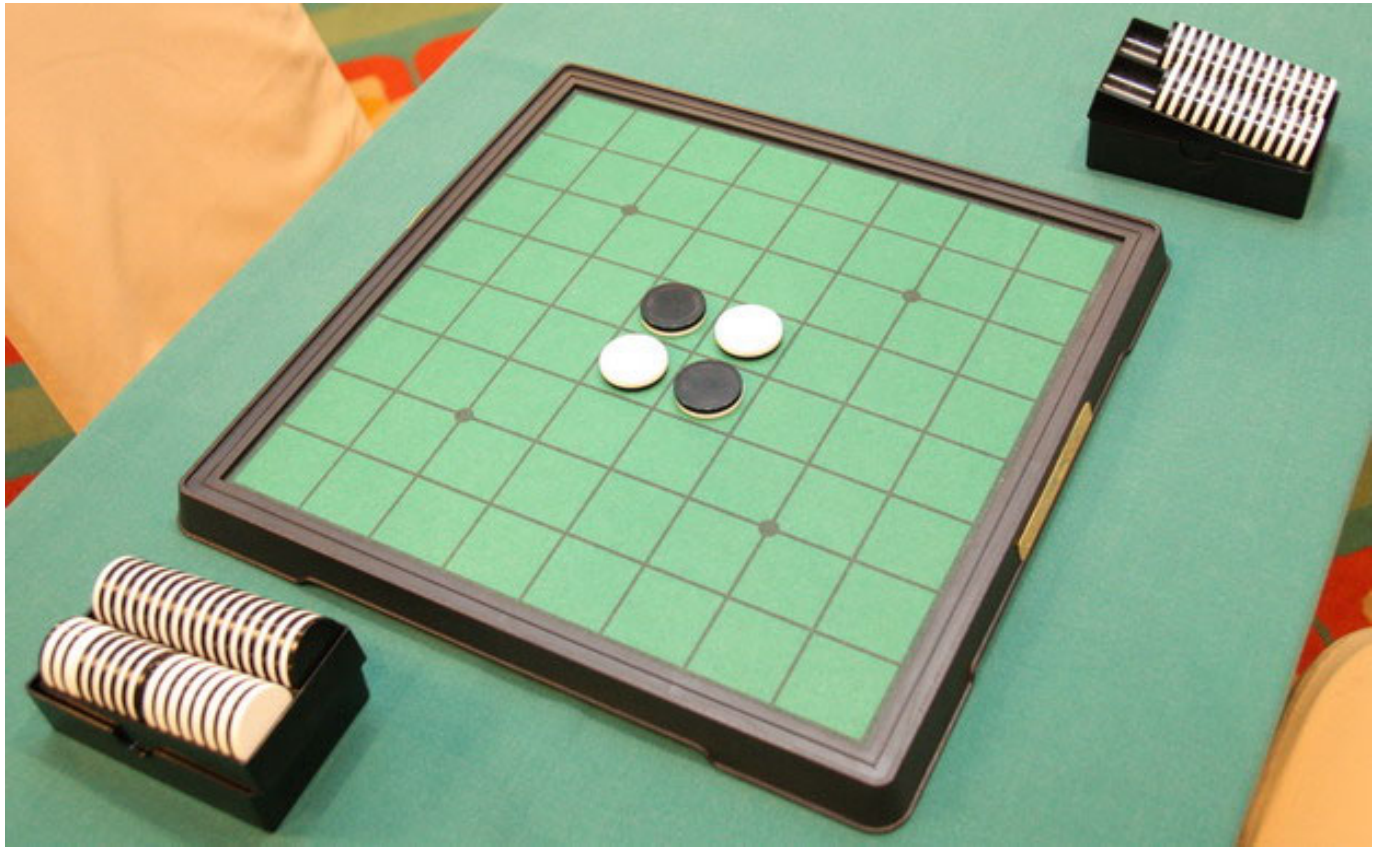
Writing a Computer Player?

- a brief introduction to artificial intelligence

(exam excluded but project included)

Before you start…



If you don’t know how to play this game: http://www.othelloonline.org/

Version 1: Dumbest AI ever!

- This is the **dumbest Othello computer player**!

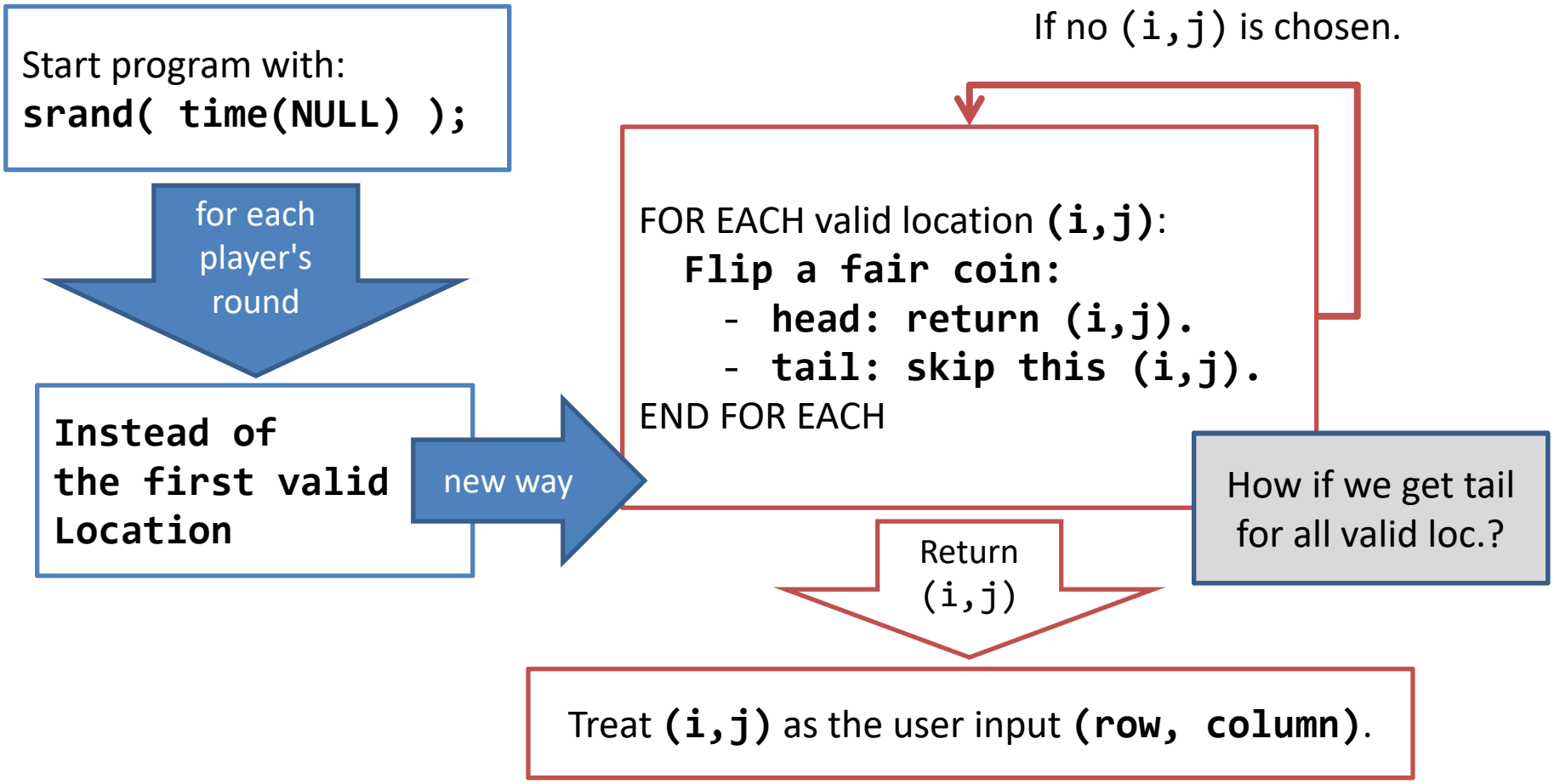
```
1  int dumb_AI_player( int  player    ,
2                      char map[8][8] )
3  {
4      int i , j ;
5      for ( i = 0 ; i < 8 ; i++ )
6          for ( j = 0 ; j < 8 ; j++ )
7              if ( map[i][j] == ' '
8                && is_valid_input( ... ) )
9              {
10                 // just the first valid choice
11
12                 return ...; // position at (i,j)
13
14                 // nothing to do anymore
15             }
16     ...
17 }  // end function
```

Function:
is_valid_input()

It should decide if the location **(i,j)** is a valid move.

Note: the function prototype here is different from that in the project spec.

Add in some randomness?



Version 2: Still Dumbest AI ever!

- In case you want some **randomness**...

```
1  int dumb_AI_player( int  player    ,
2                      char map[8][8] )
3  {
4     int i , j ;
5     while ( 1 )
6     {
7        for ( i = 0 ; i < 8 ; i++ )
8           for ( j = 0 ; j < 8 ; j++ )
9              if ( map[i][j] == ' '
10               && is_valid_input( ... ) )
11             {
12               if ( rand() % 2 == 0 )
13                  return ...; // position at (i,j)
14             }
15    } // end while
16    ...
17 }  // end function
```

Now, you should know why we need "**while(1)**" here?

rand() % 2

Produce the result of "*flipping a fair coin*": either 0 or 1.

How dumb the AI player is...

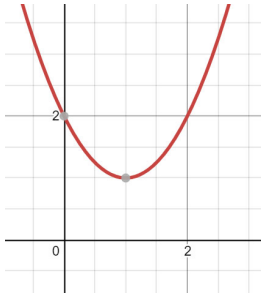
- The previous AI players can play the game … but
  - They can't play intelligently… why?
- It just shows you **a way** to play the game until the end, but obviously, **NOT one of the best ways**.

What is “Optimization” ?

- Optimization is similar to the decision problem.
  - Decision: ***look for the existence of a solution***
  - Optimization: ***look for the best solution***
    - *Maximize/minimize something subject certain choices (search space)*

- What does it mean by "***the best***"?
  - In optimization problem , one usually define an ***objective function*** to measure how happy you are with a solution.
  - For each solution, pass it into the objective function.
  - Look for the ***best objective value / score***, either minimizing/maximizing this score value, e.g.,

    What is the value of x to minimize "$f(x) = (x-1)^2 + 1$”?



Optimization

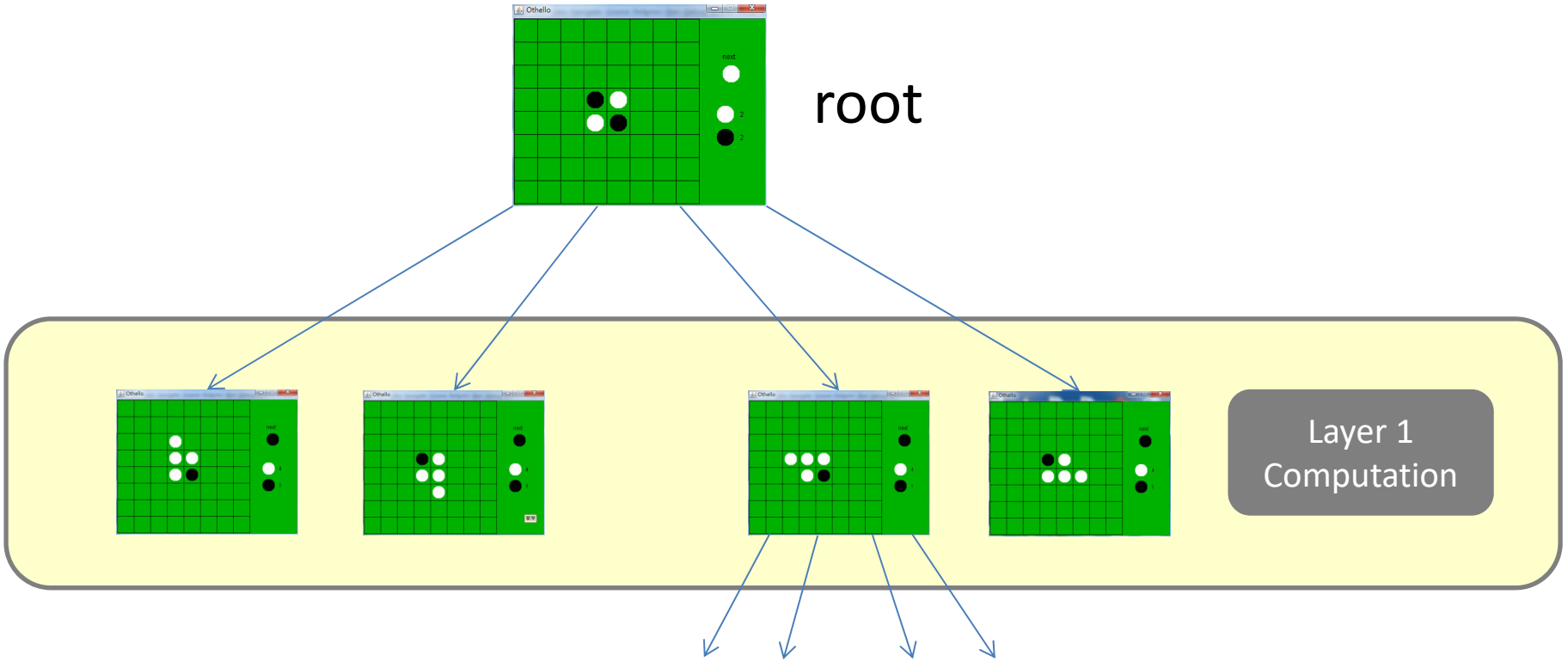
- The board-based game is a typical optimization problem.

- The challenges lie in:
  (a) The **solution space** is usually HUGE!
  (b) The definition of the **objective function** (or the **scoring** function) is usually hard! Think about chess games!!!

- Before you start to code AI player, can your code find **all possible valid moves** in the current game board!!!

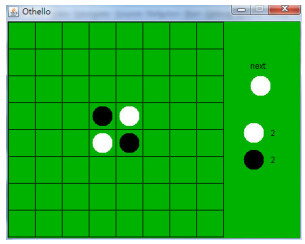
The solution space

- In a chess game, we can arrange the solution space in a layered computation as a tree structure.



The solution space

- For Othello, the worst case to consider 60 layers of computation, which is computationally intractable!

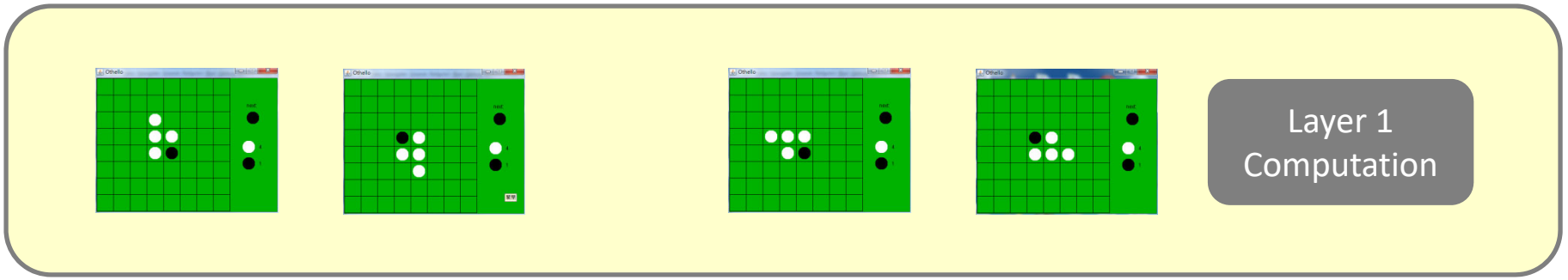


60
empty slots

- How to cut down the solution space?
  - ***Compute fewer layers***, even one layer only.
  - Not until the game end.
  - Thus, a computer player will make a choice based on **an incomplete search** (so you may still beat the AI player ☺)

The objective function

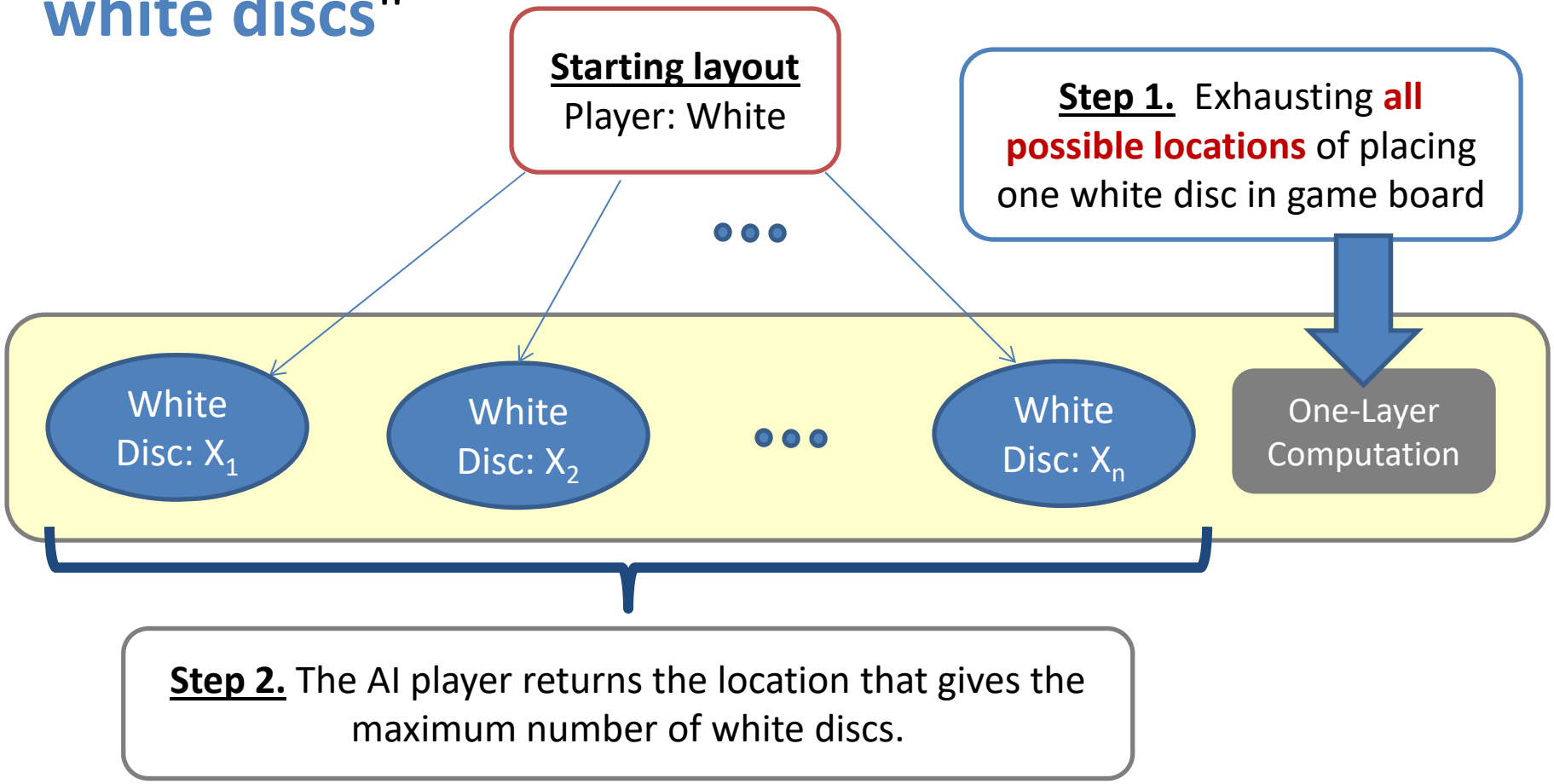
- To define how favorable a layout is?



  - In the 1st level, none of the four layouts is outstanding.
  - In other words, the objective function should ***compute the same scores*** for all the above four layouts.
  - May be: ***count the number of white discs***.

The combined solution for AI player

- "**Compute one layer**" + "**Count the number of white discs**"



A short summary

- Simulating one step is easy.

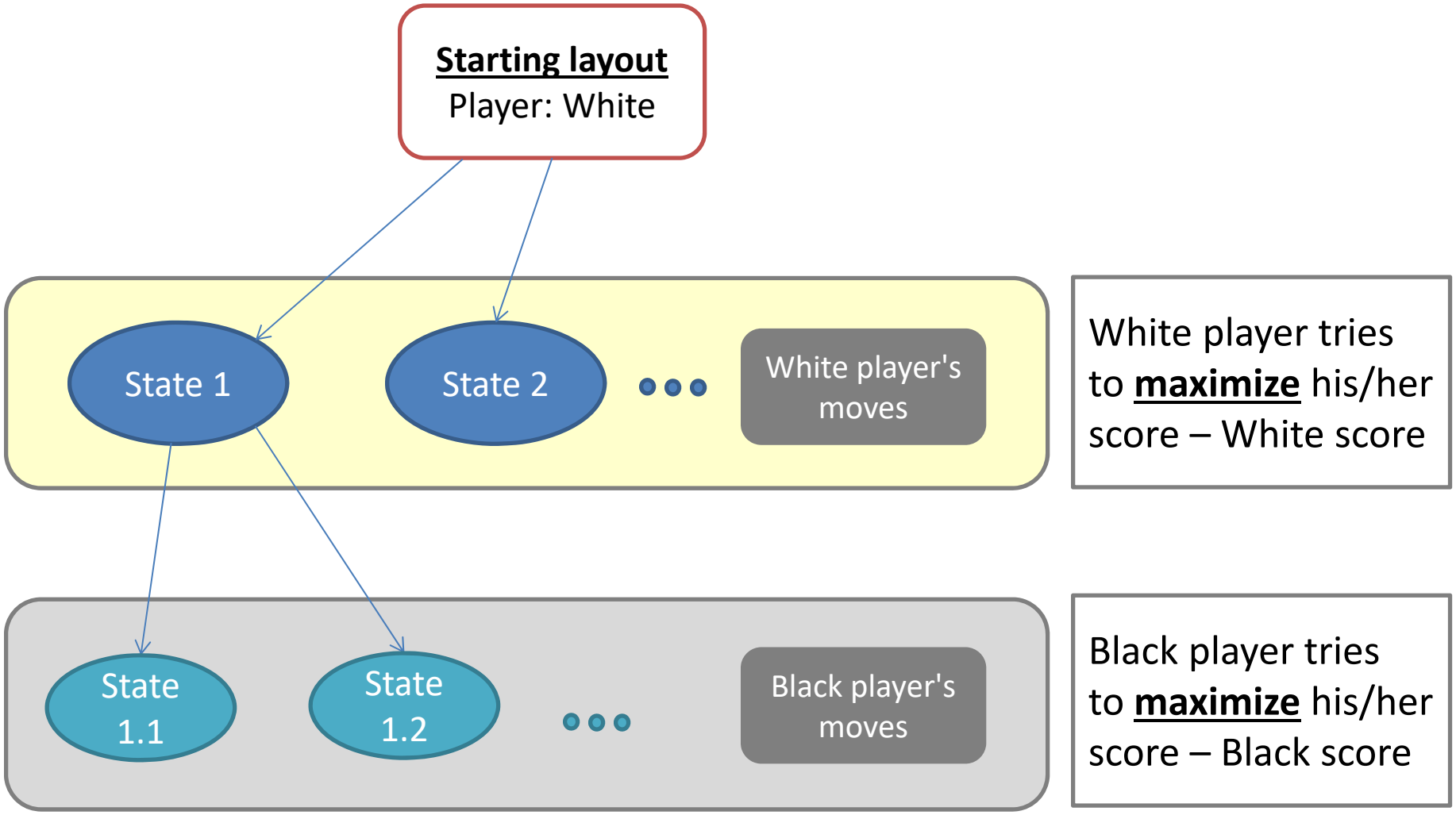
- Yet, the difficulty is how to formulate a good objective function.
  - As a matter of fact, the method we shown previously is ***a very simple one***, usually called a **greedy algorithm**!
    - Bad at the beginning of the game
    - But good, near the end of the game
  - Can you think of a better strategy and also a better objective function?

Can we have a stronger AI player?

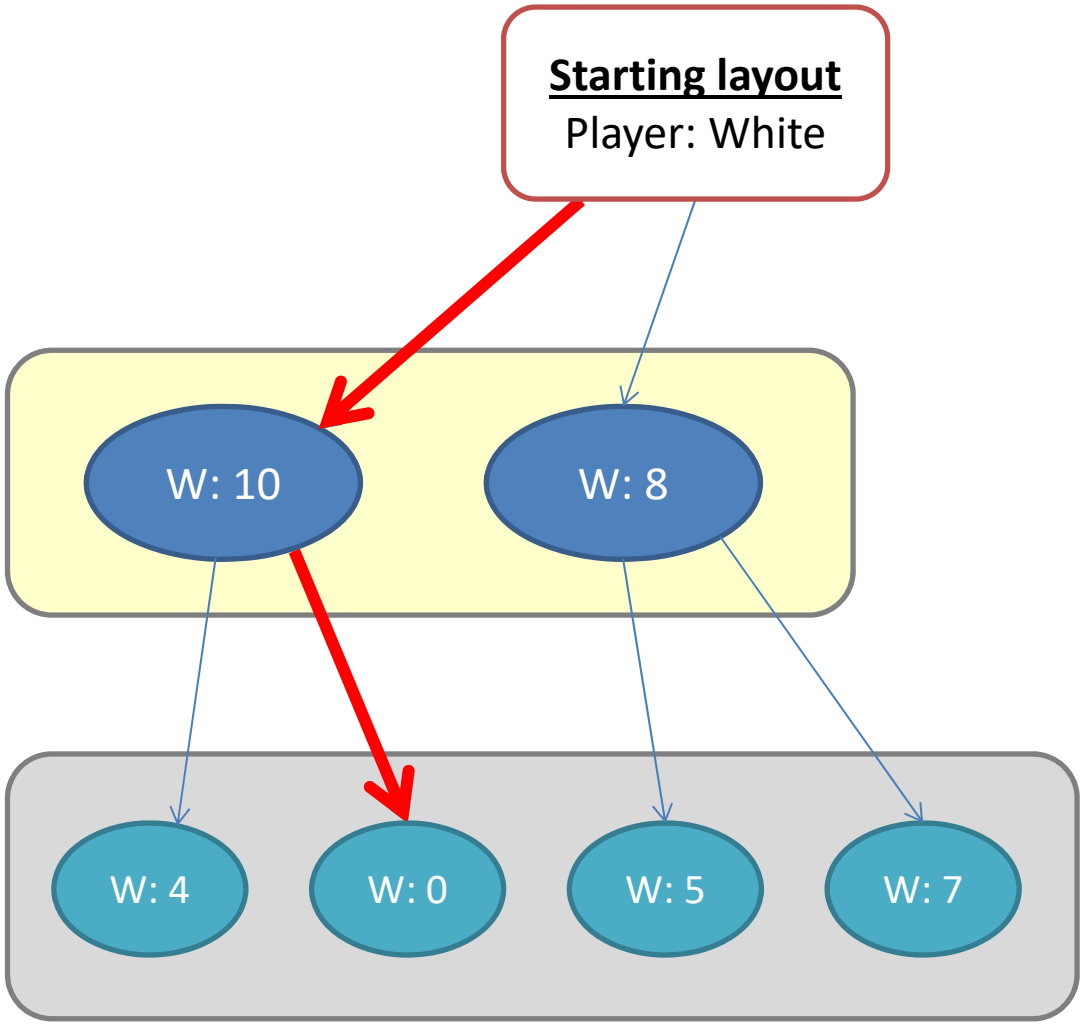
- One basic strategy is **min-max searching algorithm**.
  - In a nutshell, it ***simulates both players***: each player aims for ***the best possible move*** in each turn.
  - Technically, it tries to ***minimize the possible loss*** for a worst case (maximum loss) scenario.
  - In common words, it tries to ***avoid the opponent to make good moves***, while trying to make a good move

https://en.wikipedia.org/wiki/Minimax

Min-max searching – Basic idea



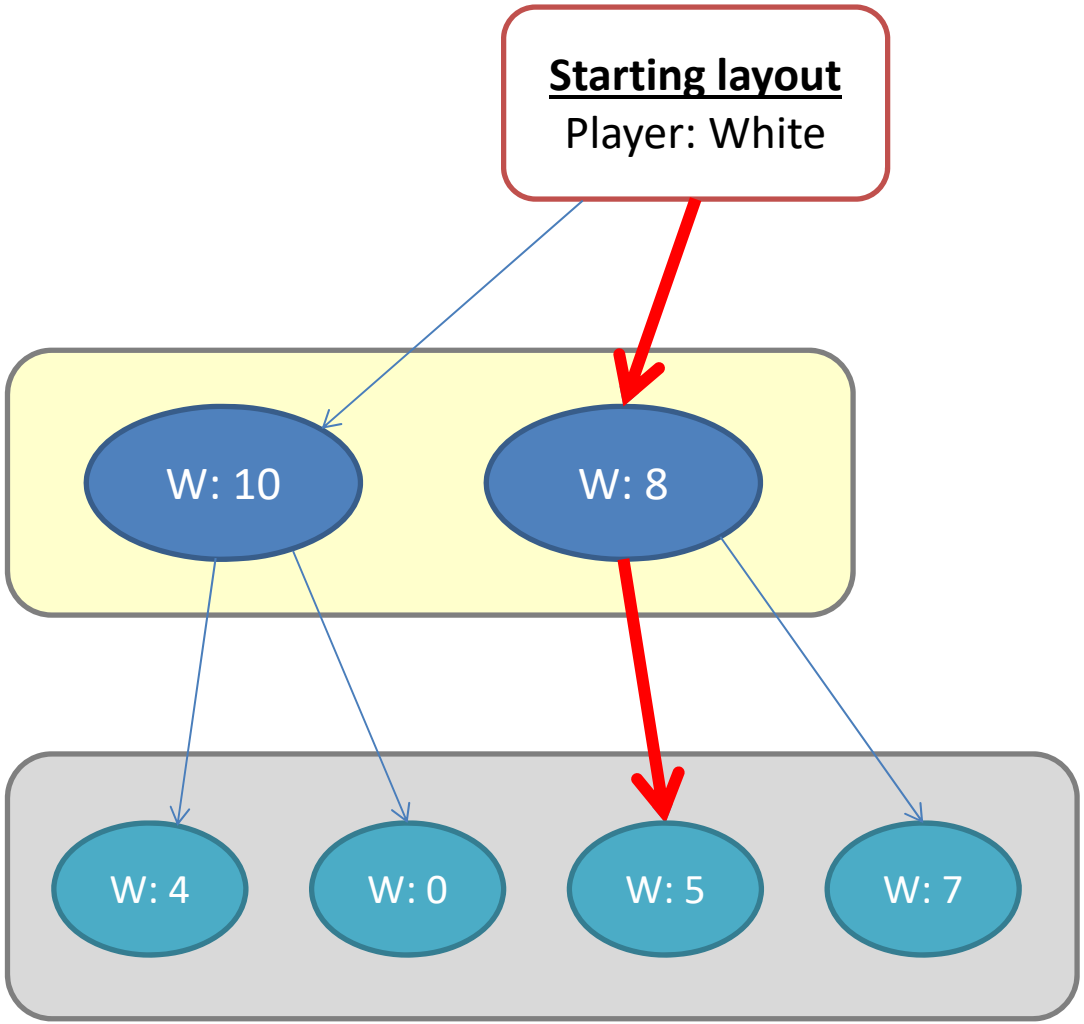
Min-max searching – Example (1 of 3)



If the white player chooses the left path:

- although he/she will get more white discs in the 1st round,
- he/she will eventually lose the game if black player picks the best move in 2nd round!

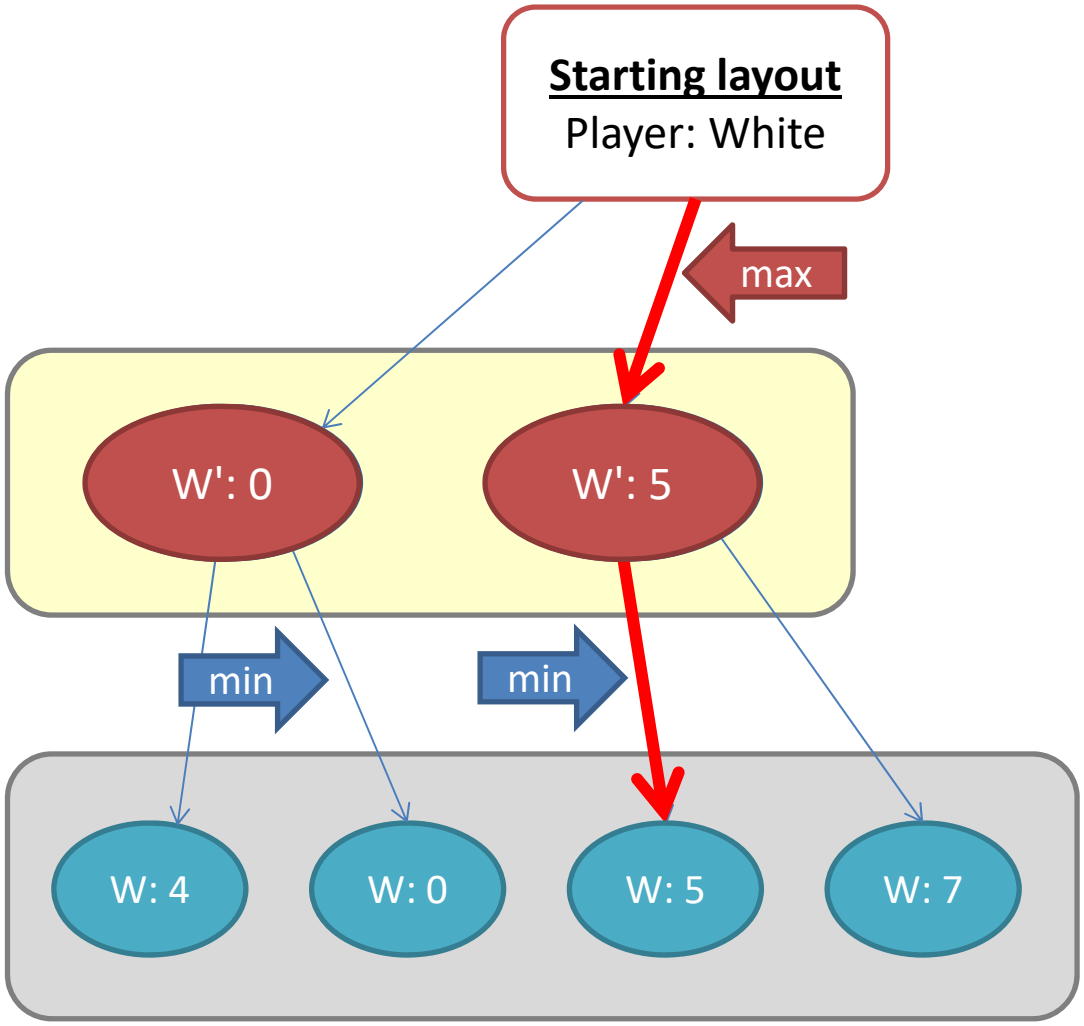
Min-max searching – Example (2 of 3)



If the white player chooses the right path:

- although he/she will get fewer white discs in the 1st round compared to that of the left path,
- he/she won't lose the game in the 2nd round!

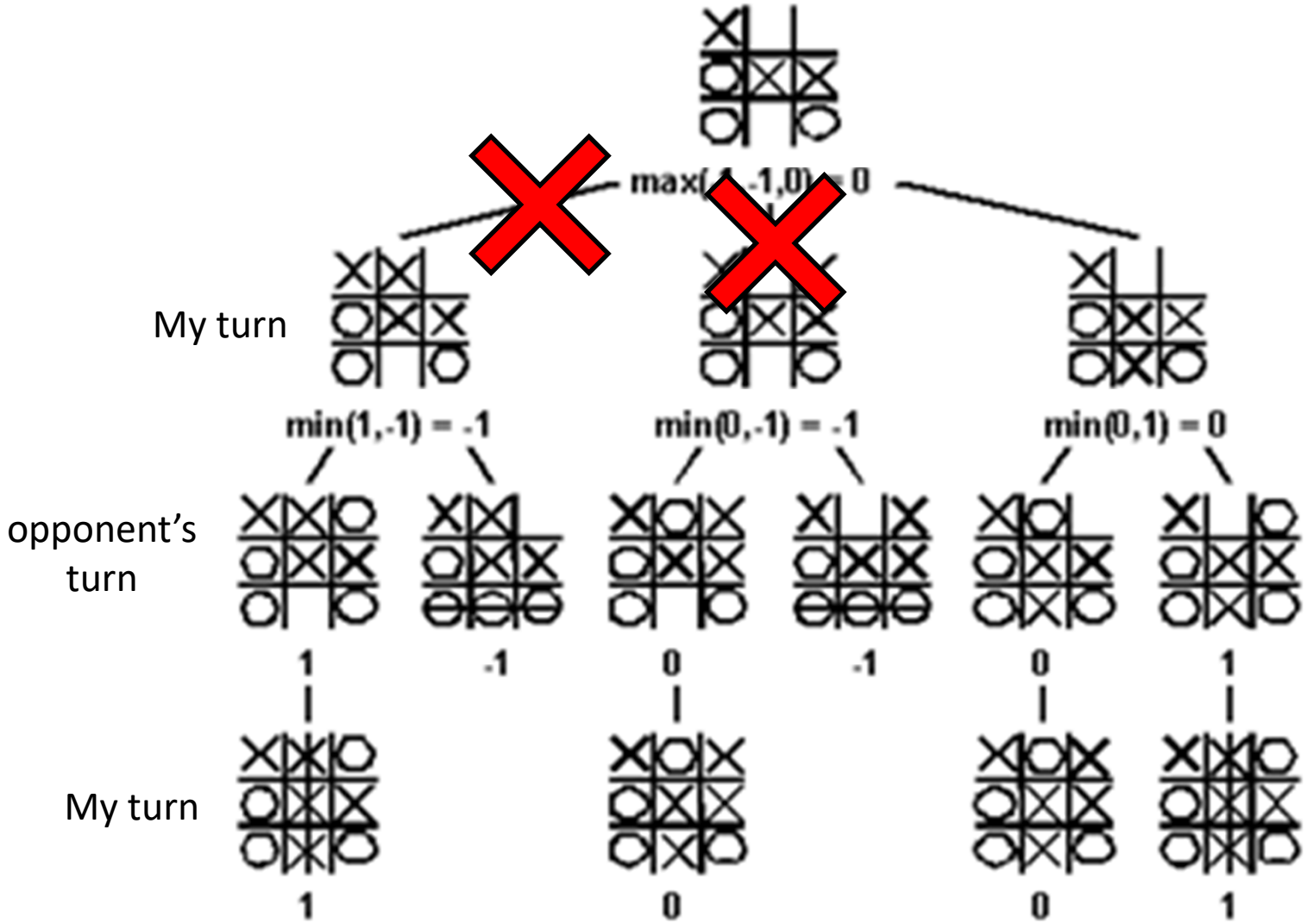
Min-max searching – Example (3 of 3)



Conclusion:

- Don't rely on the score obtained directly from the first layer.
- Instead, we *recursively* obtain the scores in the internal nodes from the bottom layers.

A simple example



From https://intellipaat.com/community/2630/tic-tac-toe-ai-how-to-make-the-tree

Summary

- Since the min-max searching algorithm is not a must in your project, we would not give you code.

- Some general hints:
  - Use **recursion** to search through different possible moves layer by layer, just like simulating different possible moves in the game play.
  - Calculate the score from bottom up with min & max.
  - Back up the game board inside the recursion function before taking any recursion to modify the game board.
    - Why? Because game board will be changed in the simulation.

If you like to know more...

- CSCI3230/ESTR3108 Fundamentals of Artificial Intelligence
  - A faster implementation on the Min-Max algorithm
  - Algorithm: Alpha-beta pruning and A* search
    - A path-finding algorithm
    - E.g., https://www.youtube.com/watch?v=DlkMs4ZHHr8
  - Rule-based, expert system;
  - Neural Networks: learning algorithms... etc.

- CSCI 3320: Fundamentals of Machine Learning
  - Looking for information in a set of data
  - It is actually a math-oriented course
  - Probability, statistics, a bit of optimization, programming, etc.

Note: searching alone is not sufficient to produce “Alpha Go”

In future, there will be tournament between AI players:

Google: “Alpha Star”: https://www.youtube.com/watch?v=6eiErYh_FeY