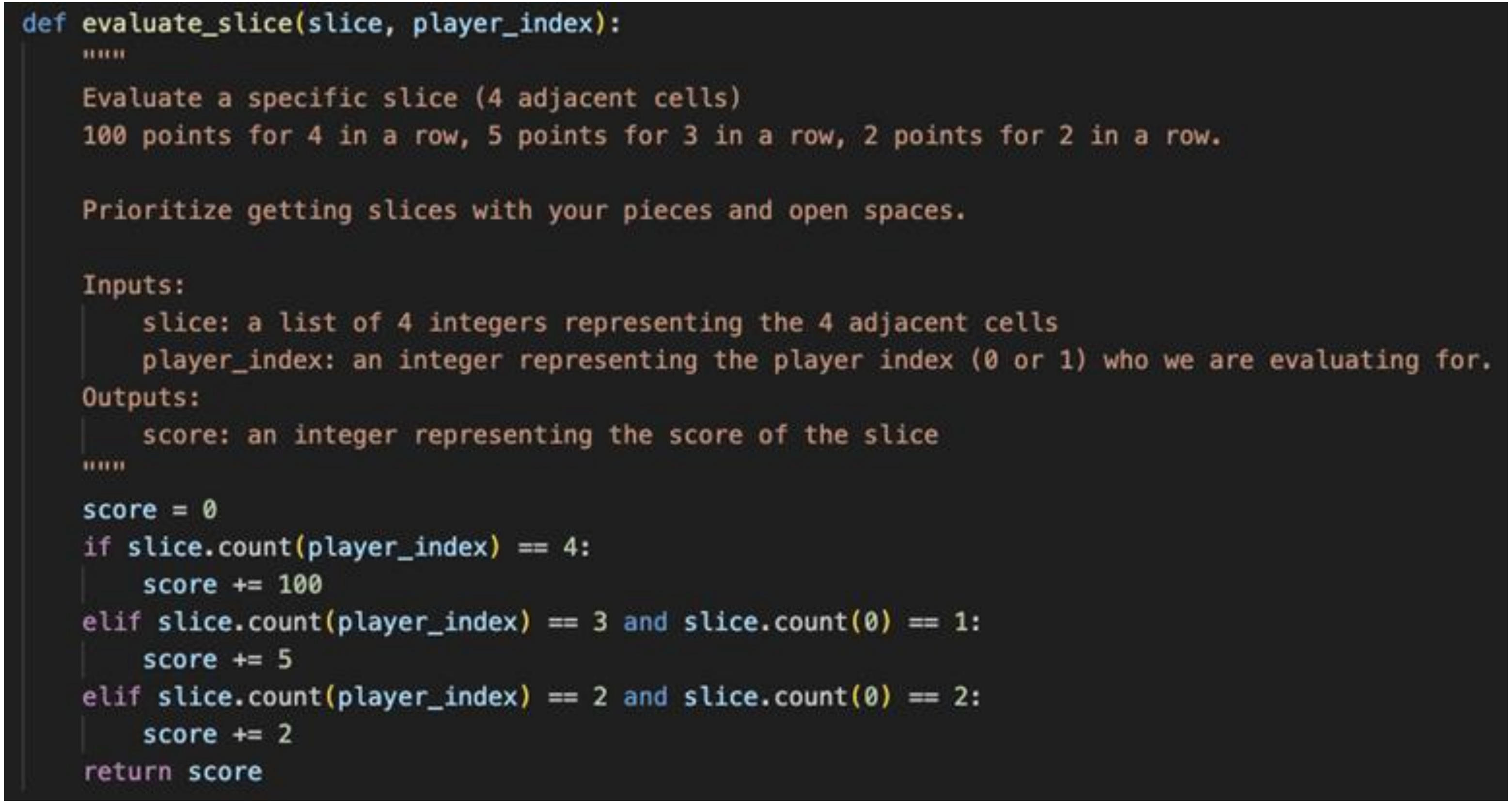
Monte Carlo Tree Search

Connect 4 Heuristic

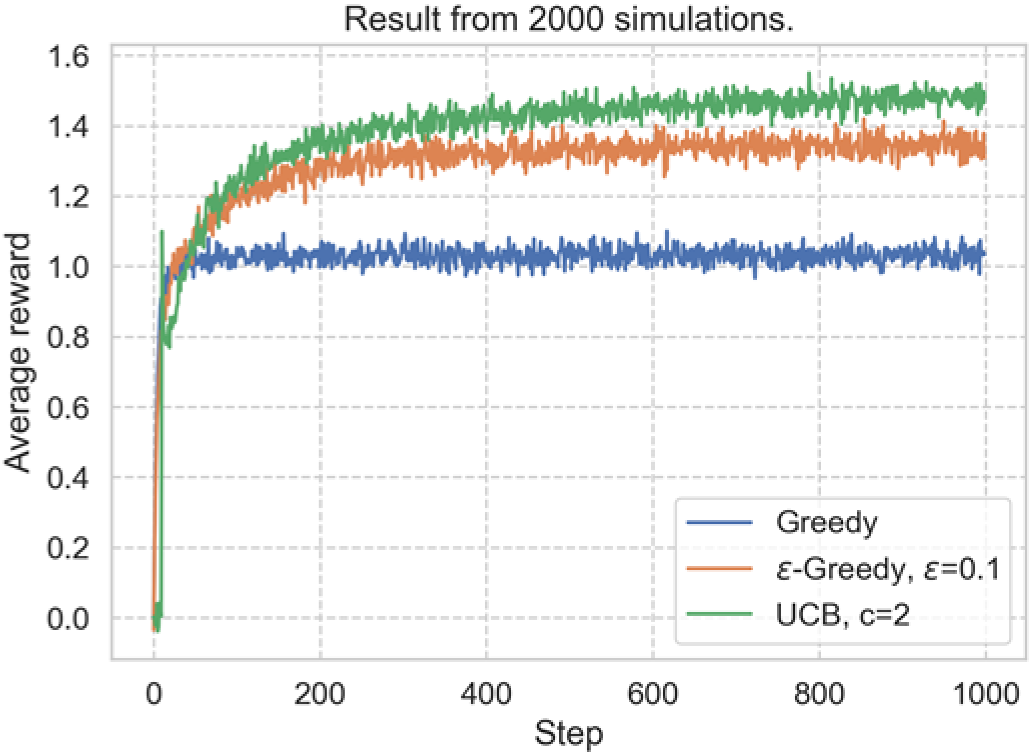


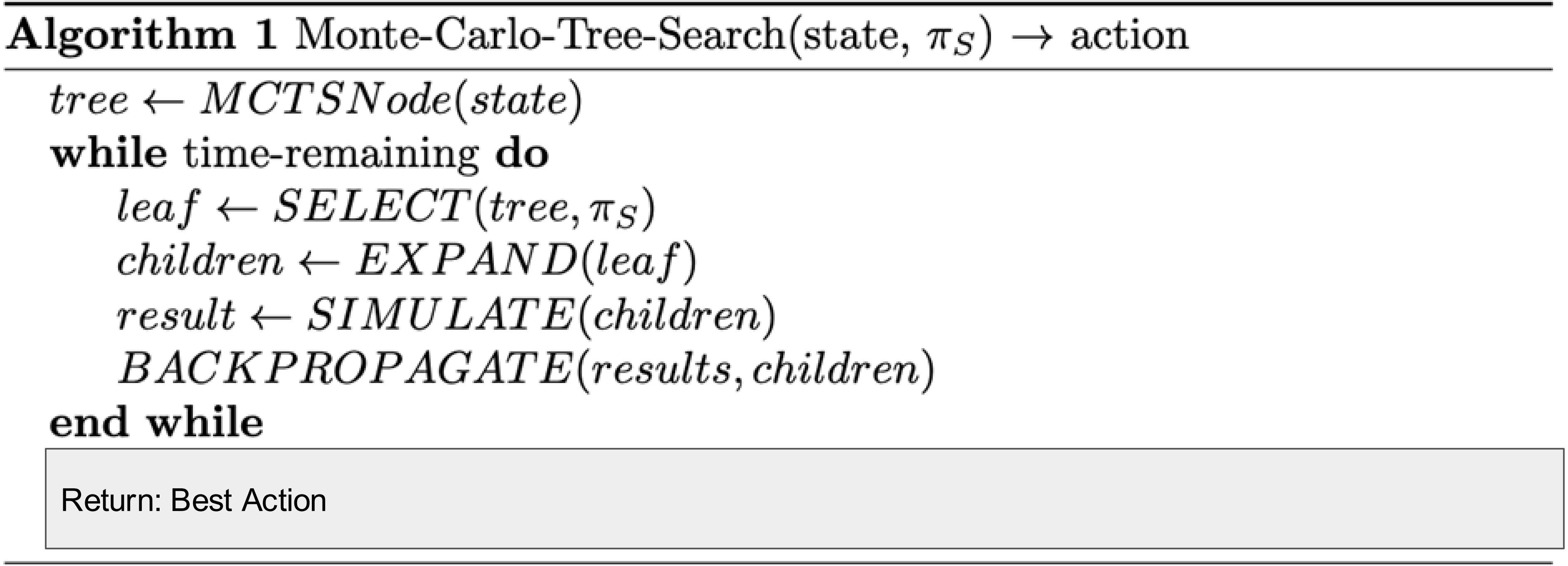
```python
def evaluate_slice(slice, player_index):
    """
    Evaluate a specific slice (4 adjacent cells)
    100 points for 4 in a row, 5 points for 3 in a row, 2 points for 2 in a row.

    Prioritize getting slices with your pieces and open spaces.

    Inputs:
        slice: a list of 4 integers representing the 4 adjacent cells
        player_index: an integer representing the player index (0 or 1) who we are evaluating for.
    Outputs:
        score: an integer representing the score of the slice
    """
    score = 0
    if slice.count(player_index) == 4:
        score += 100
    elif slice.count(player_index) == 3 and slice.count(0) == 1:
        score += 5
    elif slice.count(player_index) == 2 and slice.count(0) == 2:
        score += 2
    return score
```

Performance of Different Exploration Policies in Multi-Armed Bandits



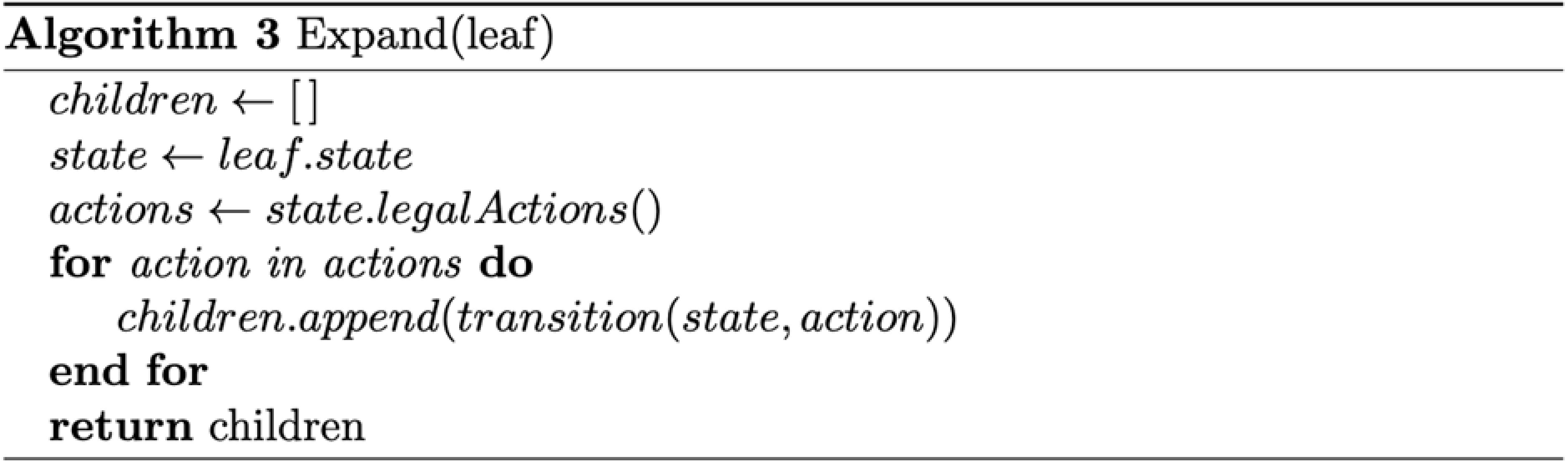


Algorithm 1 Monte-Carlo-Tree-Search(state, π_S) → action

```
$tree \leftarrow MCTSNode(state)$
**while** time-remaining **do**
    $leaf \leftarrow SELECT(tree, \pi_S)$
    $children \leftarrow EXPAND(leaf)$
    $result \leftarrow SIMULATE(children)$
    $BACKPROPAGATE(results, children)$
**end while**
Return: Best Action
```

Algorithm 2 SELECT(state, π_S)

```
  $currNode \leftarrow state$
  **while** $!isLeaf(currNode)$ **do**
      $currNode \leftarrow \pi_S(currNode.children)$
      $currNode.visits \leftarrow currNode.visits + 1$
  **end while**
  **return** $currNode$
```

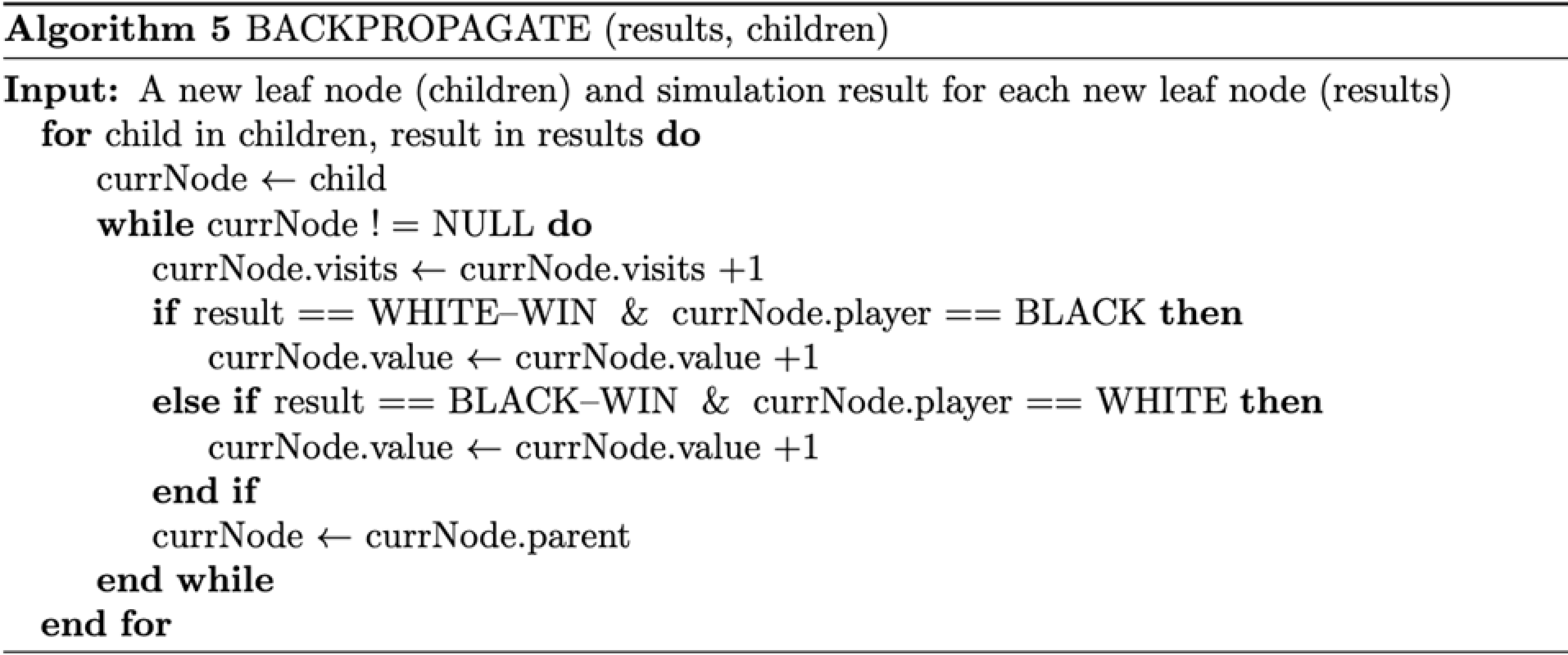


Algorithm 3 Expand(leaf)

```
$children \leftarrow []$
$state \leftarrow leaf.state$
$actions \leftarrow state.legalActions()$
for $action$ in $actions$ do
    $children.append(transition(state, action))$
end for
return children
```

Algorithm 4 SIMULATE(children)

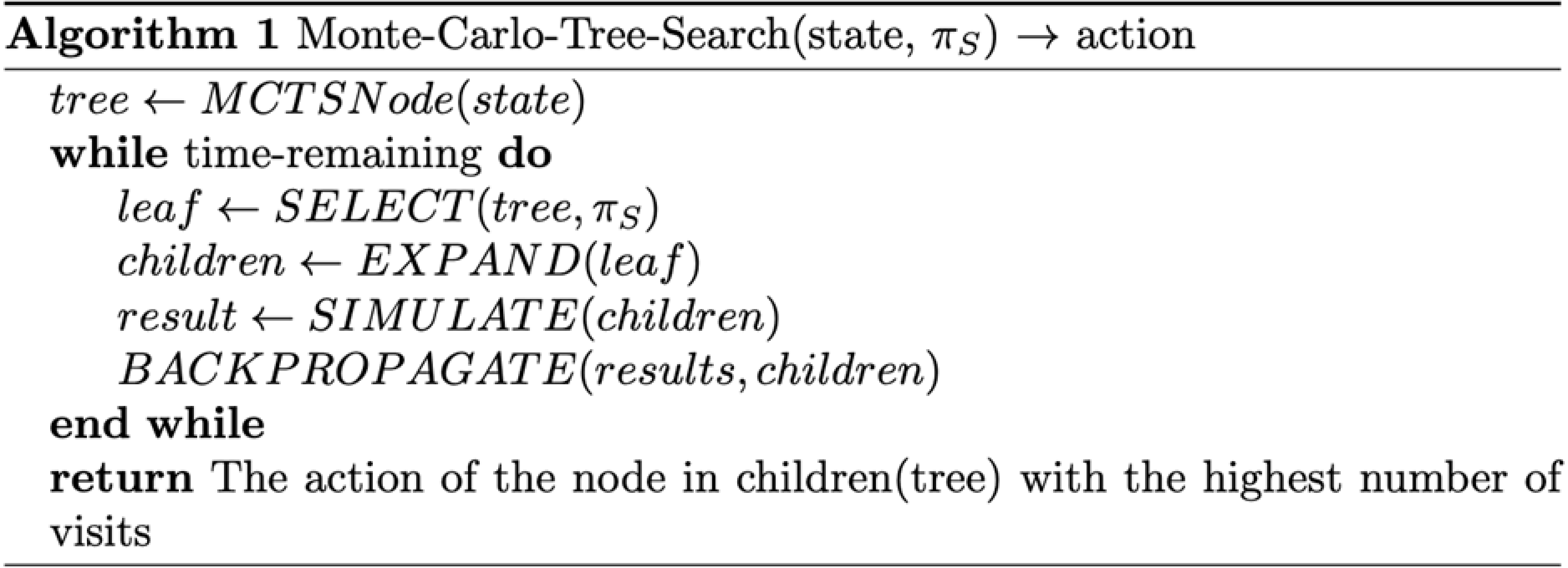
```
$results \leftarrow [\,]$
for child in children do
    $result = rollout(child)$
    $results.append(result)$
end for
return $results$
```



Algorithm 5 BACKPROPAGATE (results, children)

Input: A new leaf node (children) and simulation result for each new leaf node (results)

```
for child in children, result in results do
    currNode ← child
    while currNode ! = NULL do
        currNode.visits ← currNode.visits +1
        if result == WHITE–WIN & currNode.player == BLACK then
            currNode.value ← currNode.value +1
        else if result == BLACK–WIN & currNode.player == WHITE then
            currNode.value ← currNode.value +1
        end if
        currNode ← currNode.parent
    end while
end for
```



Algorithm 1 Monte-Carlo-Tree-Search(state, π_S) → action

```
$tree \leftarrow MCTSNode(state)$
while time-remaining do
    $leaf \leftarrow SELECT(tree, \pi_S)$
    $children \leftarrow EXPAND(leaf)$
    $result \leftarrow SIMULATE(children)$
    $BACKPROPAGATE(results, children)$
end while
return The action of the node in children(tree) with the highest number of visits
```