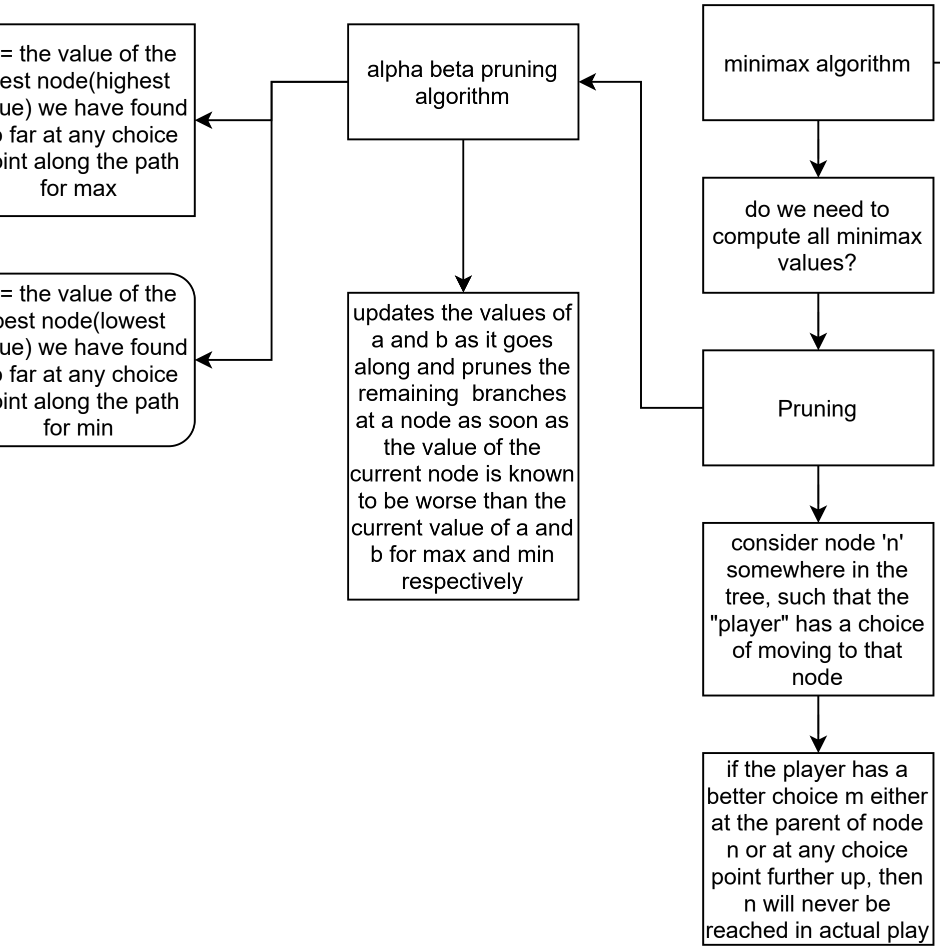
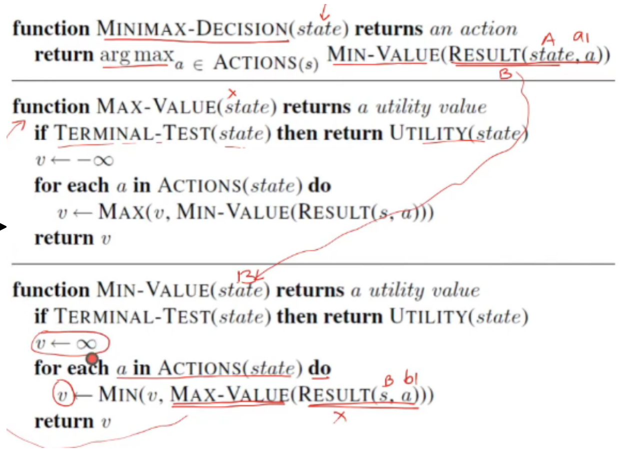
a
b
val
sc
po
b
b
val
sc
po


= the value of the
est node(highest
ue) we have found
o far at any choice
oint along the path
for max
= the value of the
est node(lowest
ue) we have found
o far at any choice
oint along the path
for min
alpha beta pruning algorithm
updates the values of a and b as it goes along and prunes the remaining branches at a node as soon as the value of the current node is known to be worse than the current value of a and b for max and min respectively
minimax algorithm
do we need to compute all minimax values?
Pruning
consider node 'n' somewhere in the tree, such that the "player" has a choice of moving to that node
if the player has a choice m either at the parent of node n or at any choice point further up, then n will never be reached in actual play

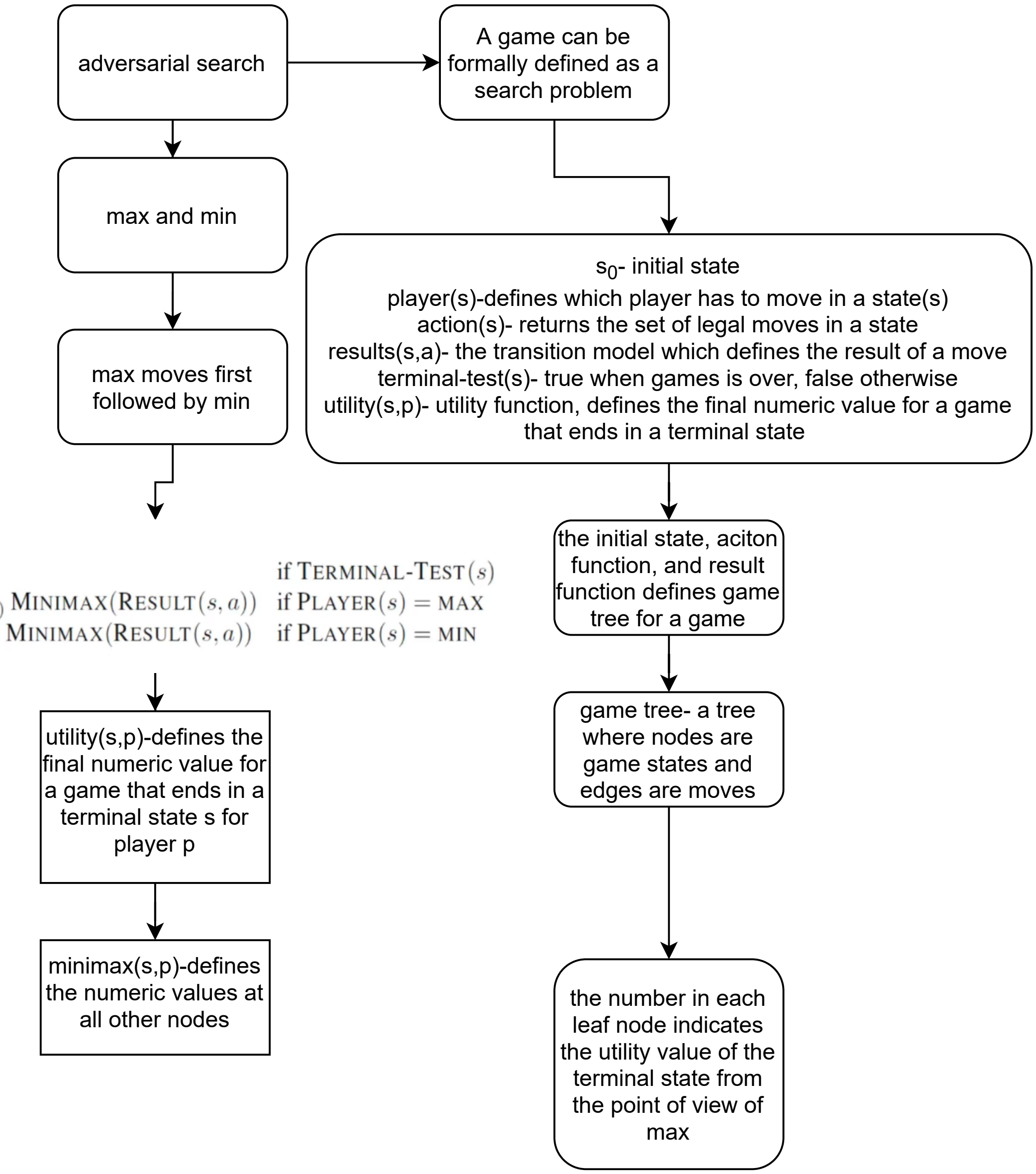


```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) 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)))
  return v

function MIN-VALUE(state) 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)))
  return v
```

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s) \\ \max_{a\in Actions(s} \\ \min_{a\in Actions(s)} \end{cases}$$



adversarial search

max and min

max moves first followed by min

if TERMINAL-TEST(s)

MINIMAX(RESULT(s, a)) if PLAYER$(s) = $ MAX

MINIMAX(RESULT(s, a)) if PLAYER$(s) = $ MIN

utility(s,p)-defines the final numeric value for a game that ends in a terminal state s for player p

minimax(s,p)-defines the numeric values at all other nodes

A game can be formally defined as a search problem

s_0- initial state
player(s)-defines which player has to move in a state(s)
action(s)- returns the set of legal moves in a state
results(s,a)- the transition model which defines the result of a move
terminal-test(s)- true when games is over, false otherwise
utility(s,p)- utility function, defines the final numeric value for a game that ends in a terminal state

the initial state, aciton function, and result function defines game tree for a game

game tree- a tree where nodes are game states and edges are moves

the number in each leaf node indicates the utility value of the terminal state from the point of view of max