Algorithm 1: MiniMax

```
if gioco finito then
    //Solo se vittoria, sconfitta o pareggio
    return (valuta(B), squeezedChildren:true);
end
if tempo sta per scadere ∨ depth ≤ 0 then
    return (valutazioneParziale(B), squeezedChildren:false);
end
if maximizing then
    int MaxValue = -∞;
    bool SqueezedNode = true;
    for c in FreeCells do
        marca cella c;
        long boardHash;
        if abbiamo hash precedente then
            boardHash = diffHash(hashPrecedente, c);
        end
        else
            boardHash = computeHash(B);
        end
        Integer boardValue = EvaluatedStates.getOrDefault(boardHash);
        if boardValue == null then
            //Dobbiamo procedere con l'algoritmo
            aggiorna WinCounters(c);
            moveVal = MiniMax(depth-1, boardHash, alpha, beta);
            boardValue = moveVal.boardValue;
            if !moveVal.squeezedChildren then
                // Se abbiamo usato una valutazione parziale piu' in profondita'
                // non possiamo considerare questo nodo come spremuto
                SqueezedNode = false;
            end
            resetta WinCounters(c);
        end
        undo marca cella c;
        if boardValue > MaxValue then
            MaxValue = boardValue;;
        end
        //Alpha beta
        alpha = Math.max(alpha, MaxValue);
        if alpha ≥ beta then
            return (MaxValue, squeezedChildren:true);
        end
        // Se a questo punto SqueezedNode e' ancora true allora
        // non abbiamo mai raggiunto una valutazione parziale
        if SqueezedNode then
            //Possiamo procedere a salvare la valutazione nelle tabelle hash
            salvaStato(nodeHash, MaxValue);
        end
        return (MaxValue, squeezedChildren:SqueezedNode);
    end
end
else
    simmetrico per minimizing
end
```