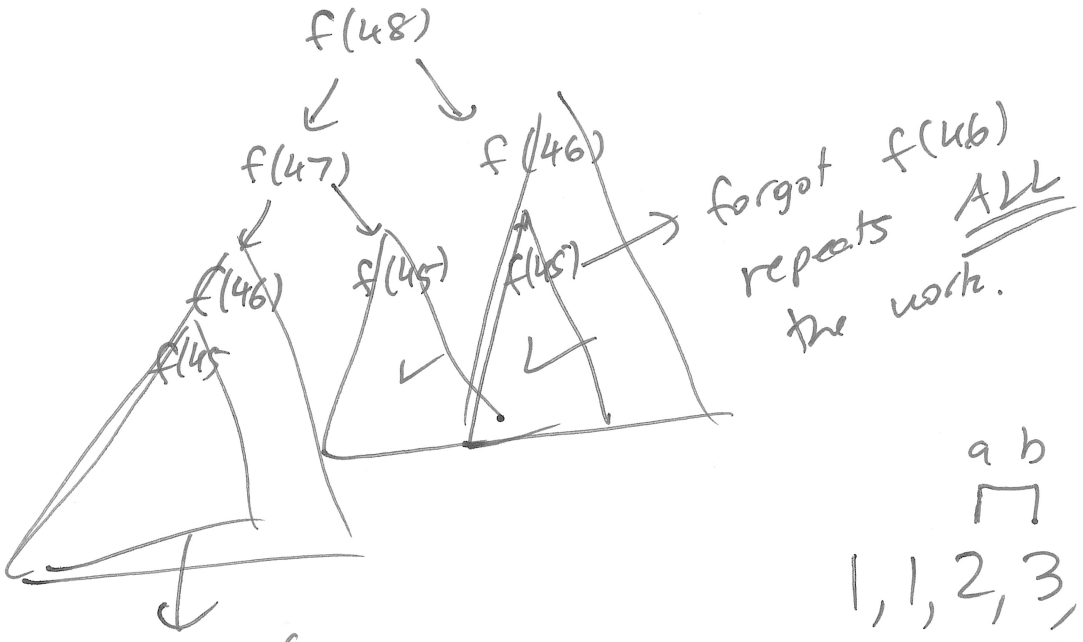
COP 3503 4/2/24



1, 1, 2, 3, 5

PROBLEM: MAKE THE EXACT SAME RECURSIVE CALL MULTIPLE TIMES!!!

SOLUTION: ADD MEMORY

memoization

(a) anticipate each unique recursive call

(b) build storage to hold the answers to each of those calls (array, hashmap)

(c) when we finish an answer store it before we return

(d) in base case, if we've done it before return the stored answer (memoized)

fib[n] depends on fib[n-1], fib[n-2]
Why not fill out my storage in order so that all lookups are already in table?

* build loop going up
* seed table w/ 2 base cases

---

longest common subsequence

input: 2 strings
output: length of the LCS of both strings

s1: GACTTAGACGATA
s2: ACGATACAGGACTA

```
if s1[n-1] == s2[m-1]
    return 1 + lcs( s1[0..n-2], s2[0..m-2] )
else
    return max( lcs(s1[0..n-2], s2[0..m-1]),
                lcs(s1[0..n-1], s2[0..m-2]) )
```

---

dp[0][0] = lcs(0 letters s1, 0 letter s2)

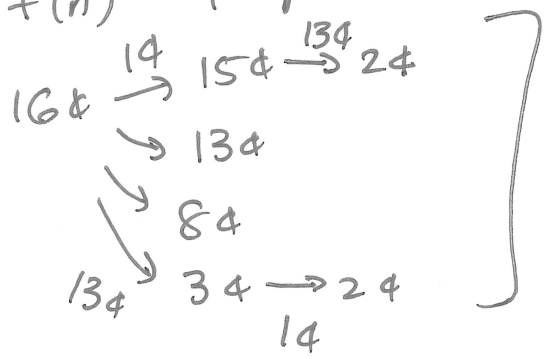
dp[n][m] = lcs(n letters s1, m letter s2)

		G	A	C	T	T	A
	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
C	0	0	1	2	2	2	2
G	0	1	1	2	2	2	2
A	0	1	2	2	2	2	3
T	0	1	2	2	3	3	3

Change Problem

1¢, 3¢, 8¢, 13¢

make change for 16¢. How many ways can I do it order you receive coins doesn't matter.



ans to # ways make change when each diff order of coin counts differently

fib like dp/memo

If you don't want to count different orders, then only count "sorted sets"

f(n, d) = # ways to make change for n cents using denomination d or lower

$$f(16, 13¢) = \underbrace{f(15, 1¢)}_{\text{give penny}} + \underbrace{f(13, 3¢)}_{\text{give 3¢}} + \underbrace{f(8, 8¢)}_{\text{give 8¢}} + \underbrace{f(3, 13¢)}_{\text{give 13¢}}$$

$$= f(16, 8¢) + \underbrace{f(3, 13¢)}_{\text{give 1 13¢ coint}}$$

CSES Prob

idx	den	0	1	2	...	n
0	1	1	0	0	...	1's in mult of smallest den
1	2	1				
2	6	1				
3	8	1				

$dp(n, k) = dp(n, k-1) + dp(n - den[k], k)$

1	1	1	1	1
1	0	1	2	3
1	1	2	3	3
1	2	4	4	7

End Path