CompMath: LaTeX-Übung 3

MiniMax Algorithm for Dummies

Richard Weiss

Technische Universität Wien

June 2019

Input and Output

Output

The *Minimax Algorithm* determinines the **optimal game strategy** for *finite*, *two-person*, *zero-sum games*, with *perfect information*. It can be **extended** on the basis of *expected values*.

Input

It operates on a *tree* of *states* (representing the game), described by:

- *Leaf Node*: Assessed state via *evaluation function*
- *Maximizing Node*: Prefers *child node* with maximal assessed value, but initial value is "worst" maximal value (i.e. $-\infty$)
- *Minimizing Node*

Example: Game Tree

Consider the following situation:

- You're playing chess and want to decide, which move to make.
- You've generated chains of moves and their states.
- You know the evaluation value of the deepest state of each move-chain.

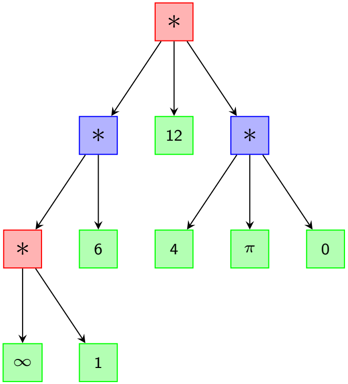
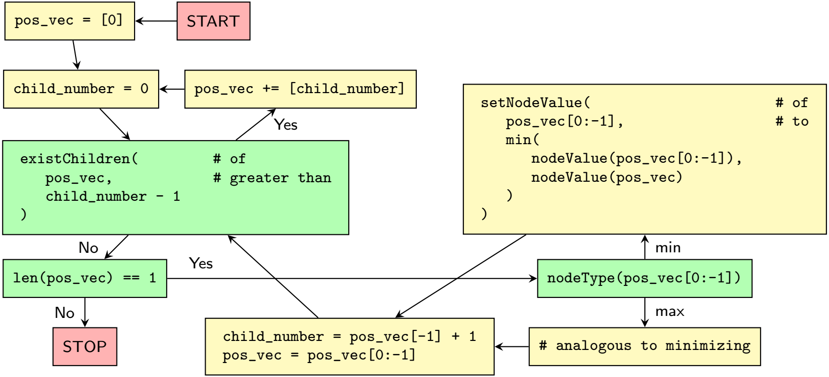


Figure: game tree

Algorithm Outline



START

```
pos_vec = [0]
```

```
child_number = 0
```

```
pos_vec += [child_number]
```

Yes

```
existChildren(          # of
   pos_vec,             # greater than
   child_number - 1
)
```

No

```
len(pos_vec) == 1
```

Yes

No

STOP

```
setNodeValue(                         # of
   pos_vec[0:-1],                     # to
   min(
      nodeValue(pos_vec[0:-1]),
      nodeValue(pos_vec)
   )
)
```

min

```
nodeType(pos_vec[0:-1])
```

max

```
# analogous to minimizing
```

```
child_number = pos_vec[-1] + 1
pos_vec = pos_vec[0:-1]
```

Figure: MiniMax Flowchart

```
int maxi( int depth ) {
    if ( depth == 0 ) return evaluate();
    int max = -oo;
    for ( all moves) {
        score = mini( depth - 1 );
        if( score > max )
            max = score;
    }
    return max;
}
```

```
int mini( int depth ) {
    if ( depth == 0 ) return -evaluate();
    int min = +oo;
    for ( all moves) {
        score = maxi( depth - 1 );
        if( score < min )
            min = score;
    }
    return min;
}
```

Figure: Recursive Implementation

Optimisation

- α-β-pruning: memorises best/worst node values and prunes (cuts) branches of game tree
- sort branches via evaluation function
- parallel computing
- ...

Sources and further reading

- https://www.chessprogramming.org/Minimax
- Programming a Computer for Playing Chess by Claude Shannon
- Building a Simple Chess AI by Brandon Yanofsky
- Simple optimisation:
  - α-β-pruning
  - Razoring , more advanced α-β-pruning
- Computer Chess Compendium by David Levy (all classic papers on computer chess)