Minimax, Alpha-Beta Pruning, and Expectimax Algorithms

Algorithm 1 Minimax Algorithm

```
1: function MINIMAX(node, depth, maximizingPlayer)
2:     if depth == 0 or node is a terminal node then
3:         return Evaluate(node)
4:     end if
5:     if maximizingPlayer then
6:         bestValue ← −∞
7:         for each child in node's children do
8:             value ← MINIMAX(child, depth - 1, false)
9:             bestValue ← max(bestValue, value)
10:        end for
11:        return bestValue
12:    else
13:        bestValue ← ∞
14:        for each child in node's children do
15:            value ← MINIMAX(child, depth - 1, true)
16:            bestValue ← min(bestValue, value)
17:        end for
18:        return bestValue
19:    end if
20: end function
```

Algorithm 2 Alpha-Beta Pruning Algorithm

```
1: function AlphaBeta(node, depth, α, β, maximizingPlayer)
2:     if depth == 0 or node is a terminal node then
3:         return Evaluate(node)
4:     end if
5:     if maximizingPlayer then
6:         bestValue ← −∞
7:         for each child in node's children do
8:             value ← AlphaBeta(child, depth - 1, α, β, false)
9:             bestValue ← max(bestValue, value)
10:            α ← max(α, bestValue)
11:            if β ≤ α then
12:                break                                  ▷ Beta cutoff
13:            end if
14:        end for
15:        return bestValue
16:    else
17:        bestValue ← ∞
18:        for each child in node's children do
19:            value ← AlphaBeta(child, depth - 1, α, β, true)
20:            bestValue ← min(bestValue, value)
21:            β ← min(β, bestValue)
22:            if β ≤ α then
23:                break                                  ▷ Alpha cutoff
24:            end if
25:        end for
26:        return bestValue
27:    end if
28: end function
```

Algorithm 3 Expectimax Algorithm

```
1: function EXPECTIMAX(node, depth, maximizingPlayer)
2:     if depth == 0 or node is a terminal node then
3:         return Evaluate(node)
4:     end if
5:     if maximizingPlayer then
6:         bestValue ← −∞
7:         for each child in node's children do
8:             value ← EXPECTIMAX(child, depth - 1, false)
9:             bestValue ← max(bestValue, value)
10:        end for
11:        return bestValue
12:    else
13:        sumValues ← 0
14:        numChildren ← number of children
15:        for each child in node's children do
16:            value ← EXPECTIMAX(child, depth - 1, true)
17:            sumValues ← sumValues + value
18:        end for
19:        return sumValues / numChildren                ▷ Expected value
20:    end if
21: end function
```