Algorithm Design Techniques

- Induction
- Divide and Conquer
- **Dynamic Programming**
- Greedy
- Reduction

Longest Common Subsequence

- Input: Two lists of integers
  $x_1,x_2, \ldots, x_m$ and $y_1,y_2, \ldots, y_n$
- Output: A longest subsequence of X that is also a subsequence of Y

LCS - Example

- Input: X = 1, -2, 3, 4, 9, 18
  Y = 3, 9, 1, -2, 5, -2, 22, 18
- Output: Z = 1, -2, 18

LCS - Algorithm

LCS($X=x_1,x_2,\ldots,x_m$; $Y=y_1,y_2,\ldots,y_n$)

$Z_1=Z_2=Z_3=\phi$

If $m>0$ then $Z_1=LCS(x_1,x_2,\ldots,x_{m-1}; y_1,y_2,\ldots y_n)$

If $n>0$ then $Z_2=LCS(x_1,x_2,\ldots,x_m; y_1,y_2,\ldots y_{n-1})$

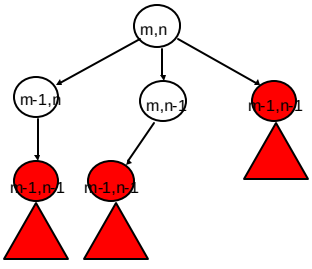
If $n>0$ and $m>0$ and $x_m=y_n$ then
$Z_3=LCS(x_1,x_2,\ldots,x_{m-1}; y_1,y_2,\ldots y_{n-1}),x_m$

Return the longest of Z_1, Z_2, Z_3

Run Time

$T(m,n) = T(m-1,n) + T(m,n-1) + T(m-1,n-1) + c$
$\geq 3T(m-1,n-1) + c$
$= \Omega(3^{\min(m,n)})$

Run Time Analysis
Many duplicated subtrees



Dynamic Programming

- A(i,j) is the length of a longest common subsequence of $x_1,x_2,\ldots,x_i$ and $y_1,y_2,\ldots,y_j$
- $A(i,0)=A(0,j)=0 \quad 0 \leq i \leq m, 0 \leq j \leq n$
- A(i,j)= maximum of
  A(i-1,j), A(i,j-1), and $A(i-1,j-1)+match(x_i,y_j)$
- A(m,n) is the length of a longest common subsequence of X and Y

A(i,j)

		1	-2	3	4	9	18
	0	0	0	0	0	0	0
3	0	0	0	1	**1**	**1**	1
9	0	0	0	1	**1**		
1	0						
-2	0						
18	0						

LCS - Algorithm

LCS($X=x_1,x_2,\ldots,x_m;Y=y_1,y_2,\ldots,y_n$)

For i=0 to m: A(i,0)=0
For i=0 to n: A(0,i)=0
For i=1 to m
  For j=1 to n
    If $x_i=y_j$ then match=1 else match=0
    A(i,j) =max(A(i-1,j),A(i,j-1),A(i-1,j-1)+match))
Return A(m,n)

A(i,j)

		1	-2	3	4	9	18
	0	0	0	0	0	0	0
3	0	0	0	1	1	1	1
9	0	0	0	1	1	2	2
1	0	1	1	1	1	2	2
-2	0	1	2	2	2	2	2
5	0	1	2	2	2	2	3
18	0	1	2	2	2	2	3

Dynamic Programming

Run Time: O(nm) to find length of LCS

Backtracking

		1	-2	3	4	9	18
	0	0	0	0	0	0	0
3	0	0	0	1	1	1	1
9	0	0	0	1	1	2	2
1	0	1	1	1	1	2	2
-2	0	1	2	2	2	2	2
18	0	1	2	2	2	2	3