

cs5050

01 16 14

Object index $i$	4												1	T	→	$s[4] = 1$
	3												2	T	↙	$s[3] = 2$
	2								3	F			5	T	↓	$s[2] = 10$
	1								4	F					↙	$s[1] = 9$
	0															
		0	1	2	3	4	5	6	7	8	9	10	11			<u>knapFit(i, j)</u>

Caching       $\xrightarrow{j}$

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

## Cs5050: Dynamic programming and caching example

```

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

```

Apply the Caching algorithm and show solutions in cache

knapFit(4,11)

Object index	4												T
	3										1 T		
	2							2 F		4 T			
	1							3 F					
	0												
		0	1	2	3	4	5	6	7	8	9	10	11
Knapsack size													S

knapFit(4,11)

Object index	0	1	2	3	4	5	6	7	8	9	10	11
4	T	T	T	T	F	F	F	F	F	T	T	T
3	T	F	T	F	F	F	F	F	F	T	T	T
2	T	F	F	F	F	F	F	F	F	T	T	F
1	T	F	F	F	F	F	F	F	F	T	F	F
0	T	F	F	F	F	F	F	F	F	F	F	F

Knapsack size  $S$

→ use  $i=3$   $i=1$

The worked through examples:

Bool Sol [n+1]

Bool findSol (i, s)

needs to  
be a  
Bool

if base case return // base cases the same

if !knapcache[i, s] // wrong solution

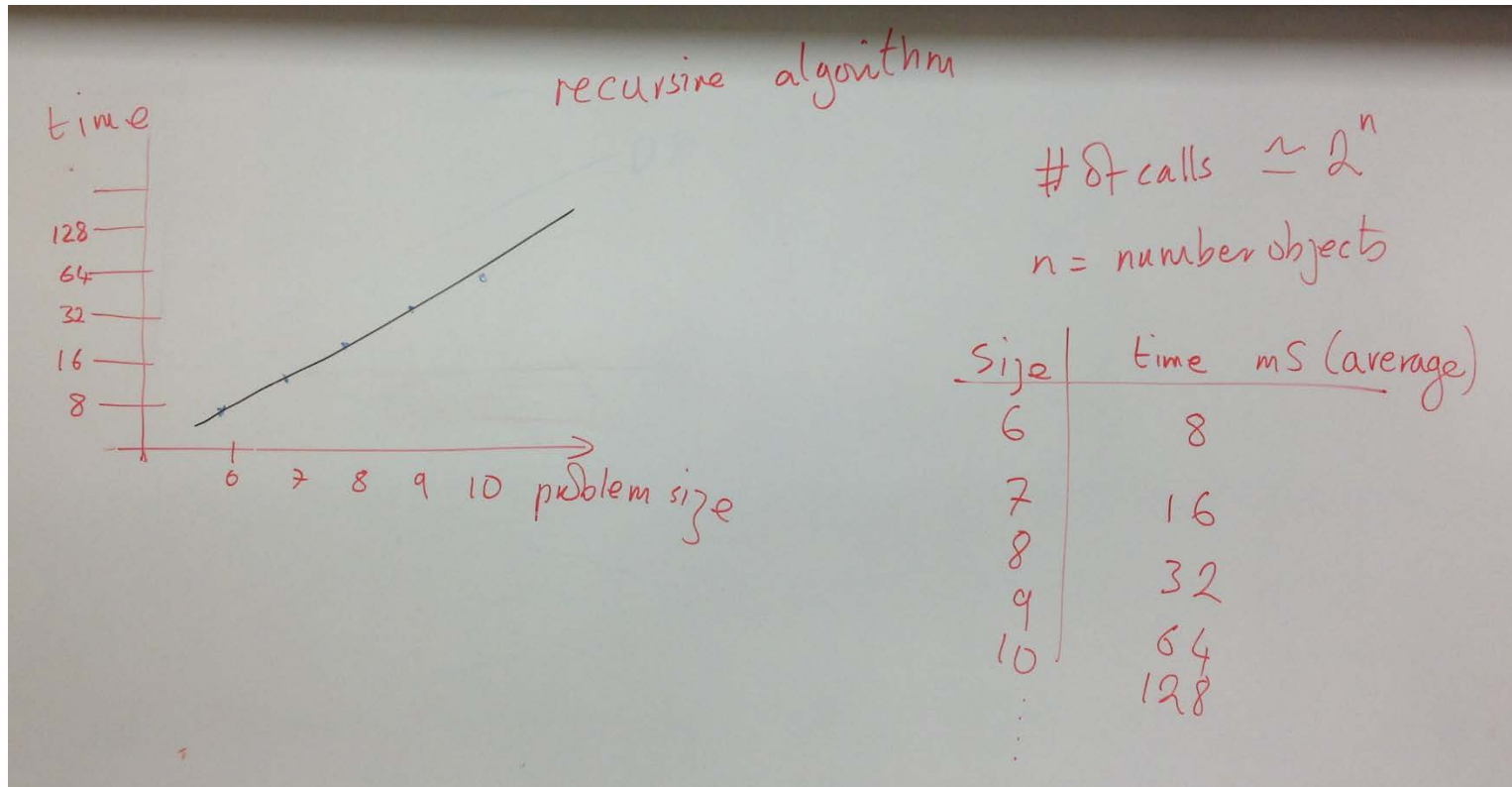
return false

if findSol (i-1, s-s[i]) // did we use this object?

then sol[i] = true // yes save

else return findSol(i-1, s) // no keep going

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

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

range of object sizes

study 1 narrow range -  
wide range

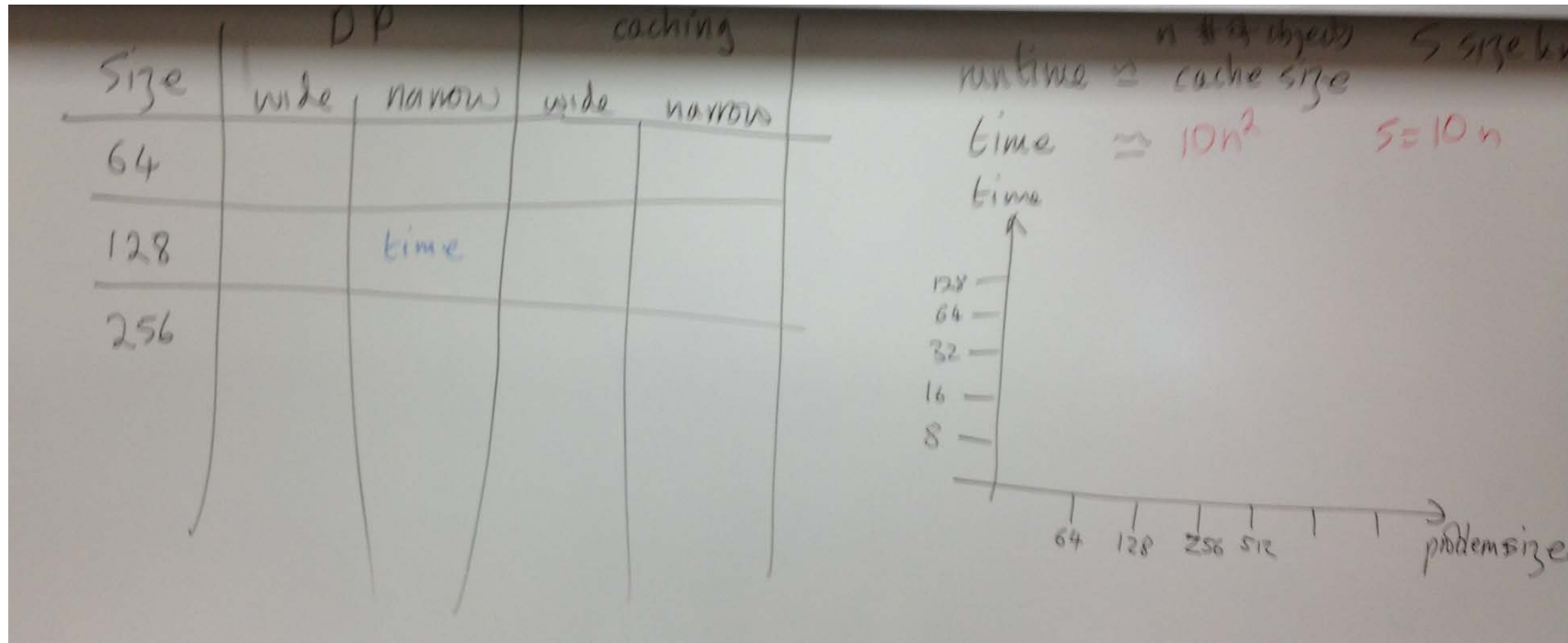
	caching	DP
pro	may compute less solutions (sparse solution)	constant, no function overhead
con	dynamic allocation	does all the solutions

## 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