Algorithm 1: Minimax Algorithm

```
Function Minimax(state, depth, maximizing = true):
    if depth = 0 or game is over then
        return evaluate(state)
    end
    if maximizing = true then
        $bestValue \leftarrow -\infty$;
        for each child in state do
            $value \leftarrow$ Minimax(child, depth $- 1$, false);
            $bestValue \leftarrow \max(bestValue, value)$;
        end
        return $bestValue$;
    end
    else
        $bestValue \leftarrow \infty$;
        for each child in state do
            $value \leftarrow$ Minimax(child, depth $- 1$, true);
            $bestValue \leftarrow \min(bestValue, value)$;
        end
        return $bestValue$;
    end
```

Algorithm 2: Minimax Algorithm with Alpha-Beta Pruning

```
Function AlphaBeta(state, depth, $\alpha$, $\beta$, maximizing = true):
    if depth = 0 or game is over then
        return evaluate(state)
    end
    if maximizing = true then
        $bestValue \leftarrow -\infty$;
        for each child in state do
            $value \leftarrow$ AlphaBeta(child, $depth - 1$, $\alpha$, $\beta$, false);
            $bestValue \leftarrow \max(bestValue, value)$;
            $\alpha \leftarrow \max(\alpha, value)$;
            if $\beta \leq \alpha$ then
                break;
            end
        end
        return $bestValue$;
    end
    else
        $bestValue \leftarrow \infty$;
        for each child in state do
            $value \leftarrow$ AlphaBeta(child, $depth - 1$, $\alpha$, $\beta$, true);
            $bestValue \leftarrow \min(bestValue, value)$;
            $\beta \leftarrow \min(\beta, value)$;
            if $\beta \leq \alpha$ then
                break;
            end
        end
        return $bestValue$;
    end
```