**Algorithm 1** myAIAlgorithm

**Input** List<positionTicTacToe> *board*, int *player*
**Return** positionTicTacToe $myNextMove$

**procedure** MYAIALGORITHM($board, player$)

initialization

$winMove \leftarrow getWinMove(player)$ ▷ If we have a win move
**if** $winMove$ exists **then**
  **return** $winMove$

$forceMove \leftarrow getForceMove(player)$ ▷ If we have a force move
**if** $forceMove$ exists **then**
  **return** $forceMove$

$coreMove \leftarrow getFirstTwoSteps(player)$ ▷ Occupy the strongest points
**if** $coreMove$ exists **then**
  **return** $coreMove$

$maxValue \leftarrow -\infty$
$positionTicTacToe\ myNextMove$

**do** ▷ Progressive deepening
  **for** `<each available move` $curMove$`>` **do**
    `<make current move>`
    $newVale \leftarrow miniMax(depth, player, false, -\infty, +\infty)$
    **if** $newValue > maxValue$ **then**
      $maxValue \leftarrow newValue$
      $myNextMove \leftarrow curMove$
    `<cancel current move>` ▷ Backtracking
**while** `<time is still enough>`

**Algorithm 2** miniMax

**Input int** *depth*, **int** *player*, **boolean** *maximizingPlayer*, **int** *alpha*, **int** *beta*
**Return int** *value*
**procedure** MINIMAX
  **if** *depth* == 0 **then** ▷ search finish
    **return** *evaluation*(*player*) ▷ evaluate the board configutaion
  **if** *maximizingPlayer* **then** ▷ Maximizer
    *value* ← −∞
    **for** <each available move *curMove*> **do**
      *winMove* ← *getWinMove*(*player*) ▷ win move pruning
      **if** *winMove* exists **then**
        <make this win move>
        *value* ← *evaluation*(*player*)
        <cancel this win move> ▷ Backtracking
        **break**
      *forceMove* ← *getForceMove*(*player*) ▷ force move pruning
      **if** *forceMove* exists **then**
        <make this force move> ▷ DFS and deepening
        *value* ← $max(value, miniMax(depth, player, false, alpha, beta))$
        <cancel this force move> ▷ Backtracking
      **else** ▷ Naive miniMax
        <make current move> ▷ DFS
        *value* ←
        $max(value, miniMax(depth - 1, player, false, alpha, beta))$
        <cancel current move> ▷ Backtracking
        *alpha* ← $max(alpha, value)$
        **if** *alpha* >= *beta* **then**
          **break**
    **return** *value*
  **else** ▷ Minimizer
    *value* ← +∞
    *opponent* ←!*player*
    **for** <each available move *curMove*> **do**
      *winMove* ← *getWinMove*(*opponent*) ▷ win move pruning
      **if** *winMove* exists **then**
        <make this win move>
        *value* ← *evaluation*(*player*)
        <cancel this win move> ▷ Backtracking
        **break**
      *forceMove* ← *getForceMove*(*opponent*) ▷ force move pruning
      **if** *forceMove* exists **then**
        <make this force move> ▷ DFS and deepening
        *value* ← $min(value, miniMax(depth, player, true, alpha, beta))$
        <cancel this force move> ▷ Backtracking
      **else** ▷ Naive miniMax
        <make current move> ▷ DFS
        *value* ←
        $min(value, miniMax(depth - 1, player, true, alpha, beta))$
        <cancel current move> ▷ Backtracking
        *beta* ← $min(beta, value)$
        **if** *alpha* >= *beta* **then**
          **break**
    **return** *value*