

C# Project

Assignment 1

Outline



Overview
of Chess

The Forsyth
notation

Requirements
of
the assignment

Computing the
next states

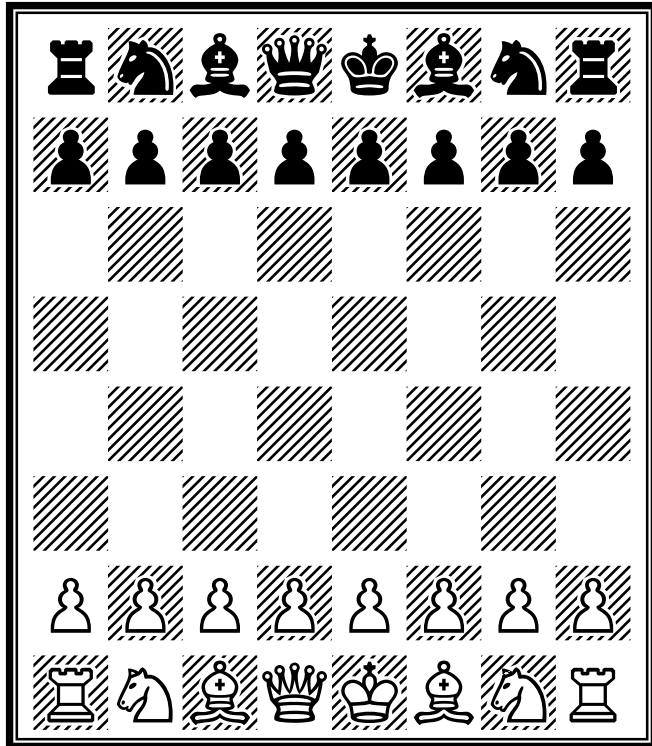
Evaluation
Function







MinMax

Conclusion

Questions?

Overview of Chess

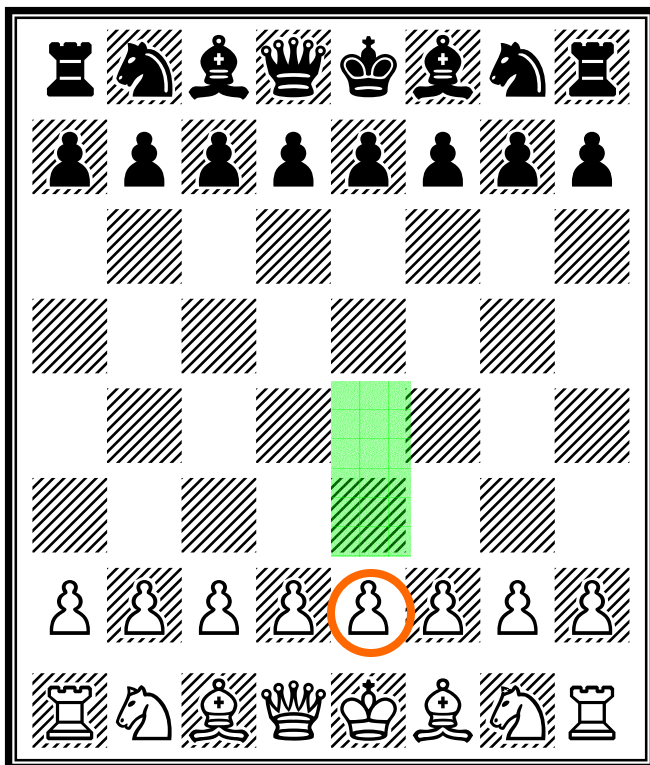


- Board: 8x8 squares
- Two players: White, Black
- 32 pieces (16w, 16b)
-  Rock
-  Knight
-  Bishop
-  Queen
-  King
-  Pawn

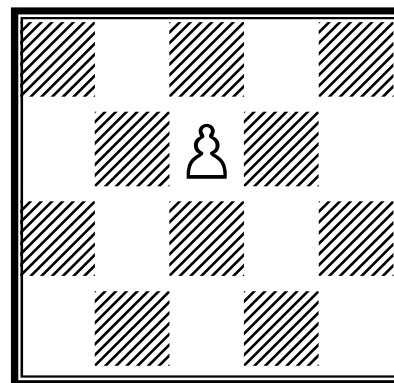
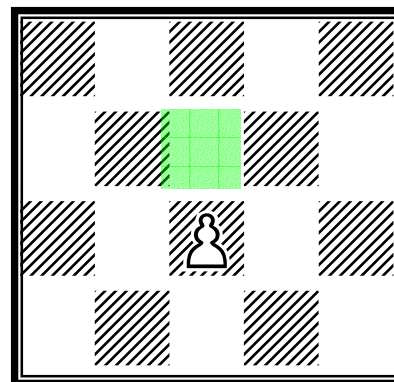
Goal: checkmate your opponent's King

The Pawn

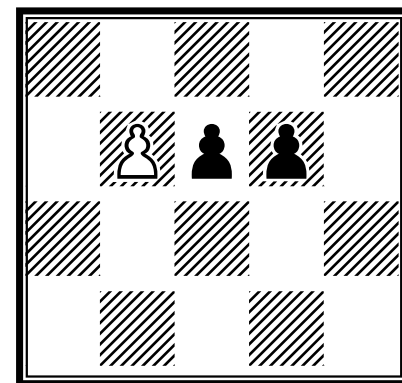
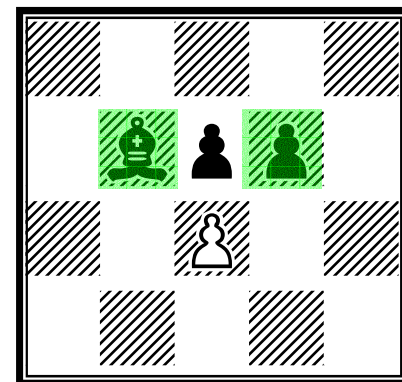
Initial Move



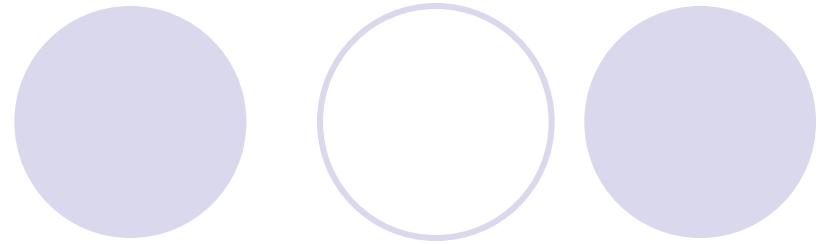
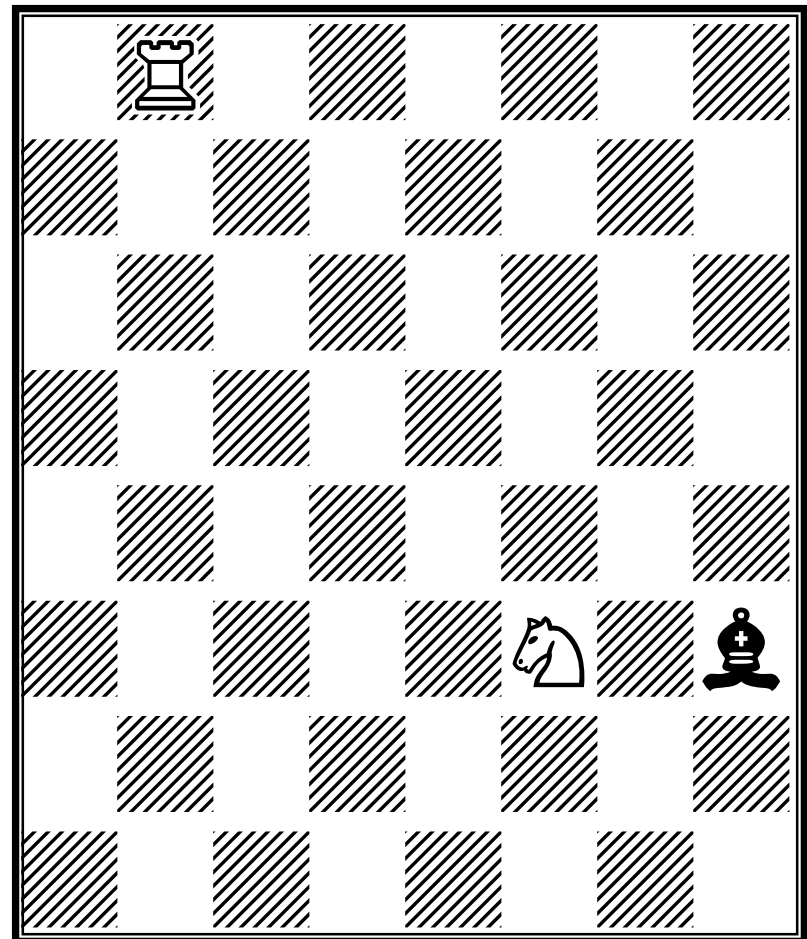
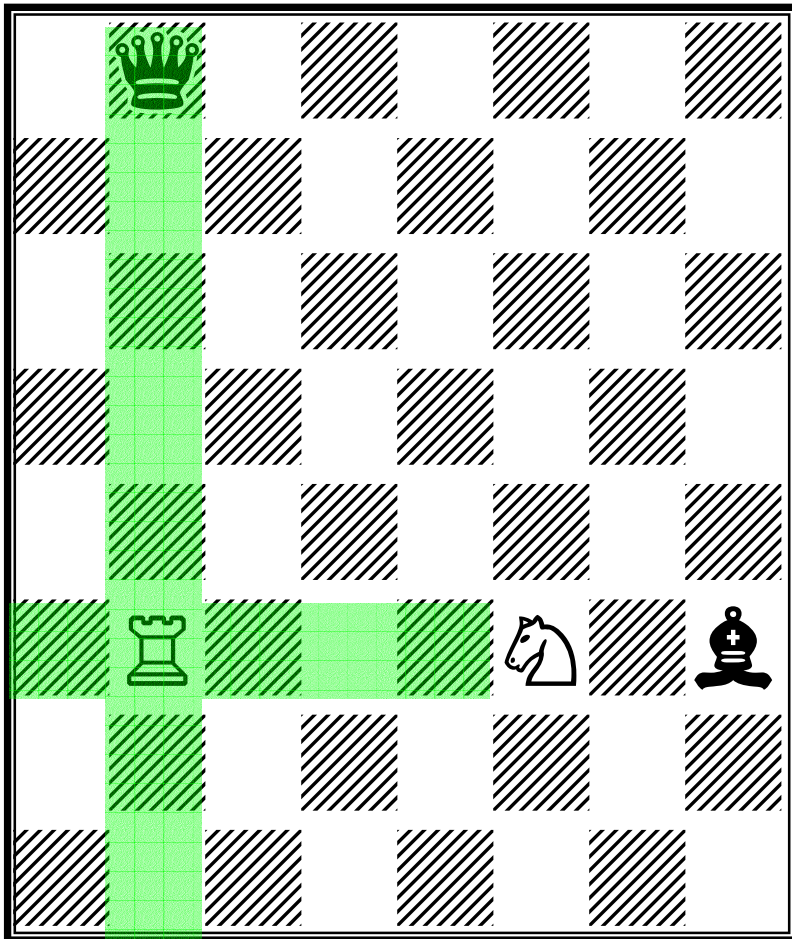
Normal Move



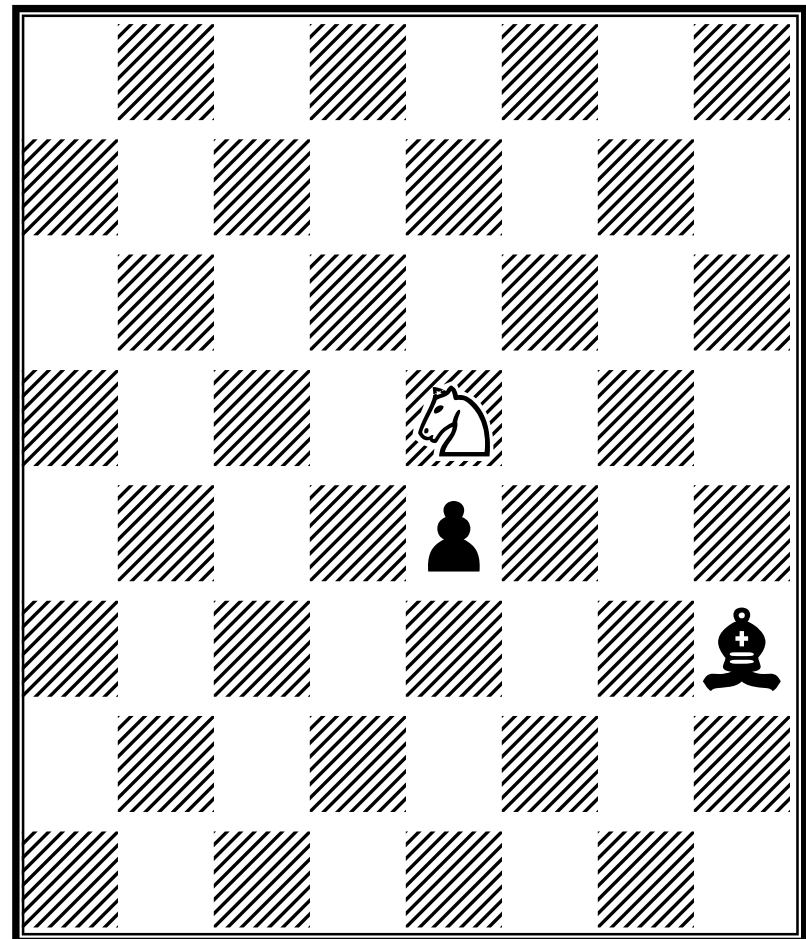
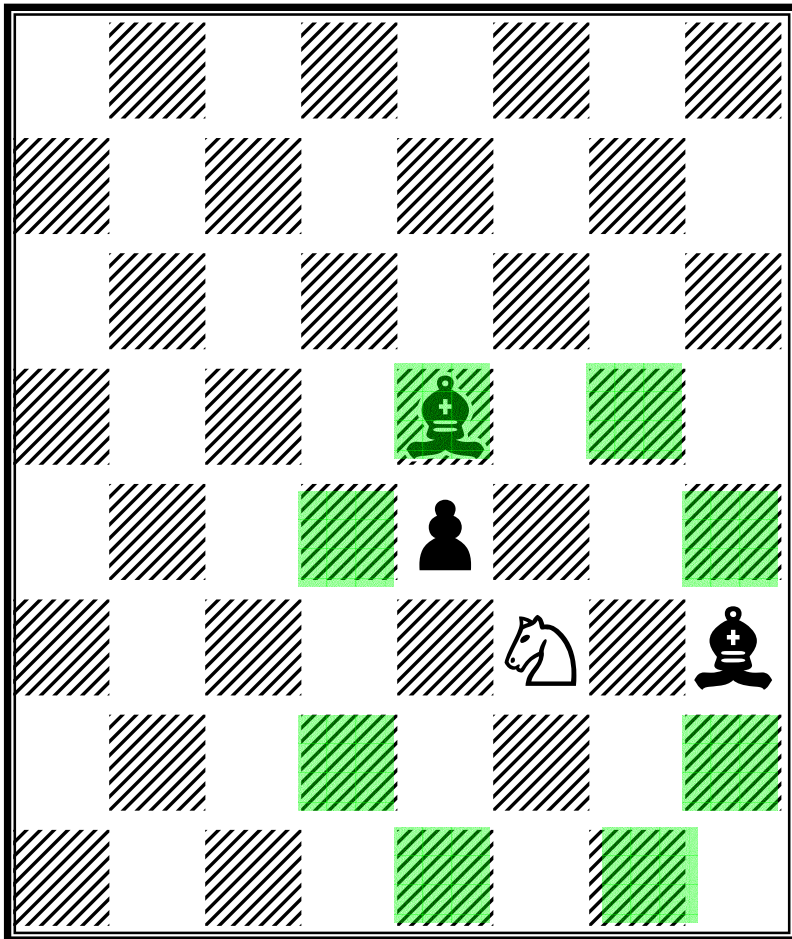
Attack



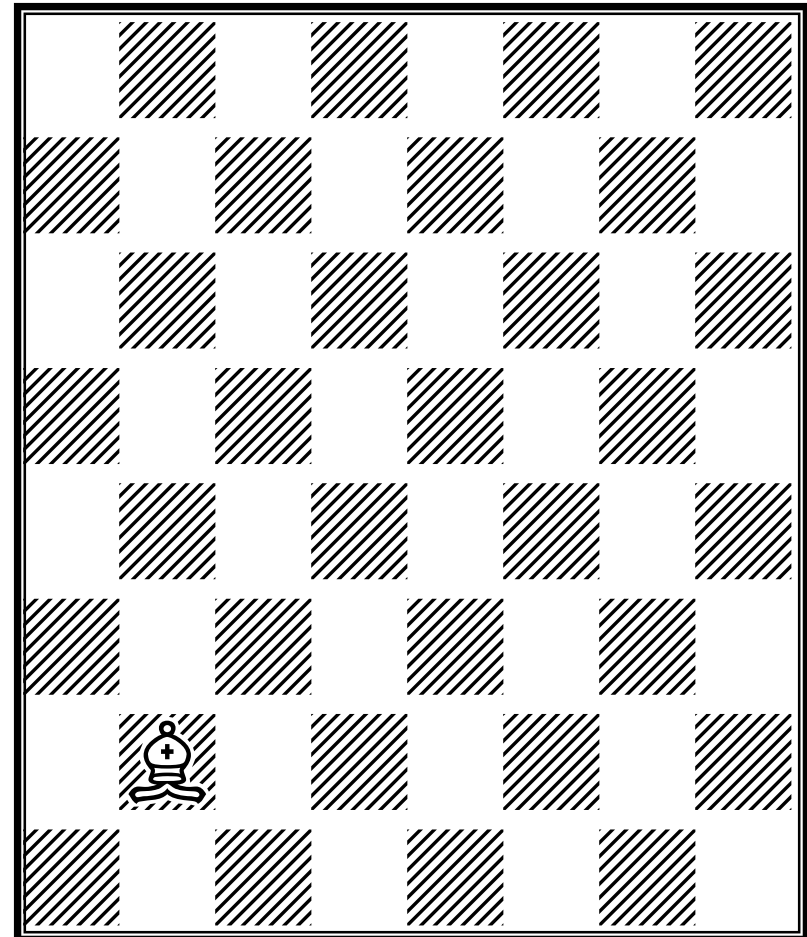
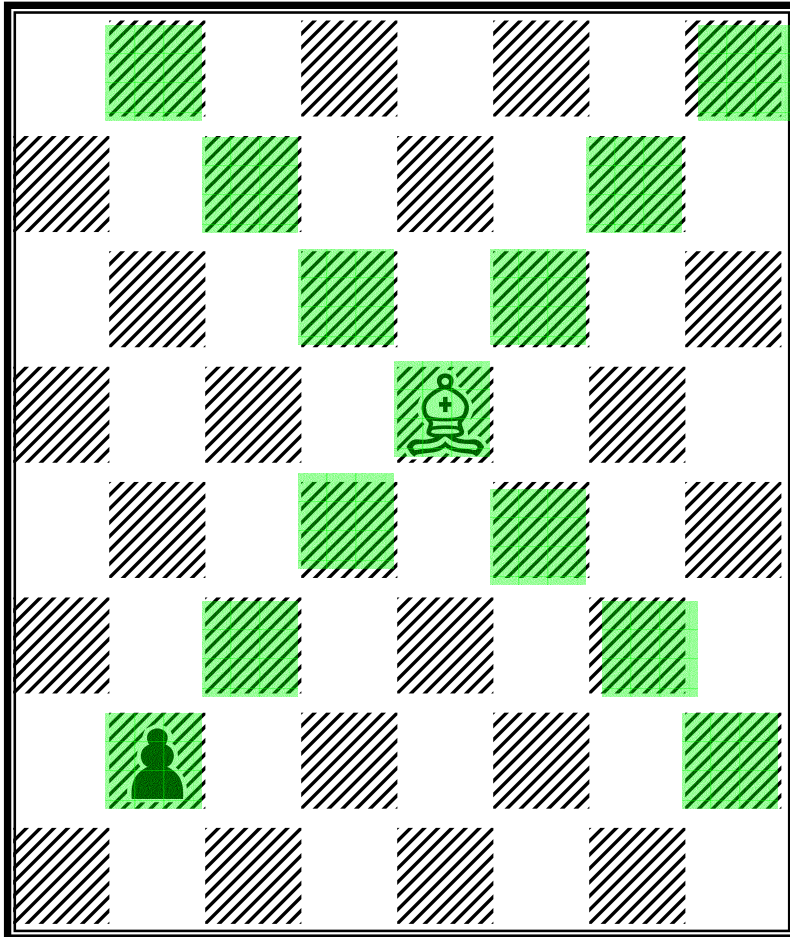
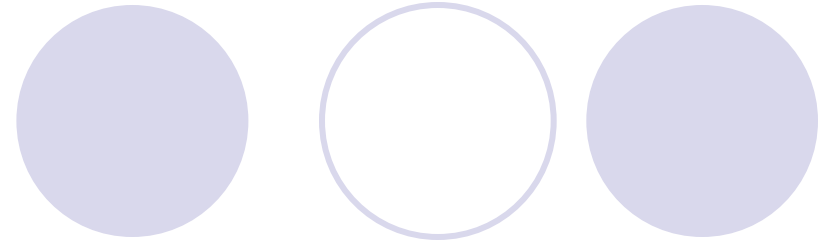
The Rock



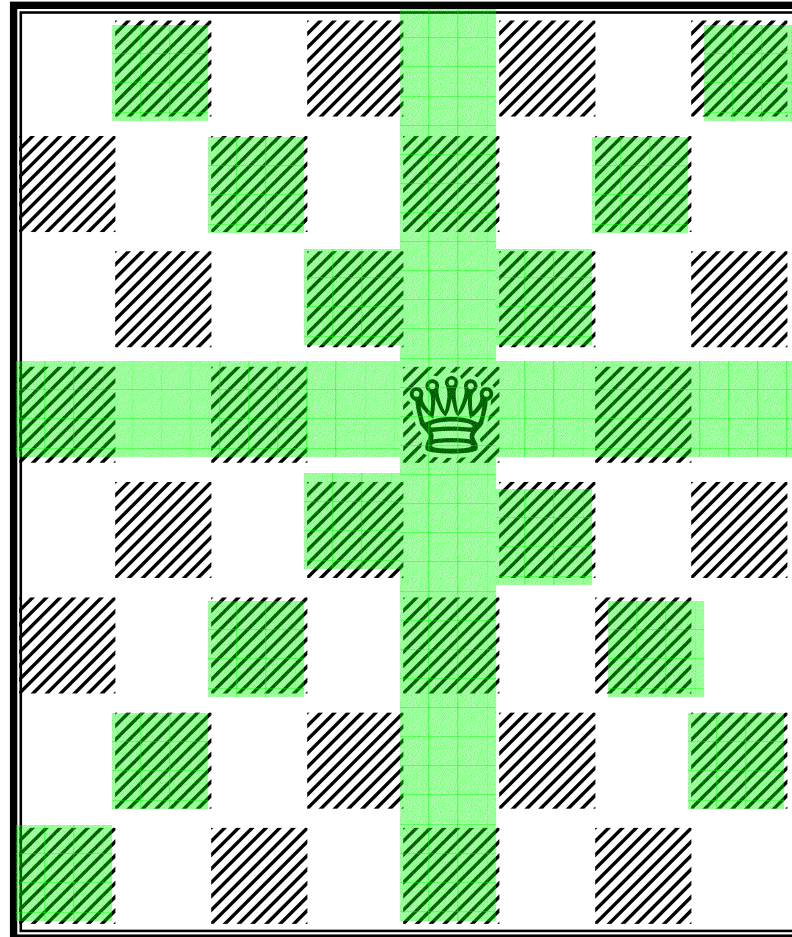
The Knight



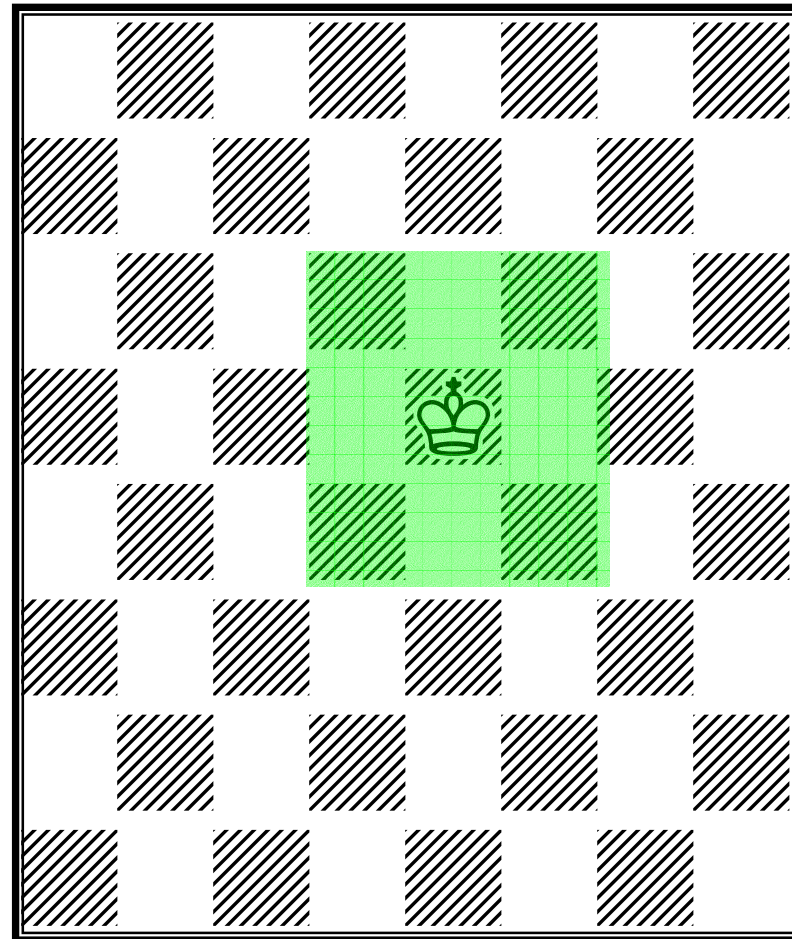
The Bishop



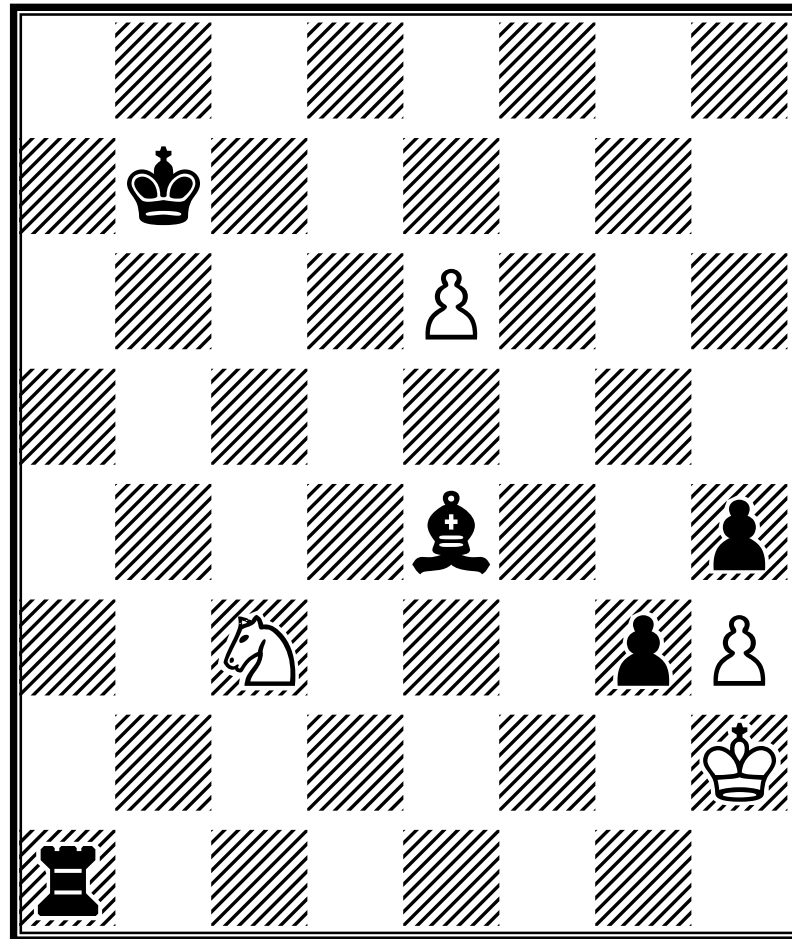
The Queen









The King

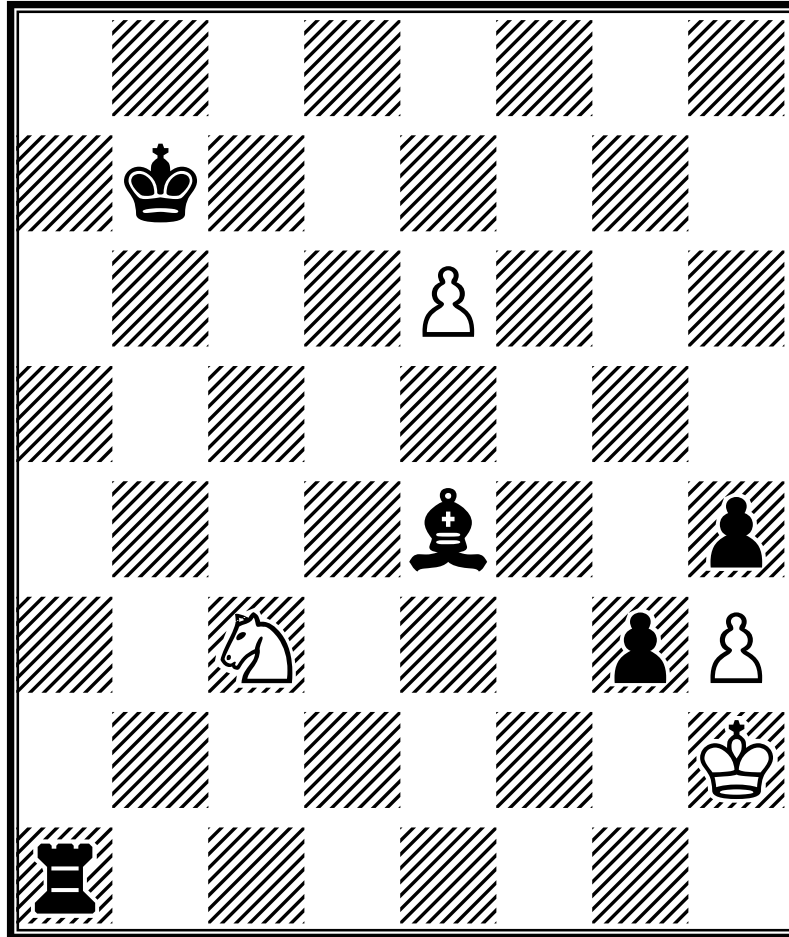


Goal of Chess: checkmate



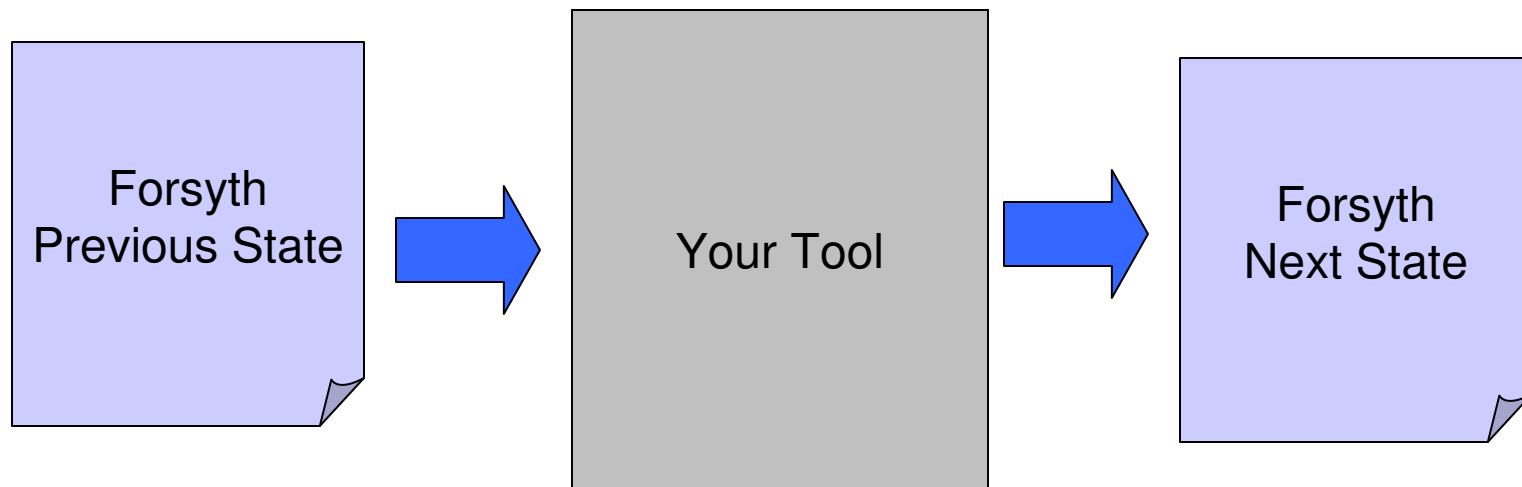
The Forsyth notation

-  r
-  n
-  b
-  q
-  k
-  p



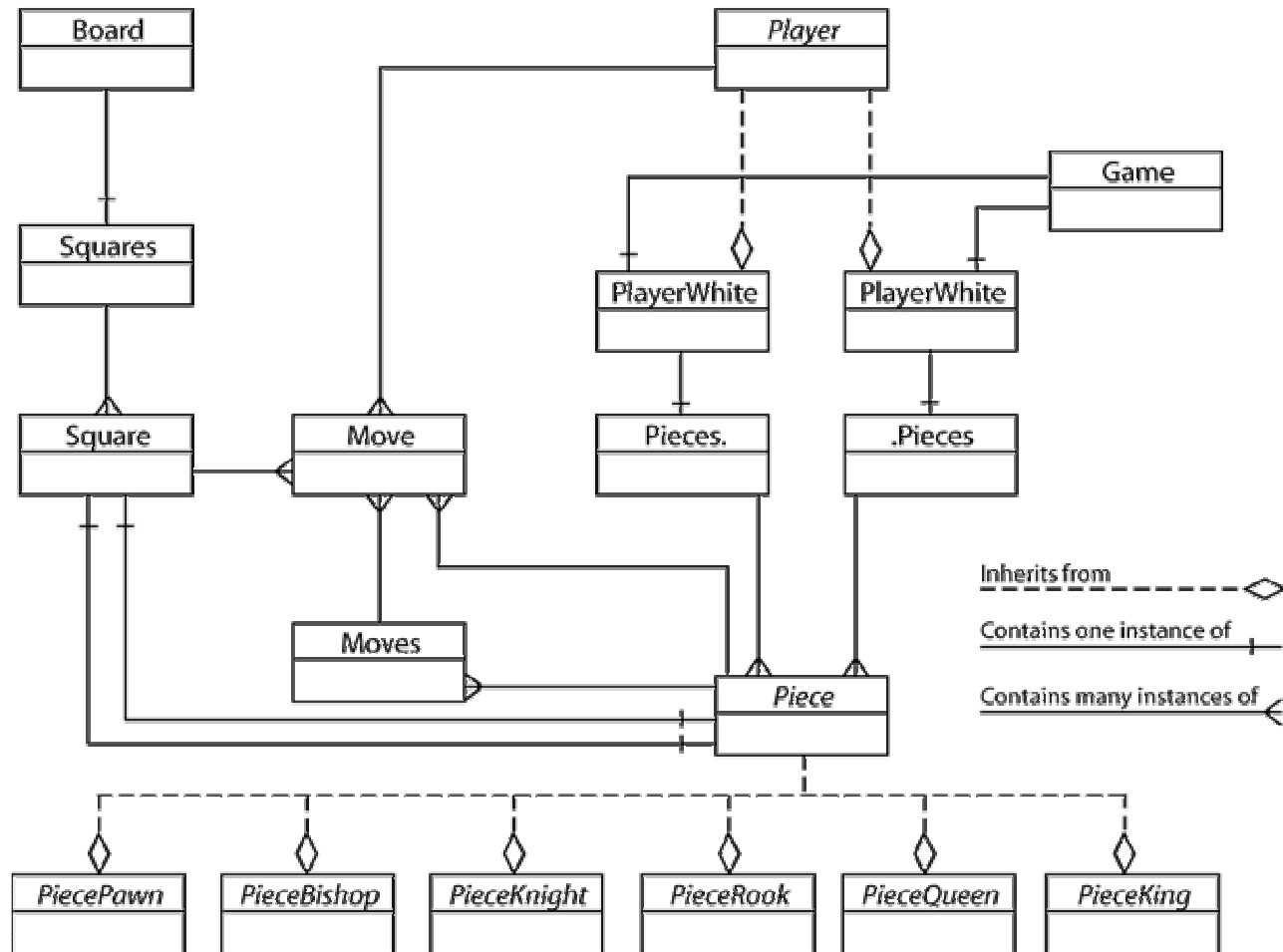
8;
 1K6;
 4p3;
 8;
 4B2P;
 2n3Pp;
 7k;
 R7.

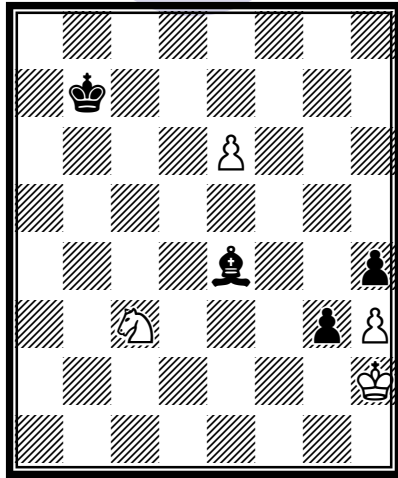
Requirements of the assignment



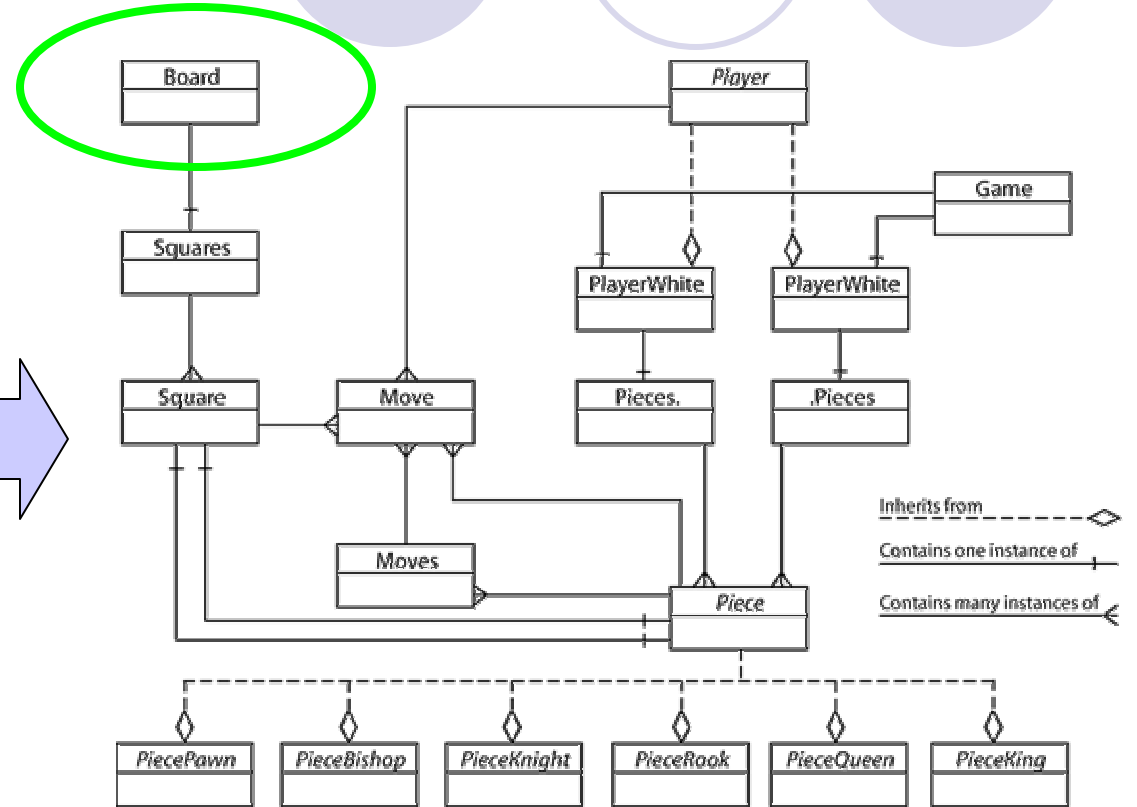
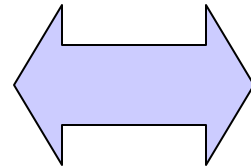
- **Play for the whites**
- Legal Move
- MinMax
- Computation time reasonable

Recommended Model for Chess

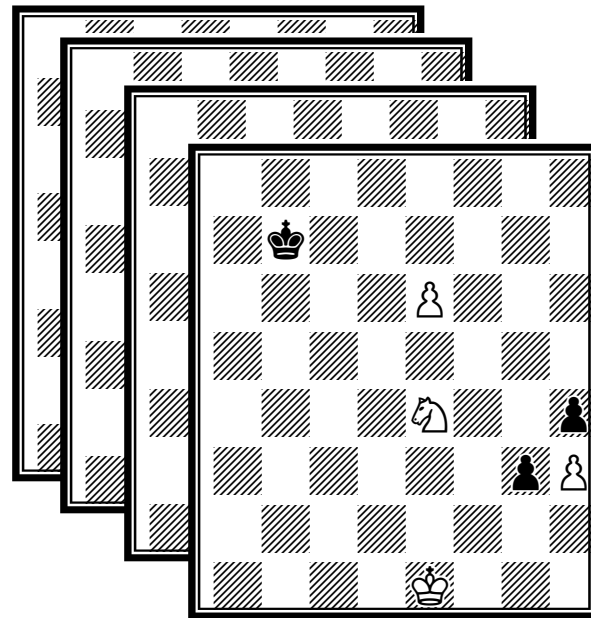
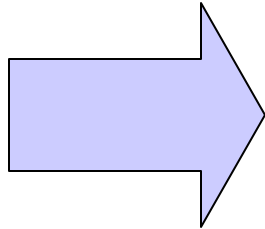
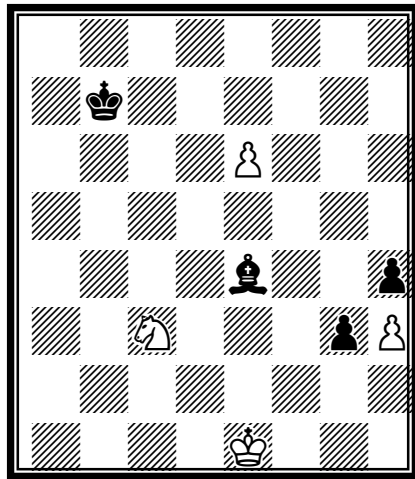




State of the game

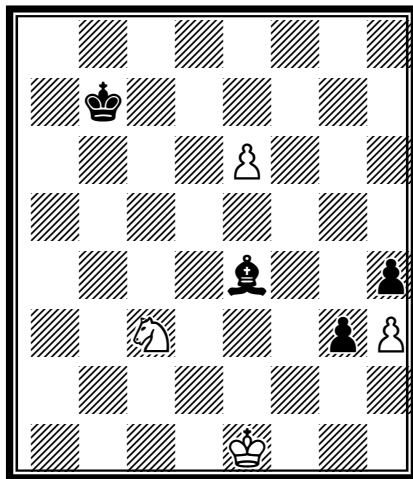


Next-State Computation

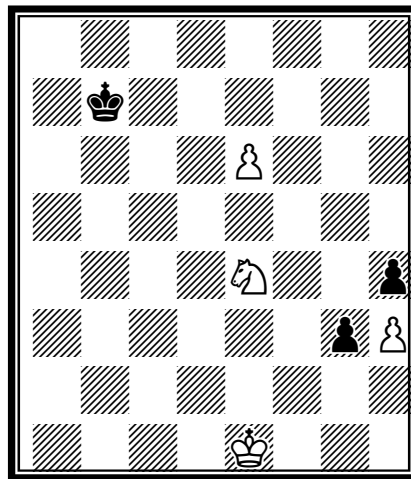


Evaluation Function

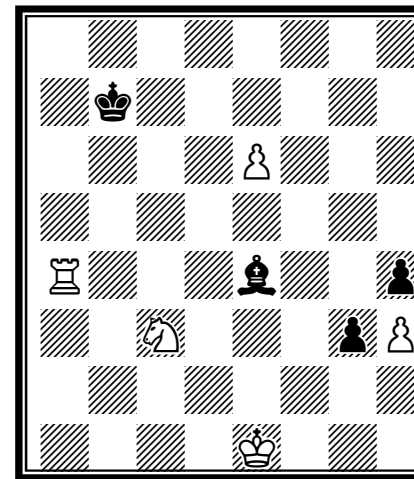
0



3



5



Infinity



3



9



3

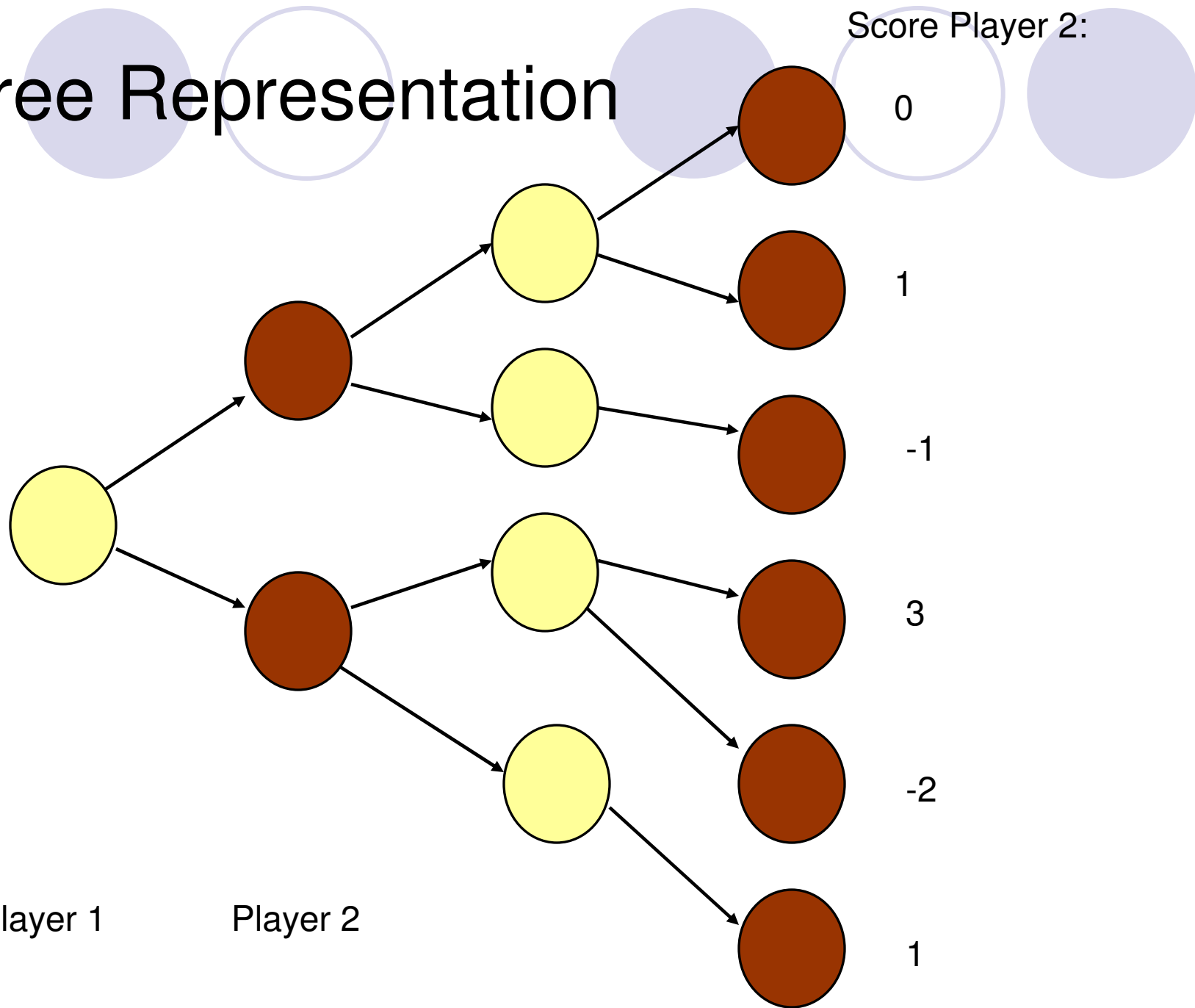


5



1

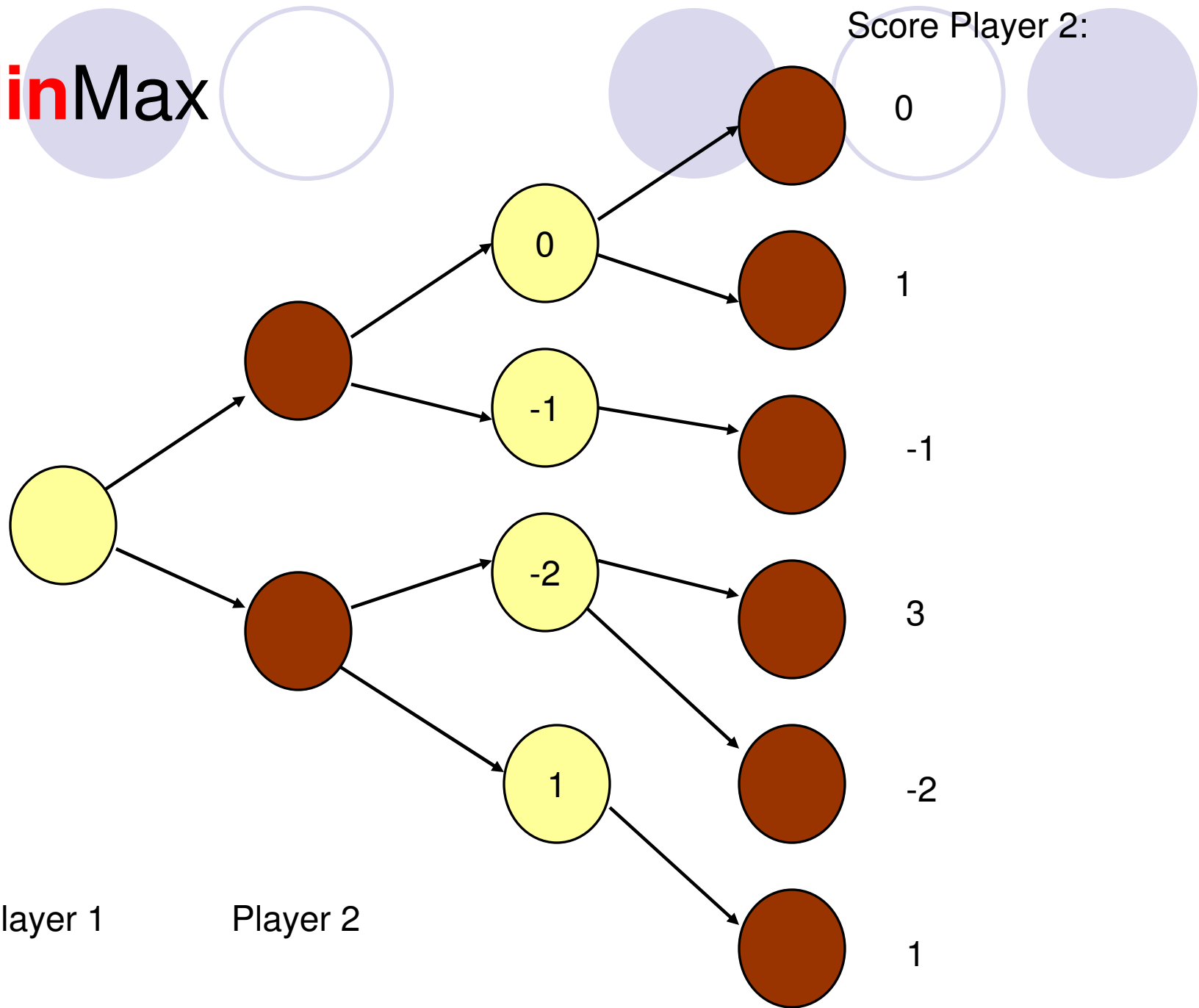
Tree Representation



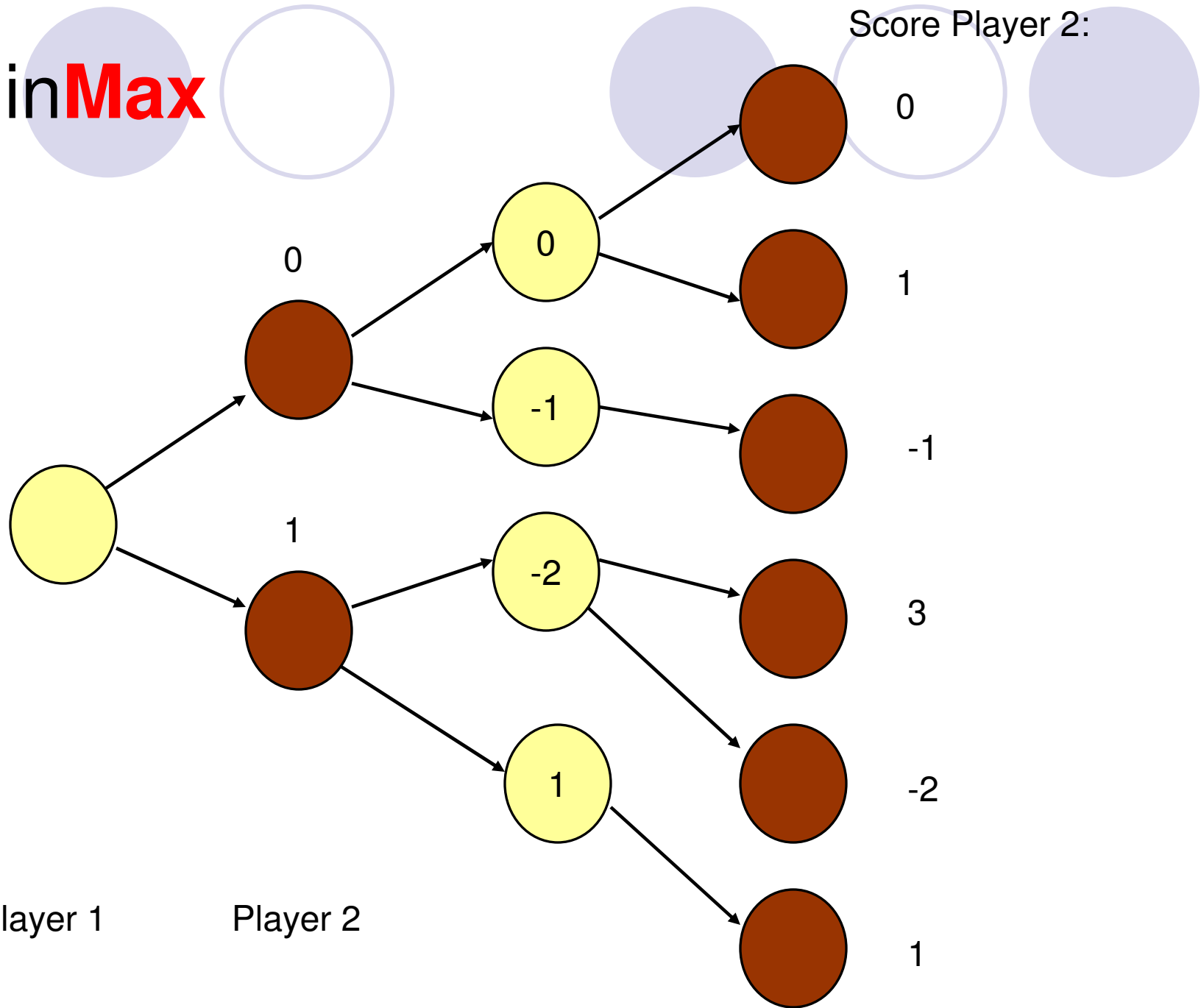
MinMax: *minimizing the maximal lost*

```
int minmax(node, depth)
{
    if (node.IsTerminal || depth == 0)
        return evaluate(node);
    if (node.WePlay)
    {
        int e = MAXINT;
        foreach child of node
            e = min(e, minmax(child, depth-1))
        return e
    } else // It's the turn of the opponent
    {
        int e = MININT;
        foreach child of node
            e = max(e, minmax(child, depth-1))
        return e
    }
}
```

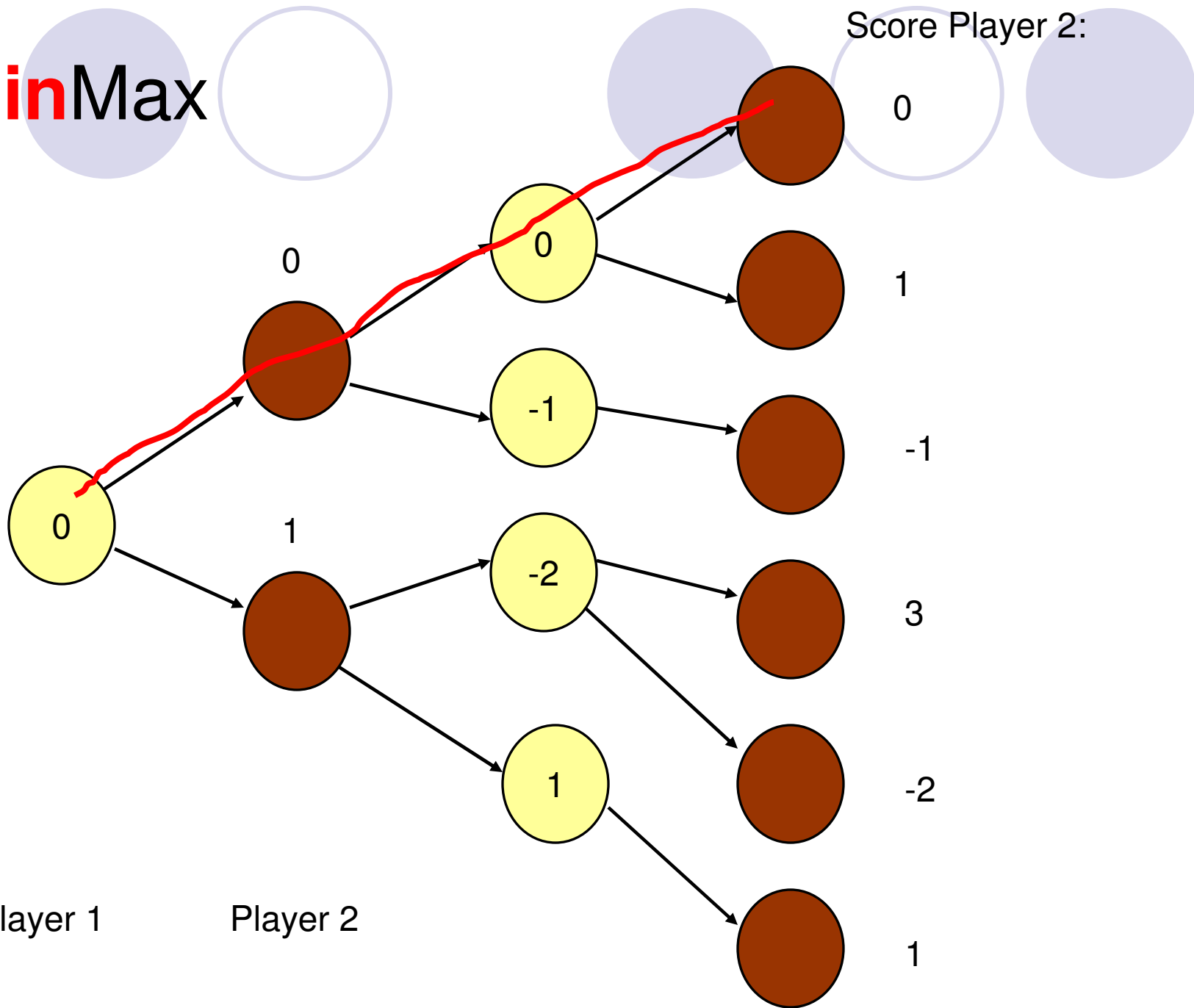
MinMax



MinMax



MinMax



Improvements



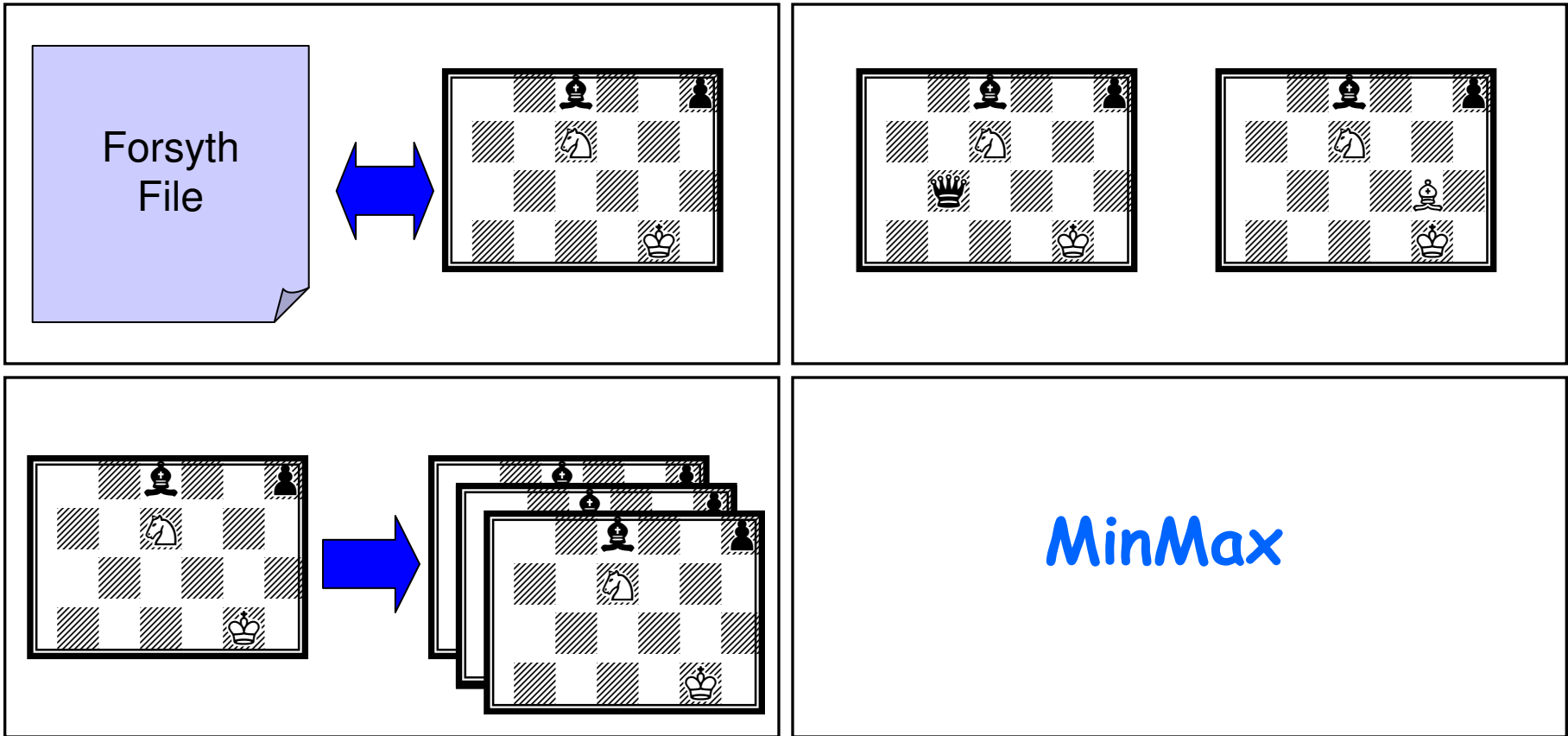
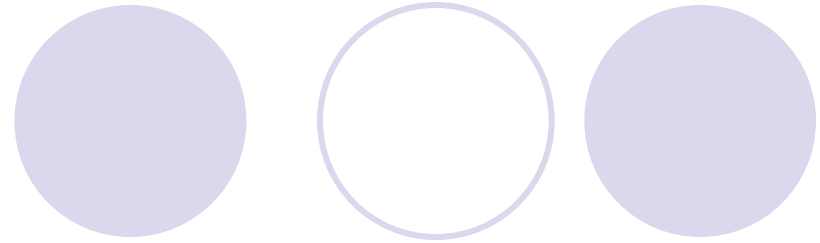
- Exploration depth: deeper = better

Huge state space:

- Alpha-beta algorithm
- + promising childs first
- Pruning techniques

- It's your job to google.

Conclusion



Questions?

