Algorithm 1: Minimax Algorithm

```
Function Minimax(state, depth, maxPlayer):
    if depth = 0 or game is over then
        return evaluate(state)
    end
    if maxPlayer then
        bestScore ← −∞;
        foreach possible move move in state do
            score ← Minimax(makeMove(state, move), depth − 1,
             false);
            bestScore ← max(bestScore, score);
        end
        return bestScore;
    end
    else
        bestScore ← ∞;
        foreach possible move move in state do
            score ← Minimax(makeMove(state, move), depth − 1, true);
            bestScore ← min(bestScore, score);
        end
        return bestScore;
    end
```

Algorithm 2: Minimax Algorithm with Alpha-Beta Pruning

```
Function AlphaBetaPruning(state, depth, α, β, maxPlayer):
    if depth = 0 or game is over then
        return evaluate(state)
    end
    if maxPlayer then
        bestScore ← −∞;
        foreach possible move move in state do
            score ← AlphaBetaPruning(makeMove(state, move),
              depth − 1, α, β, false);
            bestScore ← max(bestScore, score);
            α ← max(α, score);
            if β ≤ α then
                break;
            end
        end
        return bestScore;
    end
    else
        bestScore ← ∞;
        foreach possible move move in state do
            score ← AlphaBetaPruning(makeMove(state, move),
              depth − 1, α, β, true);
            bestScore ← min(bestScore, score);
            β ← min(β, score);
            if β ≤ α then
                break;
            end
        end
        return bestScore;
    end
```