COMPSCI 330 Lecture 5 Dynamic Programming (continued)

Wednesday, September 7, 2016 4:25 PM

- Longest Common Subsequence (2-d tables)
- Longest Increasing Subsequence (Define new subproblem)
- Knapsack Problem

– Example: Longest Common Subsequence (LCS)

Input: two sequences $a = 1,2,3,2,1$ $b = 2,3,1,4,1$

Subsequence: subset of elements in the same order (not necessarily continuous)

e.g. 1,2,3 ✓ 1,3,2 ✓ 1,2,1 ✓ 1,1,2 ✗

$a = 1,2,3,2,1$

problem: find (the length of) the longest common subsequence of a, b.

(in this case 2,3,1)

(recall: look at the last step of the solution)

$a = 1,2,3,2,① \quad len(a) = n$

$b = 2,3,1,4,① \quad len(b) = m$

Q: Do a_n, b_m belong to the LCS.

case ① because $a_n = b_m$ (if $a_n \neq b_m$ this case is impossible)

it is possible both of them are in LCS

$LCS = [\ ?\ ] ① \leftarrow +1$

$[\ ?\ ] = LCS(a[1..n-1], b[1..m-1])$

case ② a_n is not in LCS

$LCS(a,b) = LCS(a[1..n-1], b[1..m])$

case ③ b_m is not in LCS

$LCS(a,b) = LCS(a[1..n], b[1..m-1])$

Recap: subproblem: Longest Common Subsequence for first i elements of a, and first j elements of b.

1. Define a "table" to store the result of subproblems

$f[i,j]$ = length of Longest Common Subsequence for first i elements of a, and first j elements of b.

2. Writing a recursive formula

assume we already know $f[i-1,j-1]$, $f[i-1,j]$, $f[i,j-1]$

if $a_i = b_j$ $f[i,j] = \max f[i-1,j-1]+1, f[i-1,j], f[i,j-1]$

if $a_i \neq b_j$ $f[i,j] = \max f[i-1,j], f[i,j-1]$

3. boundary condition $f[0,j]=0$ $f[i,0]=0$

initialize $f[0,j], f[i,0]=0$ $\forall i,j$

for $i = 1$ to n

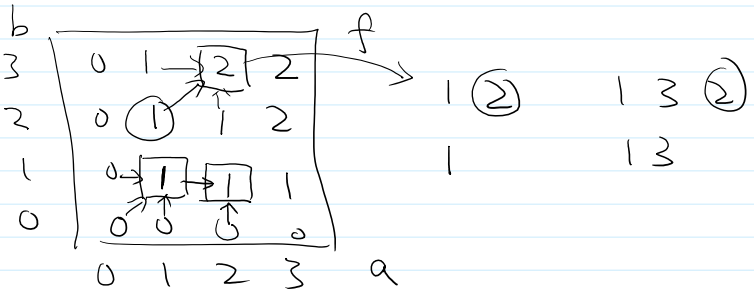
for $j = 1$ to m

$f[i,j] = \max(f[i-1,j], f[i,j-1])$ cases ②③

if $a_i = b_j$ and $f[i-1,j-1]+1 > f[i,j]$ then check for case ①

$f[i,j] = f[i-1,j-1]+1$

$a = 1\ 2\ 3$ $b = 1\ 3\ 2$



1 ② 1 3 ②

1 1 3

4. reconstruct the solution

if $f[i,j] = f[i-1,j-1]+1$ and $a_i = b_j$

$LCS(i,j) = LCS(i-1,j-1), a_i$

if $f[i,j] = f[i-1,j]$

$LCS(i,j) = LCS(i-1,j)$

output (i,j)

if $f[i,j]=0$ return

if $f[i,j] = f[i-1,j]$ } case ②

output $(i-1,j)$

if $f[i,j] = f[i,j-1]$ } case ③

output $(i,j-1)$

output $(i-1,j-1)$ } case ①

print (a_i)

output(3,3) → output(2,3) → output(1,2) → output(1,1) → output(0,0)

1,2,3 1,3,2 1,2 1,3,2 1 1,3 1 1 ∅ ∅

1,2 1

Longest Increasing Subsequence (LIS)

Input: sequence $a = 1, 3, 2, 5, 6, 4$

Output: longest increasing subsequence 1, 2, 5, 6

try: find subproblems

look at a_n, is a_n in the LIS

no ⇒ LIS = LIS($a[1, \dots n-1]$)

yes ⇒ find an increasing subsequence of $a[1, \dots, n-1]$ whose last element < 4 (a_n)

$f[i]$ = LIS of $a[1..i]$ ✗ not working because do not know last element.

alternative subproblem

$f[i]$ = length of LIS for $a[1..i]$ that stops at $a[i]$

recursion

$$f[i] = \max_{\substack{j<i \\ a[j]<a[i]}} f[j] + 1$$

boundary condition if for any $j<i$, $a[j] \geq a[i]$

$f[i] = 1$

```
for i = 1 to n
    f[i] = 1
    for j = 1 to i-1
        if a[j] < a[i] and f[j]+1 > f[i] then
            f[i] = f[j]+1
            prev[i] = j
```

increasing subsequence

maximize length of sequence

computing $\max_{j<i} f[j]+1$

store the arrows

$a = 1, 3, 2, 5, 6, 4$

$f = 1, 2, 2, 3, 4, 3$

$f[i]$ = LIS that ends at $a[i]$

Knapsack Problem: Given n objects with weights $w_1, w_2, \cdots, w_n$
A Knapsack can hold at most weight m
Goal: find a subset of objects so that total weight w
$w \le m$, w is also as close m as possible

subproblem either the n-th object is not in Knapsack
remaining problem: capacity m, $n-1$ objects
or the n-th object is in Knapsack
capacity $m - w_n$, $n-1$ objects

$f[i,j]$ = max possible weight if capacity $= i$, first j objects.

$$f[i,j] = \max f[i,j-1],\ f[i-w_j, j-1] + w_j$$

boundary $f[i,0] = 0$ $f[0,j] = 0$

object i has value v_i, want to find a subset of objects
total weight $\le m$, total value maximized.

$m = 10$ $w_1 = 6$ $v_1 = 8$ ✓
$w_2 = 7$ $v_2 = 6$

$$f[i,j] = \max f[i,j-1],\ f[i-w_j, j-1] + w_j$$

$w_j \downarrow v_j$