Tutorial #6

Thursday, October 19, 2017 4:13 PM

Dynamic Programming

What is it?

- similar to Divide & Conquer
  - solves problems by combining soln's to subproblems
- when subproblems overlap
- solve subproblem only once
  - store in memory (table)
- optimization problems
  - many solutions → want the optimal one

4 Steps in Dynamic Programming

1. Characterize the structure of an optimal soln.
2. Recursively define the value of an optimal soln
3. Compute the value of an optimal soln (typically Bottom up)
4. Compute optimal soln from computed information

Longest common sequence

- given 2 sequences
  1. $X = \{x_1, \ldots, x_m\}$
  2. $Y = \{y_1, \ldots, y_n\}$

Goal: Find a subsequence common to both whose length is longest. A subsequence doesn't have to be consecutive, but it has to be in order

example: horseback

snowflake

Brute Force Approach:

→ For every subsequence of X, check whether it's a subsequence

of Y.

ex: $X = \{a, b, c, d\}$

subsequences: a, b, c, d
ab, ac, ad, bc, bd, cd
abc, acd, abd, bcd
abcd

Total = $16 = 2^m$

→ each subsequence takes $\Theta(n)$ time to check:
- check $y_1, y_2, \ldots y$

Total = $\Theta(n \cdot 2^m)$

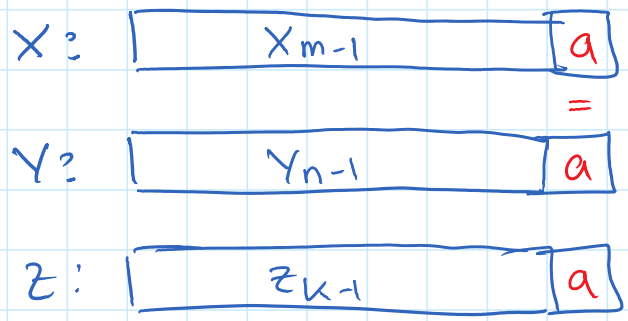
Dynamic Programming Approach

X_i = prefix $\langle x_1, \ldots, x_i \rangle$
Y_i = prefix $\langle y_1, \ldots, y_i \rangle$

Thm: Let $Z = \langle z_1, \ldots, z_k \rangle$ be any LCS of X & Y

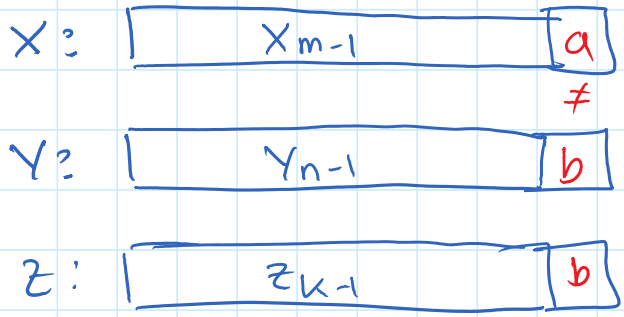
① If $x_m = y_n$: $z_k = x_m = y_m$ and $Z_{k-1} = LCS(X_{m-1}, Y_{n-1})$
② If $x_m \neq y_n$: $z_k \neq x_m$ and $Z_{k-1} = LCS(X_{m-1}, Y)$
③ If $x_m \neq y_n$: $z_k \neq y_n$ and $Z_{k-1} = LCS(X, Y_{m-1})$

Visual Interpretation

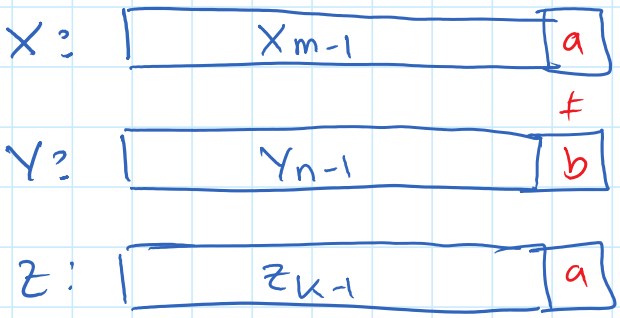
①


$Z_{k-1} = LCS(X_{m-1}, Y_{n-1})$

$z_{k-1} = LCS(x_{m-1}, y_{n-1})$

② 

$z_{k-1} = LCS(x_{m-1}, Y)$

③ 

$z_{k-1} = LCS(x, y_{n-1})$

Recursive Formulation

Define (memory) $c[i,j]$ = length of $LCS(x_i, y_j)$

Want: $c[m,n]$

$$c[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_i \\ \max(c[i-1,j], c[i,j-1]) & \text{if } i,j>0 \text{ and } x_i \neq y_i \end{cases}$$

Longest common sequence algorithm:

```
LCS-LENGTH(X, Y, m, n)
for i ← 1 to m
    do c[i, 0] ← 0
for j ← 0 to n
    do c[0, j] ← 0
for i ← 1 to m
    do for j ← 1 to n
          do if x_i = y_j
                then c[i, j] ← c[i − 1, j − 1] + 1
                     b[i, j] ← "↖"
                else if c[i − 1, j] ≥ c[i, j − 1]
                        then c[i, j] ← c[i − 1, j]
                             b[i, j] ← "↑"
                        else c[i, j] ← c[i, j − 1]
                             b[i, j] ← "←"
return c and b
```

O(n) O(m·n)

X = {A B C B D A B} m = 7
Y = {B D C A B A} n = 6
Z = {B, C, B, A} k = 4

		0	1	2	3	4	5	6
		y_i	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	0	1
2	B	0	1	1	1	1	0	1
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

for i 1:7
  for j 1:6
(1, 1) : 0
(1, 2) : 1
(1, 3) :

Sol": ① : B C B A
② : B C A B

→ Naive approach: exponential ($O(2^m \cdot n)$)
→ Dynamic programming: polynomial ($O(m \cdot n)$)

* Read Chapter 15: Dynamic Programming.