# Minimax

**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** $v$

**Figure 5.3** An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation $\text{argmax}_{a \in S} f(a)$ computes the element $a$ of set $S$ that has the maximum value of $f(a)$.

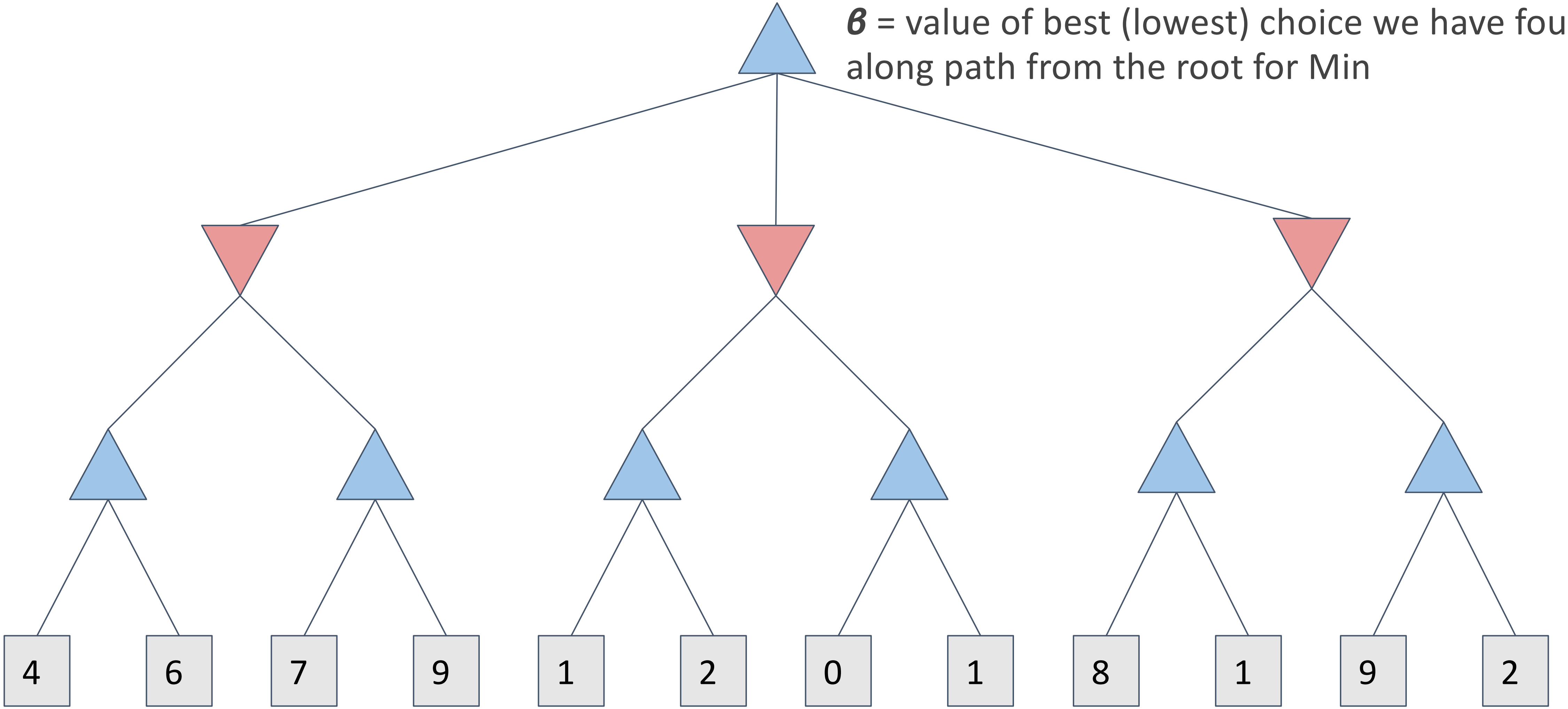
# Alpha–Beta Pruning

```
def alphabeta_search(state):
    alpha = -infinity
    beta  = +infinity
    value = max_value(state, alpha, beta)
    best_action = action that has utility=value to Max
    return

def max_value(state, alpha, beta):
    if terminal_state(state):
        return utility(state)
    value = -infinity
    for action in all available actions:
        value = max(value, min_value(result(action, state), alpha, beta))
        if value >= beta: return value
        alpha = max(value, alpha)
    return value

def min_value(state, alpha, beta):
    if terminal_state(state):
        return utility(state)
    value = +infinity
    for action in all available actions:
        value = min(value, max_value(result(action, state), alpha, beta))
        if value <= alpha: return value
        beta = min(value, beta)
    return value
```

# Alpha–Beta Pruning

$\alpha$ = value of best (highest) choice we have found so far along path from the root for Max

$\beta$ = value of best (lowest) choice we have found so far along path from the root for Min

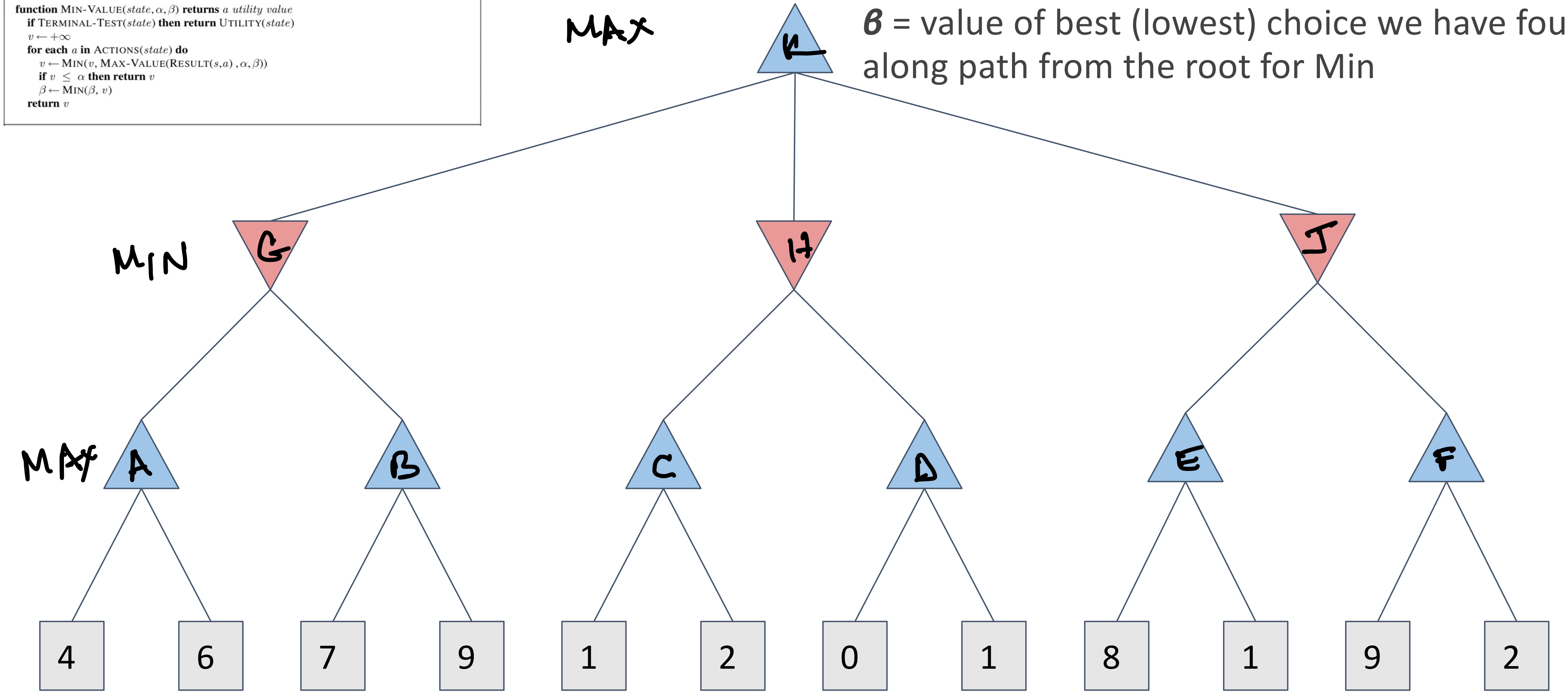


# Alpha-Beta Pruning

```
function ALPHA-BETA-SEARCH(state) returns an action
   v ← MAX-VALUE(state, −∞, +∞)
   return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for each a in ACTIONS(state) do
      v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v

function MIN-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← +∞
   for each a in ACTIONS(state) do
      v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

***α*** = value of best (highest) choice we have found so far along path from the root for Max

***β*** = value of best (lowest) choice we have found so far along path from the root for Min



# Alpha-Beta Pruning

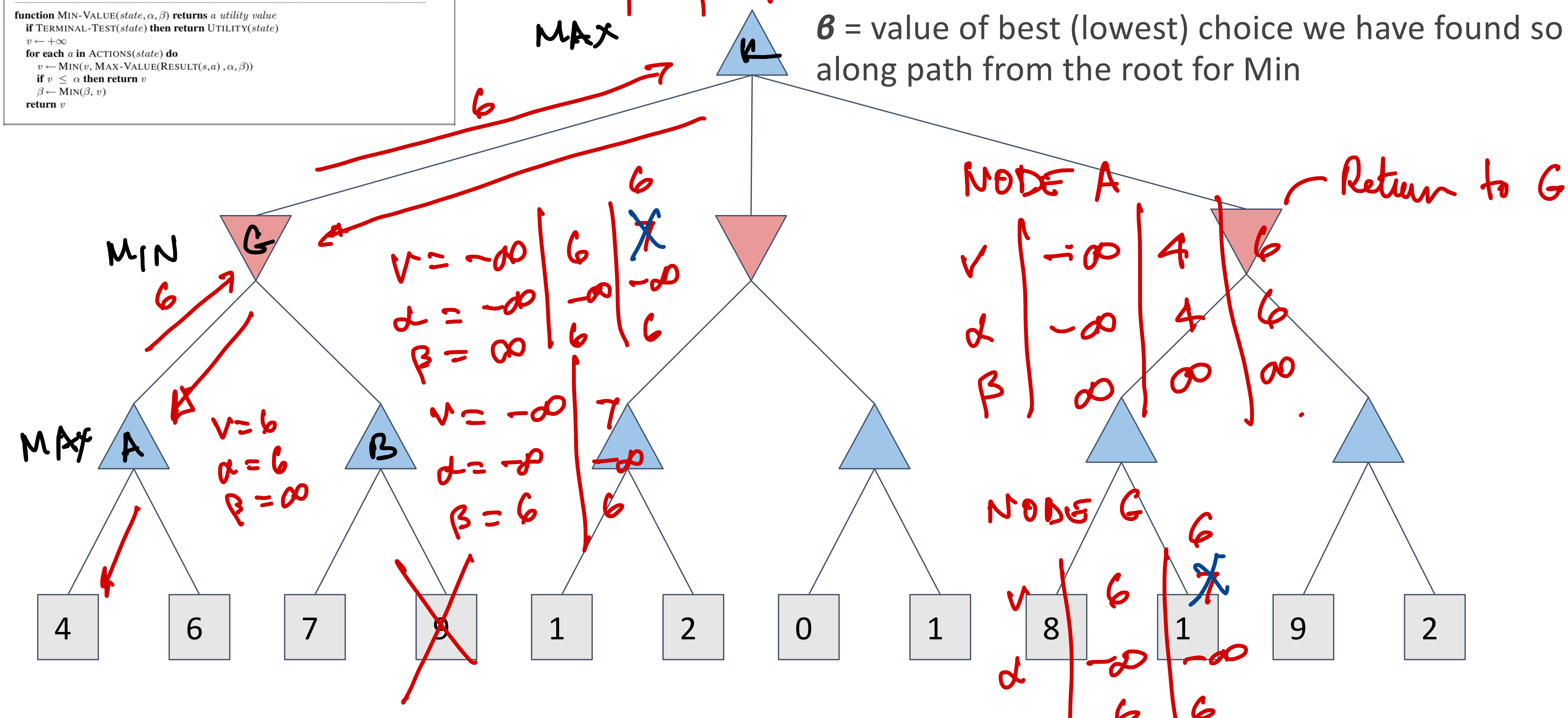
**function** ALPHA-BETA-SEARCH(*state*) **returns** an action
  $v \leftarrow$ MAX-VALUE(*state*, $-\infty, +\infty$)
  **return** the *action* in ACTIONS(*state*) with value $v$

**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** $v$

***α*** = value of best (highest) choice we have found so far along path from the root for Max

***β*** = value of best (lowest) choice we have found so far along path from the root for Min



# Alpha-Beta Pruning

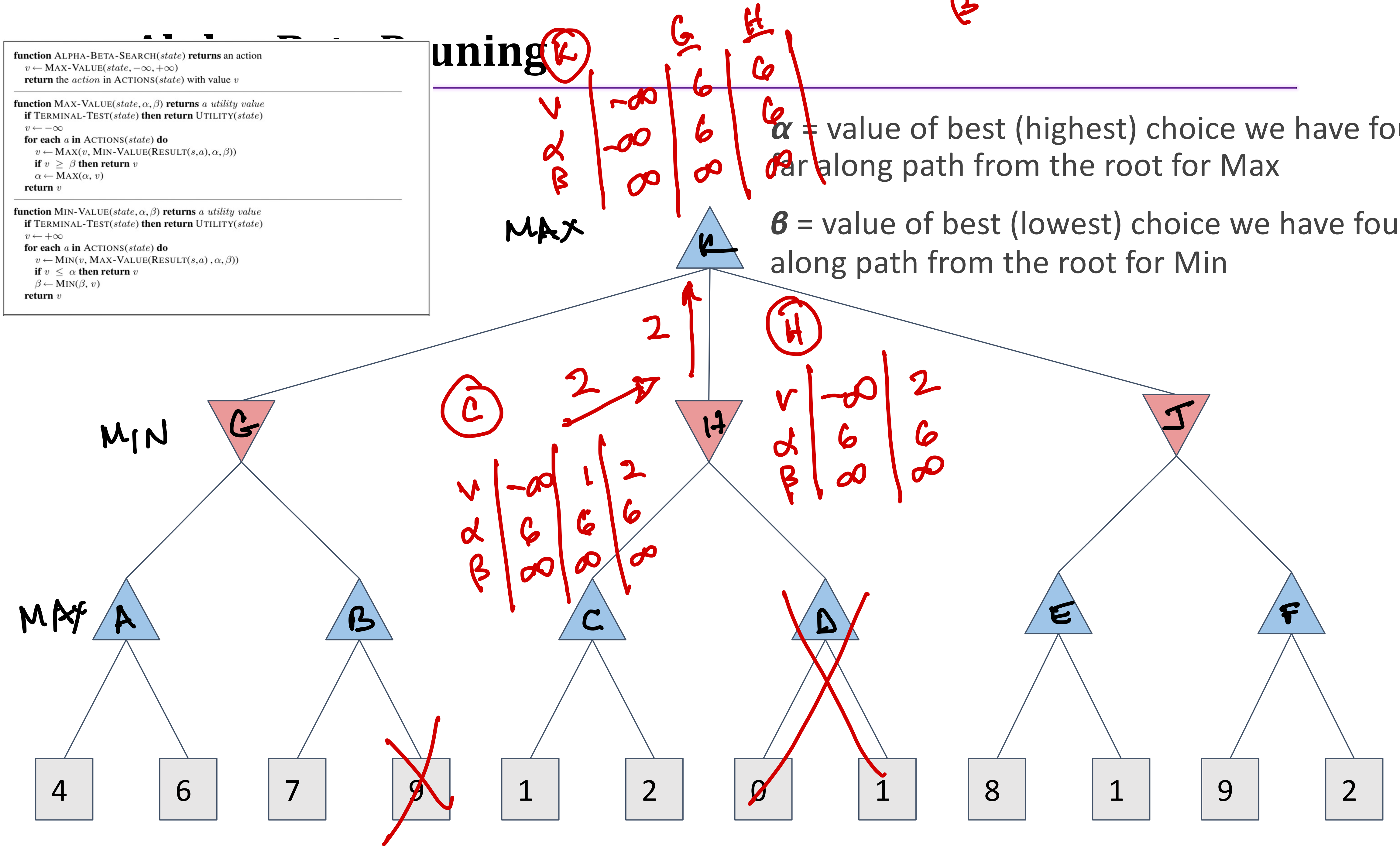
**function** ALPHA-BETA-SEARCH(*state*) **returns** an action
  $v \leftarrow$ MAX-VALUE(*state*, $-\infty, +\infty$)
  **return** the *action* in ACTIONS(*state*) with value $v$

**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** $v$

$\boldsymbol{\alpha}$ = value of best (highest) choice we have found so far along path from the root for Max

$\boldsymbol{\beta}$ = value of best (lowest) choice we have found so far along path from the root for Min

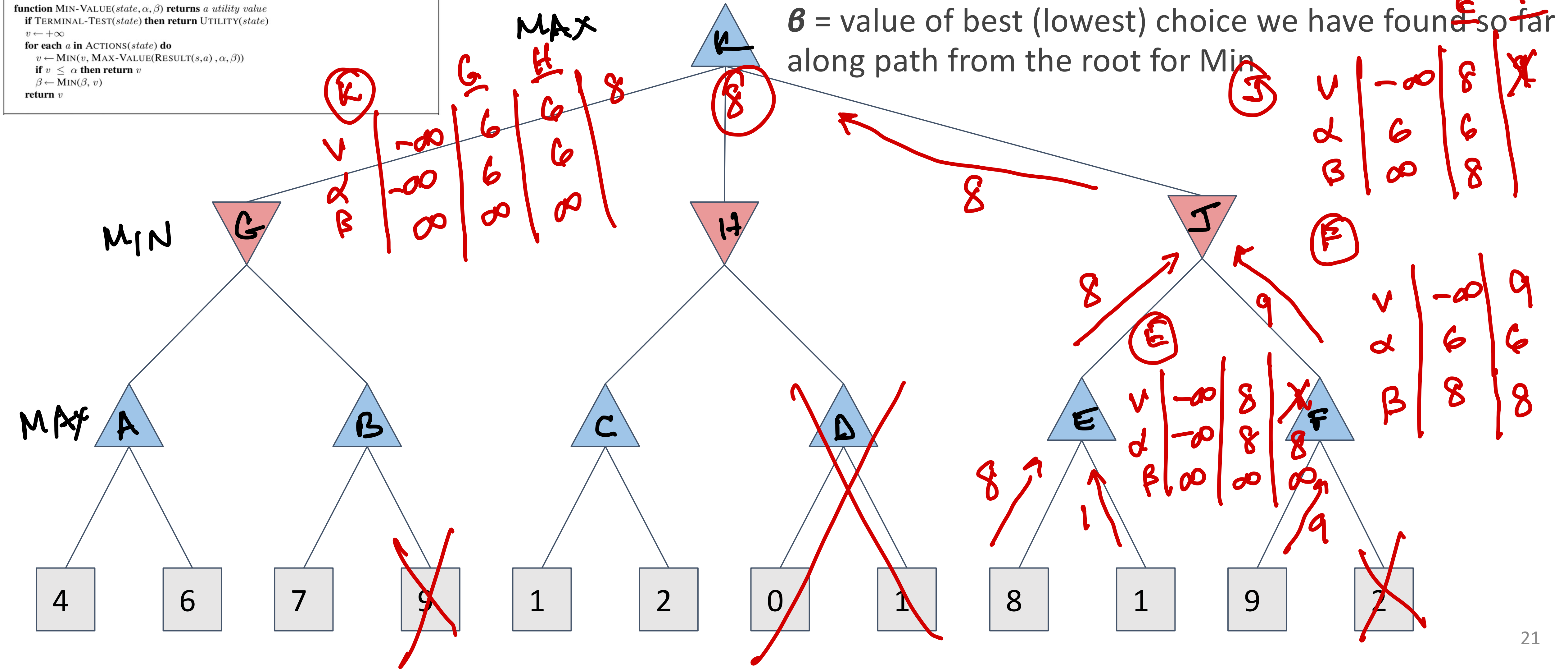


# Alpha-Beta Pruning

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

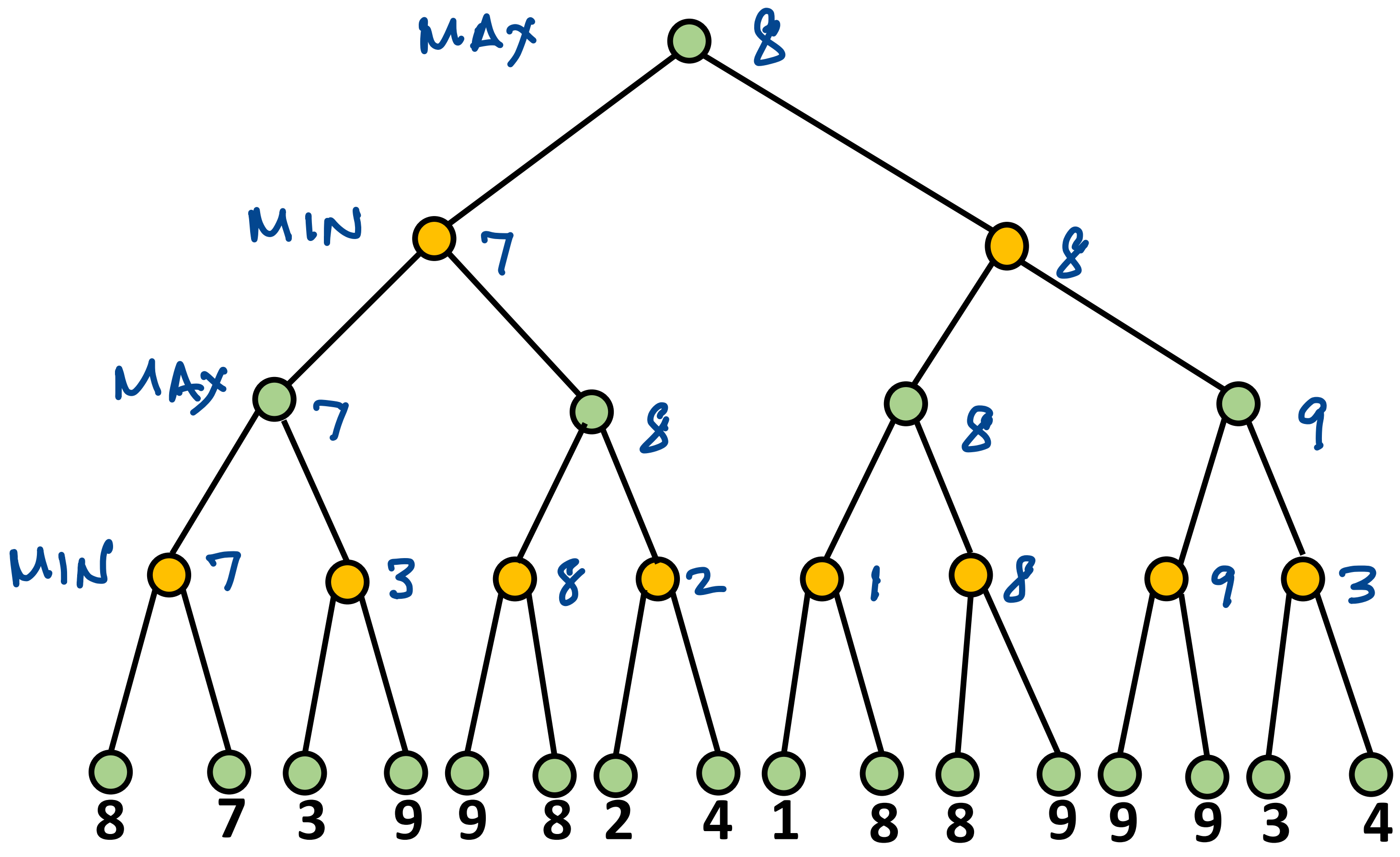
***α*** = value of best (highest) choice we have found so far along path from the root for Max

***β*** = value of best (lowest) choice we have found so far along path from the root for Min



# Alpha_Beta Pruning

**Example**: Consider the Game Tree to below. Find the resulting value at the root node by following the minimax algorithm.



# Alpha–Beta Pruning

**Example**: Consider the Game Tree to below. Find the resulting value at the root node by following the minimax algorithm.