Subsequence

A subsequence of a sequence/string $X = < x_1, x_2, \ldots, x_m >$ is a sequence obtained by deleting 0 or more elements from X.

Example: "sudan" is a subsequence of "sesquipedalian".

So is "equal".

$\boxed{\text{There are } 2^m \text{ subsequences of } X.}$

A common subsequence Z of two sequences X and Y is a subsequence of both.

Example: "ua" is a common subsequence of "sudan" and "equal".

Longest Common Subsequence

Input: $X = < x_1, x_2, \ldots, x_m >$

$Y = < y_1, y_2, \ldots, y_n >$

Output: a *longest common subsequence* (LCS) of X and Y.

Example a) X = abcbdab Y = bdcaba

LCS_1 = bcba LCS_2 = bdab

b) X = enquiring Y = sequipedalian

LCS = equiin

c) X = empty bottle Y = nematode knowledge

LCS = emt ole

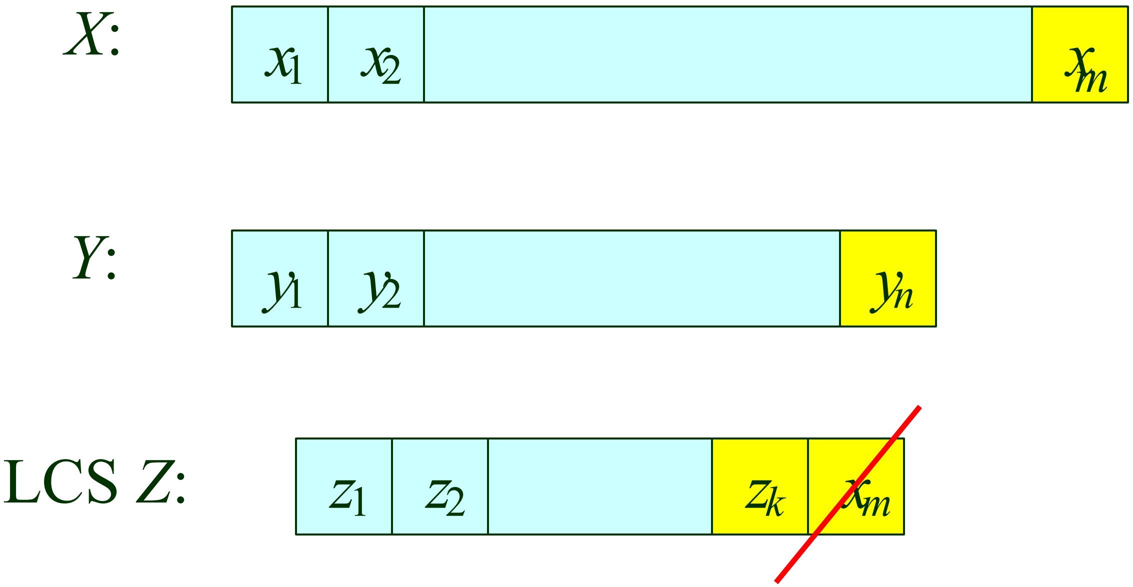
Brute-force Algorithm

For every subsequence of X, check if it's a subsequence of Y.

Worst-case running time: $\Theta(\, n\, 2^m\,)$!

- 2^m subsequences of X to check.
- Each check takes $O(n)$ time – scanning Y for first element, then from there for next element ...

Optimal Substructure of LCS



Case 1: $x_m = y_n$

Then $z_k = x_m = y_n$

Otherwise, LCS has length $\geq k + 1$, a contradiction.

Optimal Substructure (cont'd)

Case 2: $x_m \neq y_n$

X:	x_1	x_2		x_m
Y:	y_1	y_2		y_n

Either LCS of

x_1	x_2		x_{m-1}
y_1	y_2		y_n

or LCS of

x_1	x_2		x_m
y_1	y_2		y_{n-1}

A Recursive Formula

$X = < x_1, x_2, \ldots, x_m >, \quad Y = < y_1, y_2, \ldots, y_n >$

$c[i, j]$ = length of an LCS of $< x_1, \ldots, x_i >$ and $< y_1, \ldots, y_j >$.

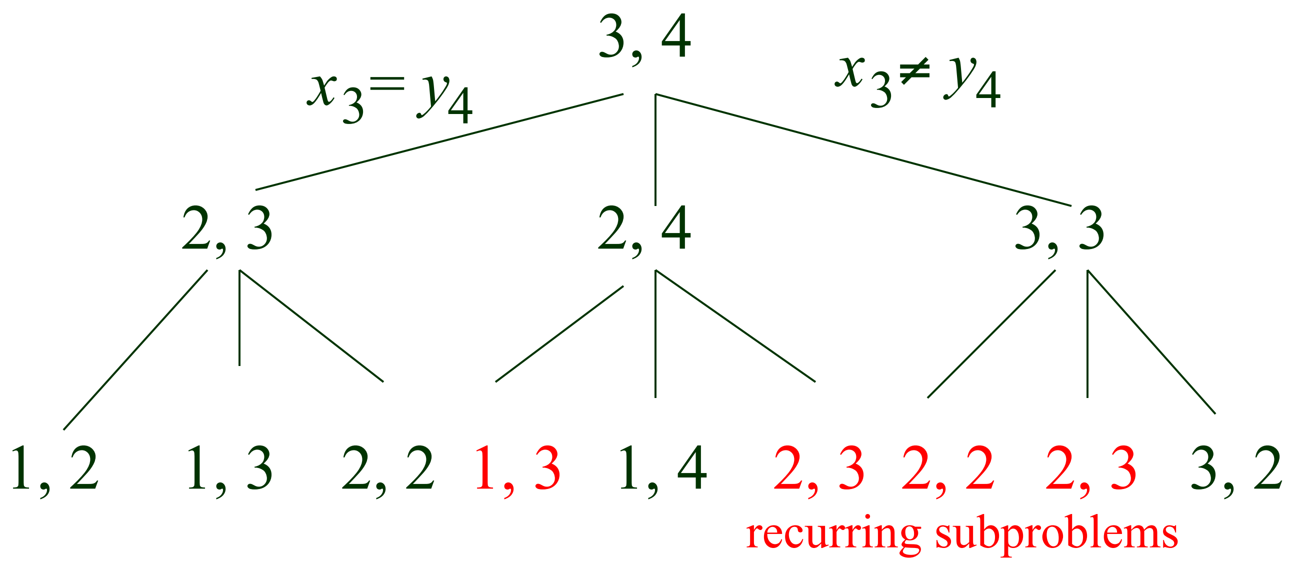
Then $c[m,n]$ = length of an LCS of X and Y.

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

A total of $(m + 1)(n + 1)$ subproblems.

Recursion Tree

$m = 3$ and $n = 4$



Depth of the tree $\leq m+n$ since at each level i and/or j reduce by 1

Branches by at most 3 at each node.

Amount of work for top-down recursion: $3^{\Theta(m+n)}$

Constructing an LCS

1. Initialize entries $c[i, 0]$ and $c[0, j]$

$c[\ ,\ ]$		s	e	s	q	u	i	p	e	d	a	l	i	a	n
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0														
n	0														
q	0														
u	0														
i	0														
r	0														
i	0														
n	0														
g	0														

$m = 9$

$n = 14$

Constructing an LCS (cont’d)

2. Fill out entries row by row. Save pointers in $b[1..m, 1..n]$.

$c[\,,\,]$		s	e	s	q	u	i	p	e	d	a	l	i	a	n
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
n	0														
q	0														
u	0														
i	0														
r	0														
i	0														
n	0														
g	0														

$c[1, 2] = 1 + c[0, 1] = 1$

$c[1, 7] = \max(c[1, 6], c[0, 7]) = 1$

LCS Length Determined

c[,]		s	e	s	q	u	i	p	e	d	a	l	i	a	n
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
n	0	0	1	1	1	1	1	1	1	1	1	1	1	1	2
q	0	0	1	1	2	2	2	2	2	2	2	2	2	2	2
u	0	0	1	1	2	3	3	3	3	3	3	3	3	3	3
i	0	0	1	1	2	3	4	4	4	4	4	4	4	4	4
r	0	0	1	1	2	3	4	4	4	4	4	4	4	4	4
i	0	0	1	1	2	3	4	4	4	4	4	4	5	5	5
n	0	0	1	1	2	3	4	4	4	4	4	4	5	5	6
g	0	0	1	1	2	3	4	4	4	4	4	4	5	5	6

LCS has length 6.

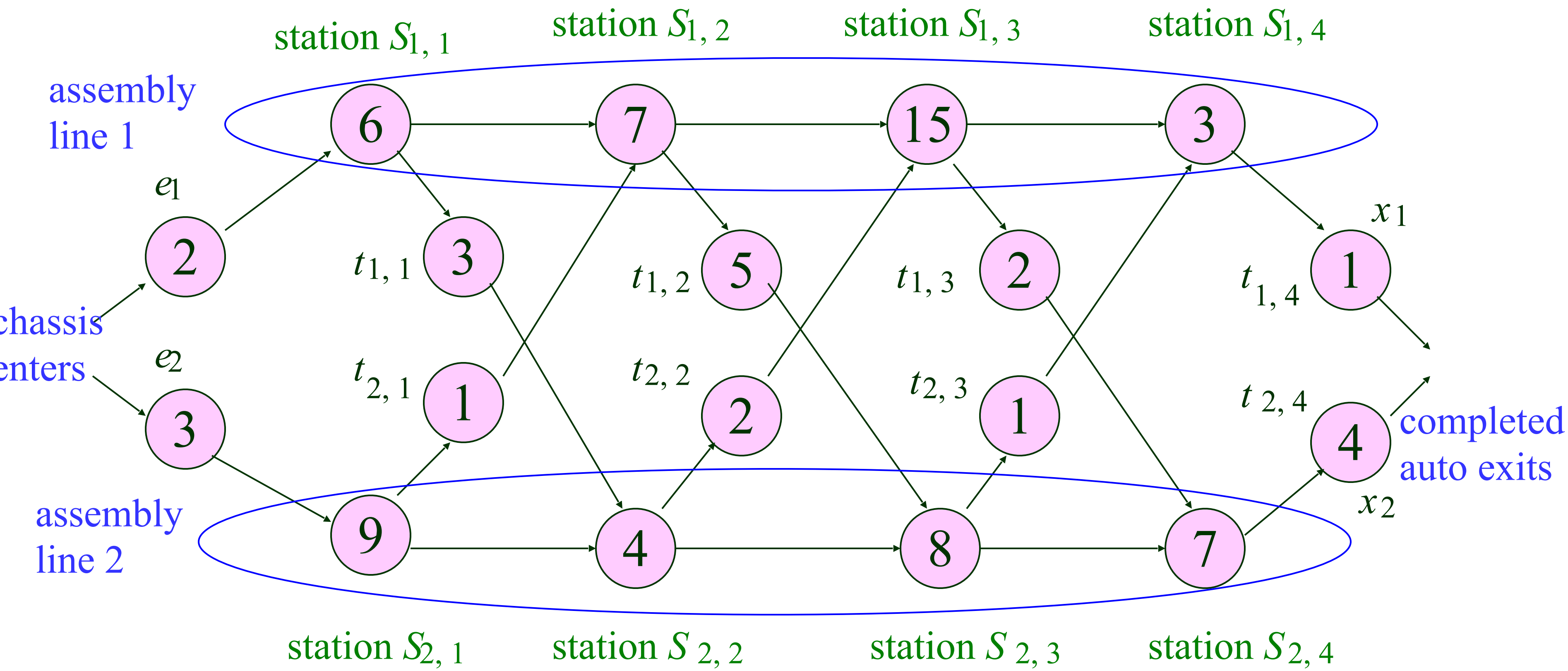
Reconstructing an LCS

3. Starting from the lower-right corner, follow the pointers in *b*[,].
   Whenever encounters an upper-left pointer, print the labeling char.

c[,]		s	e	s	q	u	i	p	e	d	a	l	i	a	n	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
e	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	Print “e”
n	0	0	1	1	1	1	1	1	1	1	1	1	1	1	2	
q	0	0	1	1	2	2	2	2	2	2	2	2	2	2	2	Print “q”
u	0	0	1	1	2	3	3	3	3	3	3	3	3	3	3	Print “u”
i	0	0	1	1	2	3	4	4	4	4	4	4	4	4	4	Print “i”
r	0	0	1	1	2	3	4	4	4	4	4	4	4	4	4	
i	0	0	1	1	2	3	4	4	4	4	4	4	5	5	5	Print “i”
n	0	0	1	1	2	3	4	4	4	4	4	4	5	5	6	Print “n”
g	0	0	1	1	2	3	4	4	4	4	4	4	5	5	6	

LCS = equiin

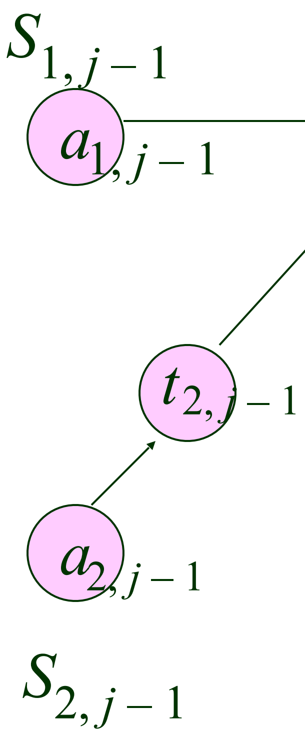
Assembly-Line Scheduling



Minimize the total time through the factory for one auto.

Optimal Substructure

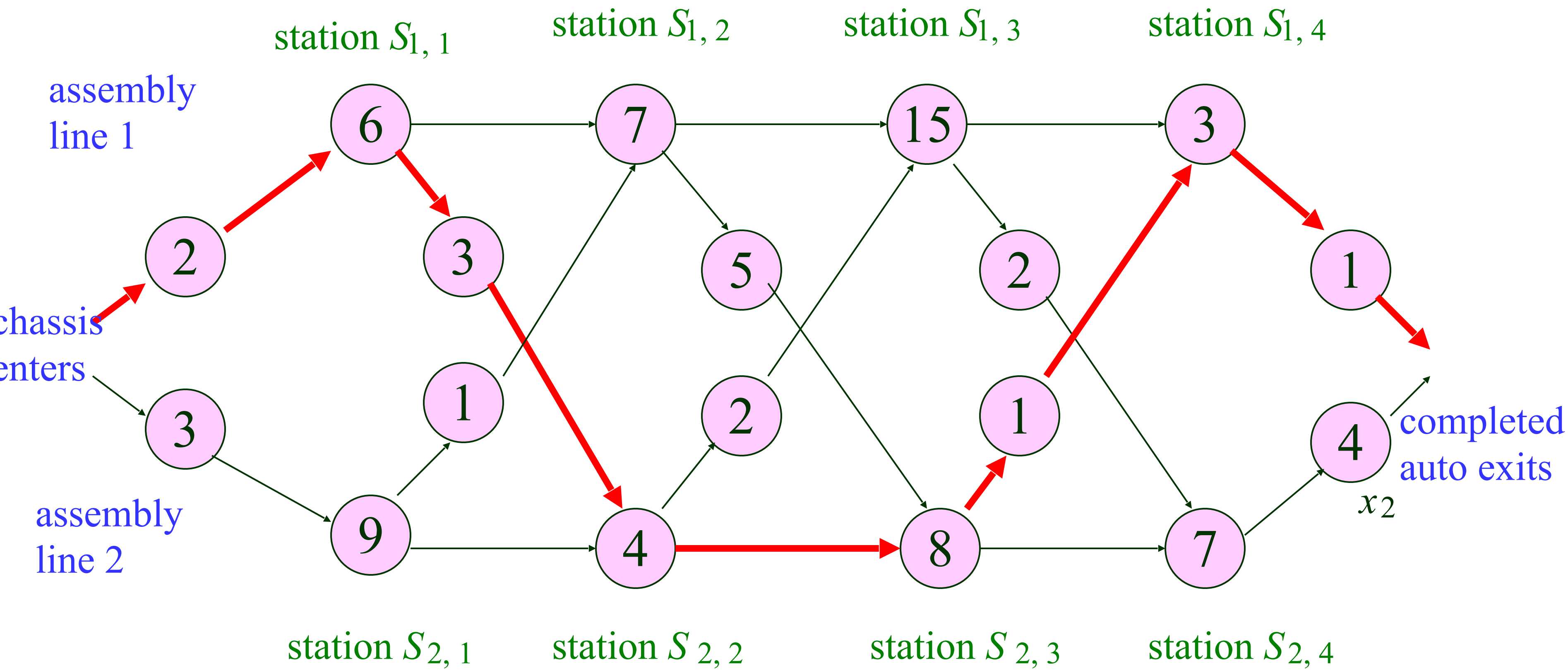
$f_i(j)$: the fastest possible time to get a chassis from the starting point through station $S_{i,j}$.



$$f_1(j) = \begin{cases} e_1 + a_{1,1} & \text{if } j = 1 \\ \min(f_1[j-1] + a_{1,j},\ f_2[j-1] + t_{2,j-1} + a_{1,j}), & \text{if } j > 1 \end{cases}$$

$$f^* = \min(f_1[n] + x_1,\ f_2[n] + x_2).$$

The DP Solution



j	1	2	3	4
$f_1[j]$	8	15	30	27
$f_2[j]$	12	15	23	30

$f^* = 28$

j	2	3	4
$l_1[j]$	1	1	2
$l_2[j]$	1	2	2

from line 2