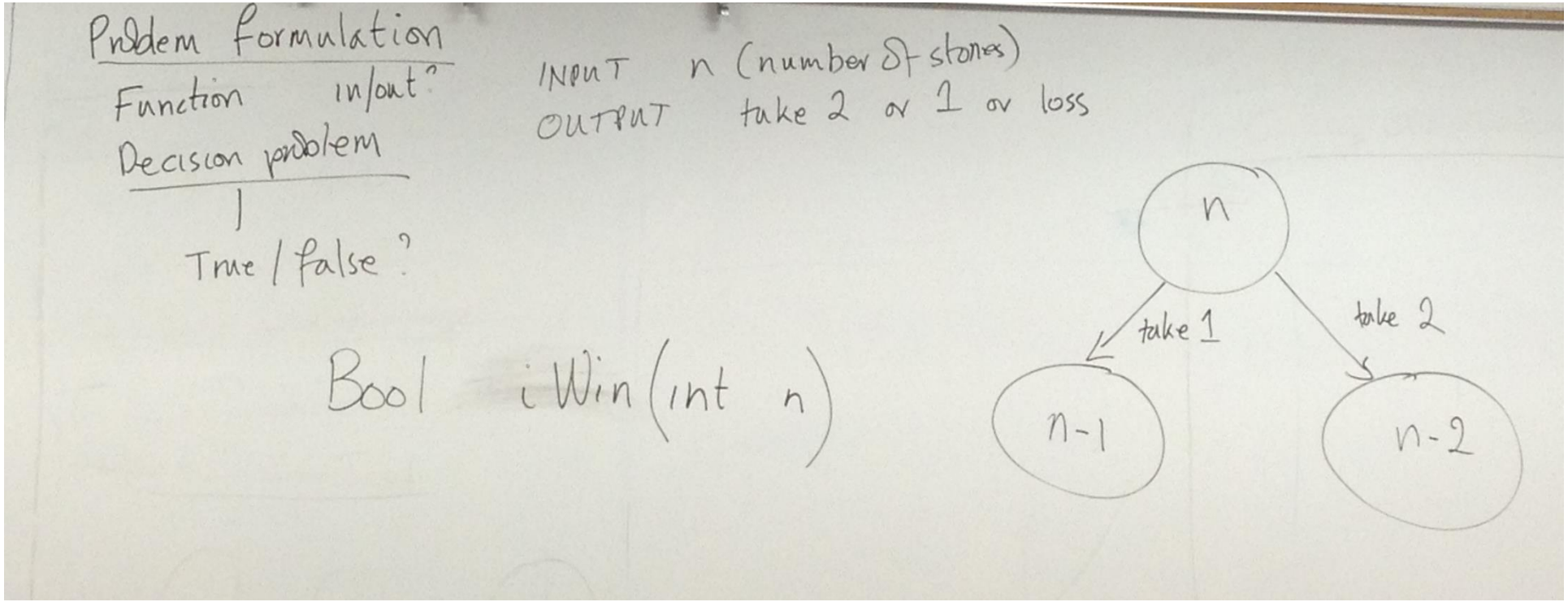
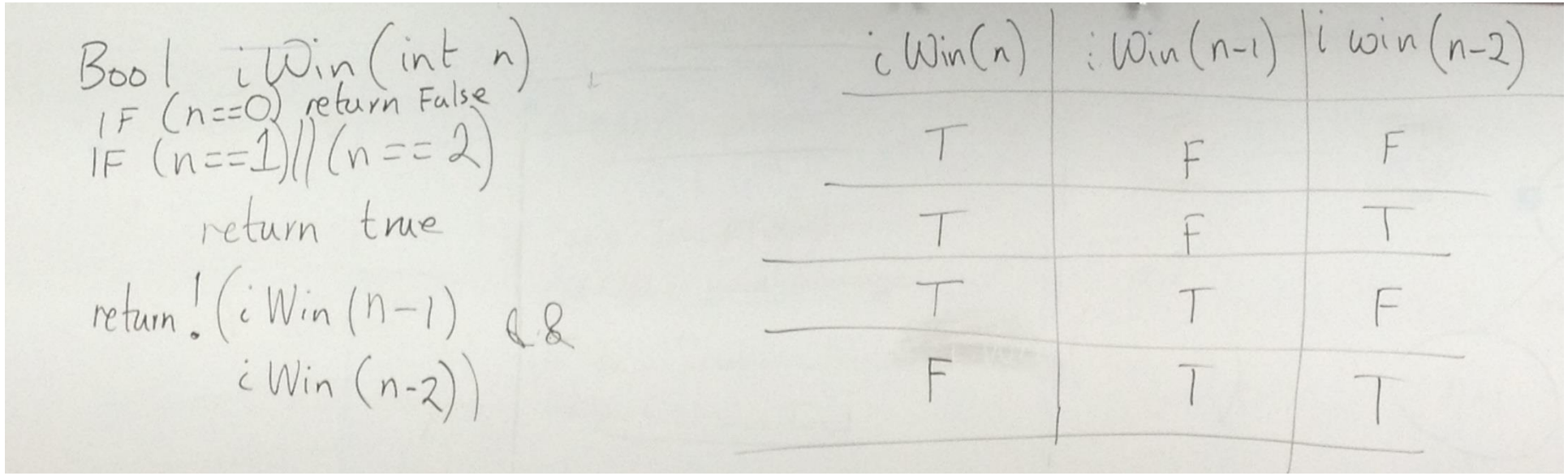
CS 5050

01-07-14

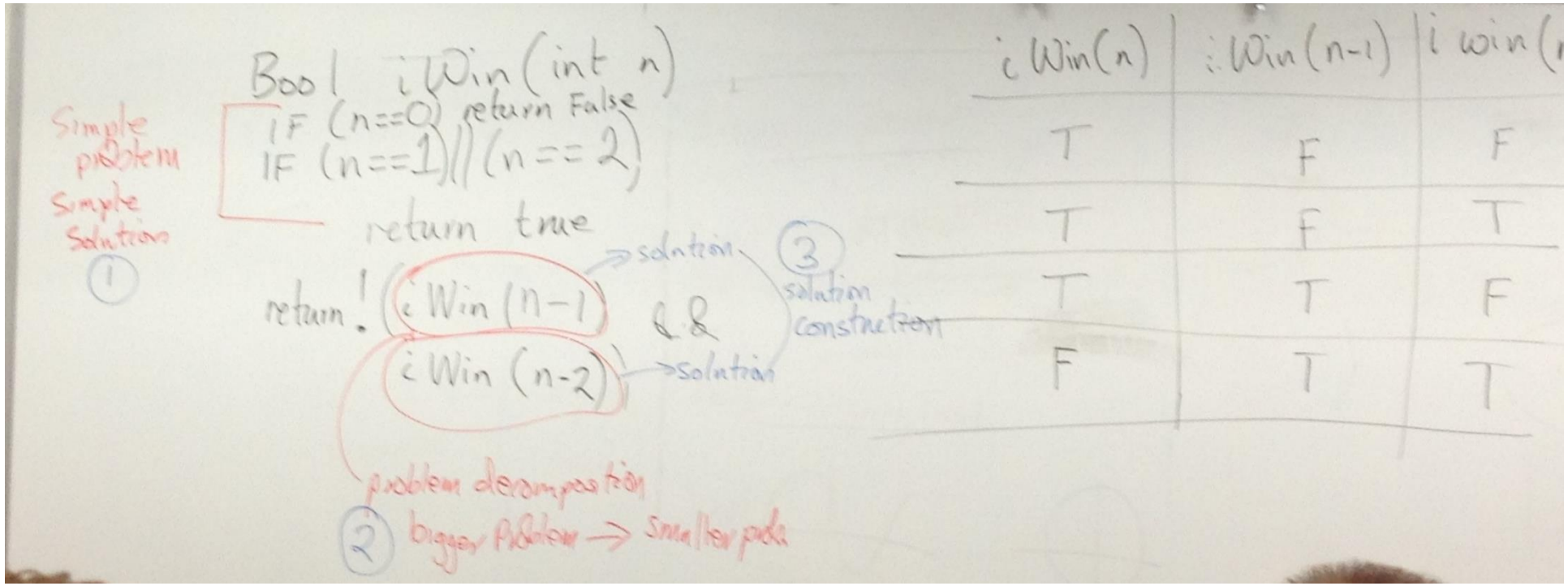


Nim: problem definition.
Simplify the problem of finding the best move to that of determining whether the number of stones Is lost or won.



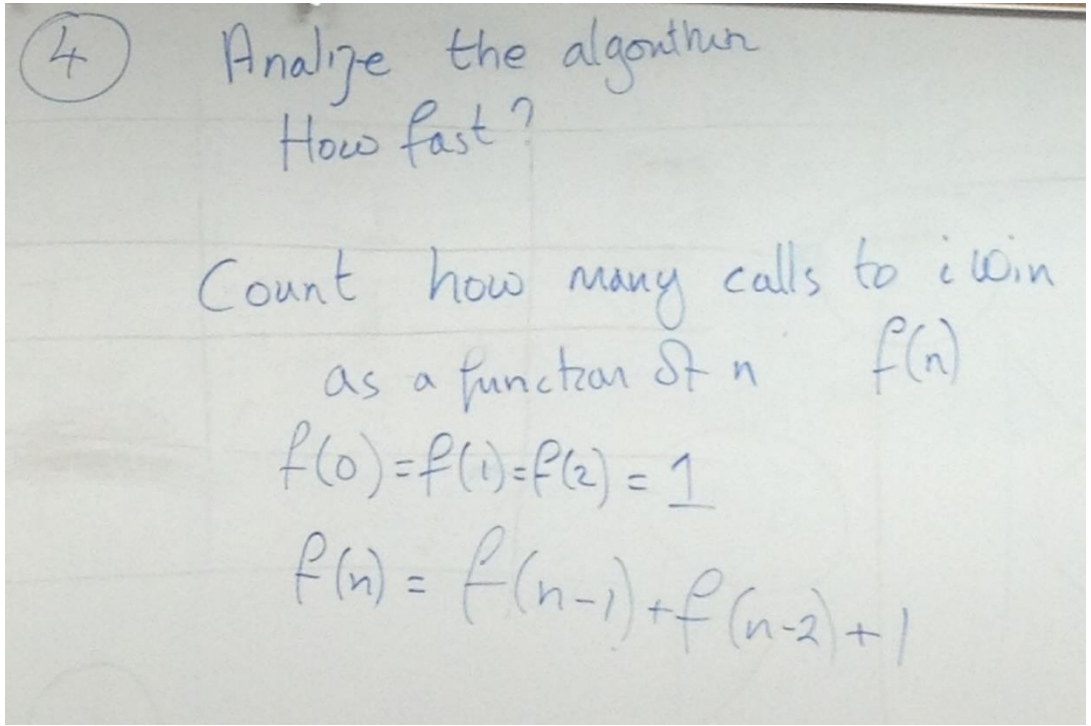
Bool iWin(int n) is a function that takes as input the number of stones on the board and returns true if the game can be won and false if the game will be lost. The result is for the player who is to move.

Recursive solution is developed from the base cases and the possible outcomes of taking either one stone or two stones

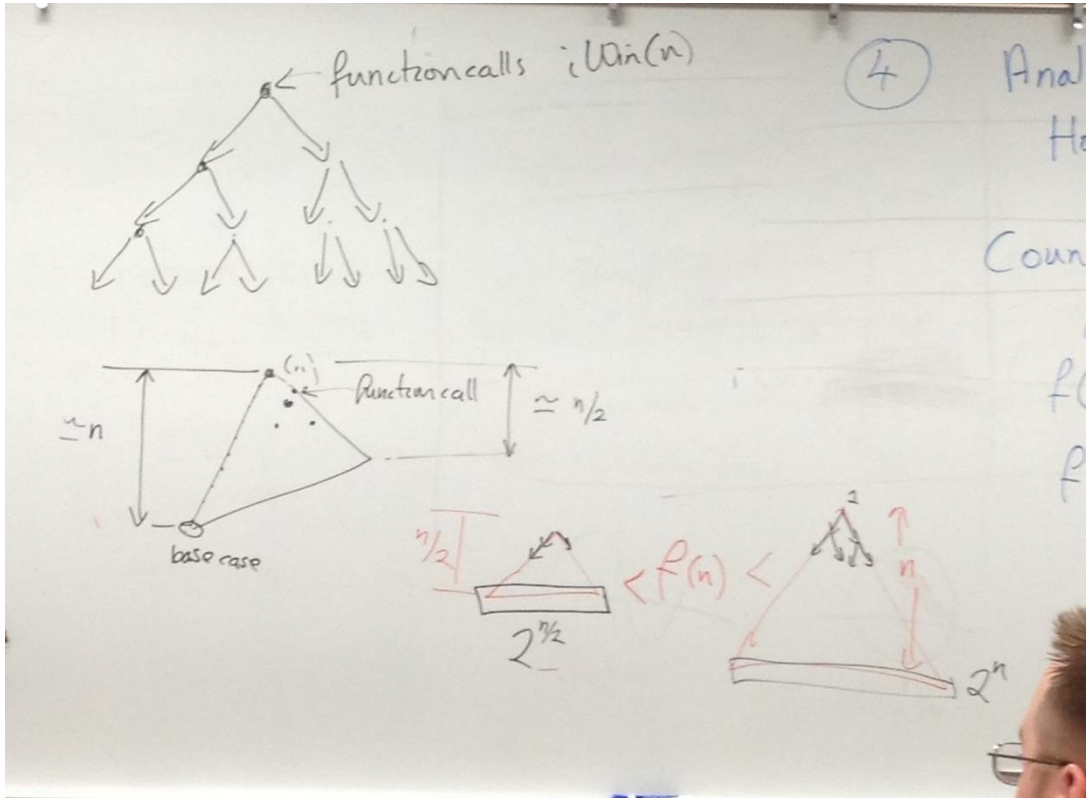


To design the solution algorithm:

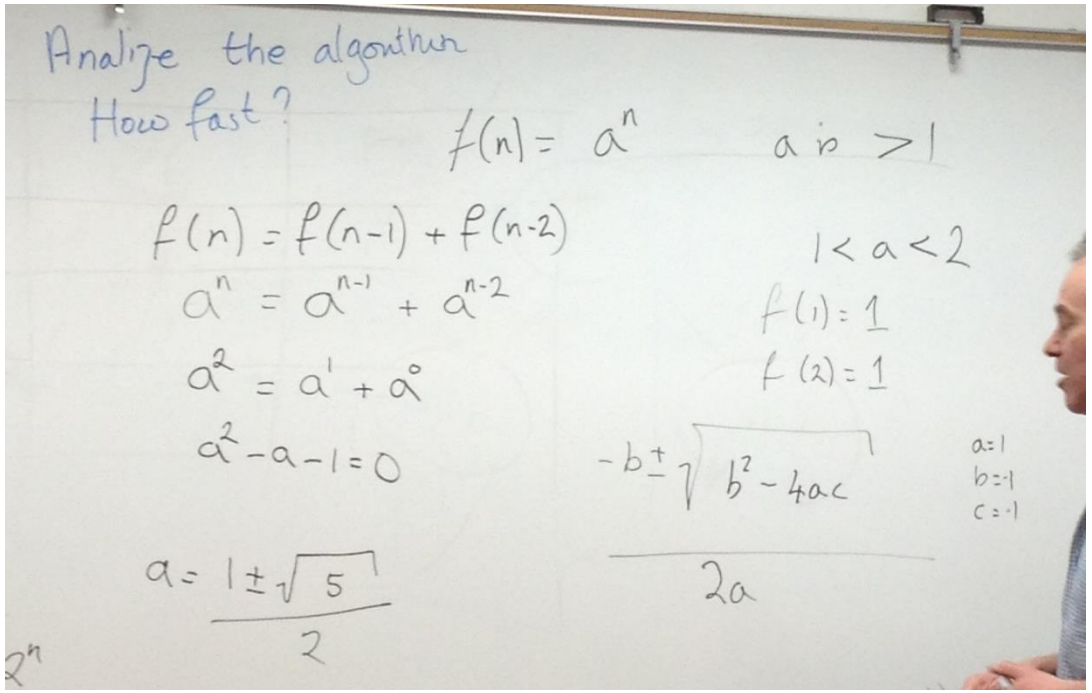
1) Identify the simplest problem instances and then return the solutions
2) Identify smaller sub-problems that can be solved using the same function
3) Once the sub-problems are solved, determine how to determine “my” solution
   From the solutions to the sub-problems



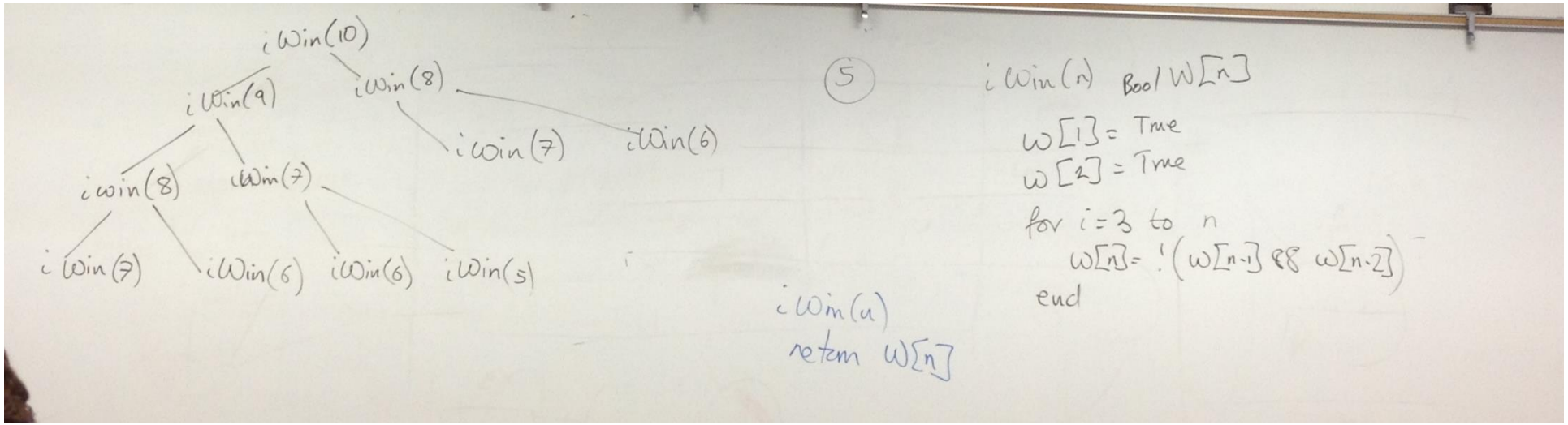
Once the algorithm is designed we need to determine its efficiency
Define a function that takes the number of stones (n) and returns the number of recursive calls generated by the algorithm. This function is int f(n)
“How many calls are needed to solve a problem of size n-1?”
answer f(n-1)



Estimating the solution for f(n). The calling tree branches into two, one side is of depth approx. n, the other is depth approx. n/2 so $2^{n/2} < f(n) < 2^n$



Guess the solution is exponential in n with base a,
$1 < a < 2$
Insert $f(n) = a^n$ simplify and solve for a



Found that the algorithm is worthless since the run time is exponential in n

Study the behavior of the algorithm and note that many calls are duplicated
In fact, since the input to the function is an integer n and all calls reduce n by 1 or 2
For a problem of size n, there are only n unique calls!!!!!!

We can use a technique call dynamic programming to precompute all values of iWin(n) into a table in linear time