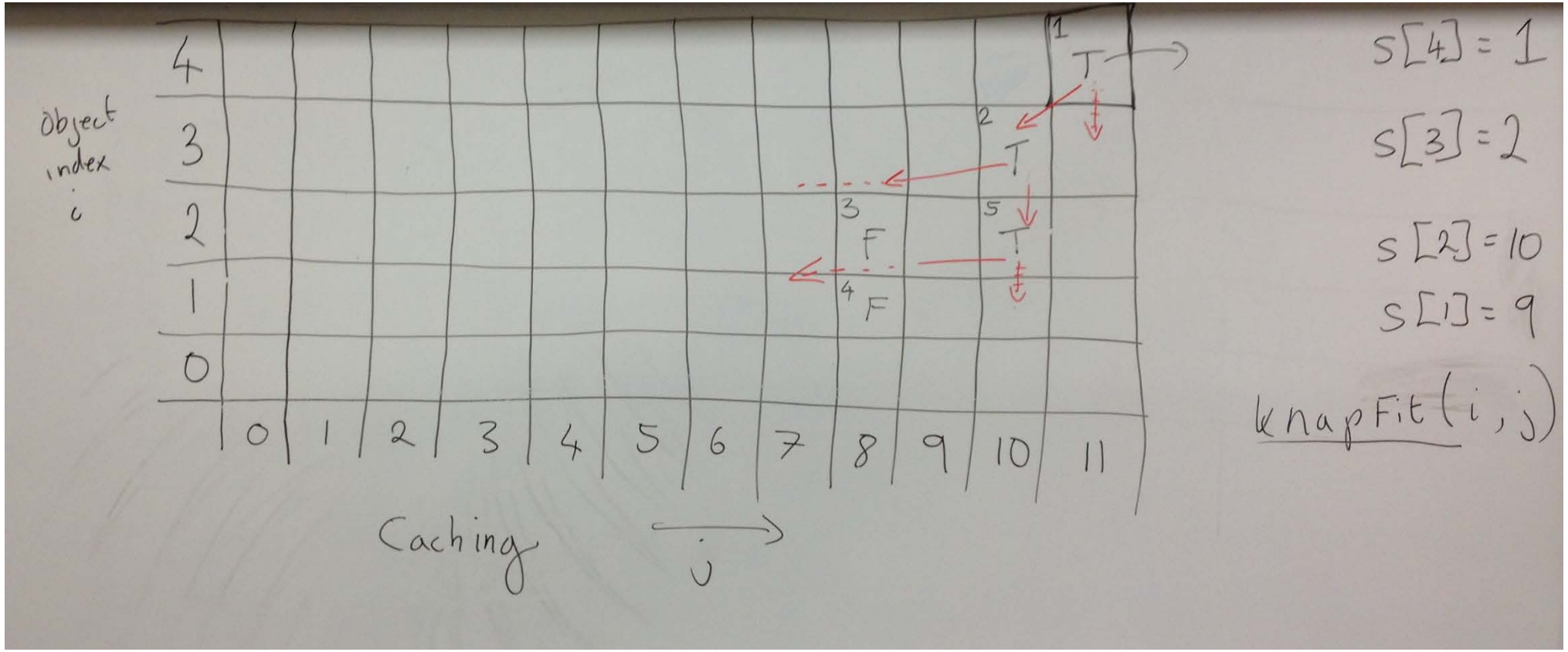
cs5050

01 16 14



T is true, F is false. Illustrated are the only solutions computed by the caching algorithm

KEY

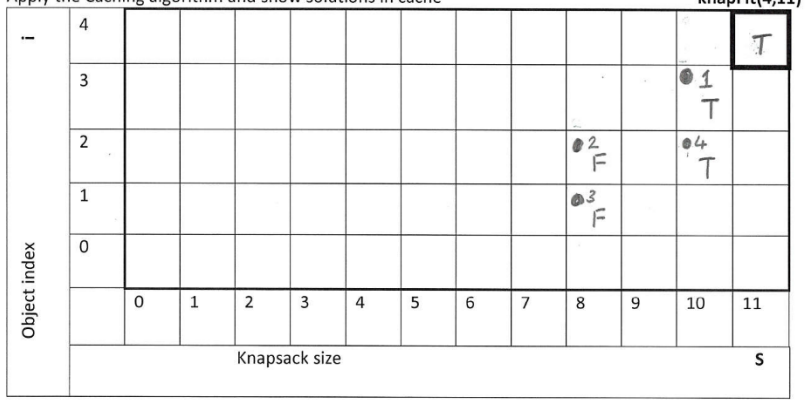
Cs5050: Dynamic programming and caching example

Bool knapFit(int i, int s) // i is item index <= n, s is the current capacity of knapsack

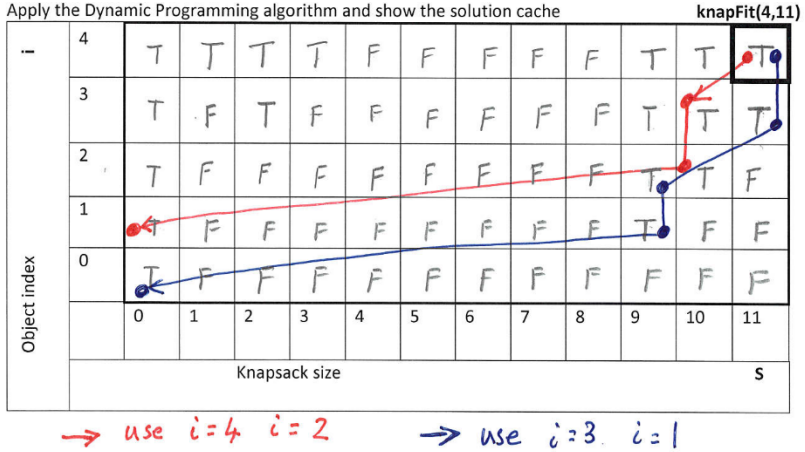
If(s==0) return true;
If(i==0 && s>0) return false;
If(s<0) return false;
return(knapFit(i-1, s-s[i]) || knapFit(i-1, s) // use or don't use the item

Problem instance: S=11 s[1]=9 s[2]=10 s[3]=2 s[4]=1

Apply the Caching algorithm and show solutions in cache **knapFit(4,11)**

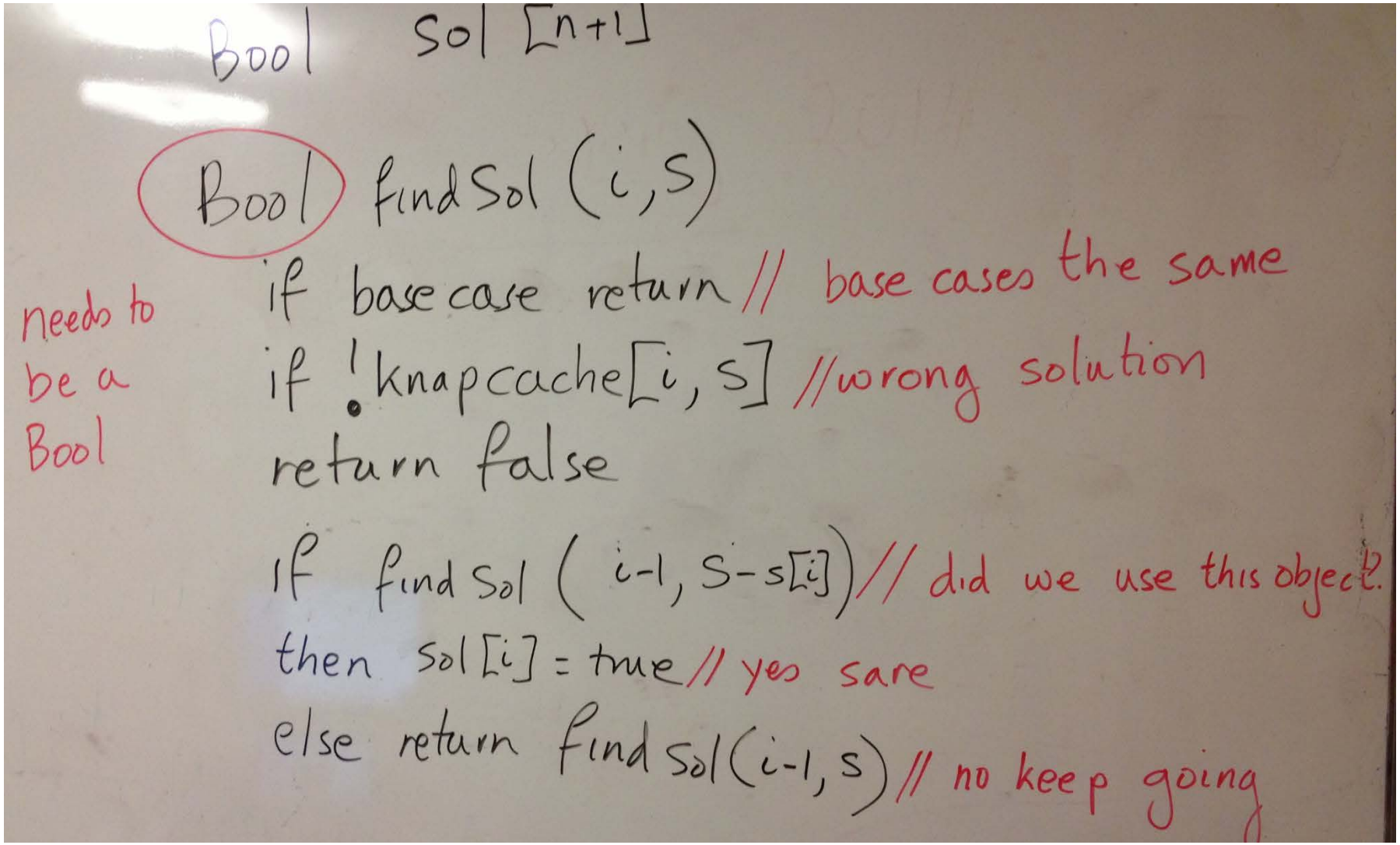


Apply the Dynamic Programming algorithm and show the solution cache **knapFit(4,11)**

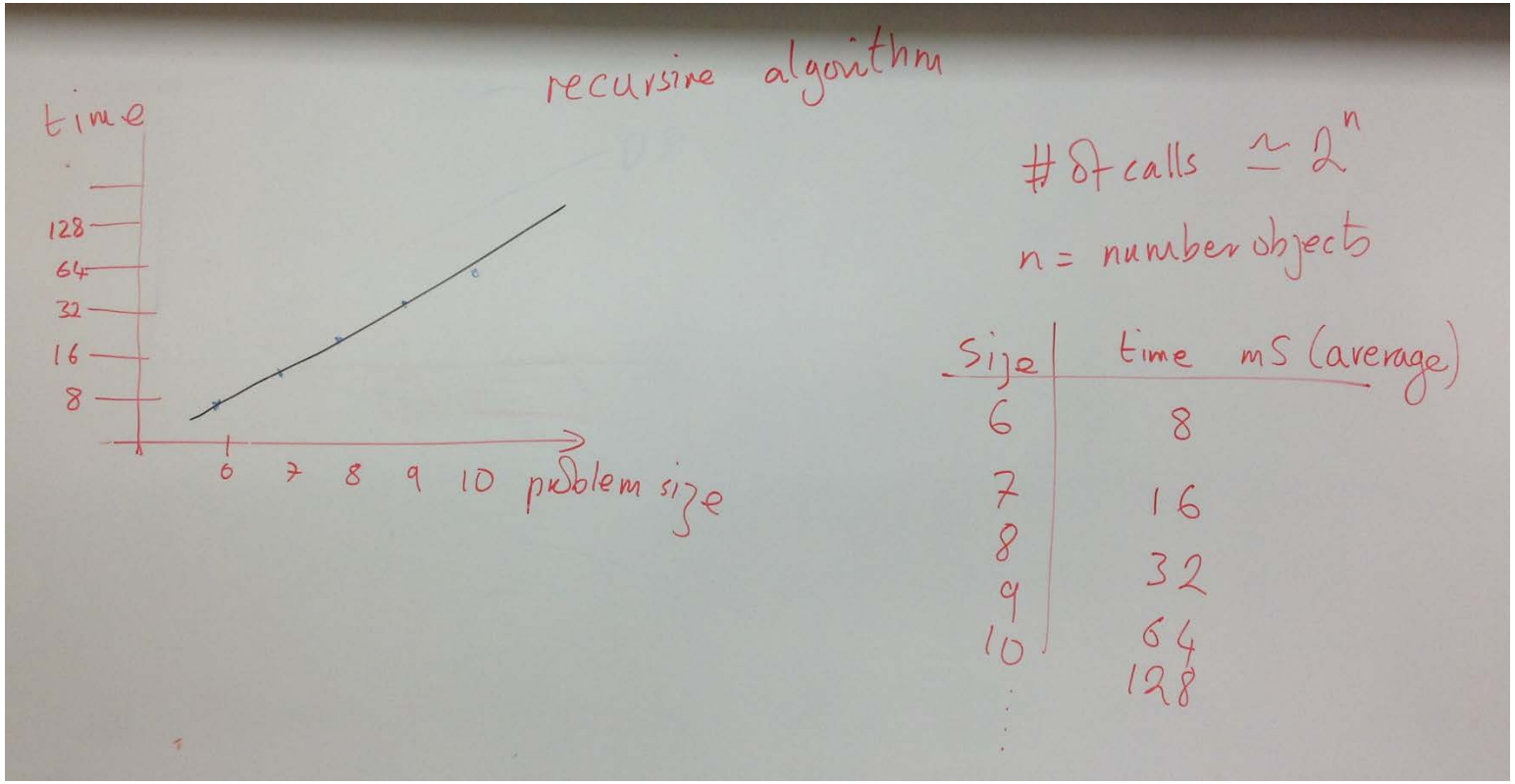


→ use i=4 i=2 → use i=3 i=1

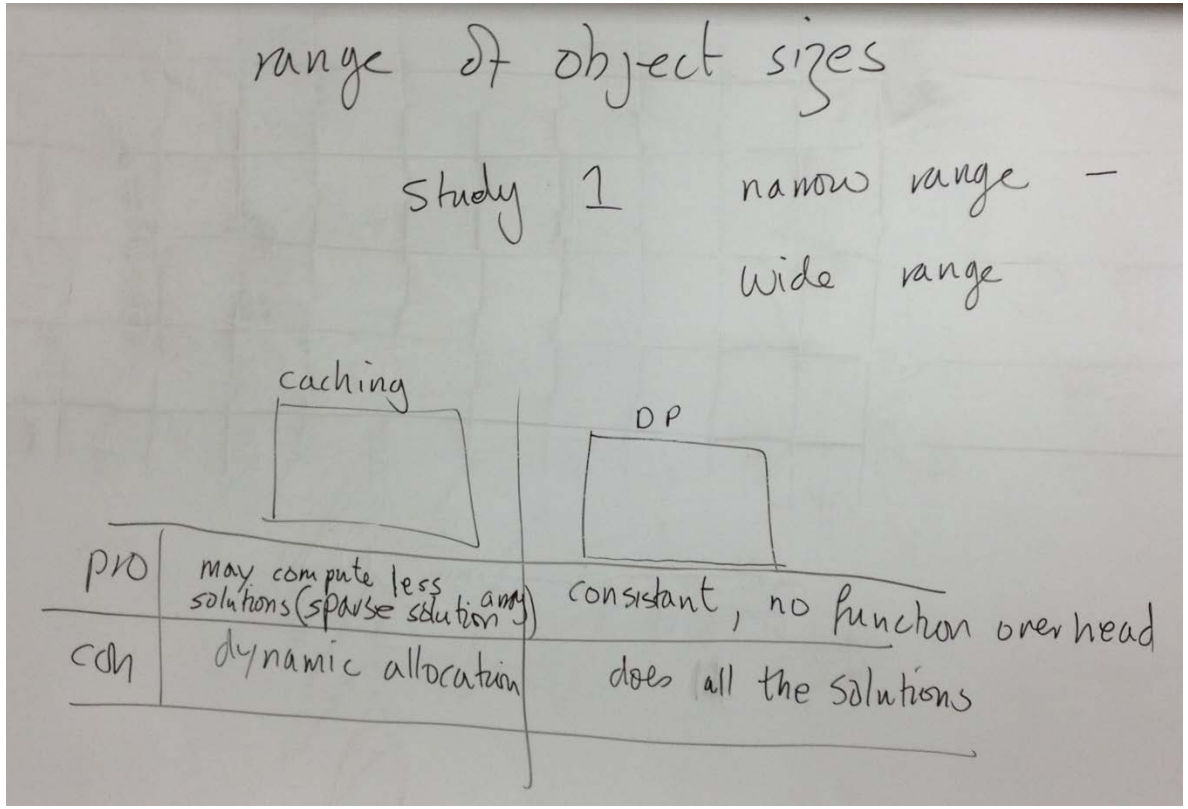
The worked through examples:



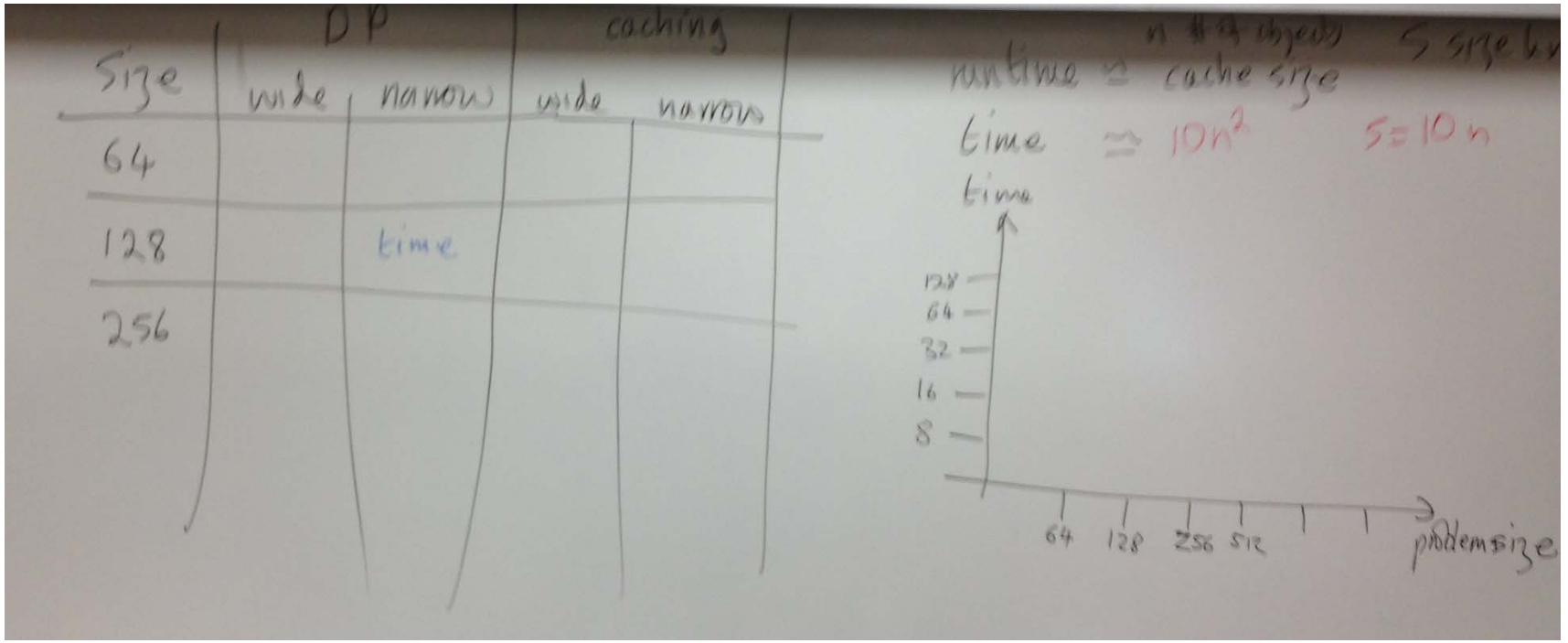
The correct trace back algorithm that sets Sol[i] = T if object I is used in the solution



First experiment. Run the simple recursive algorithm on increasing problem sizes
For each size run a set of random repeats and average the resulting time
Plot the results on a semi-log graph and you should see a near straight line
The slope will be the estimated base of the exponential function mapping problem size to time



What is the answer to the question “Which algorithm is better caching or DP?”
Depends
Each algorithm has pros and cons
The best algorithm will depend upon the problem object size distribution and the efficiency of function calling



This experiment seeks to investigate the question of which algorithm is better by running both algorithms
On two distributions of object sizes. For each problem size (number of objects) record the average running
Time of the four combinations of two size distributions and the two algorithms
Plot the graph on a log log graph