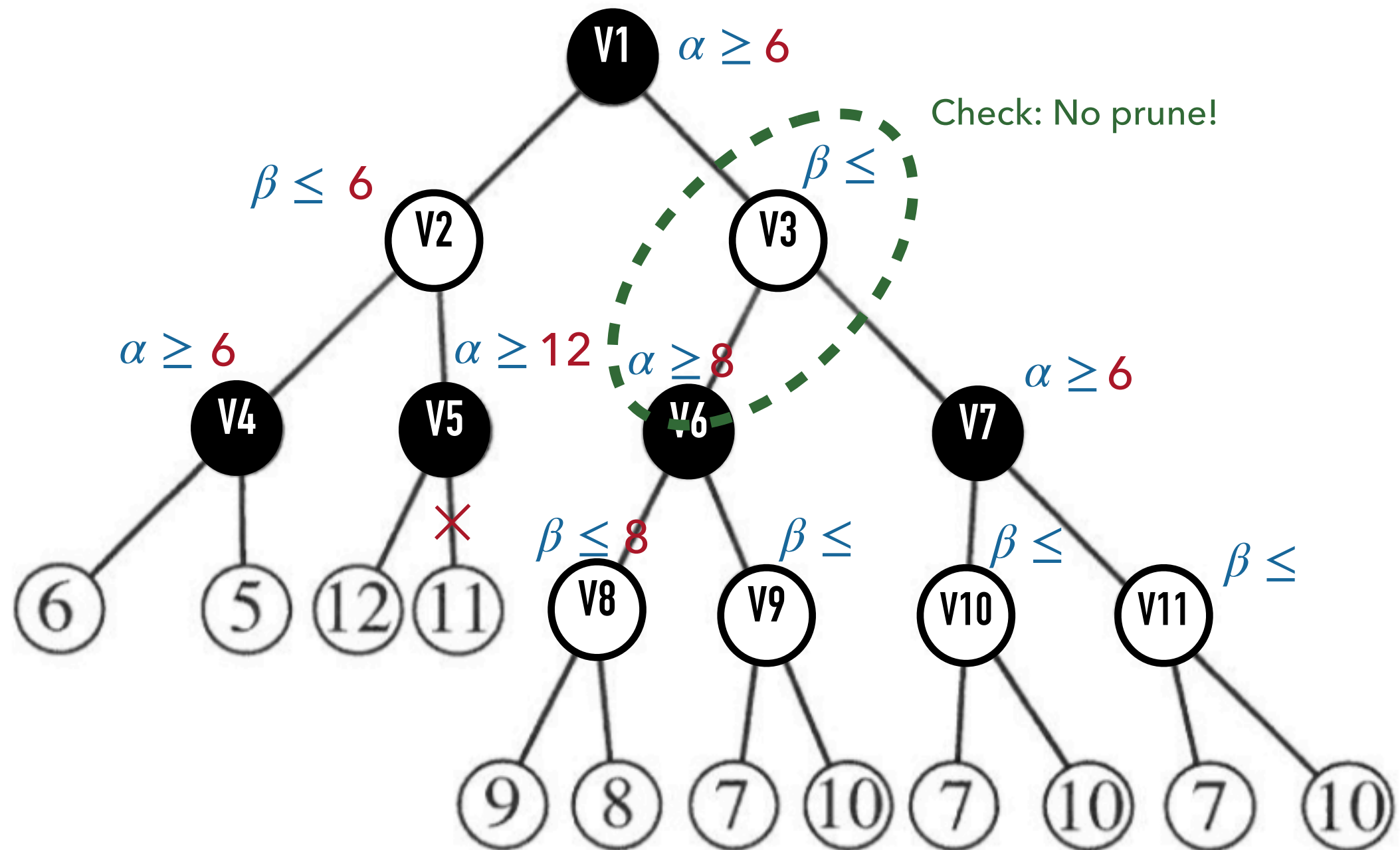
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining

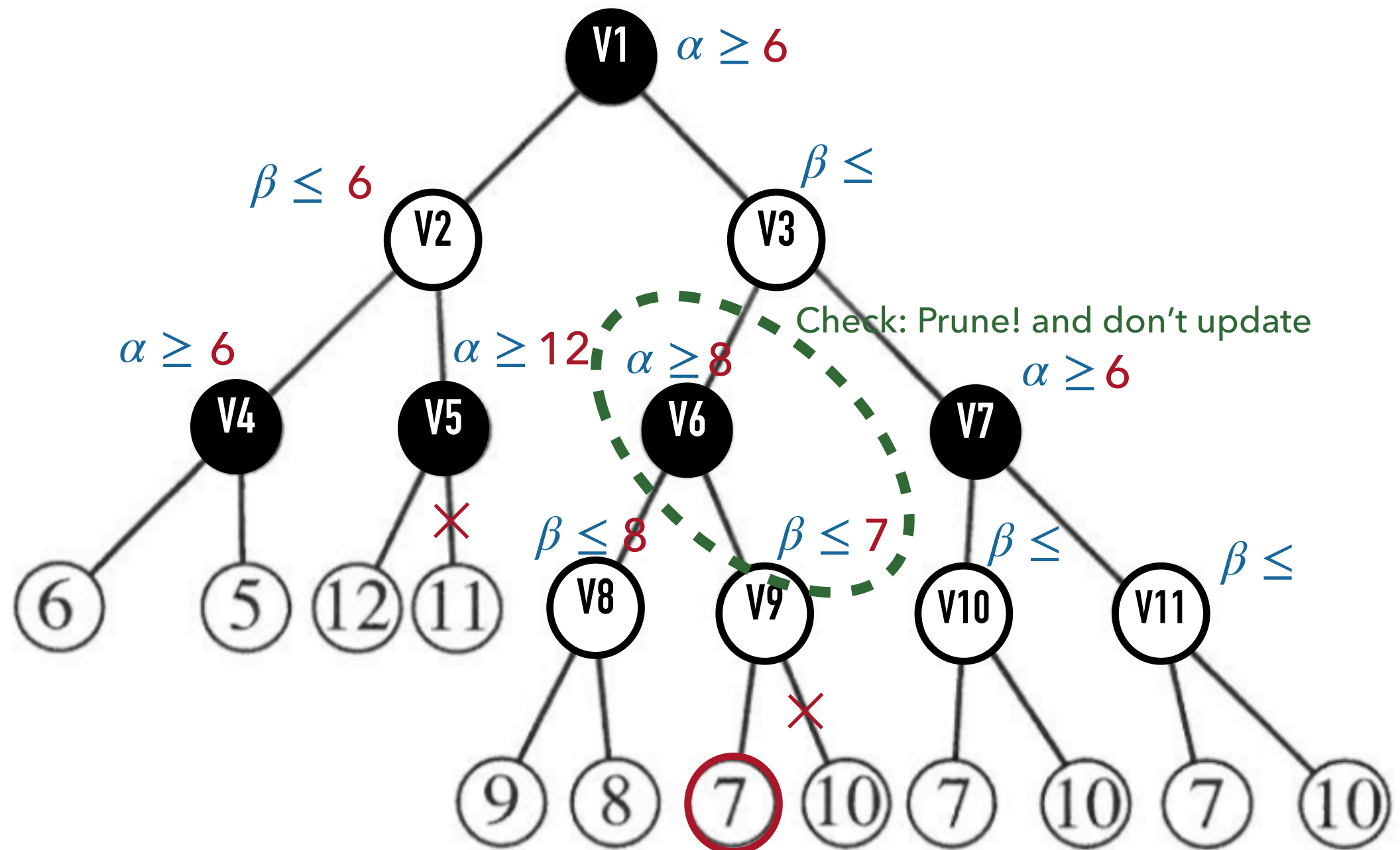
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

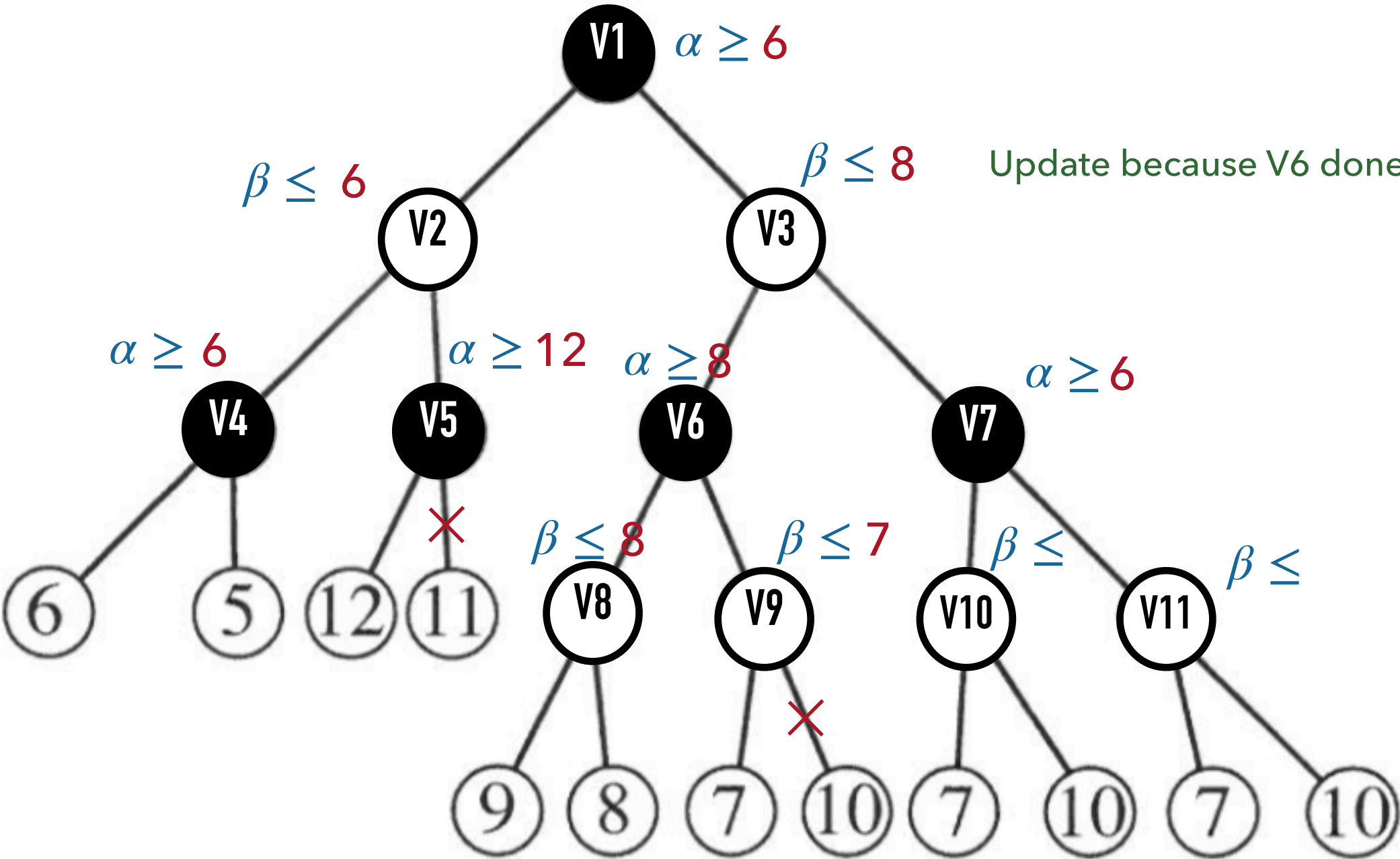
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



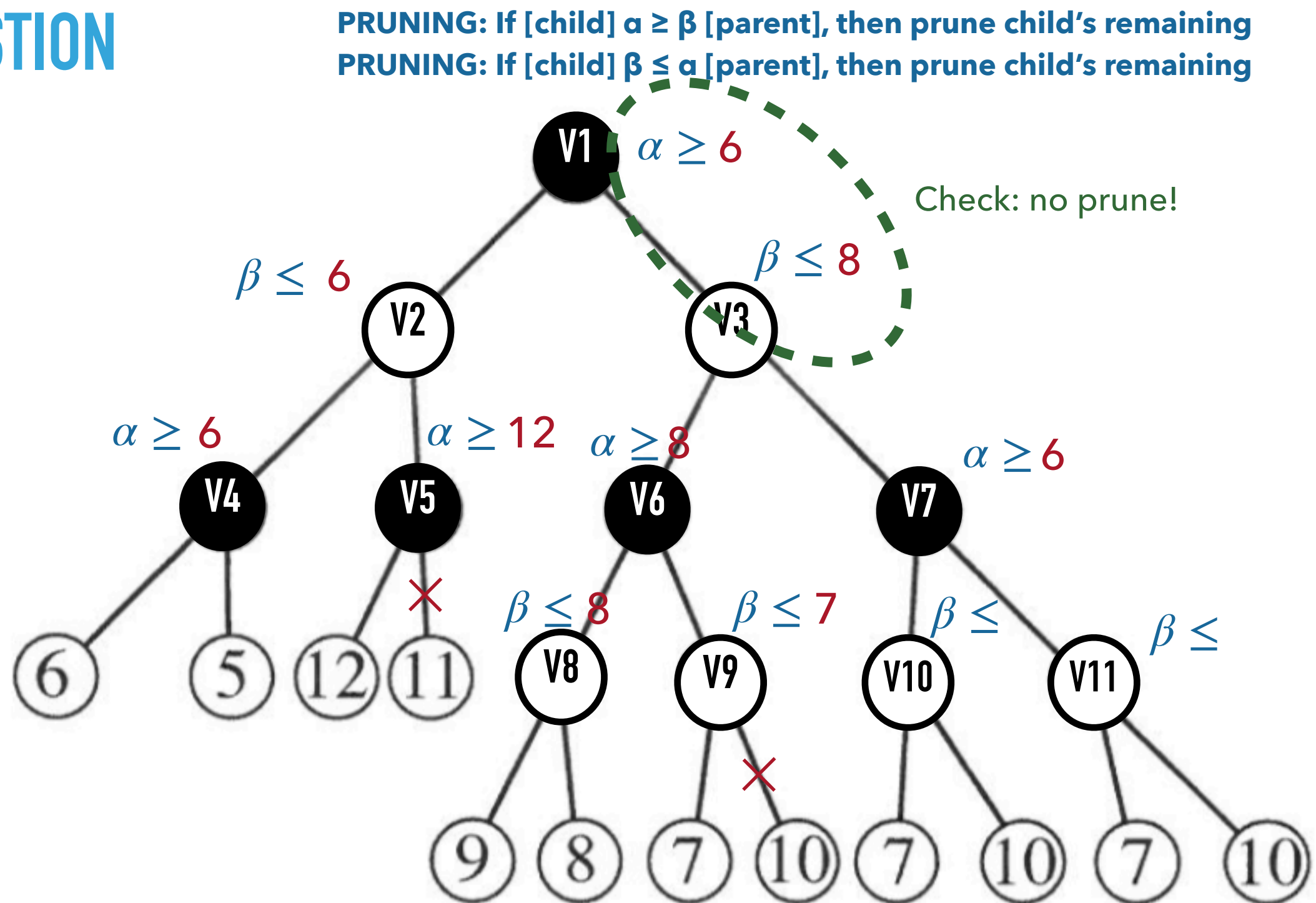
EXTRA QUESTION

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

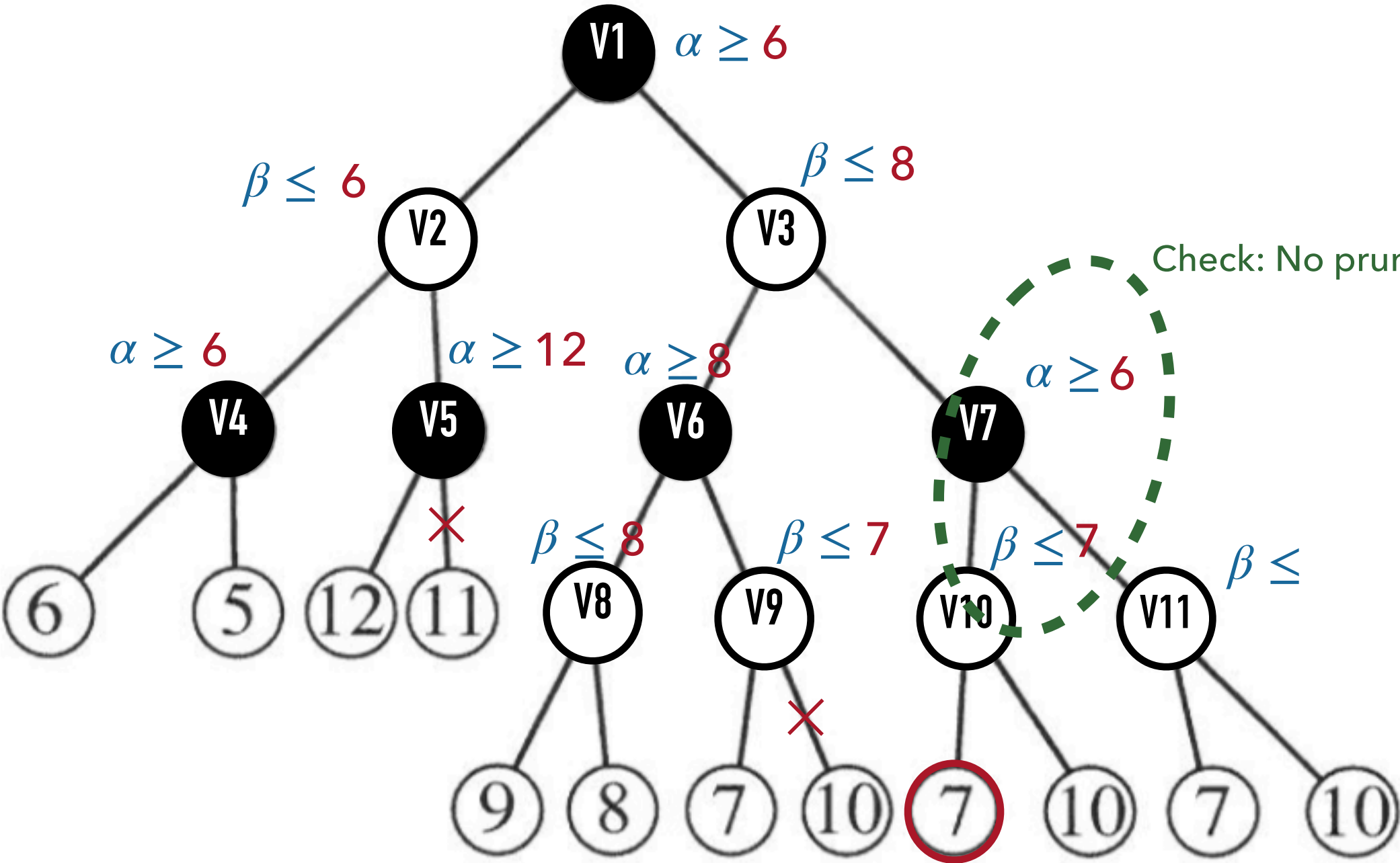
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

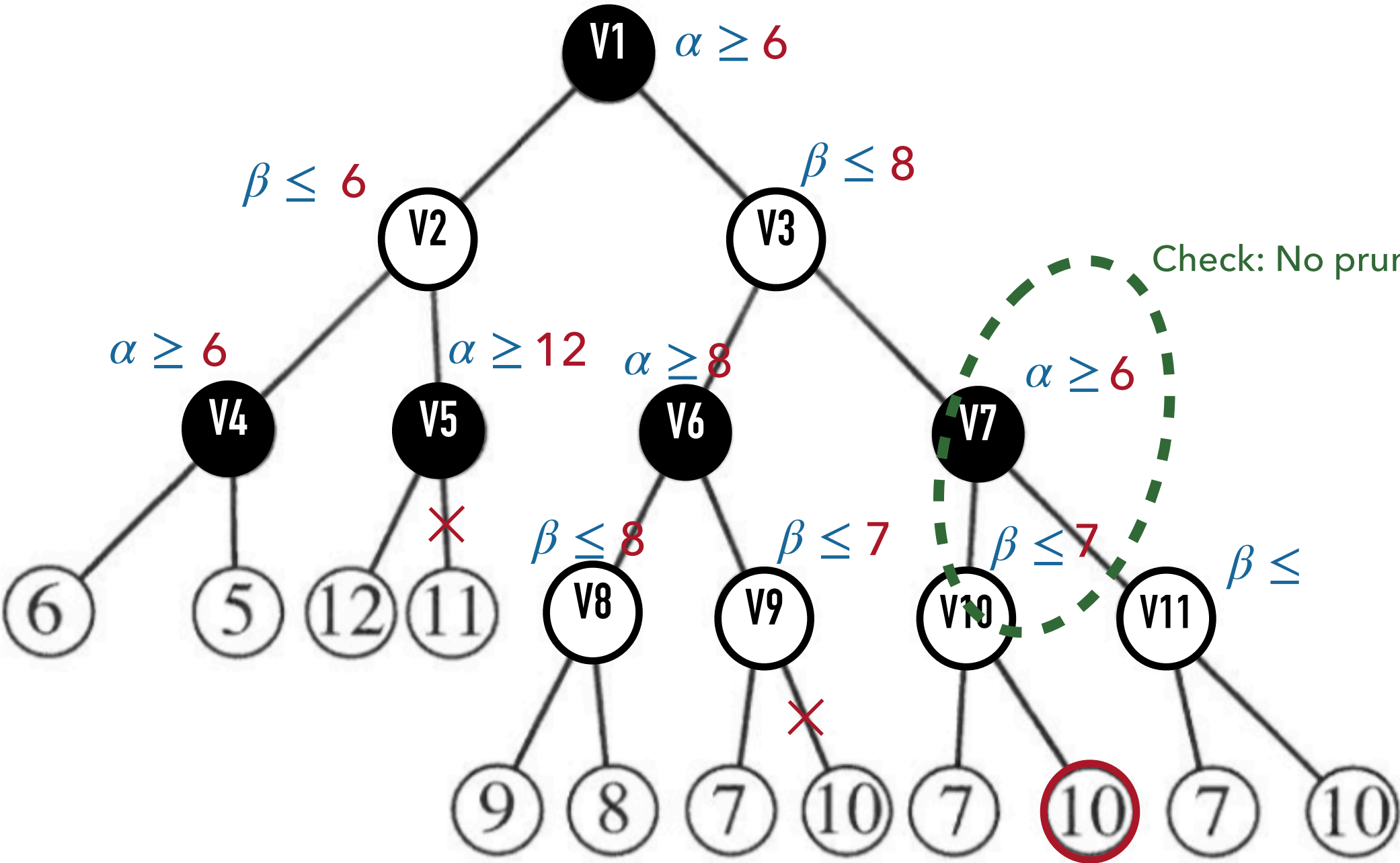
MIN



EXTRA QUESTION

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX
MIN
MAX
MIN



Show which arcs are pruned

EXTRA QUESTION

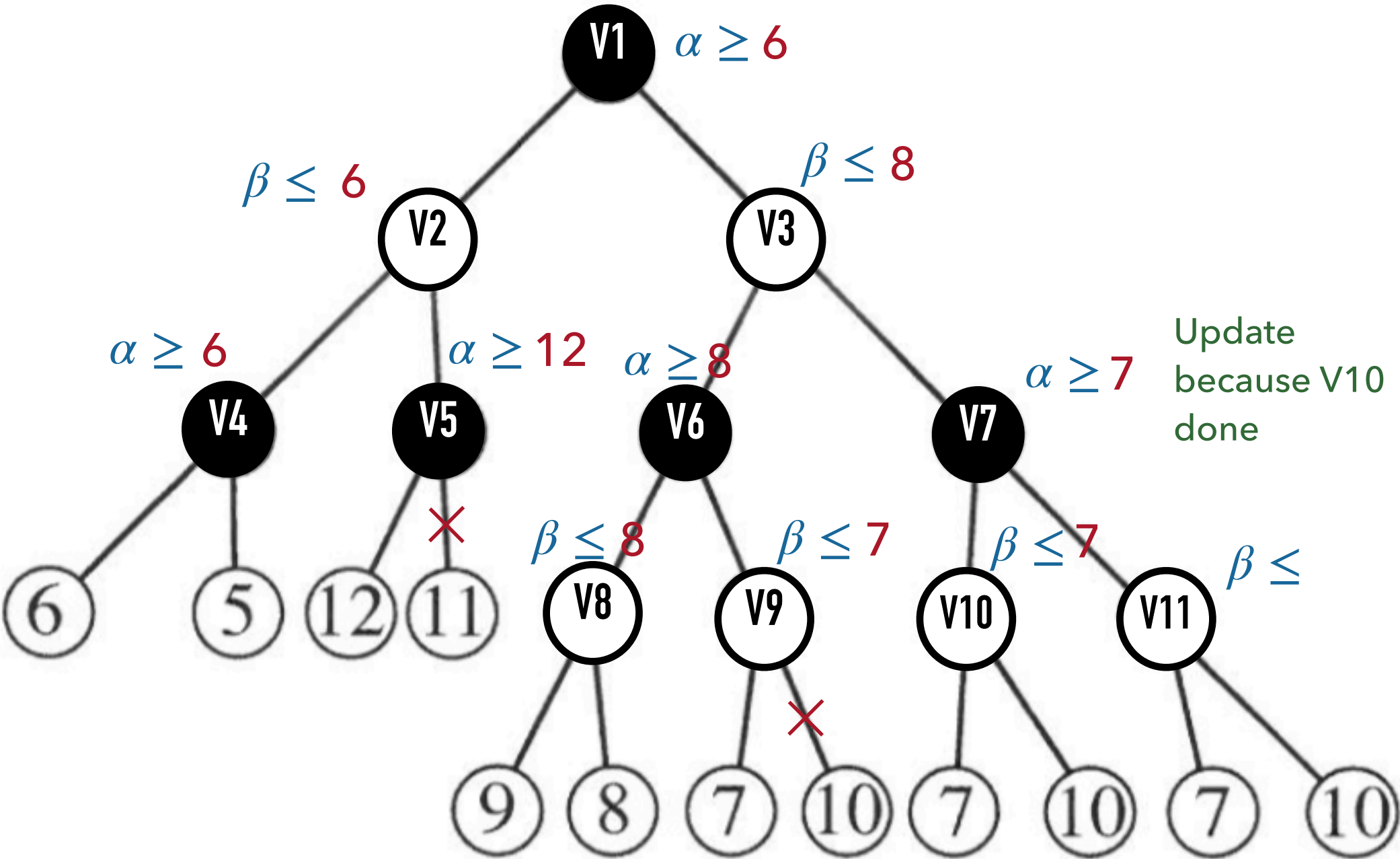
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN

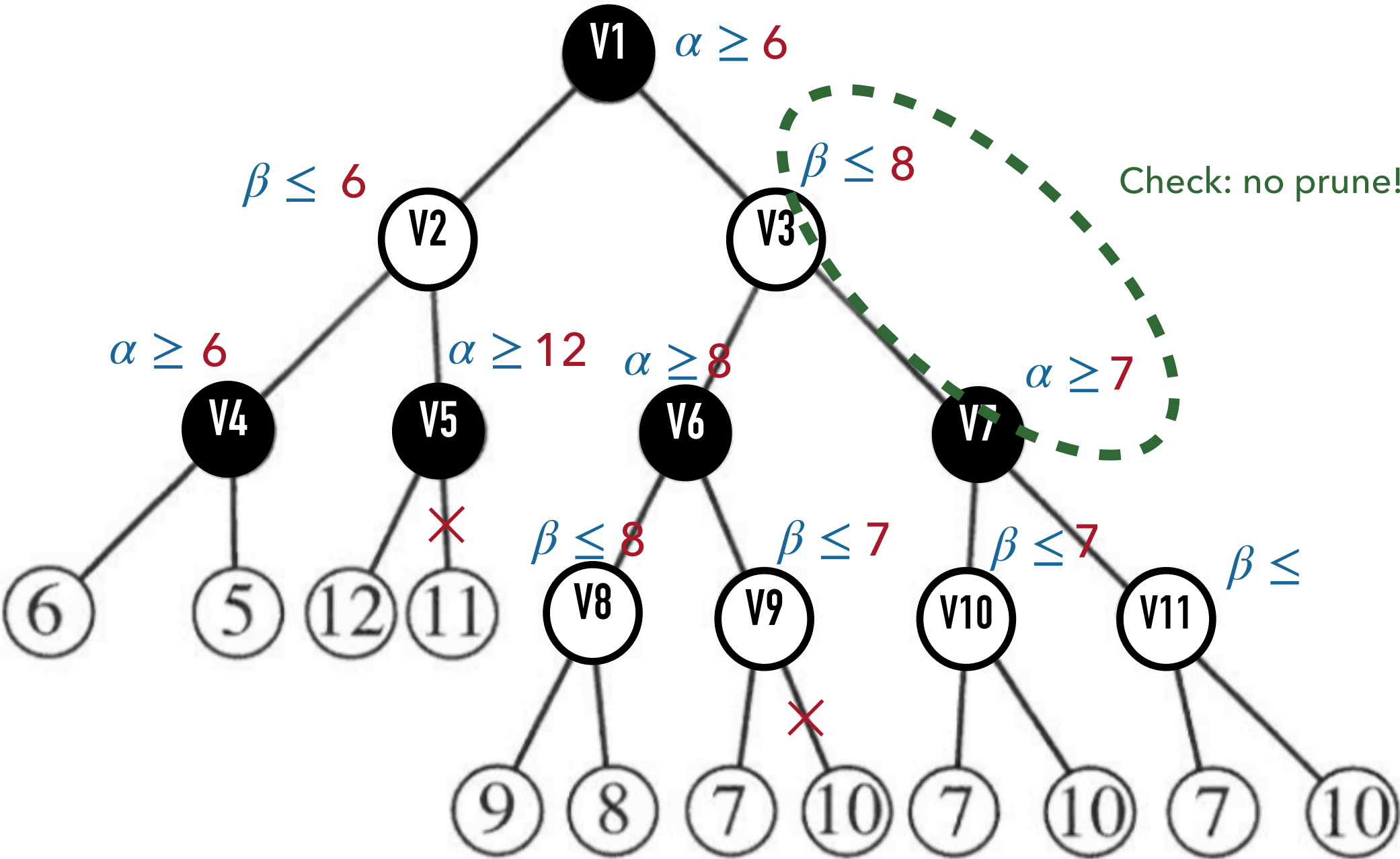


Show which arcs are pruned

EXTRA QUESTION

PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX
MIN
MAX
MIN



Show which arcs are pruned

EXTRA QUESTION

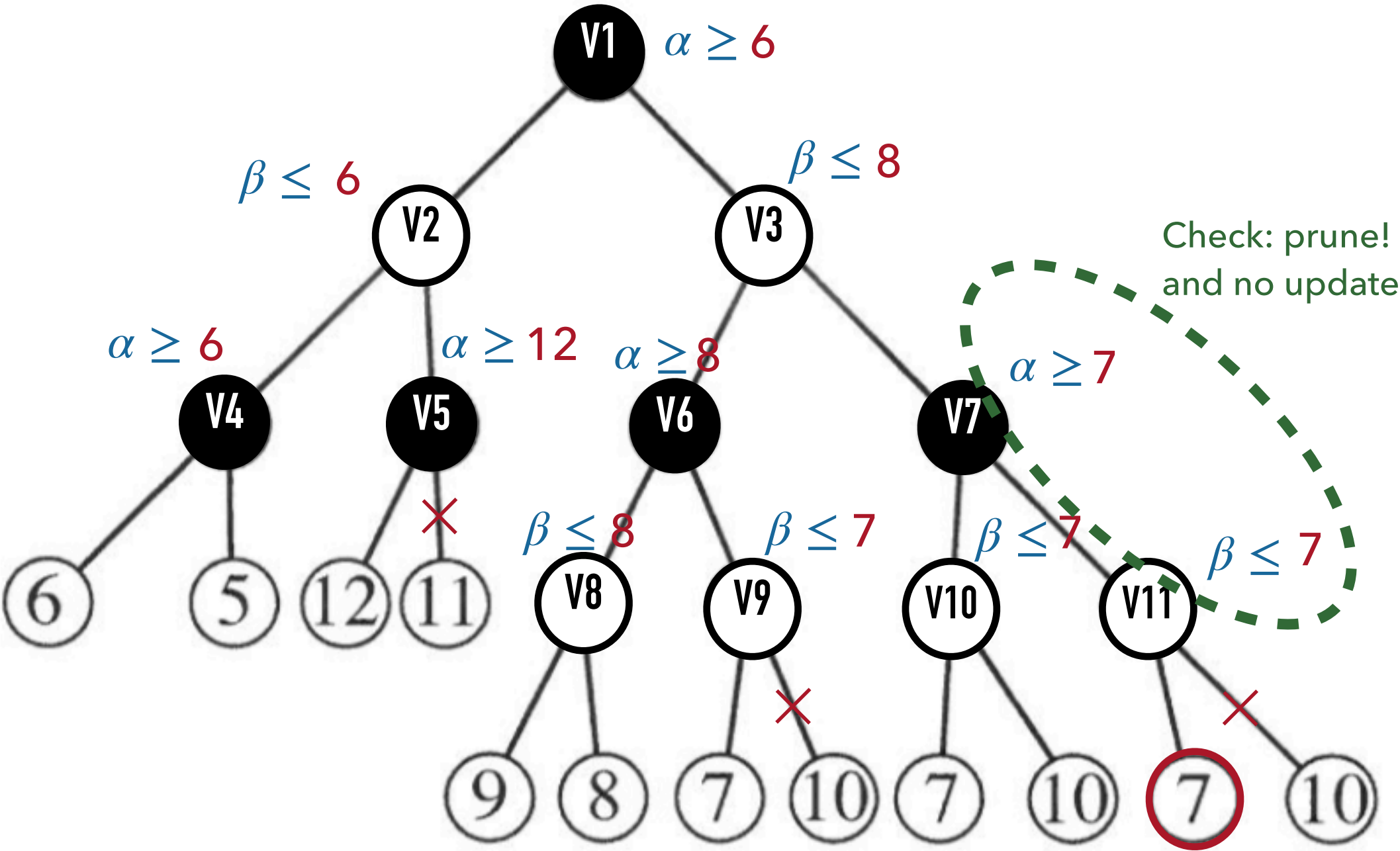
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

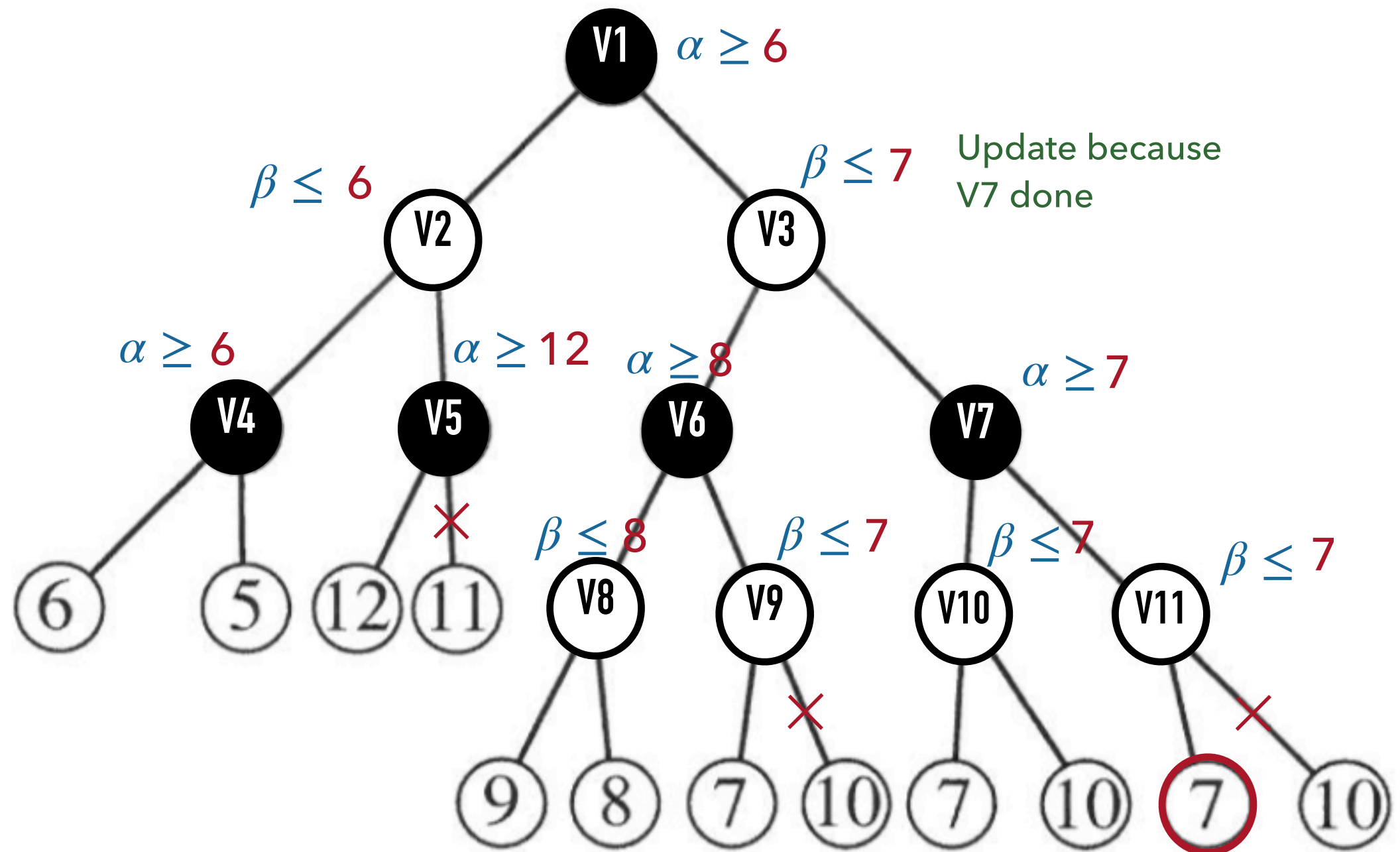
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

EXTRA QUESTION

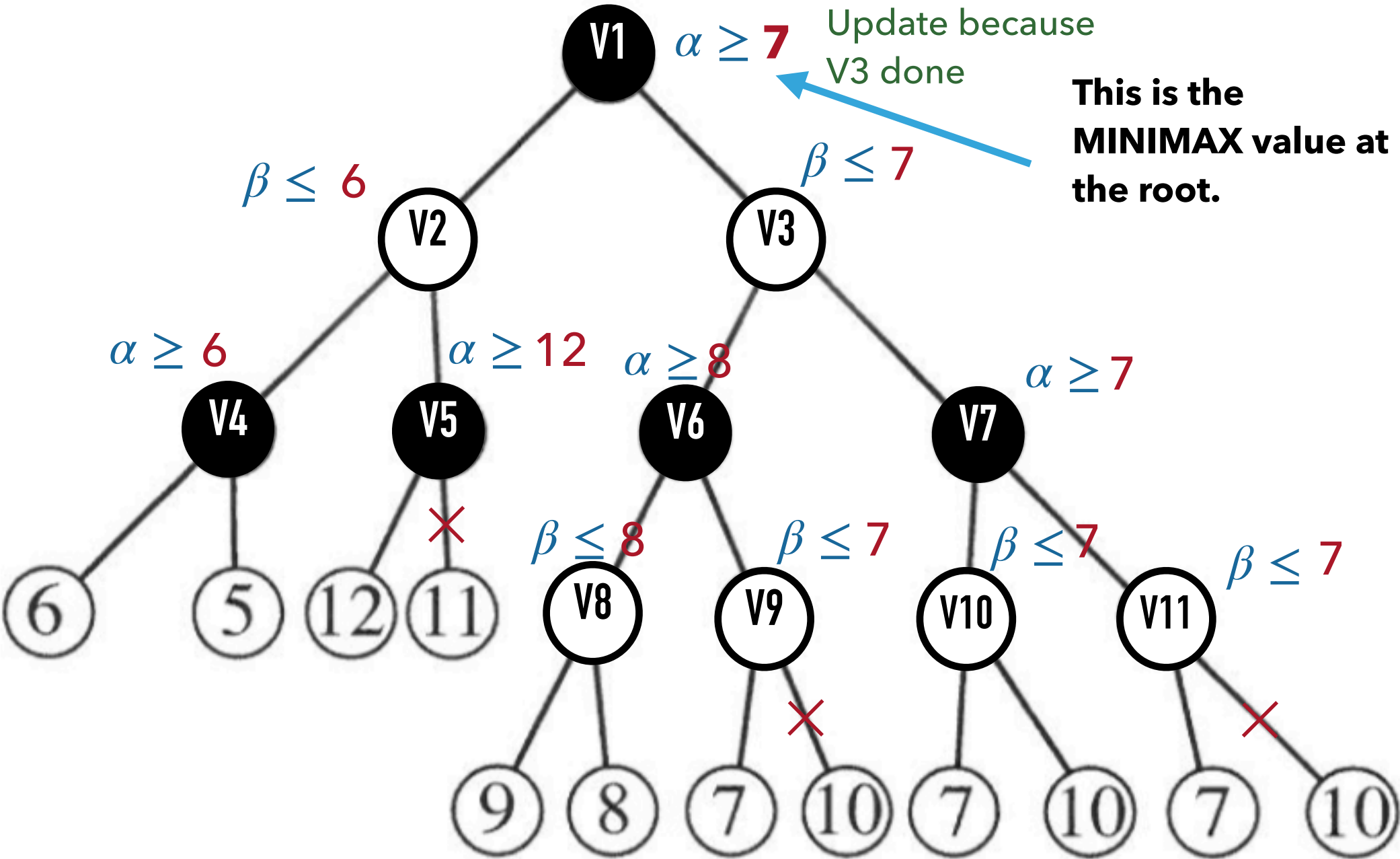
PRUNING: If [child] $\alpha \geq \beta$ [parent], then prune child's remaining
PRUNING: If [child] $\beta \leq \alpha$ [parent], then prune child's remaining

MAX

MIN

MAX

MIN



Show which arcs are pruned

TUTORIAL 5 QUESTION 2

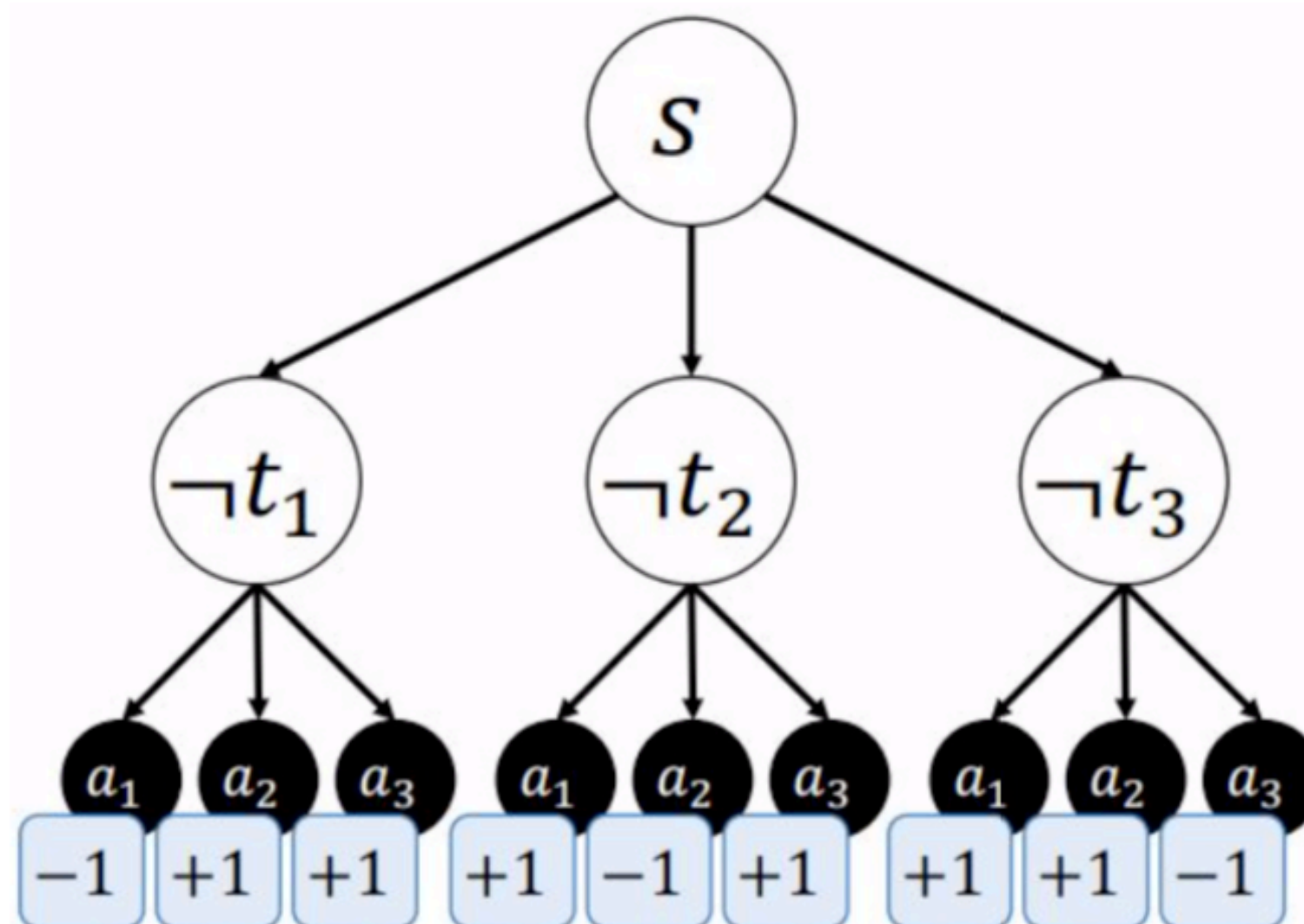
(Stackelberg Security Games)

- ▶ Consider the following game: we have an attacker looking at three targets: t_1 , t_2 and t_3 . A defender must choose which of the two targets it will guard; however, the attacker has an advantage: it can observe what the defender is doing before it chooses its move. If an attacker successfully attacks it receives a payoff of 1 and the defender gets a payoff of -1 .

(a) Model this problem as a minimax search problem. Draw out the search tree. What is the defender's payoff in this game?

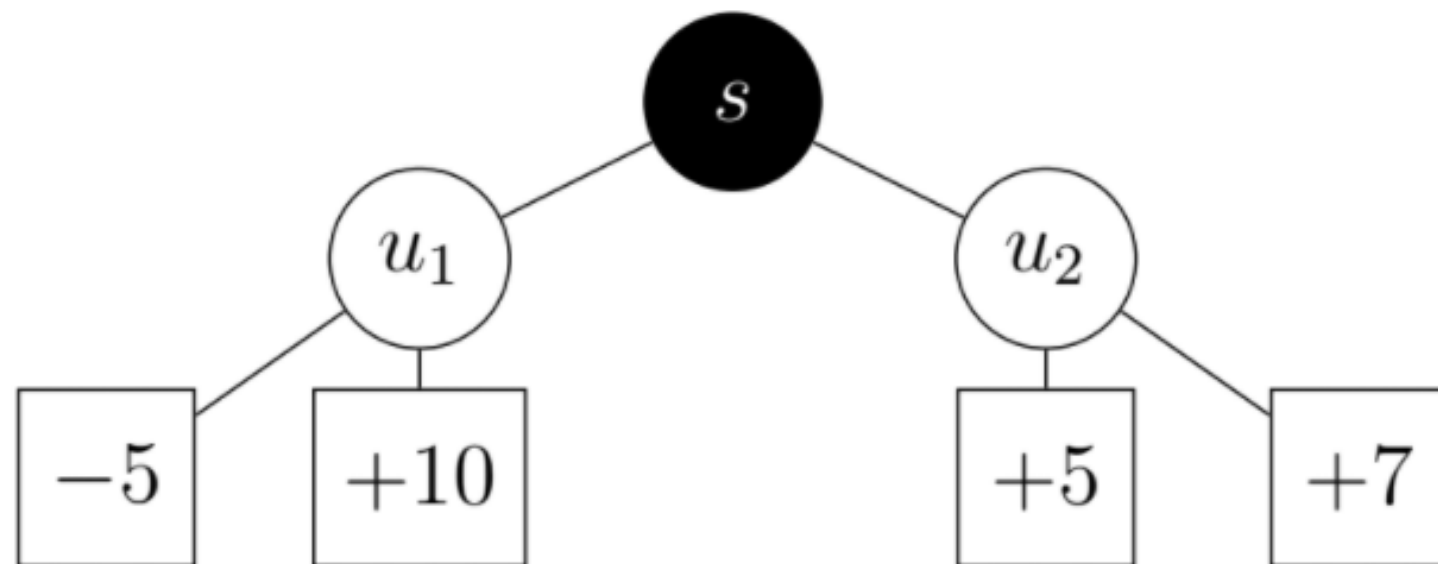
TUTORIAL 5 QUESTION 2

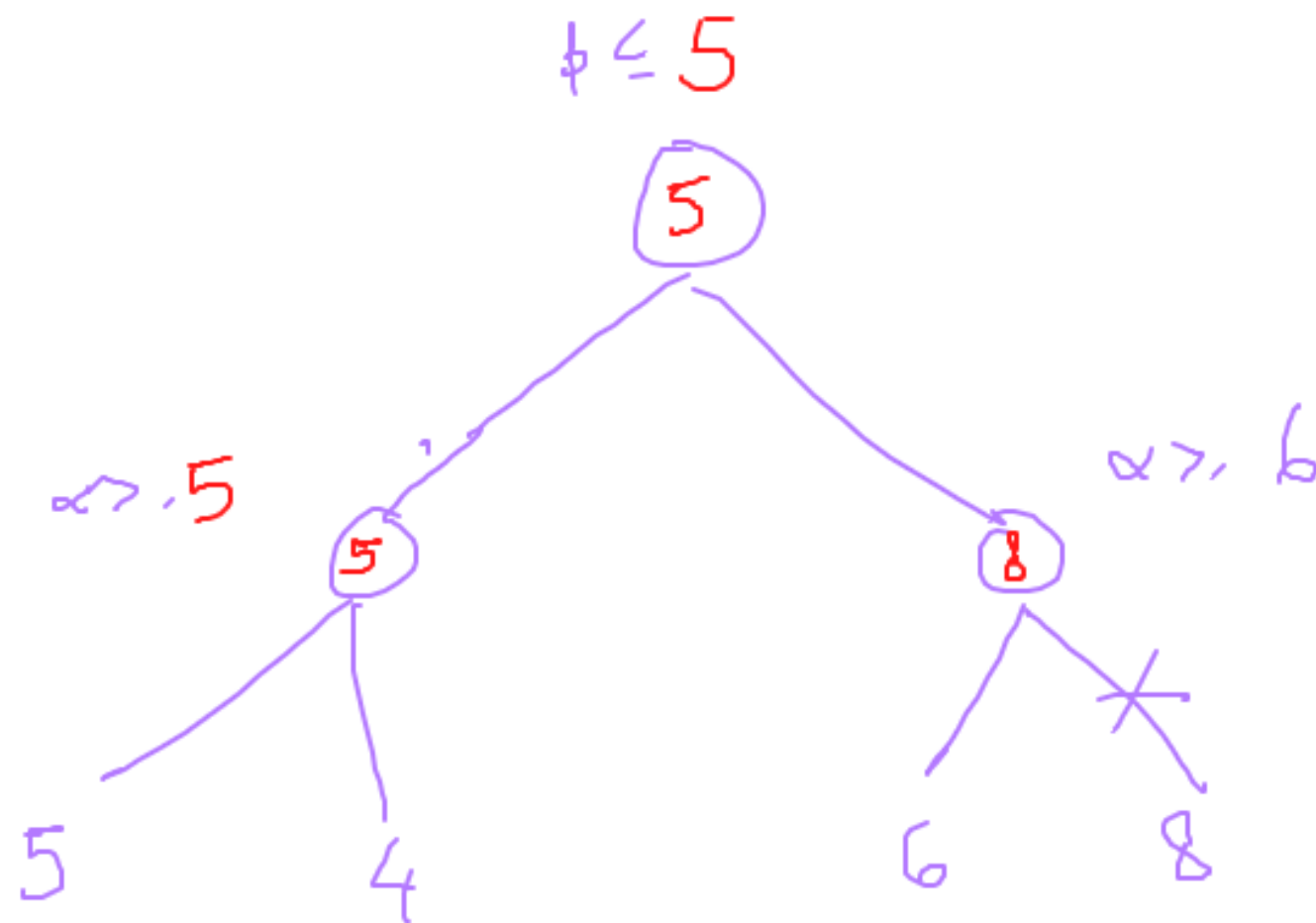
- ▶ **Model this problem as a minimax search problem. Draw out the search tree. What is the defender's payoff in this game?**



TUTORIAL 5 QUESTION 3

- ▶ **Construct an example where, should the MIN player play sub-optimally, the MINIMAX algorithm makes a sub-optimal move.**





MIN

MAX