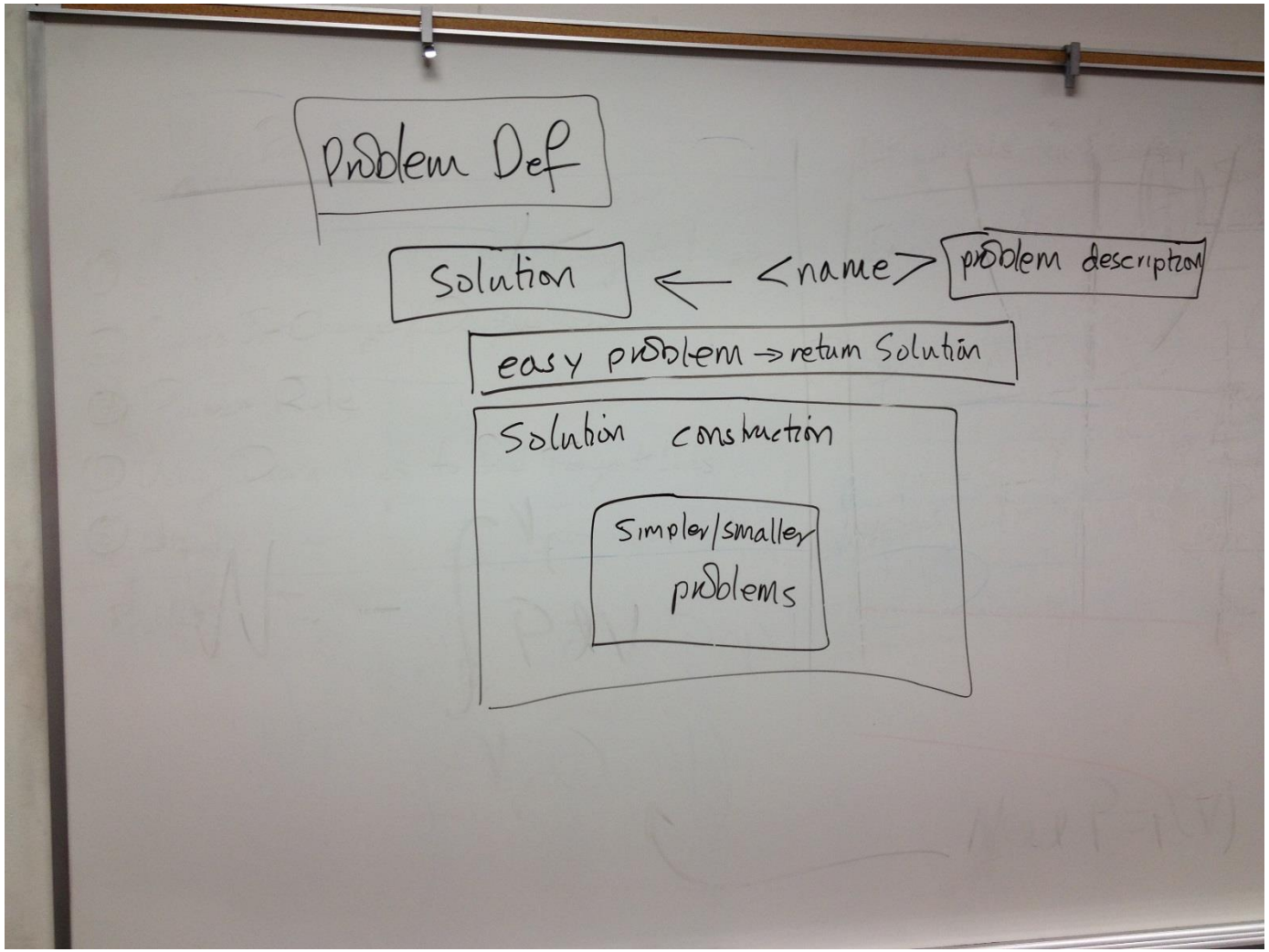
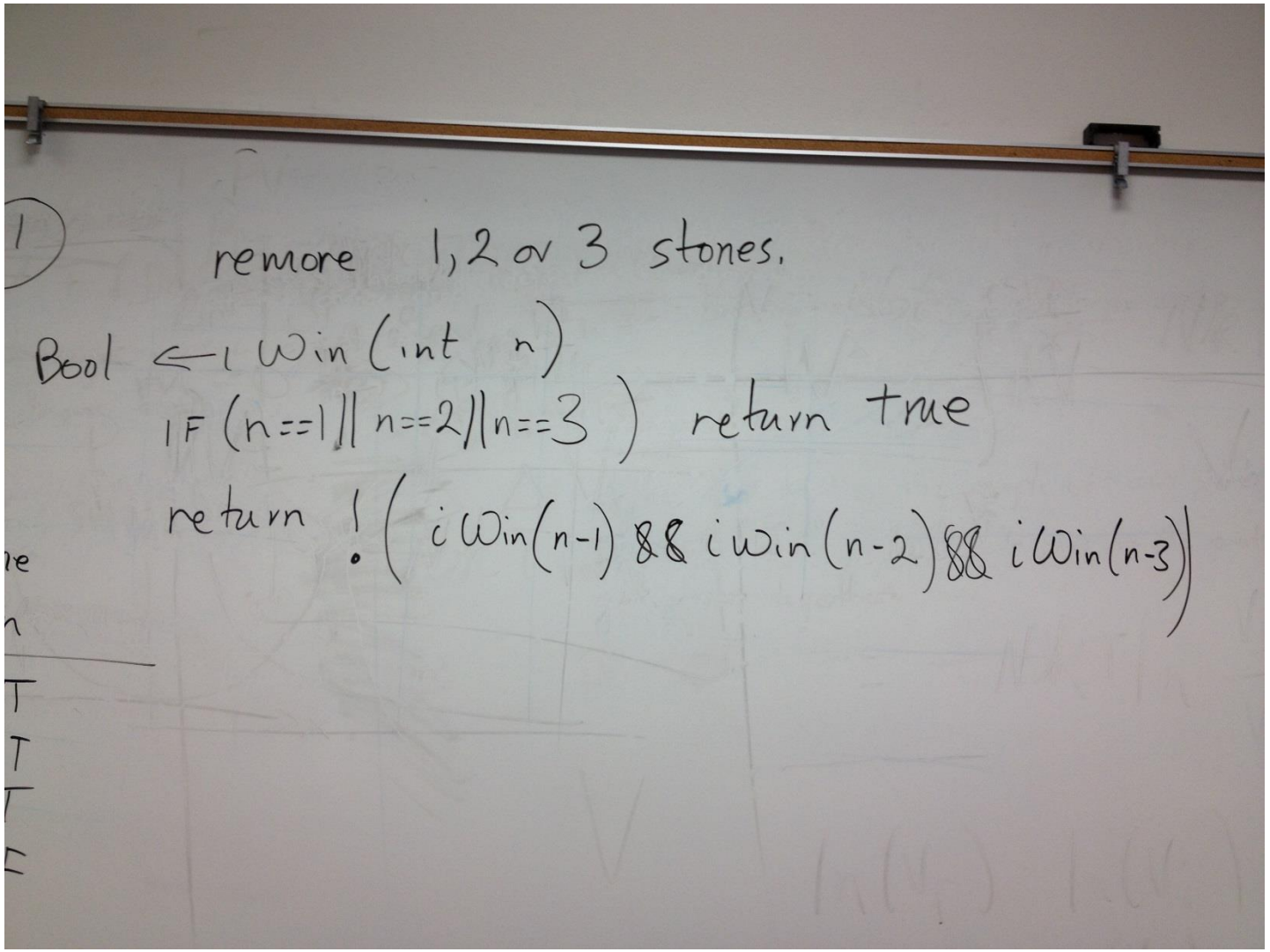
cs5050

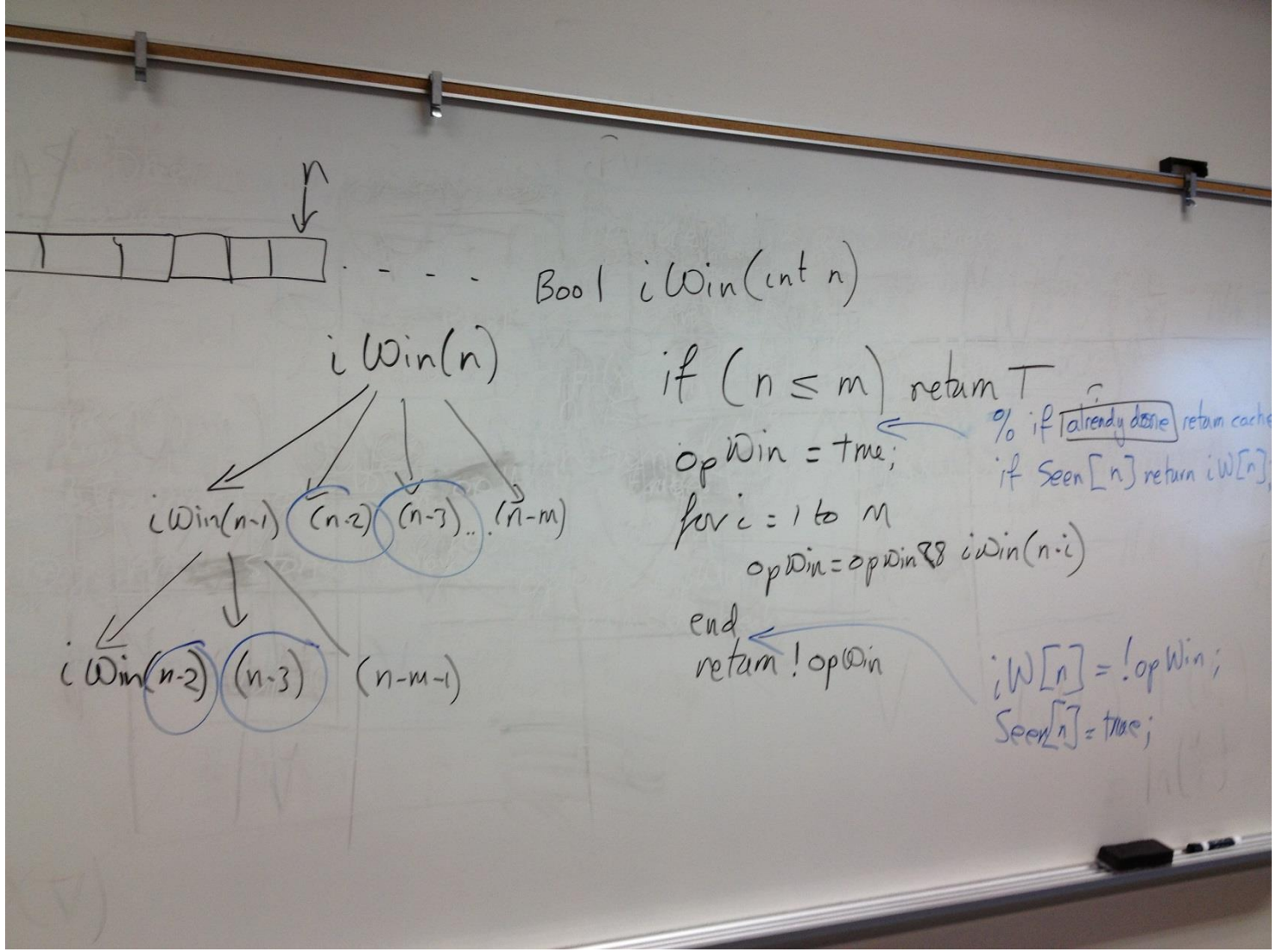
01 09 14



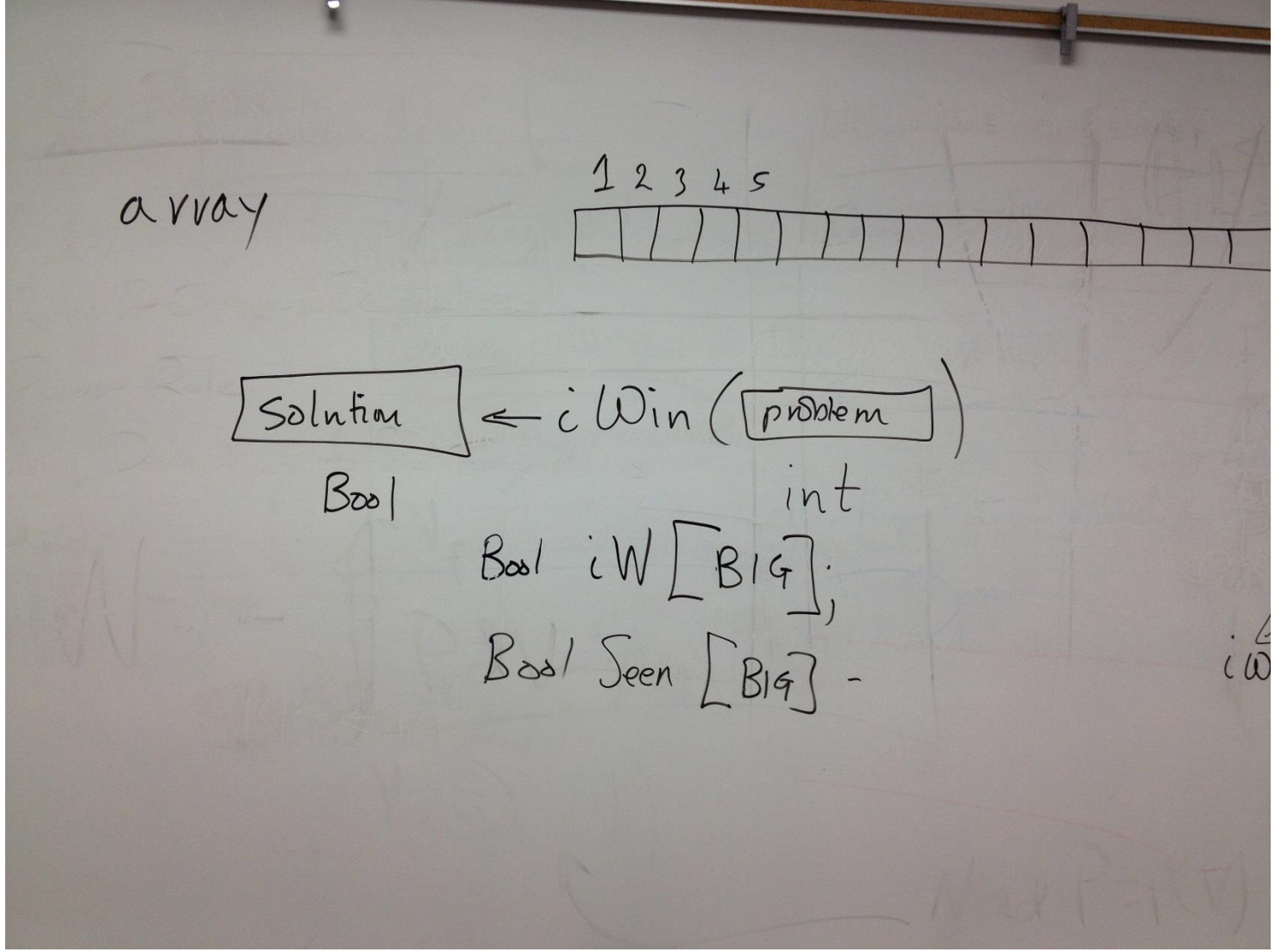
Solution scheme. We just have to fill in each component



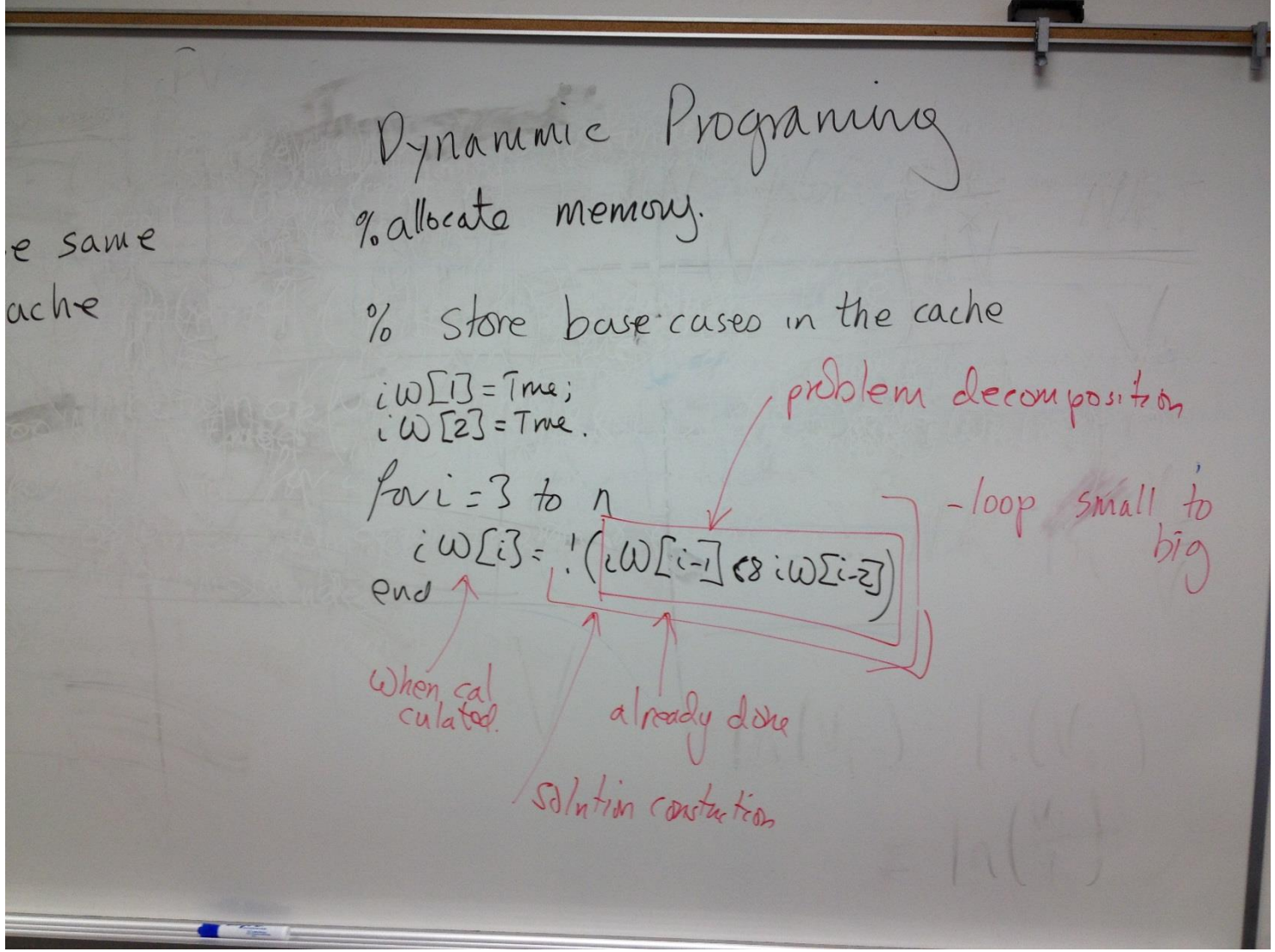
Solutions to similar problems are easy when we understand the logic



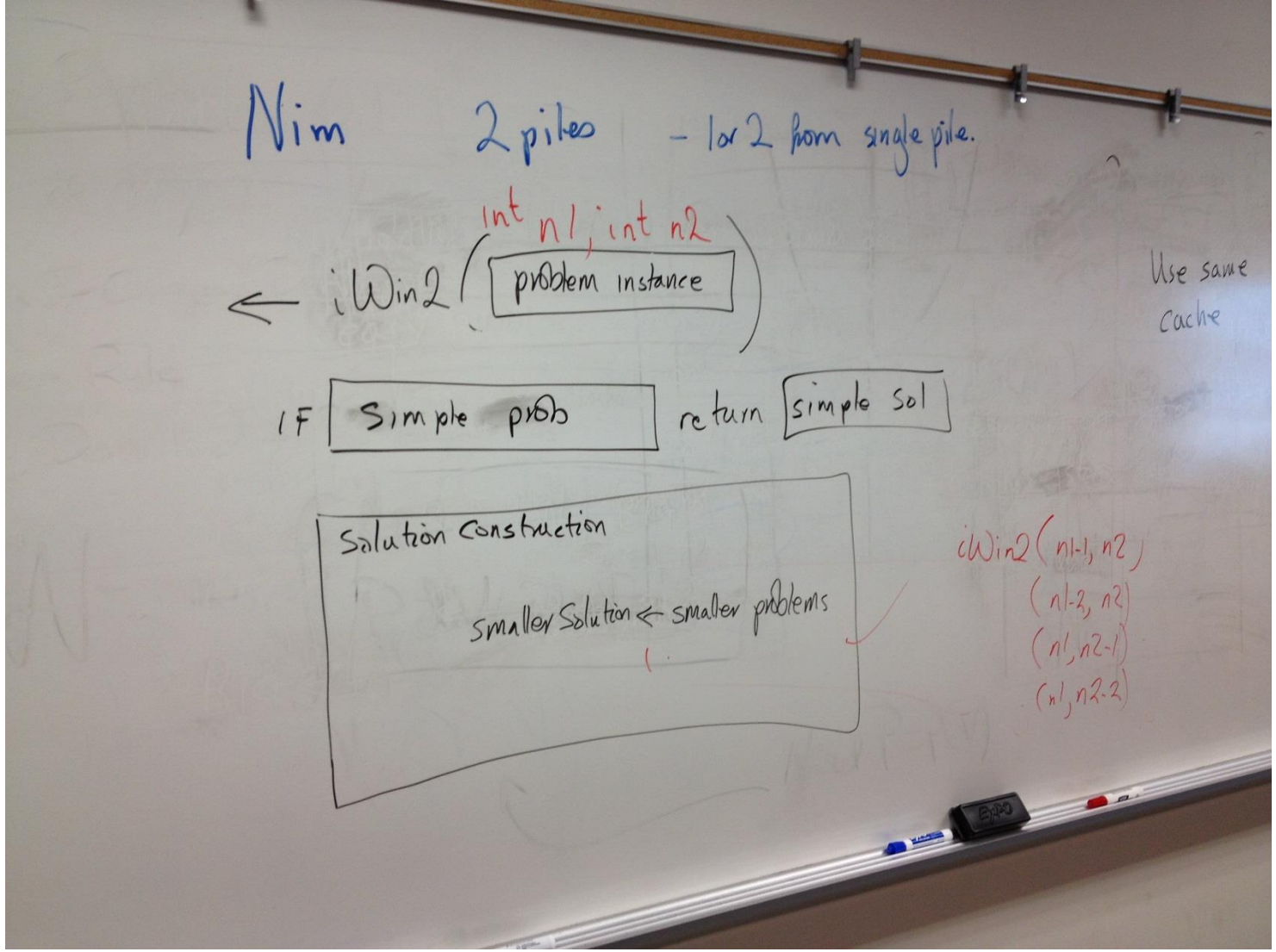
Expanding the meta algorithm to multiple stones in Nim is simple
Calling tree makes the redundant calls clear
Note that with depth first function calling, the depth of the calling tree will be n-m
Blue shows changes needed to cache then repeatedly reuse solution



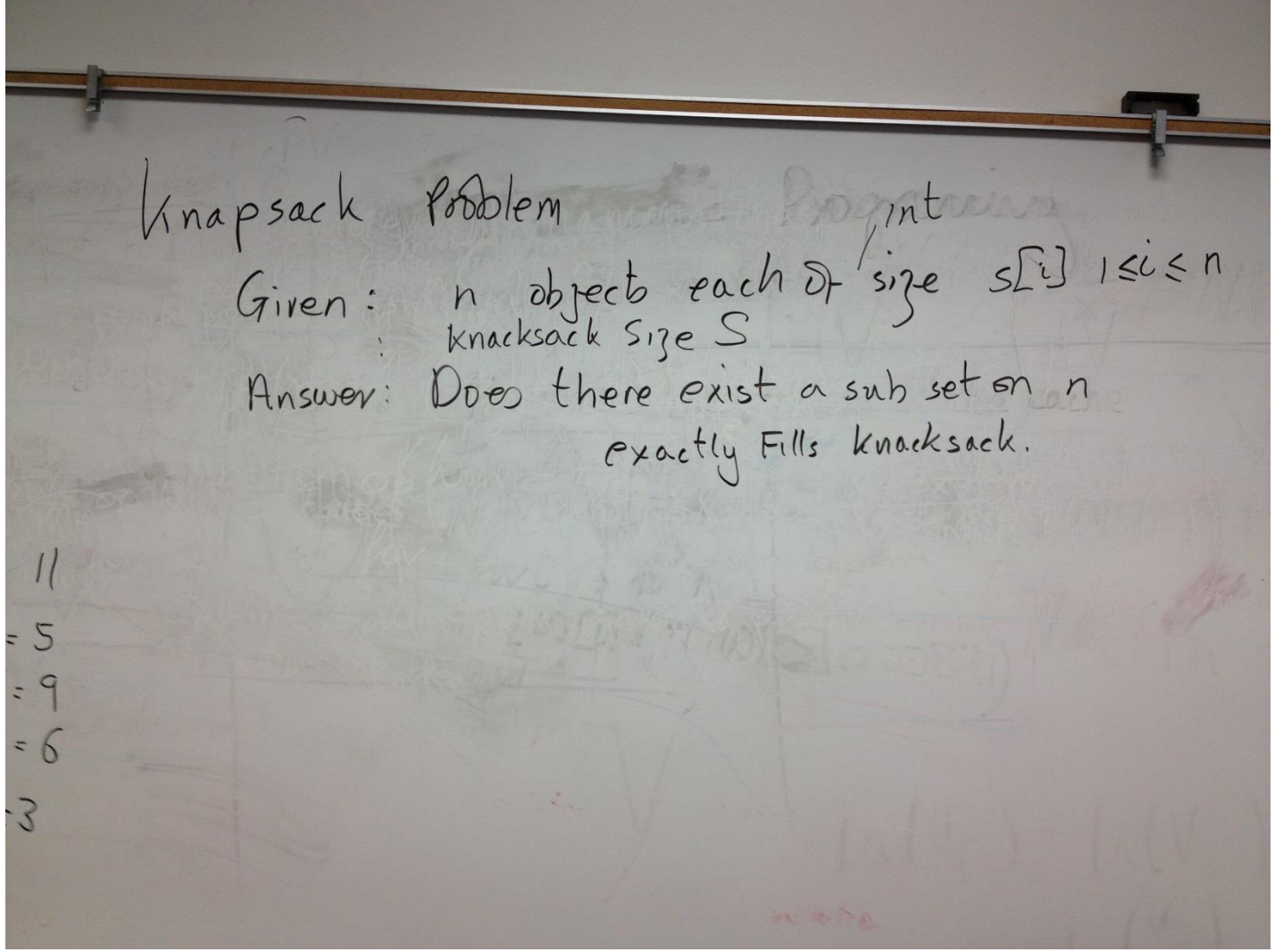
The data structure for the cache must be able to store and reteive the same mapping as the function
In this case bounded integer in, Bool out. We need a parallel array to keep track of which solutions
We have already computed



Dynamic programming solution eliminates the need for recursion and the subsequent dynamic memory allocation. The cache is filled “bottom up” or smallest solution first
Note how base cases, problem decomposition and solution construction are the same



Extending Nim to two piles is easy if we follow the meta algorithm
Here a problem instance is the count in two piles, so we need two inputs
There are four possible ways the problem can be reduced to subproblems



The Knapsack problem is optimizing value under limited resources
Many practical problems
This is the very simplest decision (true or false) problem