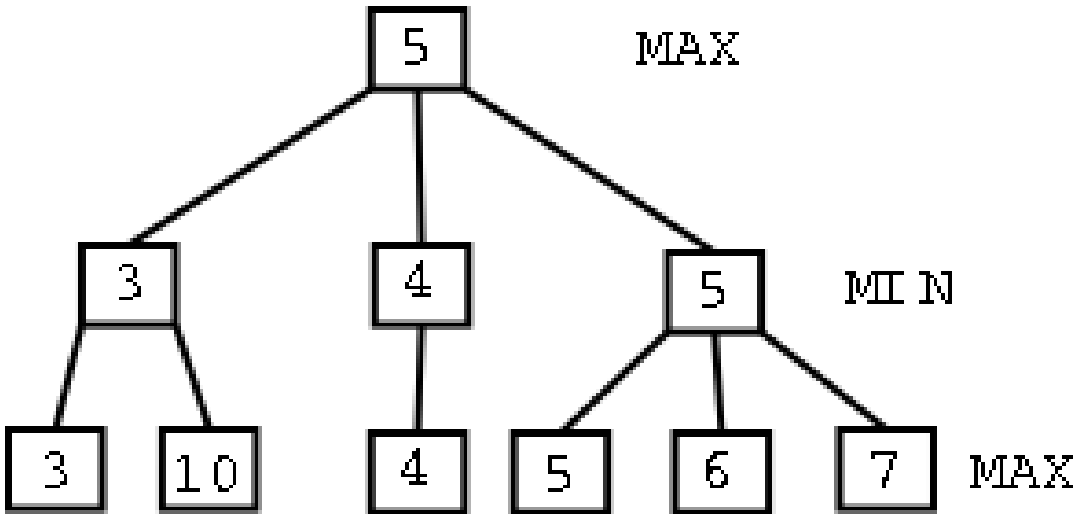
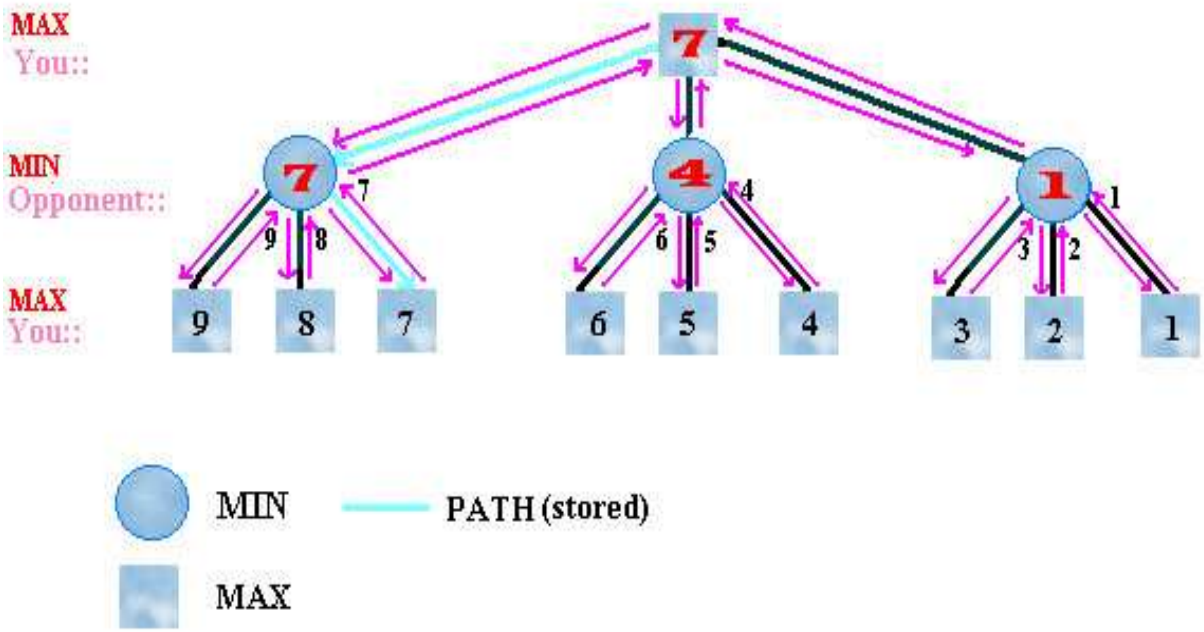
Algoritmus MiniMaxu

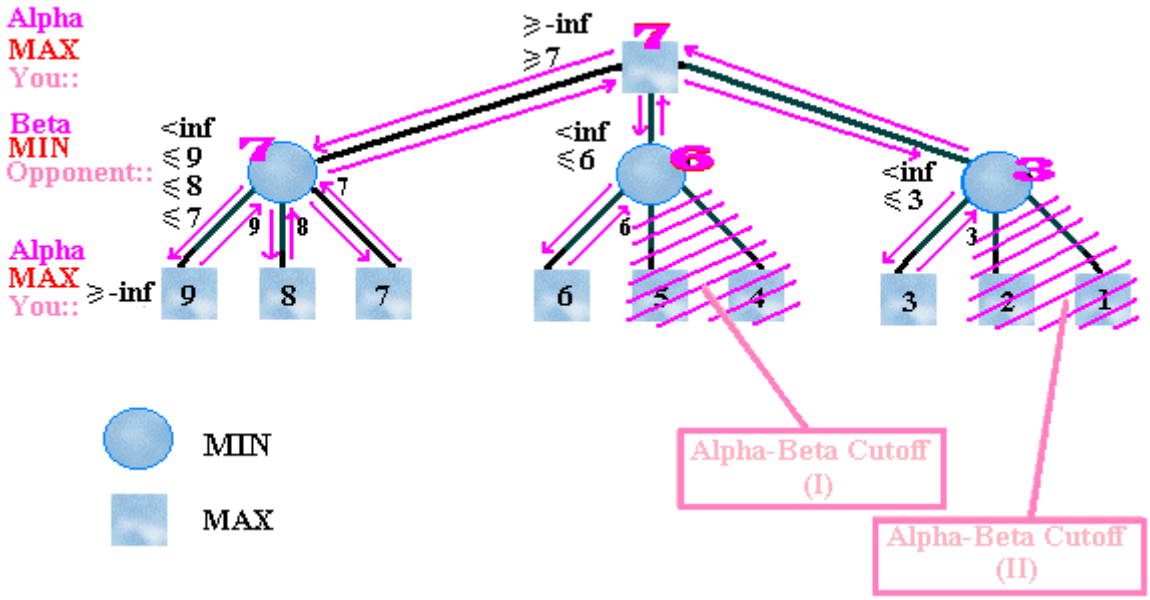



```
function MINIMAX(N) is
 begin
  if N is a leaf then
     return the estimated score of this leaf
  else
     Let N1, N2, .., Nm be the successors of N;
     if N is a Min node then
       return min{MINIMAX(N1), .., MINIMAX(Nm)}
     else
       return max{MINIMAX(N1), .., MINIMAX(Nm)}
    end MINIMAX;
```




```
minmax(u)
 {   //   u is the node you want to score
  if  u is a leaf  return score of u;
  else
   if  u in a min node
     for all childern of u: v1, .. vn;
      return min{minmax(v1),...,minmax(vn)}
    else
     for all childern of u: v1, .. vn;
      return max{minmax(v1),...,minmax(vn)}
 }
```

Alfa – Beta prořezávání




```
function MINIMAX-AB(N, A, B) is ;;
                //  Here A is always less than B
 begin
   if N is a leaf then
     return the estimated score of this leaf
   else
     Set Alpha value of N to -infinity
           and Beta value of N to +infinity;
     if N is a Min node then
      For each successor Ni of N loop
       Let Val be MINIMAX-AB(Ni,A,Min{B,Beta of N});
       Set Beta value of N to Min{Beta value of N,Val};
       When A >= Beta value of N then
          Return Beta value of N
      endloop;
      Return Beta value of N;
     else
      For each successor Ni of N loop
       Let Val be MINIMAX-AB(Ni,Max{A,Alpha value of N},B);
       Set Alpha value of N to Max{Alpha value of N, Val};
       When Alpha value of N >= B then
          Return Alpha value of N
      endloop;
      Return Alpha value of N;
 end MINIMAX-AB;
```

Jiný algoritmus, který využívá dvou funkcí:

```
evaluatemin (u, B)    //  u is a Min node
 {
   alpha = +infinity;
   if  u = leaf  return the score;
   else
    for all children v of u
     {
       Val = evaluatemax (v,  B);
       alpha = Min {alpha, Val};
       if  alpha <= beta  then  exit loop;
     }
     return alpha;
 }

evaluatemax (u, B)   //  u is a Max node
 {
   alpha = -infinity;
   if  u = leaf  return the score;
   else
    for all children v of u
     {
       Val = evaluatemin (v, B);
       alpha = Max {alpha, Val};
       if  alpha >= beta  then  exit loop;
     }
     return alpha;
 }
```

Minimax Psuedocode

```
minimax(in game board, out score chosen_score, out move chosen_move)
begin
        best_score = -infinity;
        Gt_generate_moves(current_mimx_data,moves_list);
        for (i = 0 to moves_list.num_moves-1) do
            new_board = board;
            Gt_move(board, moves_list[i]);
            minimax(board, the_score,the_move);
            if (the_score > best_score) then
                best_score = the_score;
                best_move = moves_list[i];
            endif
         enddo
         chosen_move = best_move;
         Gt_evaluate(current_mimx_data,chosen_score,best_score);
end.
```

Typically Gt_evaluate will look something like this:

```
Gt_evaluate(gt_mimx_data current_mimx_data,gt_score chosen_score,
                                                    gt_score best_score)
begin
    if(current_mimx_data->depth==maxdepth)
        chosen_score=evaluate_board(current_mimx_data->board);
    else
        chosen_score=best_score
end.
```

Alpha-Beta Pseudocode

Note: again, the arrows (< >) are relative.

```
minimax(in game board, in best_opposing_score, out score chosen_score,
                                                   out move chosen_move)
begin
        best_score = -infinity;
        Gt_generate_moves(current_mimx_data,moves_list);
        for (i = 0 to moves_list.num_moves-1) do
            new_board = board;
            Gt_move(board, moves_list[i],the_unmove);
            minimax(board, the_score,the_move);
            Gt_unmove(board,the_unmove);
            if (the_score > best_score) then
               best_score = the_score;
               best_move = moves_list[i];
               if (best_score > best_opposing_score) then
                 cut;/* the opponent will not allow you to reach this node
                          in real life so there is no since in continuing*/
               endif
            endif
         enddo
         chosen_move = best_move;
         Gt_evaluate(current_mimx_data,chosen_score,best_score);
end.
```