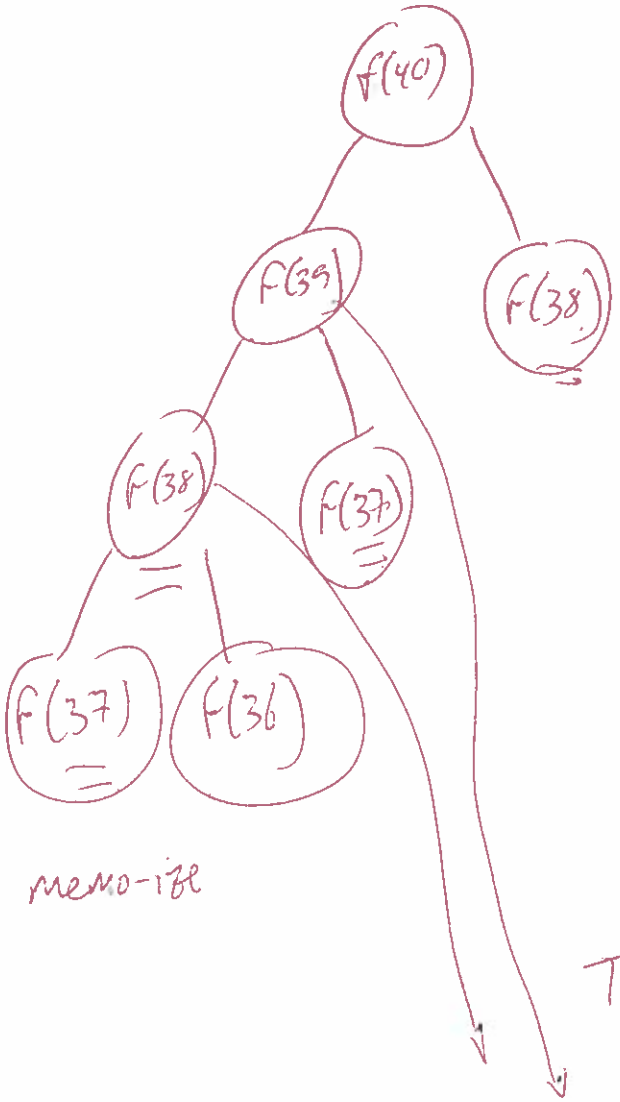
CPSC 320
2010/03/16



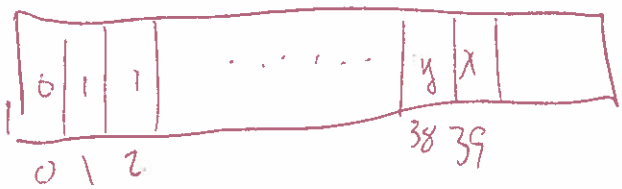
memo-ize

MEMO-IZATION:

~~take "memos" of subproblems' solutions~~

TAKE "MEMOS" OF SUBPROBLEMS' SOL'NS AND REUSE THEM IF WE RESOLVE THE SUBPROBLEM.

TRADE TIME FOR MEMORY



ITERATIVE DYNAMIC PROGRAMMING:

```
M = [0,1]
FOR i = 2 to n:
    M[i] = M[i-1] + M[i-2]
RETURN M[n]
```

OFTEN WE CAN "COLLAPSE" ONE DIMENSION OF THE TABLE

```
FIB1 = 0; FIB2 = 1
FOR i = 2 to n:
    FIB3 = FIB1 + FIB2
    FIB1 = FIB3
    FIB2 = FIB1
```

LONGEST COMMON SUBSEQUENCE (LCS)

PROBLEM:

GIVEN TWO STRINGS S & T, find THE LCS OF S AND T.

FIND THE SMALLEST # OF DELNS FROM EACH OF S & T NEEDED TO CREATE A COMMON RESULT STRING R.

SAMPLE:

FANTASTIC
ATINYSET
} → ANST

BRUTE FORCE: HOW MANY SUBSEQUENCES ARE THERE?

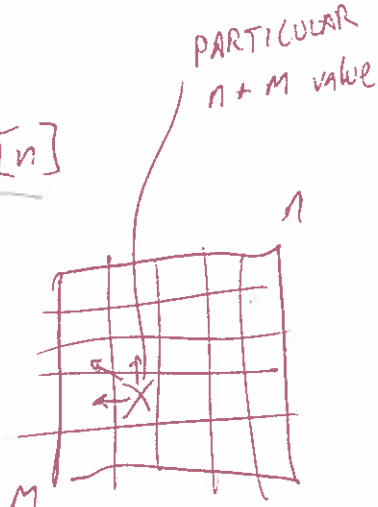
ROUGHLY 2^n

EXAMPLES:

BEST
WORST
← T MUST BE INCLUDED IN AN OPTIMAL SOL'N

```
LCS(S,T,n,m):
    IF n==0 or m==0: return ""
ELSE IF S[n] = T[m]:
        return LCS(S,T,n-1,m-1) + S[n]
    ELSE:
        LCS1 = LCS(S,T,n-1,m)
        LCS2 = LCS(S,T,n,m-1)
        IF LEN(LCS1) ≥ LEN(LCS2):
            RETURN LCS1
        ELSE RETURN LCS2
```

MAGICALLY MEMO-IZE



O(nM) OR $O(n^2)$

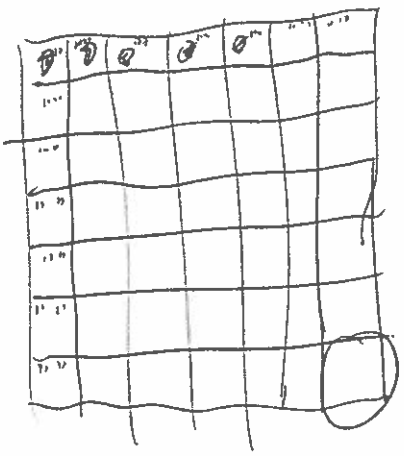
Initialize GLOBAL table w/empty entries of size $|S| \times |T|$

For all $0 \le i \le |T|$, table[0][i] = ""

For all $1 \le j \le |S|$, table[j][0] = ""

```
LCS(S, T, n, m):
    IF table[n][m] is empty:
        // calc. a value.
        IF S[n] = T[m]:
            table[n][m] = LCS(S, T, n-1, m-1) + S[n]

        ELSE:
            LCS1 = LCS(S, T, n-1, m)
            LCS2 = LCS(S, T, n, m-1)
            IF LEN(LCS1) > LEN(LCS2):
                table[n][m] = LCS1
            ELSE
                table[n][m] = LCS2
    return table[n][m]
```



```
FOR i = 1 to n
    For j = 1 to m
        IF S[i] = T[j]:
            table[i][j] = table[i-1][j-1] + 1
        ELSE
            table[i][j] = max( table[i-1][j],
                               table[i][j-1] )
return table[n][m]
```

Matrix Chain Multiplication

Suppose we need to multiply

$$A \times B \times C \times D$$

A: 50×20, B: 20×1, C: 1×10, D: 10×100

Cost of $X \times Y$ where X is $m \times n$ and Y is $n \times p$ is prop. to mnp

(not necessarily commutative) $X \cdot Y \neq Y \cdot X$

Associative: $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$ ✓

Try $((A \times B) \times C) \times D$

cost: $50 \cdot 20 \cdot 1 = 1000$

prod: 50×1

cost: $50 \cdot 1 \cdot 10 = 500$

prod: 50×10

cost: $50 \cdot 10 \cdot 100 = 50{,}000$

prod: 50×100

total cost: $51{,}500$

$(A \times B) \times (C \times D)$

$A \times B$: c: 1000, p: 50×1

$C \times D$: c: 1000, p: 1×100

C: $50 \cdot 1 \cdot 100 = 5$

p: 50×100

total cost: $7{,}000$

Given: $A_1, A_2 \ldots A_{n-1} A_n$ w/dimensions

$p_0 \times p_1, p_1 \times p_2, p_2 \times p_3, \ldots, p_{n-1} \times p_n$

Find a parenthesization that minimizes the total # of scalar multiplications req'd.

IMAGINE: an oracle says divide at $A_1 \ldots A_{43}$ and $A_{44} \ldots A_n$ and everything works.

$A_1 \ldots A_{43} \mid A_{44} \ldots A_n$

← subproblem | → subproblem

for $i = 1$ to $n-1$

try splitting at i

All subproblems generated are of the form parenthesize $A_x \ldots A_y$ where $y \geq x$