$\alpha - \beta$ pruning

```
/* Find the child state with the lowest utility value */

function Minimize(state, $\alpha$, $\beta$)
    returns Tuple of ⟨State, Utility⟩ :

    if Terminal-Test(state):
        return ⟨Null, Eval(state)⟩

    ⟨minChild, minUtility⟩ = ⟨Null, $\infty$⟩

    for child in state.children():
        ⟨ _, utility⟩ = Maximize(child, $\alpha$, $\beta$)

        if utility $<$ minUtility:
            ⟨minChild, minUtility⟩ = ⟨child, utility⟩

        if minUtility $\leq \alpha$:
            break

        if minUtility $< \beta$:
            $\beta$ = minUtility

    return ⟨minChild, minUtility⟩
```

```
/* Find the child state with the highest utility value */

function Maximize(state, $\alpha$, $\beta$)
    returns Tuple of ⟨State, Utility⟩ :

    if Terminal-Test(state):
        return ⟨Null, Eval(state)⟩

    ⟨maxChild, maxUtility⟩ = ⟨Null, $-\infty$⟩

    for child in state.children():
        ⟨ _, utility⟩ = Minimize(child, $\alpha$, $\beta$)

        if utility $>$ maxUtility:
            ⟨maxChild, maxUtility⟩ = ⟨child, utility⟩

        if maxUtility $\geq \beta$:
            break

        if maxUtility $> \alpha$:
            $\alpha$ = maxUtility

    return ⟨maxChild, maxUtility⟩
```

```
/* Find the child state with the highest utility value */

function Decision(state)
    returns State :

    ⟨child, _⟩ = Maximize(state, $-\infty$, $\infty$)

    return child
```