# 4. Adversarial Search & Pruning

Virginia Tech CS 5804
Introduction to Artificial Intelligence
Spring 2015

# Plan

- Review
- Minimax
- Pruning minimax search

# Review

- Non-adversarial search
- Strategies for ordering frontier
- Heuristics
- Completeness, optimality, time complexity, space complexity

# Game Representation

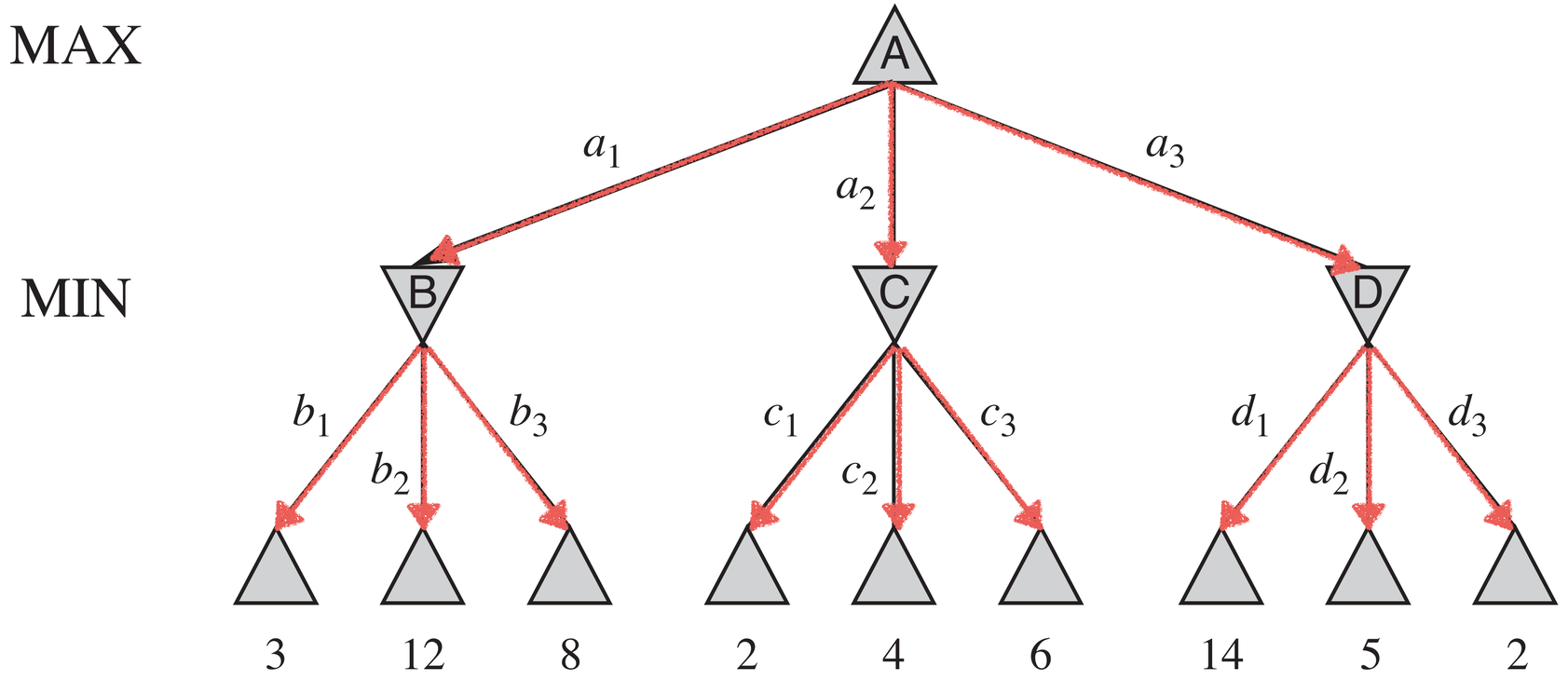
- **zero-sum** games of **perfect information**
- Two players: MAX vs MIN
- Players alternate actions: state space transitions

# Representation Elements

- PLAYER(*s*): which player chooses the action in state *s*
- ACTIONS(*s*): what actions are available from state *s*
- RESULT(*s*,*a*): the state that results from action a in state *s*
- TERMINAL-TEST(*s*): whether state *s* is a terminal state
- UTILITY(*s*,*p*): the value of state *s*, usually only if terminal

# Minimax Strategy

- Choose best move assuming opponent plays **optimally**
  - i.e., opponent also uses minimax
- MINIMAX(*s*) =
  if TERMINAL-TEST(*s*) then UTILITY(*s*)
  if PLAYER(*s*) = MAX then
  max of MINIMAX(RESULT(*s*,*a*)) for *a* in ACTIONS(s)
  if PLAYER(s) = MIN then
  min of MINIMAX(RESULT(*s*,*a*)) for *a* in ACTIONS(s)




- MINIMAX(*s*) =
  if TERMINAL-TEST(*s*) then UTILITY(*s*)
  if PLAYER(*s*) = MAX then
    max of MINIMAX(RESULT(*s*,*a*)) for *a* in ACTIONS(s)
  if PLAYER(s) = MIN then
    min of MINIMAX(RESULT(*s*,*a*)) for *a* in ACTIONS(s)

**function** MINIMAX-DECISION($state$) **returns** *an action*
  **return** $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT($state, a$))

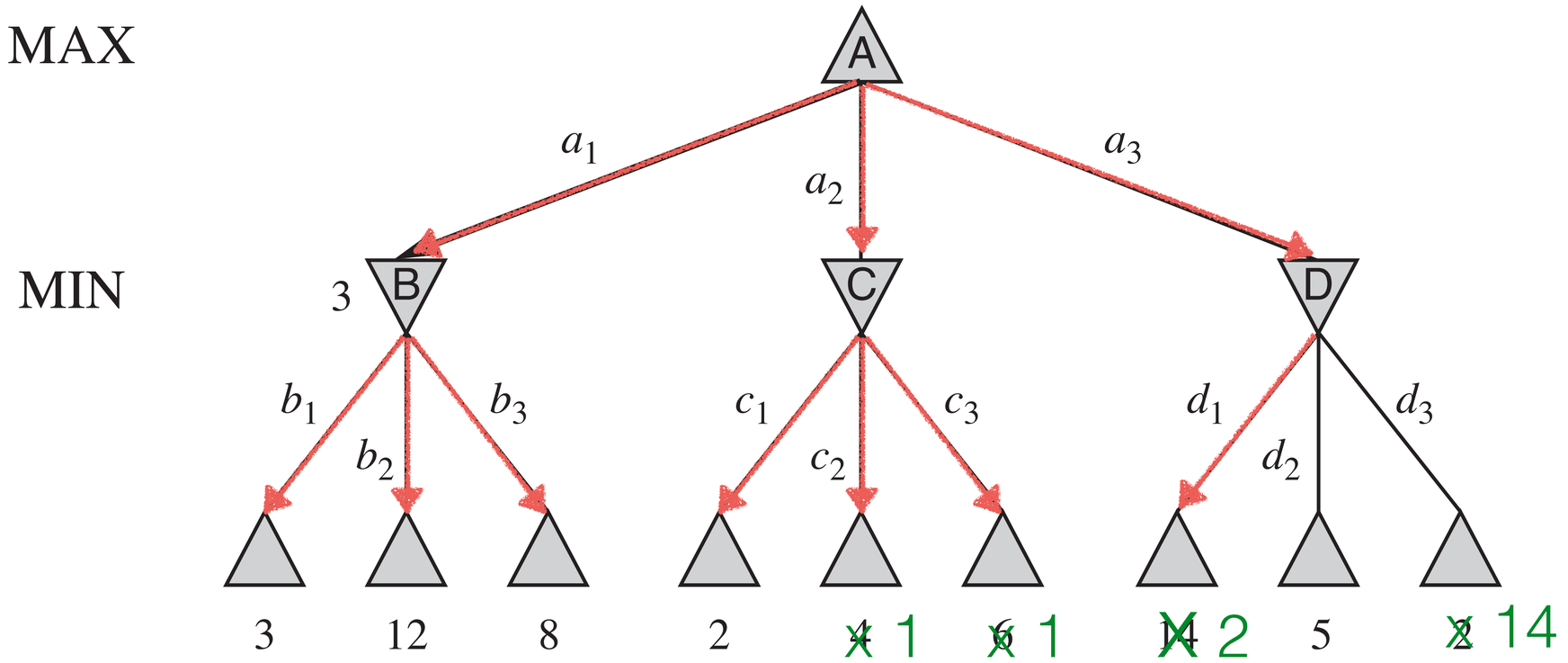
---

**function** MAX-VALUE($state$) **returns** *a utility value*
  **if** TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  **for each** $a$ **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT($s$, $a$)))
  **return** $v$

---

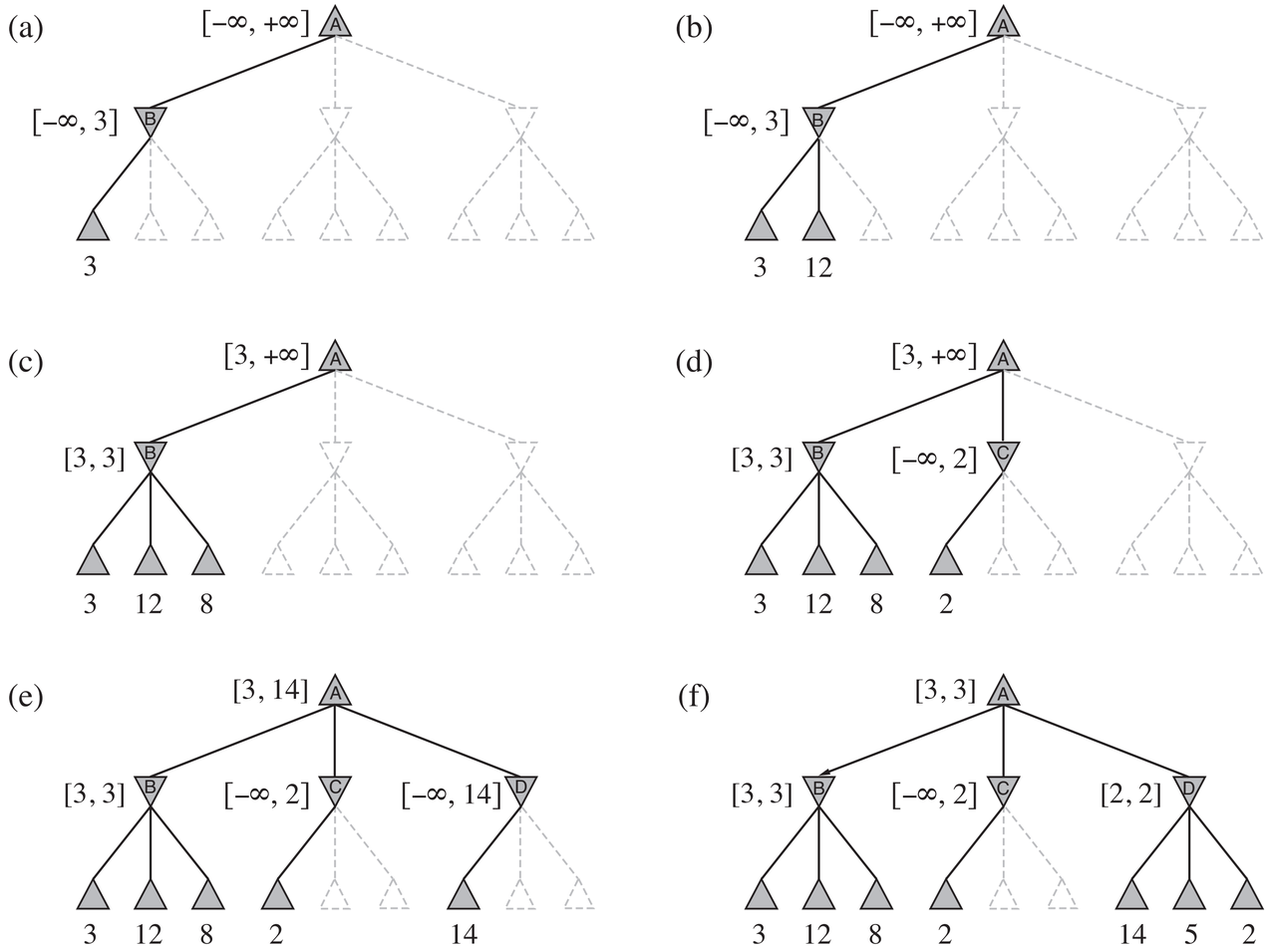
**function** MIN-VALUE($state$) **returns** *a utility value*
  **if** TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow \infty$
  **for each** $a$ **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT($s$, $a$)))
  **return** $v$

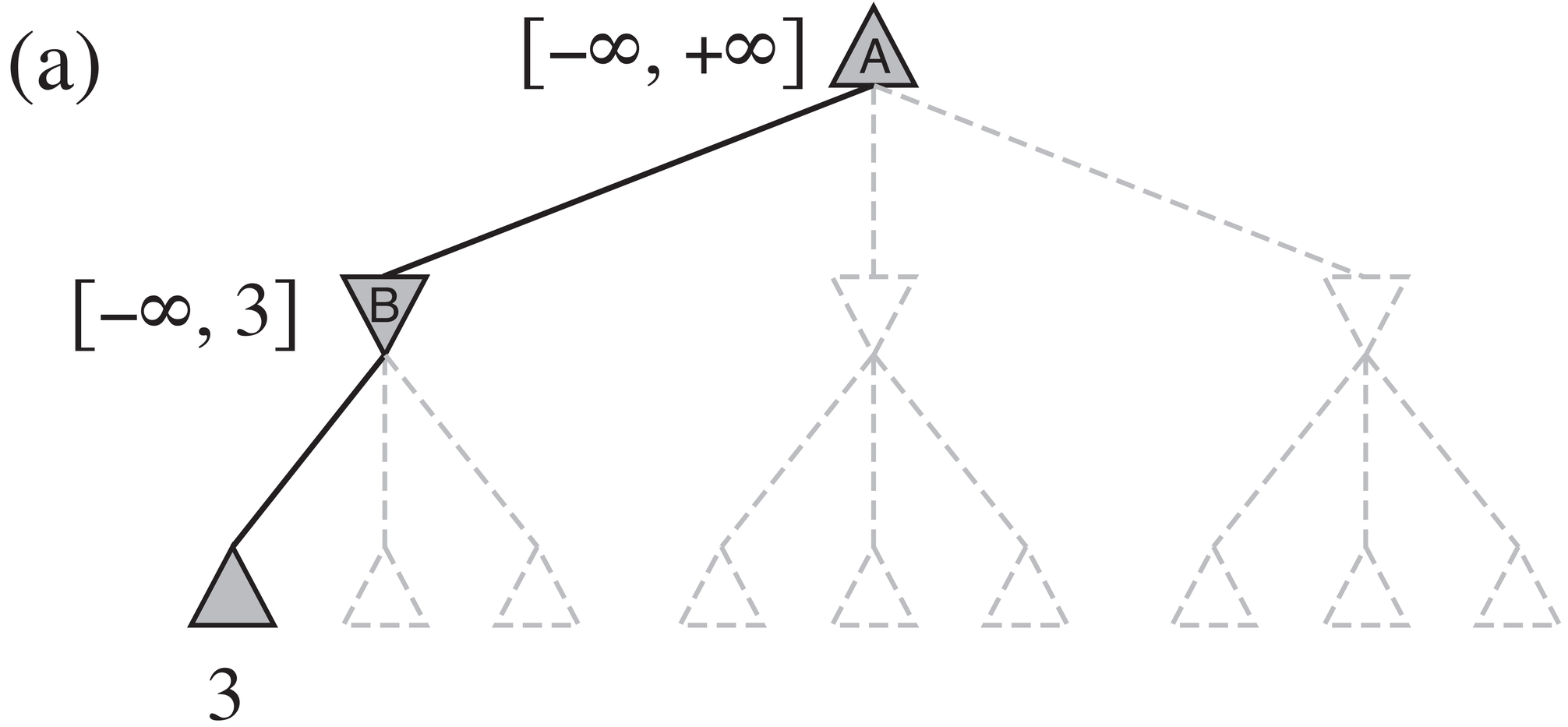
# Pruning

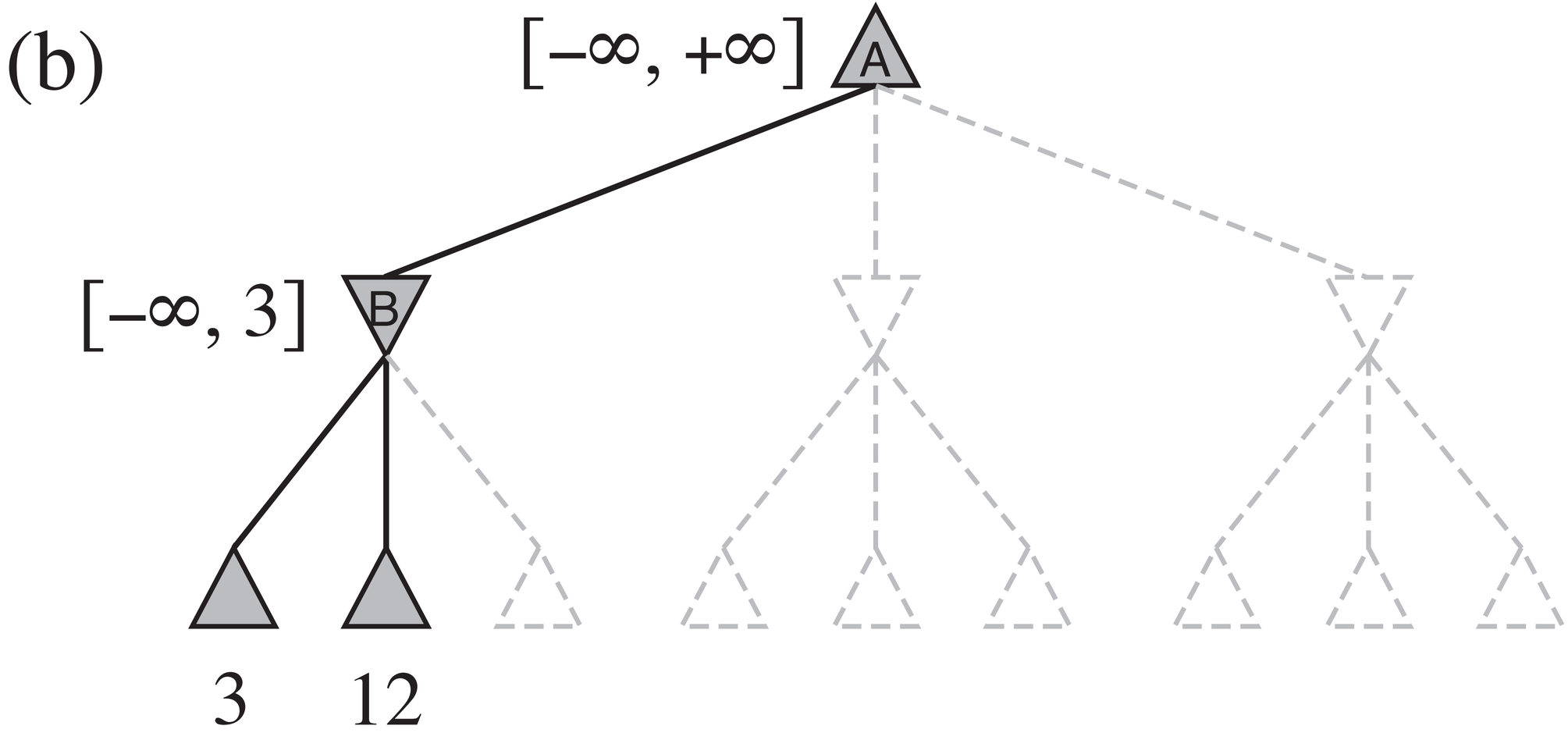

MAX
MIN
A
B
C
D
3
$a_1$
$a_2$
$a_3$
$b_1$
$b_2$
$b_3$
$c_1$
$c_2$
$c_3$
$d_1$
$d_2$
$d_3$
3
12
8
2
x 1
x 1
x 2
5
x 14

# Alpha-Beta Pruning

- [alpha, beta]
  alpha = highest-value choice along path
  beta = lowest-value choice along path

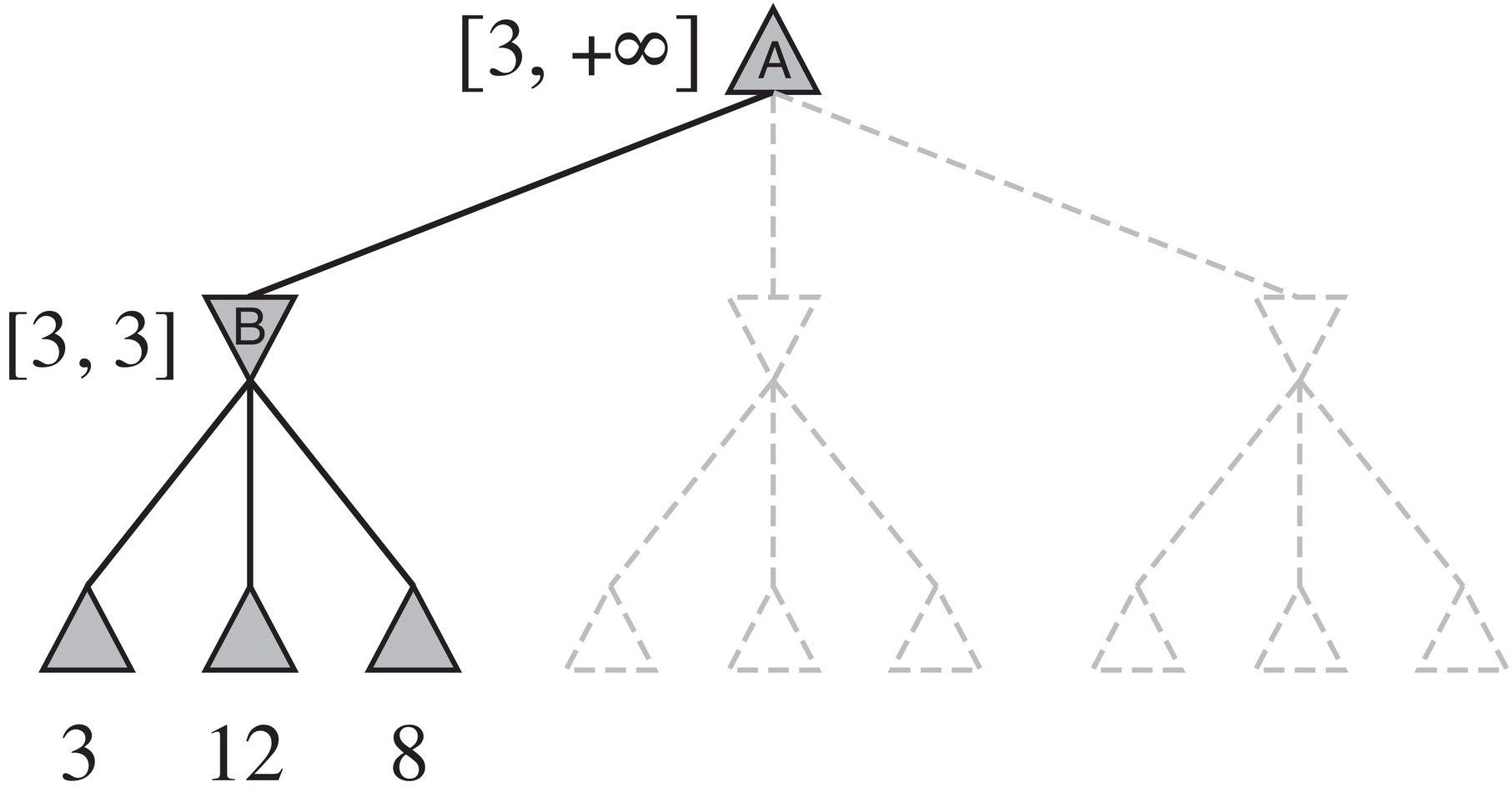



(a)
[−∞, +∞]
A
[−∞, 3]
B
3


(b)
[−∞, +∞]
A
[−∞, 3]
B
3
12


(d)
[3, +∞]
A

(c)



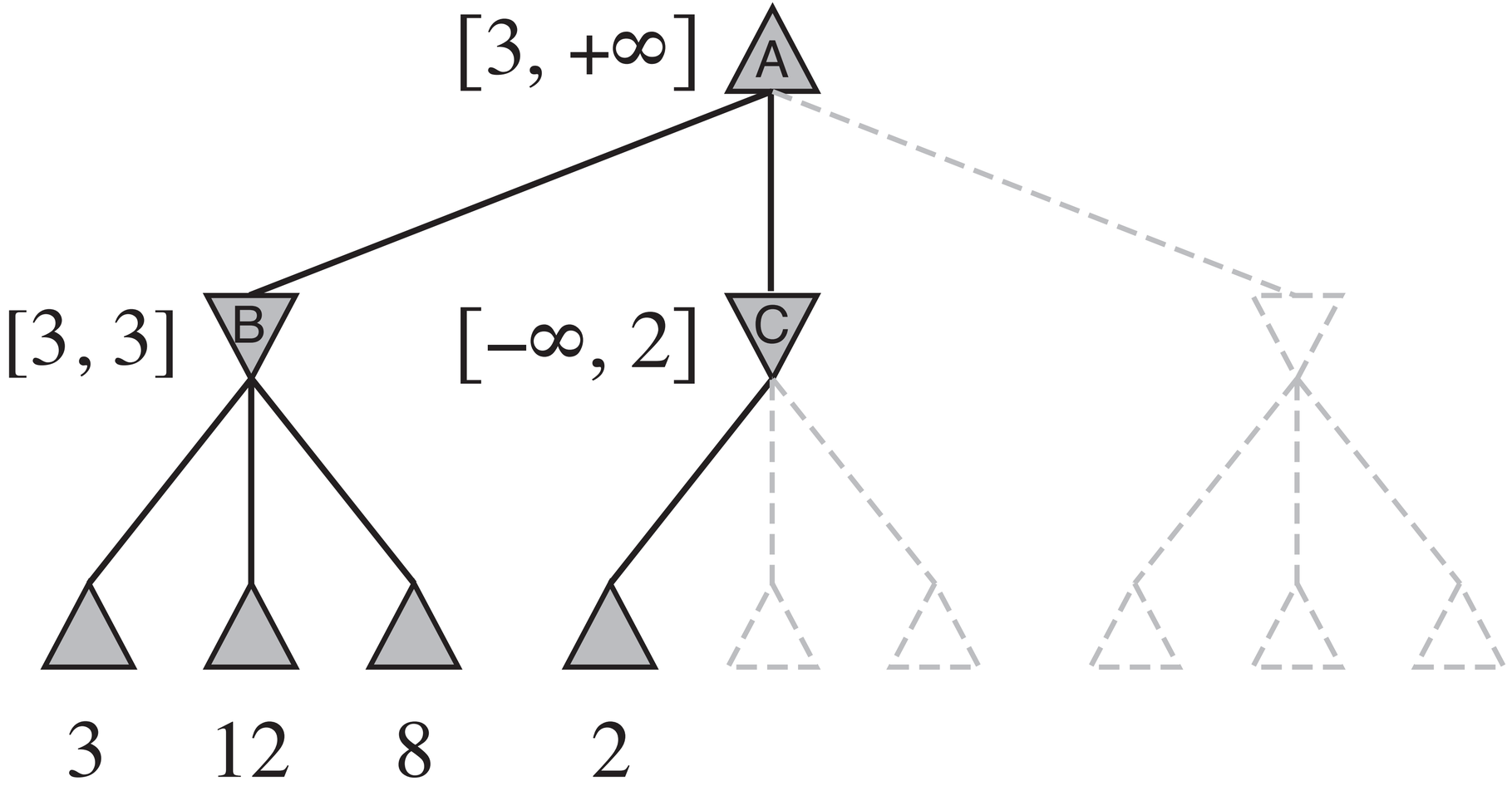

(d)

(e)

[3, 14]
A

(f)

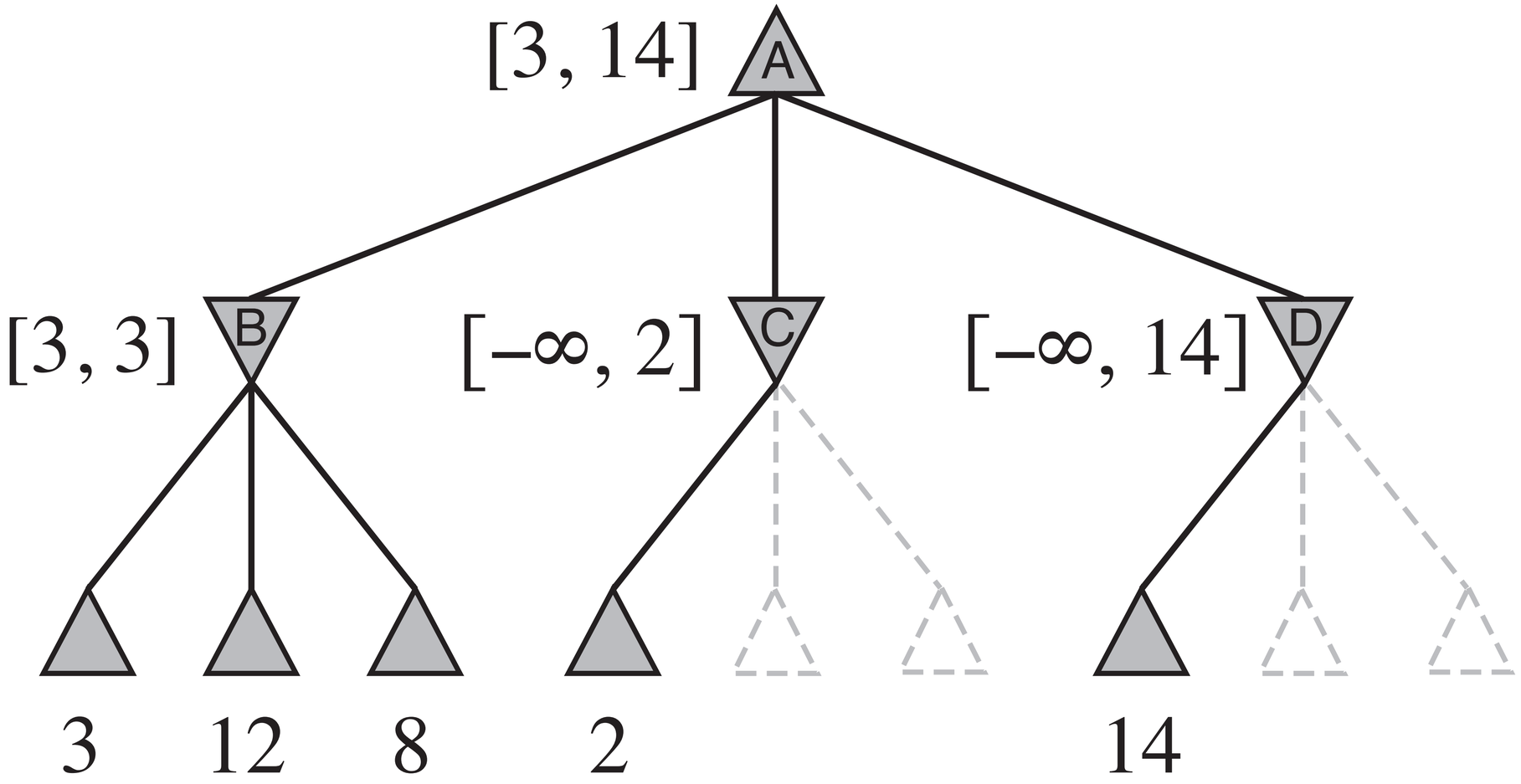
(d)




(f)

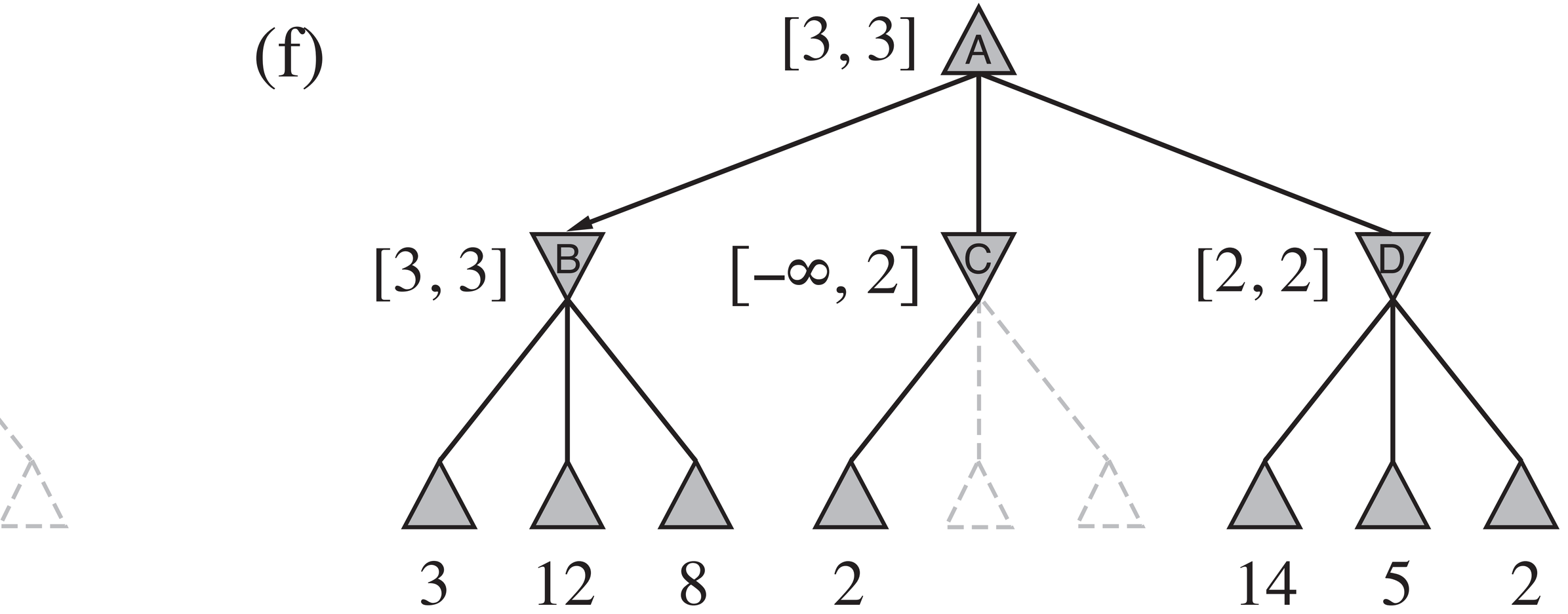
3 12 8

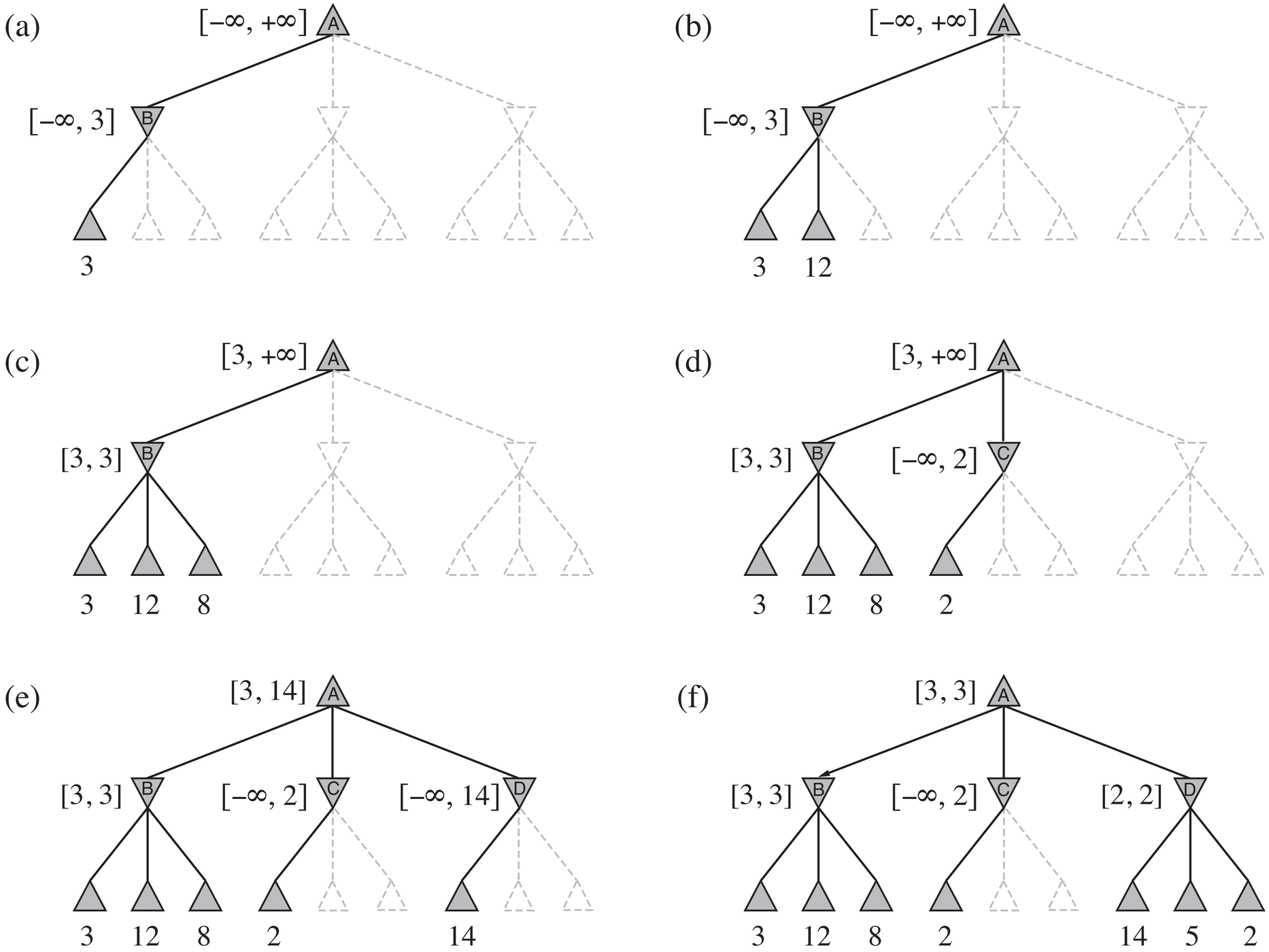
(e)



3 12 8 2

(f)




(a)
[−∞, +∞]
A
[−∞, 3]
B
3
(b)
[−∞, +∞]
A
[−∞, 3]
B
3
12
(c)
[3, +∞]
A
[3, 3]
B
3
12
8
(d)
[3, +∞]
A
[3, 3]
B
[−∞, 2]
C
3
12
8
2
(e)
[3, 14]
A
[3, 3]
B
[−∞, 2]
C
[−∞, 14]
D
3
12
8
2
14
(f)
[3, 3]
A
[3, 3]
B
[−∞, 2]
C
[2, 2]
D
3
12
8
2
14
5
2

**function** ALPHA-BETA-SEARCH($state$) **returns** an action
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  **return** the $action$ in ACTIONS($state$) with value $v$

---

**function** MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  **if** TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  **for each** $a$ **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT($s$,$a$), $\alpha, \beta$))
    **if** $v \geq \beta$ **then return** $v$
    $\alpha \leftarrow$ MAX($\alpha, v$)
  **return** $v$

---

**function** MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  **if** TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow +\infty$
  **for each** $a$ **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT($s$,$a$) , $\alpha, \beta$))
    **if** $v \leq \alpha$ **then return** $v$
    $\beta \leftarrow$ MIN($\beta, v$)
  **return** $v$

# Notes

- Transposition table: cache previously-seen states
- Maximum-depth heuristics

# Reading

- This video covers Chapter 5 up to 5.3
- Read 5.4 in addition for class