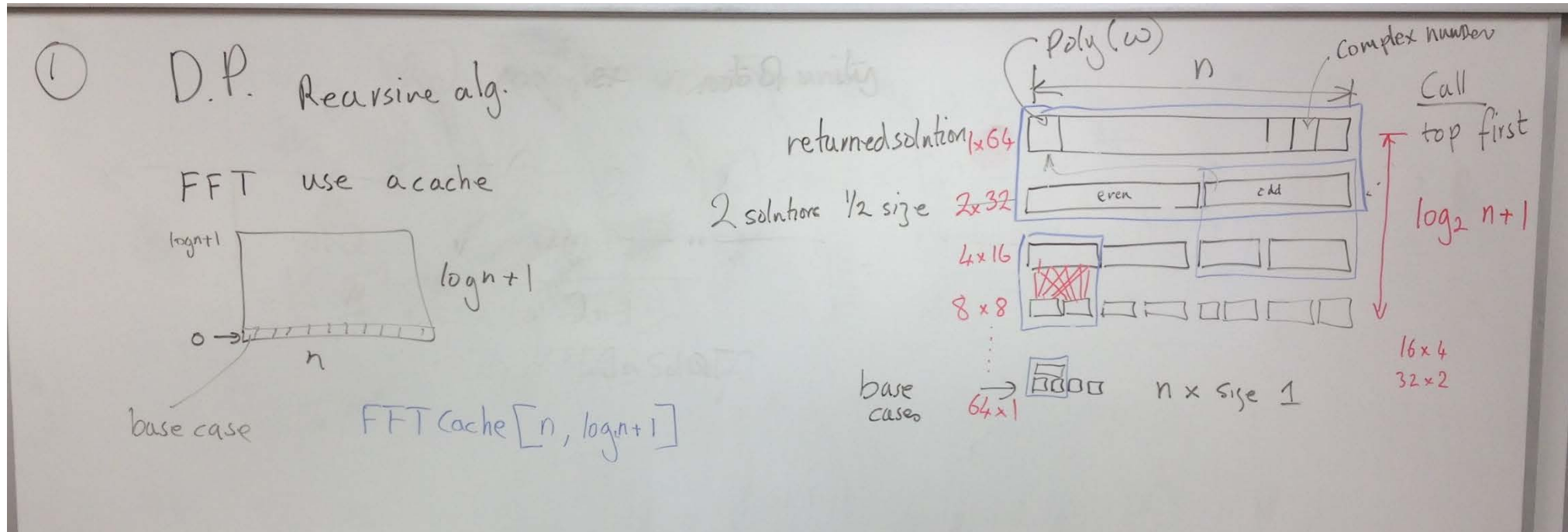


cs5050

03 06 14



Cache contains all the solutions computed.

By analysis of the calling tree of solution/subsolutions we can determine there are n by $\log_2 n + 1$ solution values computed by the recursive function.

② How do we fill in the base Case?
just the coefficients

```
for i = 0 to n-1  
    FFTCache[i, 0] = p[reverse Bit Flip Shuffle(i)]  
end
```

Base cases go in the $\text{FFTCache}[i, 0]$. Reviewing the recursive def we see that each value is simply a coefficient of the polynomial. But the i th coefficient does not go into the i th place in the array!
What is the mapping from the index of the coefficient to the index of where it should go? ReverseBitFlipShuffle

③ On the way down

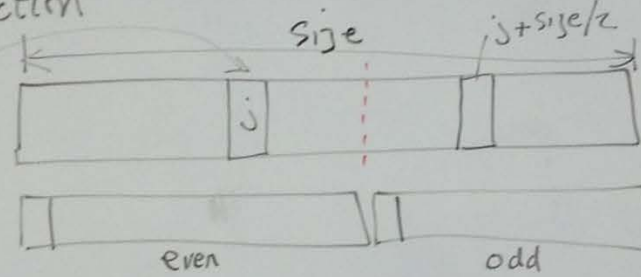
	0	1	2	3	4	5	6	7
	0	2	4	6	1	3	5	7
	0	4	2	6	1	5	3	7
write the binary code for this number →	000	100	010	110	001	101	011	111
	000	001	010	011	100	101	110	111
	0	1	2	3	4	5	6	7

Reverse bit flip shuffle example. For 8 coefficients, top line lists the coefficients
 Next lines do the even (on the left) odd (on the right) split for each recursive call
 The bottom line of numbers is where the coefficients have to be placed in the base case
 DP cache
 To determine the pattern, write the numbers in binary.
 To compute the position of the i th coefficient: represent the number i in binary, reverse
 the bits, the value of this number is the location.

④

Solution Construction

FFTCache[k, j]



level k

level k-1

for j=0 to size/2-1

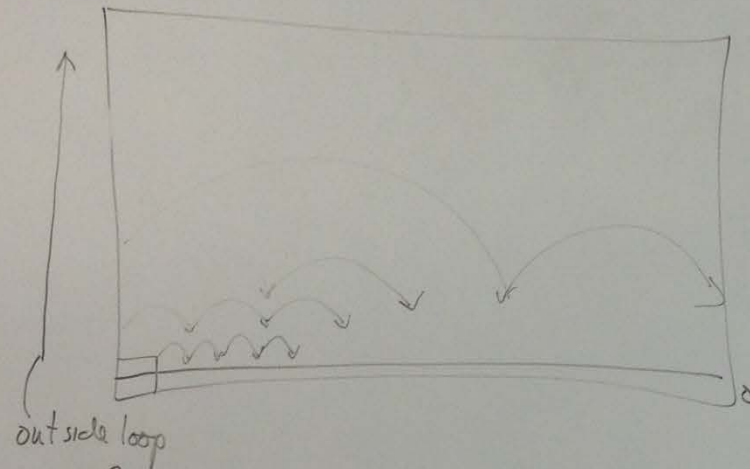
$$\text{FFTCache}[k, j] = \text{FFTCache}[k-1, j] + \text{Omega}[?] * \text{FFTCache}[k-1, j + \text{size}/2]$$

$$\text{FFTCache}[k, j + \text{size}/2] = \text{FFTCache}[k-1, j] - \text{Omega}[?] * \text{FFTCache}[k-1, j + \text{size}/2]$$

end

Study the solution construction process for one recursive call. We compute an array of size values from two sub solutions each of size/2 values. The subsolution on the left is for the evens, the one on the right is from the odds.

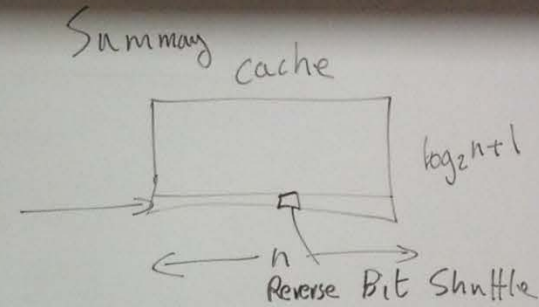
5



for $i = 1$ to $\log_2 n + 1$ ←

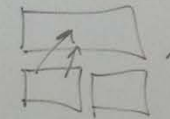
inside loop skips over each sub solution

① base case



One solution

from 2 smaller $1/2$ solution



To complete the DP calculation, nested loops must be written that scan from bottom to top taking into account that the: (a) size of the problems increase from 2 to n , (b) the number of problems reduces from $n/2$ to 1.