

Dynamic Programming

Heejin Park

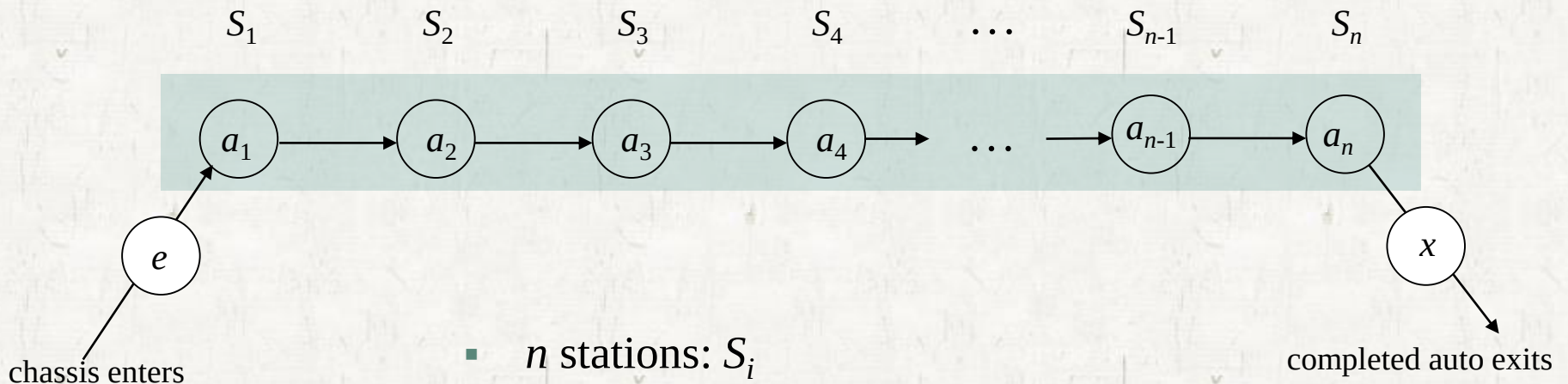
Hanyang University

Contents

- **Assembly-line scheduling**
- **Rod cutting**
- **Longest common subsequence**
- **Matrix-chain multiplication**

Assembly-line scheduling

assembly line

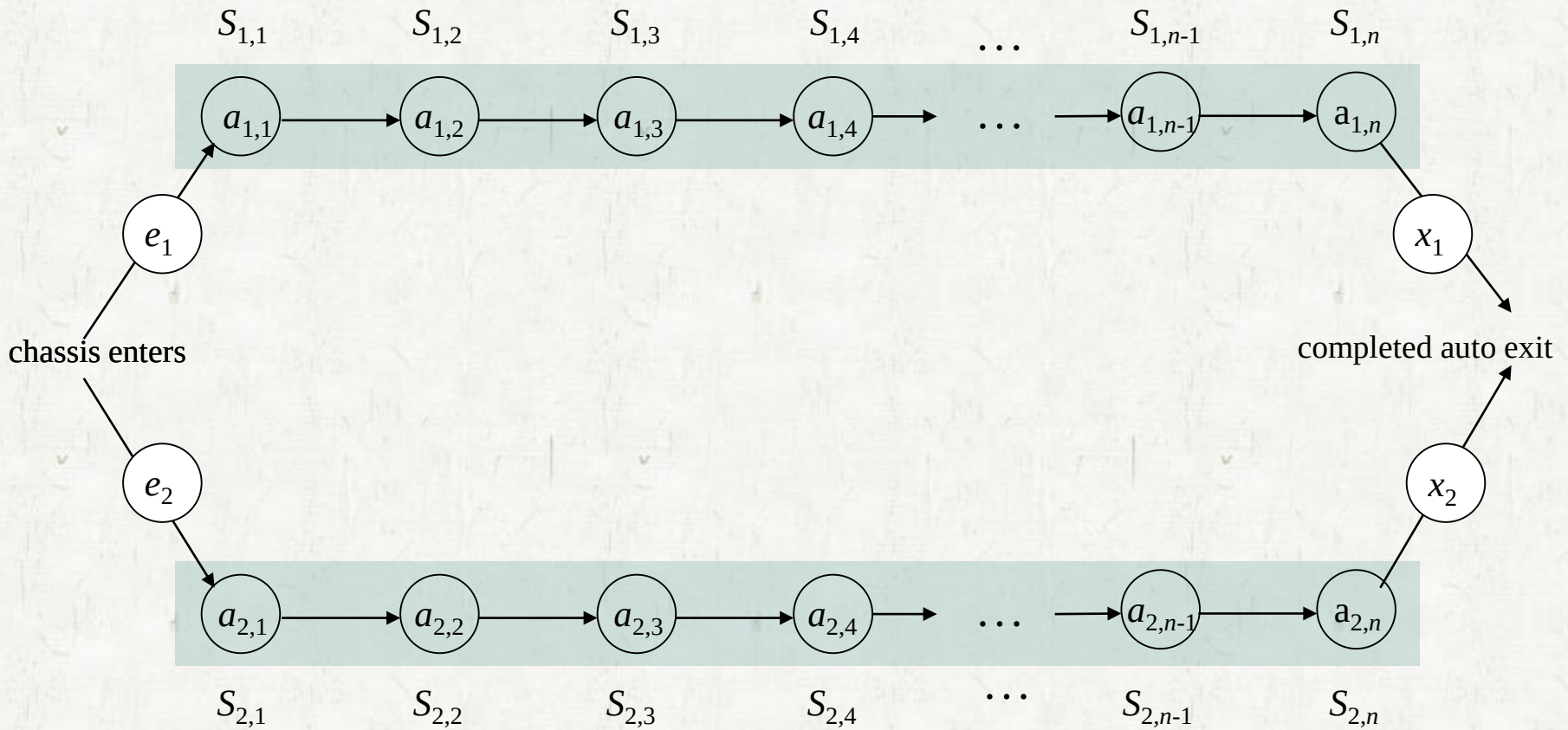


- n stations: S_i
- Assembly time in the i th station: a_i
- Entry time : e
- Exit time : x

● **The problem:** Determine the fastest assembly time

Assembly-line scheduling

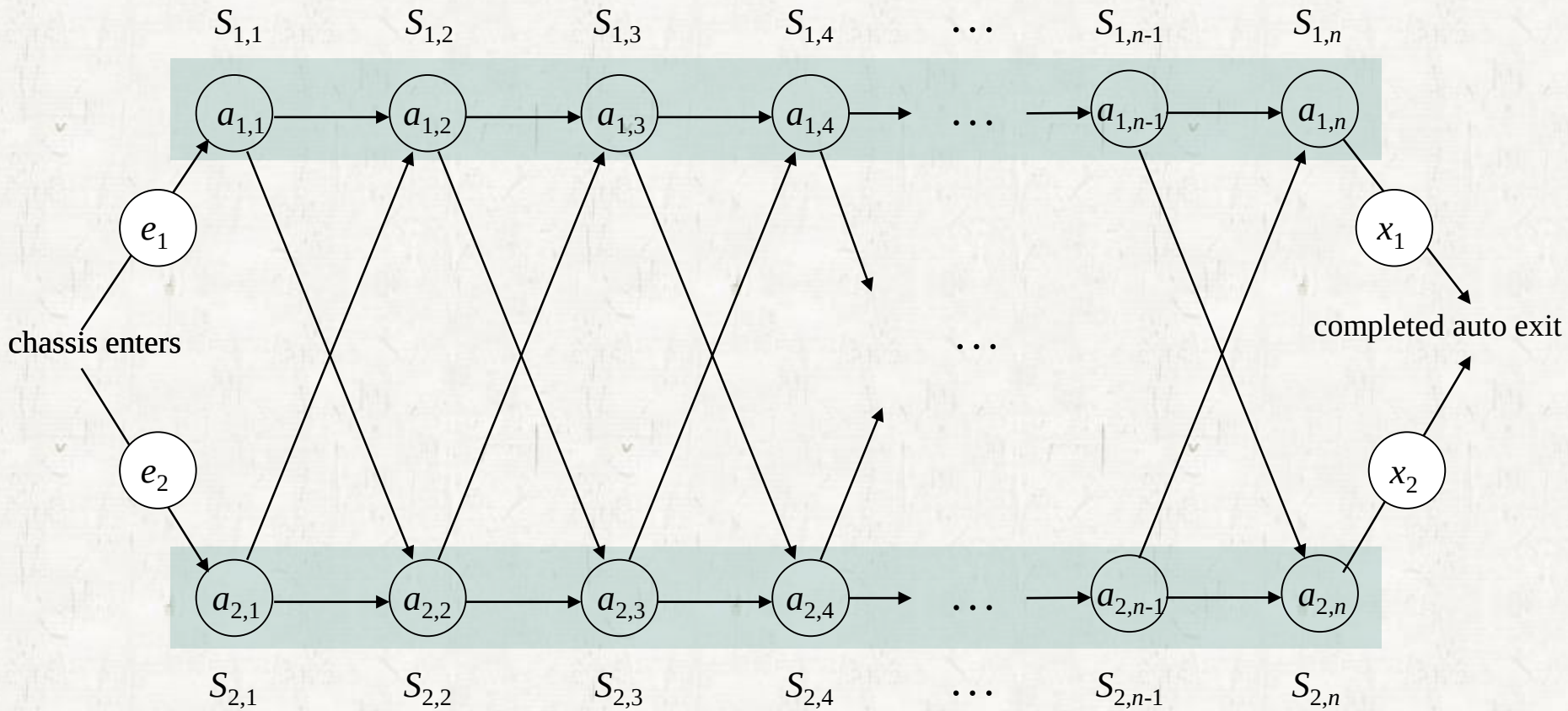
line 1



line 2

Assembly-line scheduling

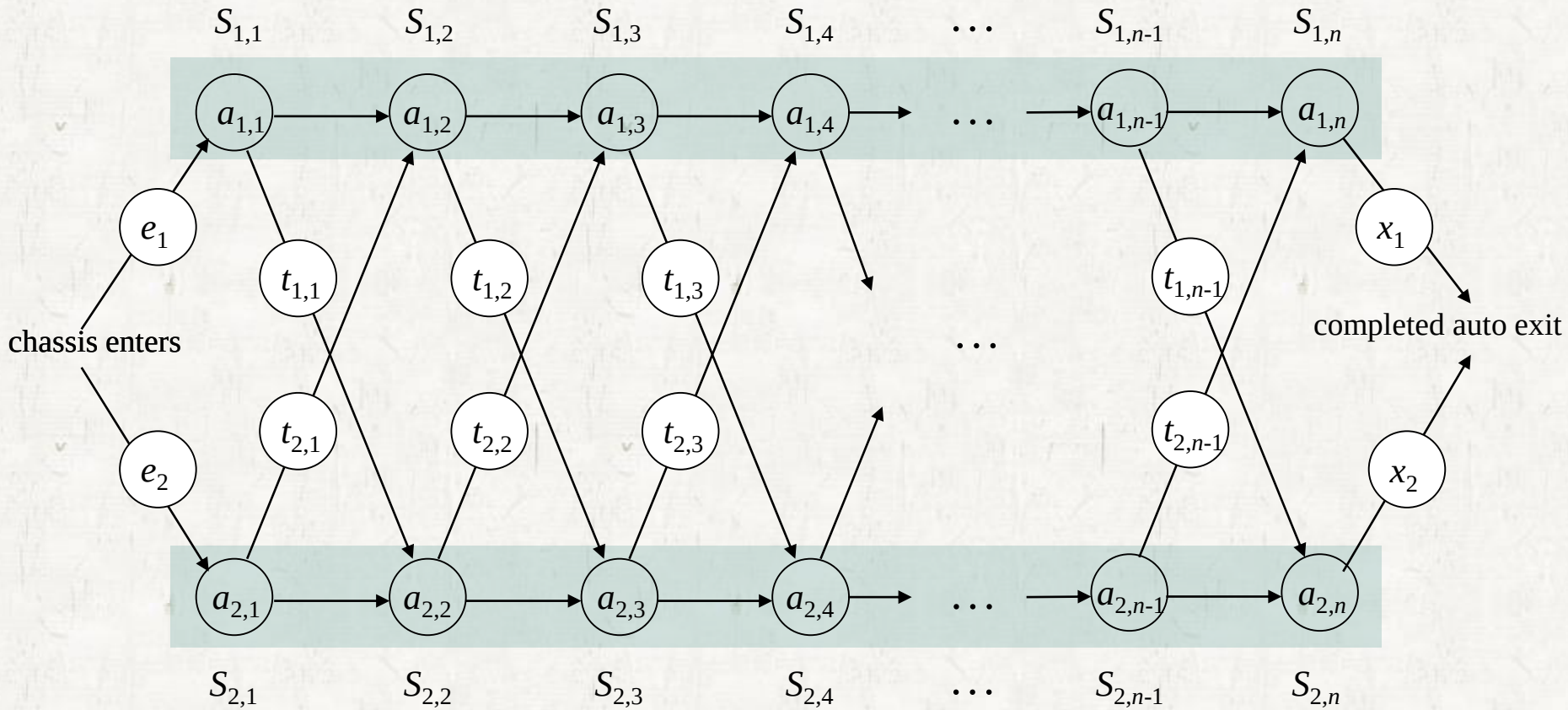
line 1



line 2

Assembly-line scheduling

line 1



line 2

- Transfer time: $t_{i,j}$

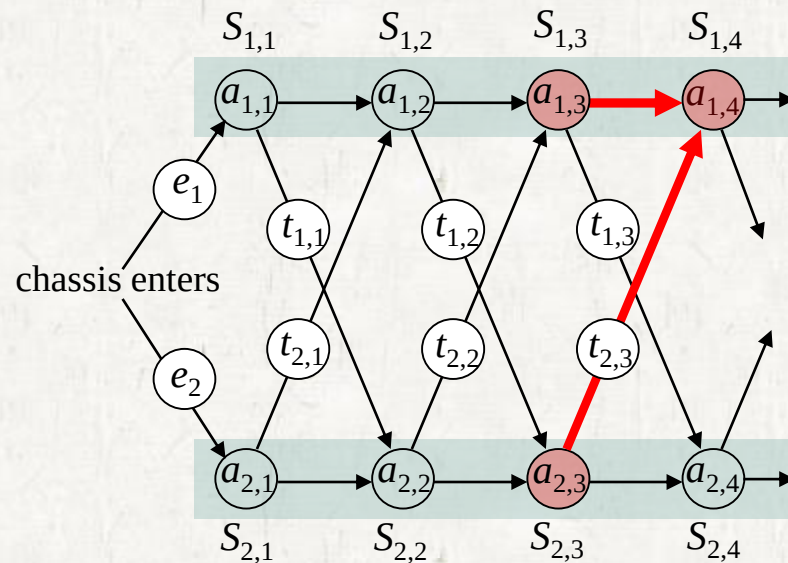
Assembly-line scheduling

- **Brute-force approach**

- Enumerate all possible ways and find a fastest way.
- There are 2^n possible ways: Too many.

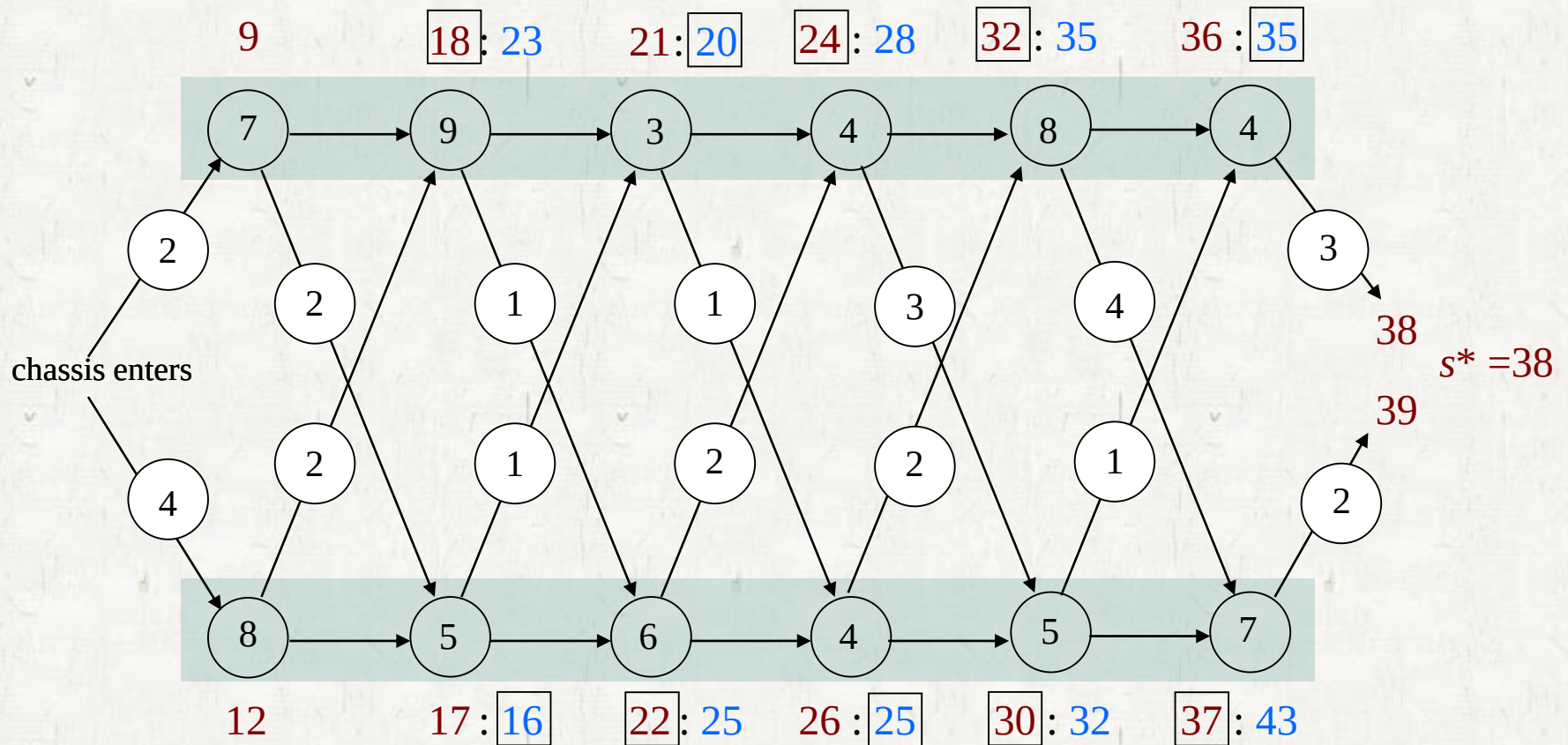
Assembly-line scheduling

The fastest way to $S_{i,j}$ goes through $S_{1,j-1}$ or $S_{2,j-1}$.

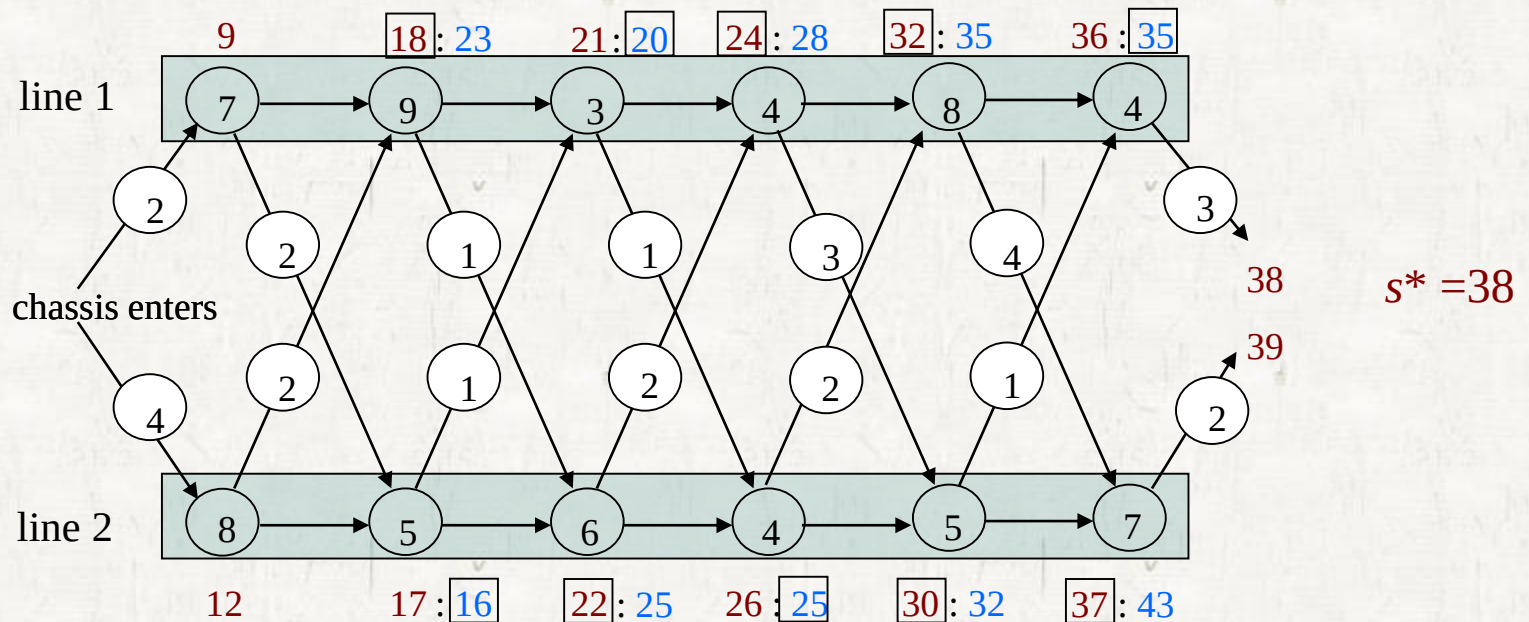


Assembly-line scheduling

- Dynamic programming



Assembly-line scheduling

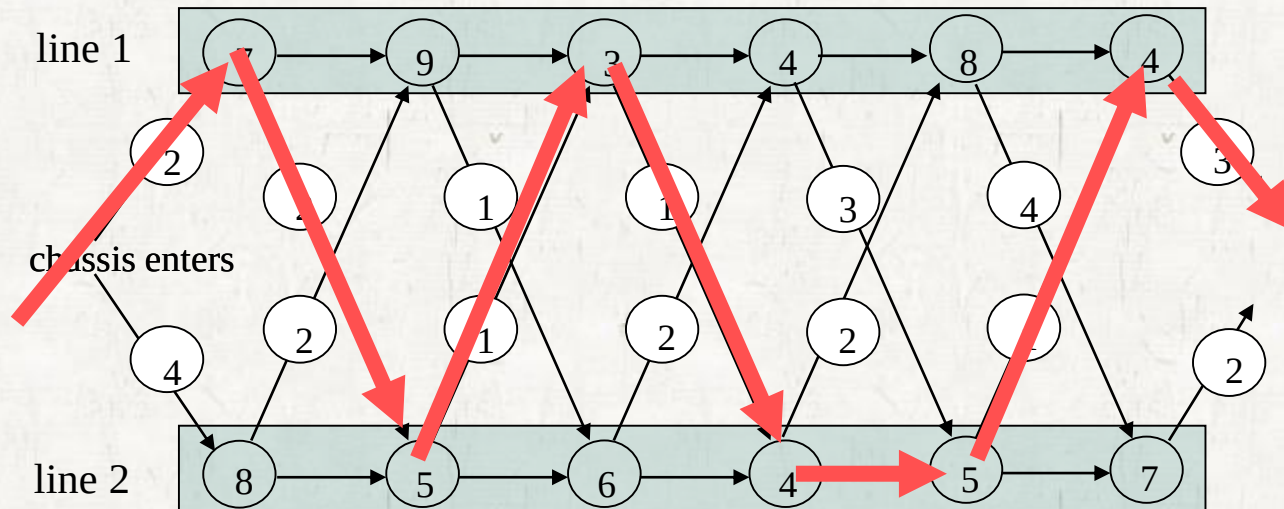


S	1	2	3	4	5	6
1	9	18	20	24	32	35
2	12	16	22	25	30	37

$s^* = 38$

● *Fastest time*

Assembly-line scheduling

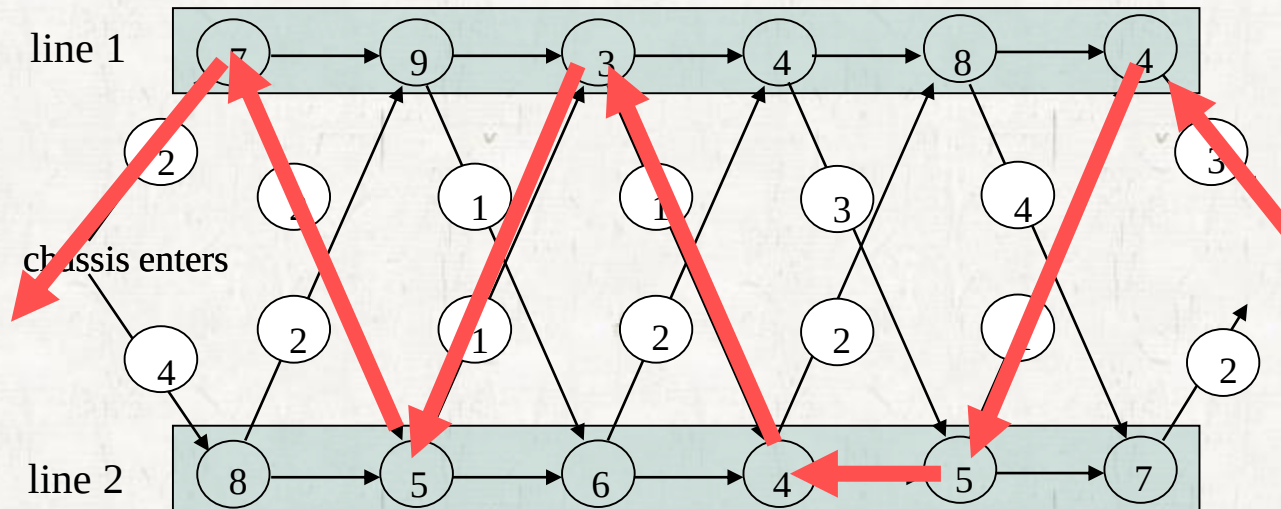


S	1	2	3	4	5	6
1	9	18	20	24	32	35
2	12	16	22	25	30	37

$$S^* = 38$$

● *Fastest way*

Assembly-line scheduling

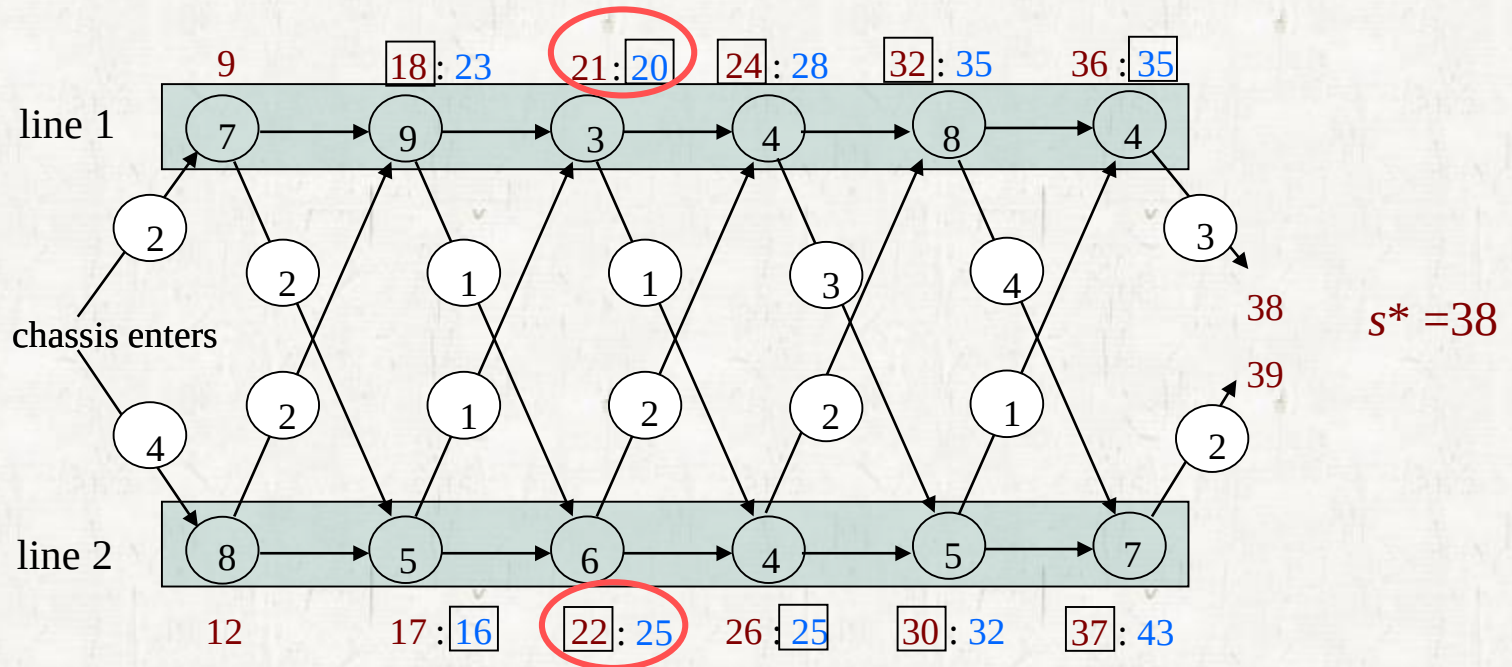


S	1	2	3	4	5	6
1	9	18	20	24	32	35
2	12	16	22	25	30	37

$$s^* = 38$$

● *Fastest way*

Assembly-line scheduling



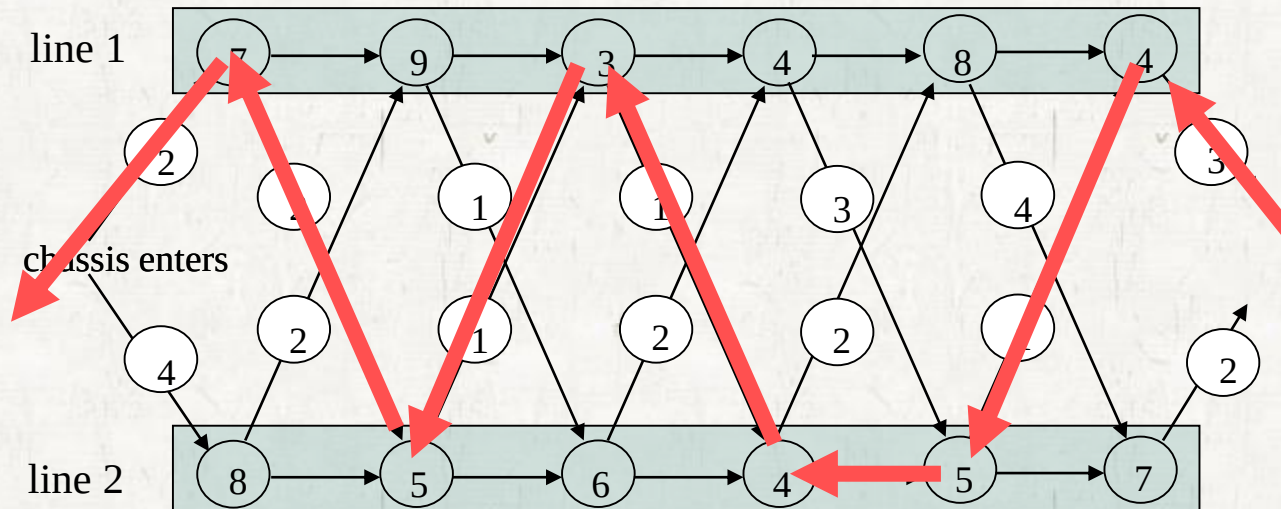
S	1	2	3	4	5	6
1	9	18	20	24	32	35
2	12	16	22	25	30	37

$$s^* = 38$$

L	1	2	3	4	5	6
1	-	1	2	1	1	2
2	-	1	2	1	2	2

$$l^* = 1$$

Assembly-line scheduling



S	1	2	3	4	5	6
1	9	18	20	24	32	35
2	12	16	22	25	30	37

$$s^* = 38$$

L	1	2	3	4	5	6
1	-	1	2	1	1	2
2	-	1	2	1	2	2

$$l^* = 1$$

Assembly-line scheduling

FASTEST-WAY (a, t, e, x, n)

```
1   $s[1][1] = e[1] + a[1][1]$ 
2   $s[2][1] = e[2] + a[2][1]$ 

3  for  $j = 2$  to  $n$ 
4      if  $s[1][j-1] \leq s[2][j-1] + t[2][j-1]$ 
5           $s[1][j] = s[1][j-1] + a[1][j]$ 
6           $l[1][j] = 1$ 
7      else  $s[1][j] = s[2][j-1] + t[2][j-1] + a[1][j]$ 
8           $l[1][j] = 2$ 

9      if  $s[2][j-1] \leq s[1][j-1] + t[1][j-1]$ 
10          $s[2][j] = s[2][j-1] + a[2][j]$ 
11          $l[2][j] = 2$ 
12     else  $s[2][j] = s[1][j-1] + t[1][j-1] + a[2][j]$ 
13          $l[2][j] = 1$ 

14 if  $s[1][n] + x[1] \leq s[2][n] + x[2]$ 
15      $s^* = s[1][n] + x[1]$ 
16      $l^* = 1$ 
17 else  $s^* = s[2][n] + x[2]$ 
18      $l^* = 2$ 
```

Assembly-line scheduling

PRINT-STATIONS (l, l^*, n)

```
1   $i = l^*$ 
2  print "line "  $i$  ", station"  $n$ 
3  for  $j = n$  downto 2
4     $i = l[i][j]$ 
5    print "line "  $i$  ", station"  $j - 1$ 
```

Result of PRINT-STATIONS

line 1, station 6
line 2, station 5
line 2, station 4
line 1, station 3
line 2, station 2
line 1, station 1

L	1	2	3	4	5	6
1	-	1	2	1	1	2
2	-	1	2	1	2	2

$l^* = 1$

Assembly-line scheduling

• Space consumption

- Table s : $2n$
- Table l : $2n$
- $\Theta(n)$ elements in total.

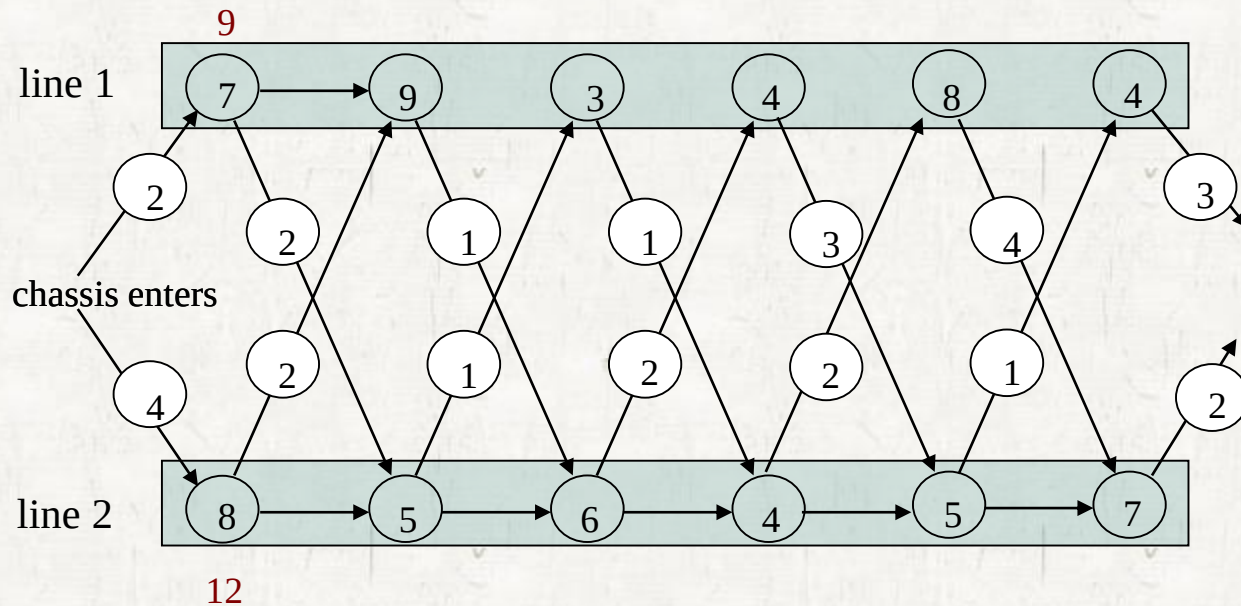
• Running time

- Computing each element requires $\Theta(1)$ time.
- $\Theta(n)$ time in total

Assembly-line scheduling

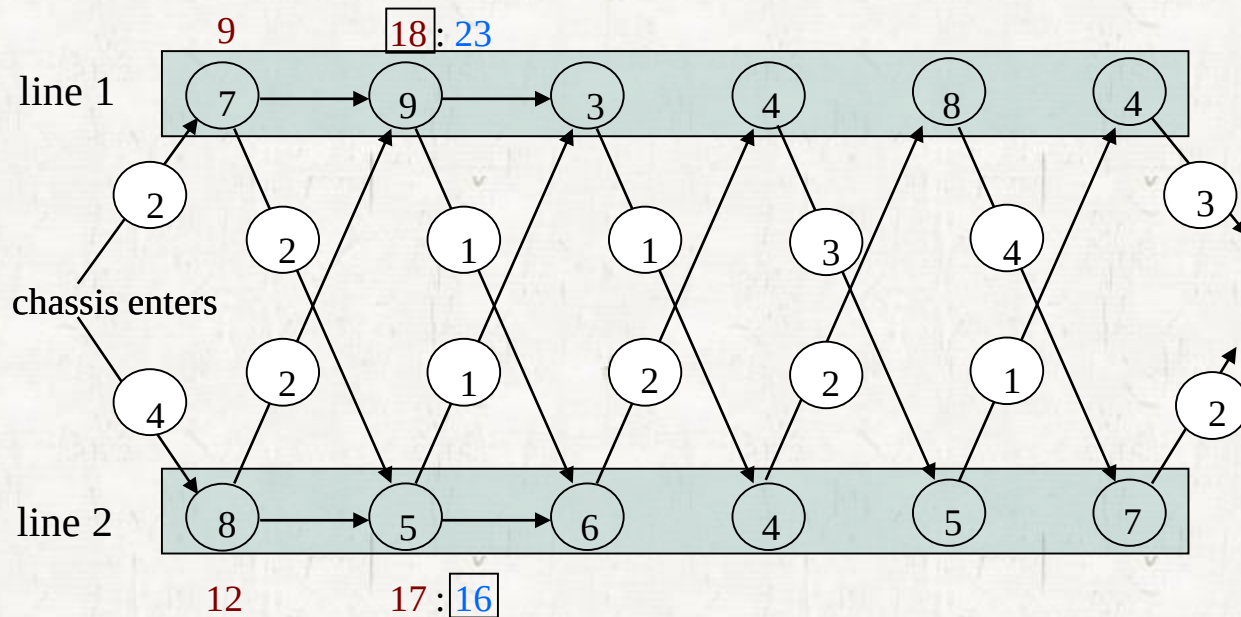
- **Fastest time only**
 - $\Theta(1)$ space
 - Table *l* is not necessary.
 - Table *s*
 - $2n$ elements $\rightarrow$ 4 elements

Assembly-line scheduling



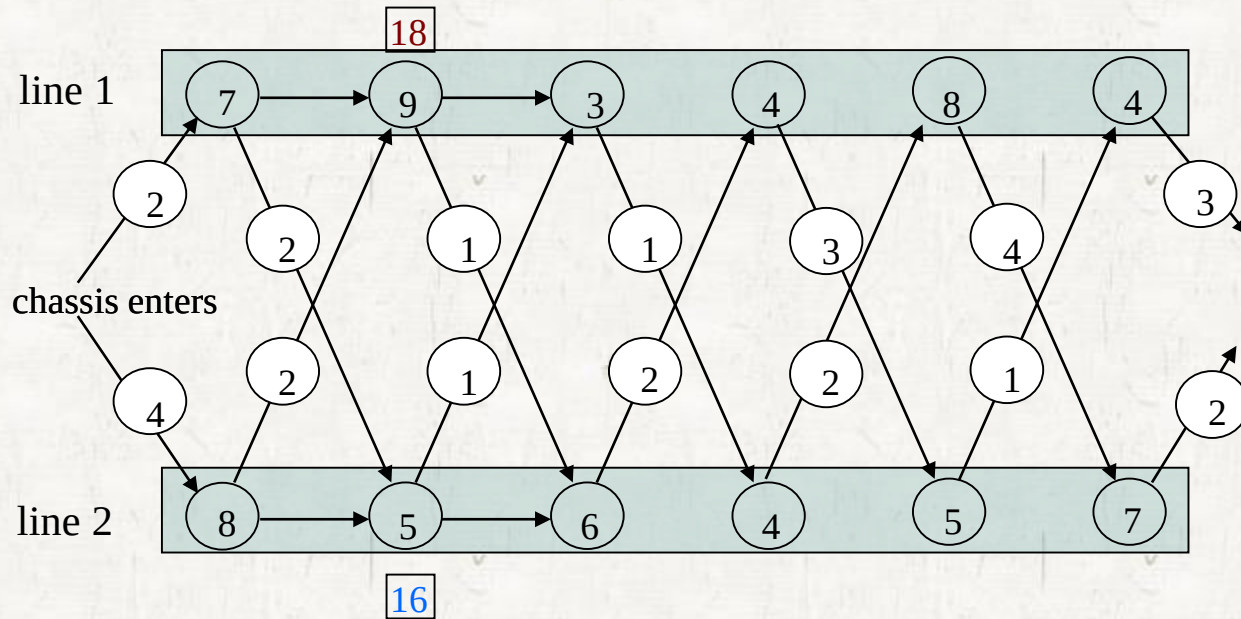
S	1	2	3	4	5	6
1	9					
2	12					

Assembly-line scheduling



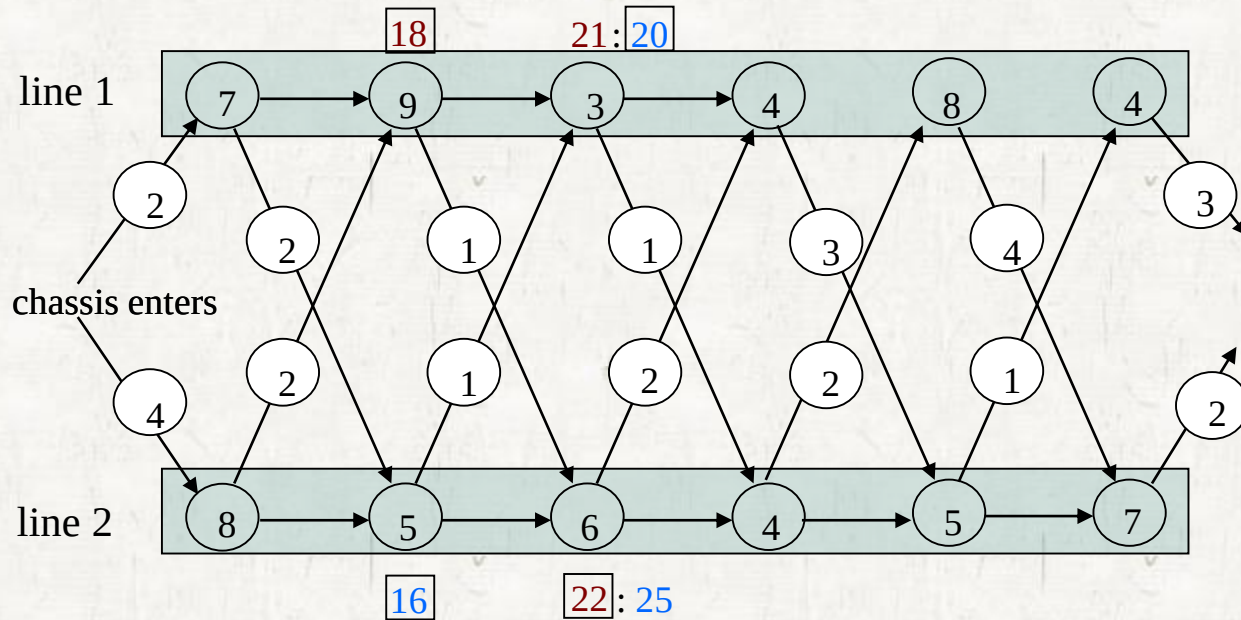
S	1	2	3	4	5	6
1	9	18				
2	12	16				

Assembly-line scheduling



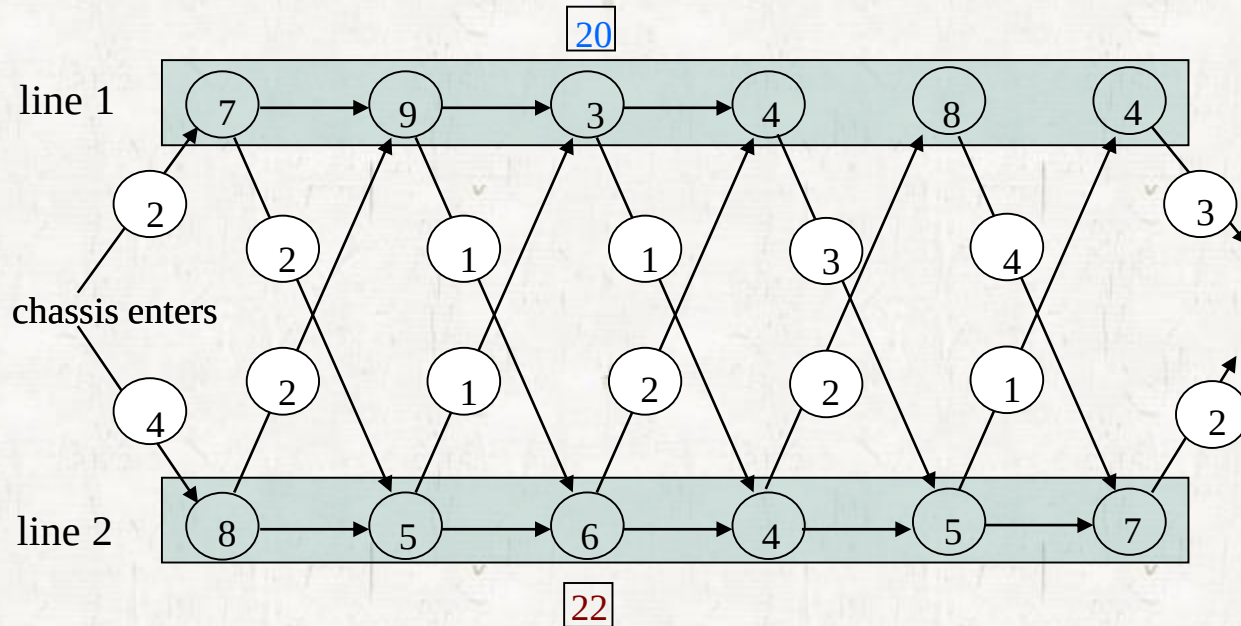
S	1	2	3	4	5	6
1		18				
2		16				

Assembly-line scheduling



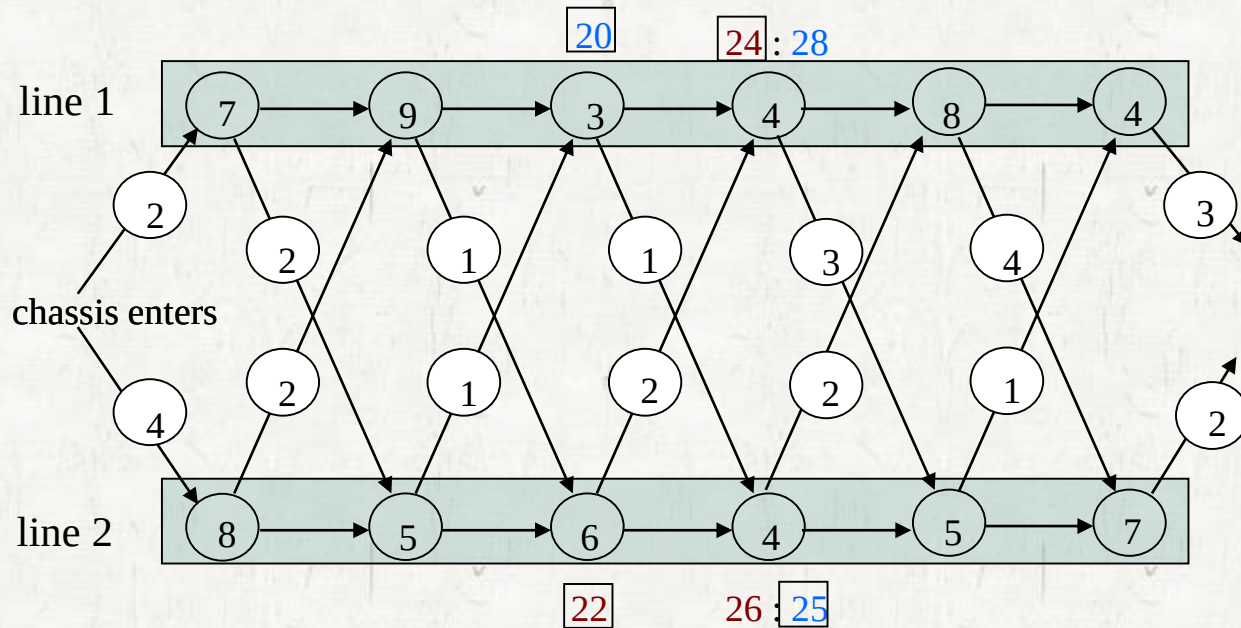
S	1	2	3	4	5	6
1		18	20			
2		16	22			

Assembly-line scheduling



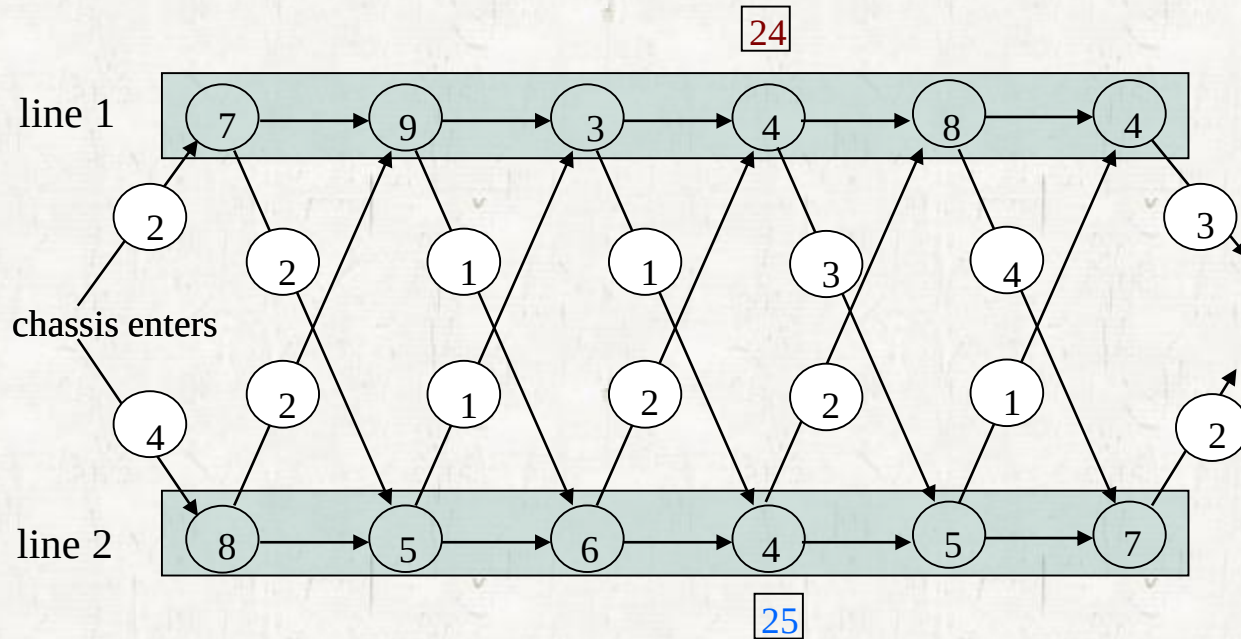
S	1	2	3	4	5	6
1			20			
2			22			

Assembly-line scheduling



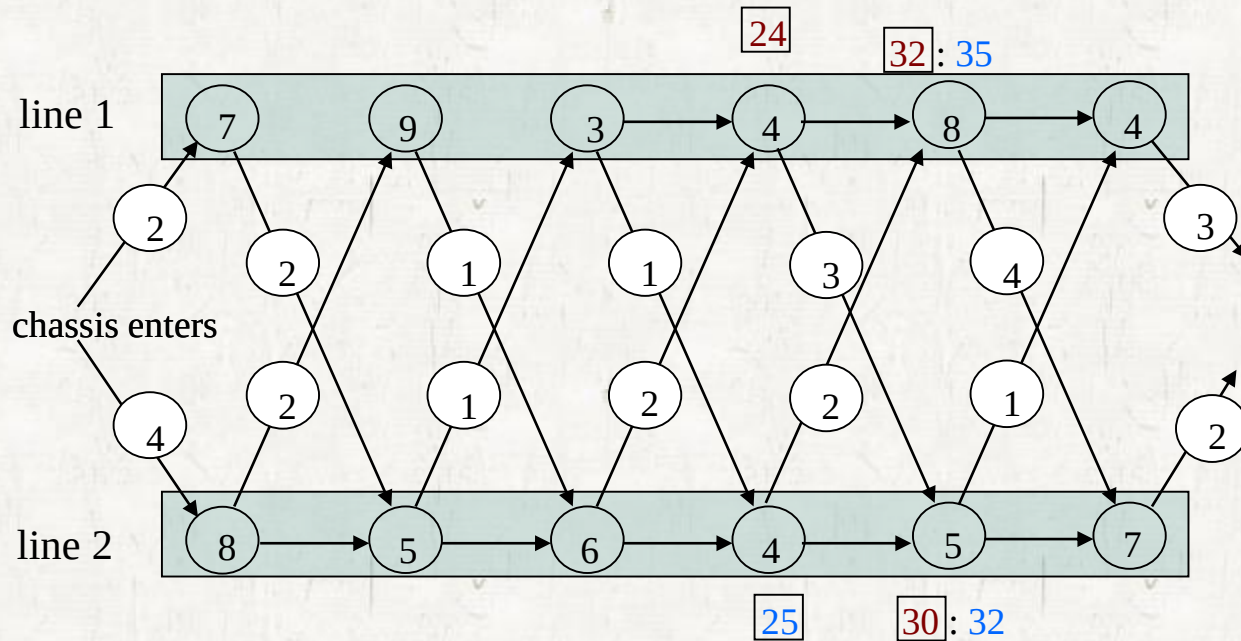
S	1	2	3	4	5	6
1			20	24		
2			22	25		

Assembly-line scheduling



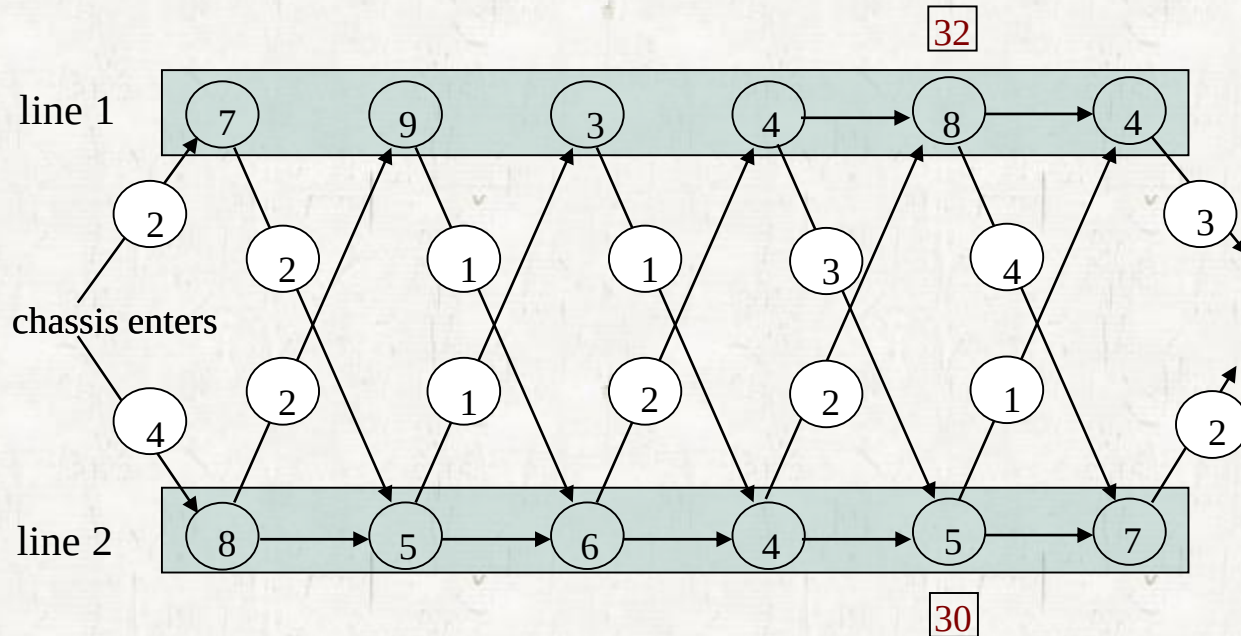
S	1	2	3	4	5	6
1				24		
2				25		

Assembly-line scheduling



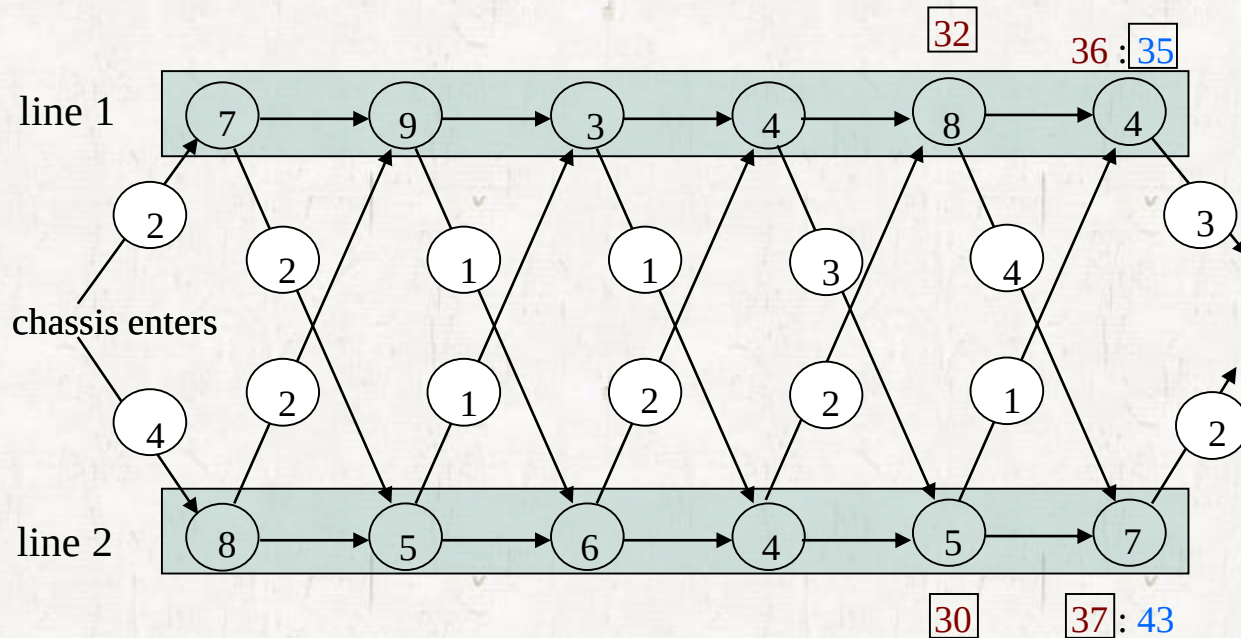
S	1	2	3	4	5	6
1				24	32	
2				25	30	

Assembly-line scheduling



S	1	2	3	4	5	6
1					32	
2					30	

Assembly-line scheduling



S	1	2	3	4	5	6
1					32	35
2					30	37

Assembly-line scheduling

- **Fastest time only**
 - $\Theta(1)$ space
 - Table s
 - $2n$ elements $\rightarrow$ 4 elements

Contents

Problems	Space	Time
Assembly-line scheduling	$\Theta(n)$	$\Theta(n)$
Rod cutting		
Longest common subsequence		
Matrix-chain multiplication		

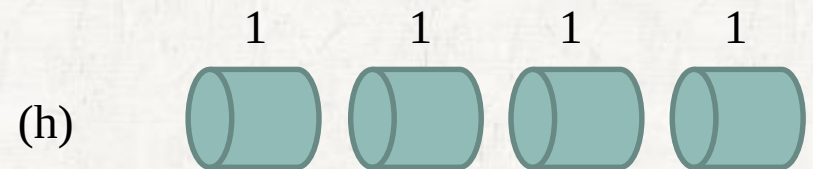
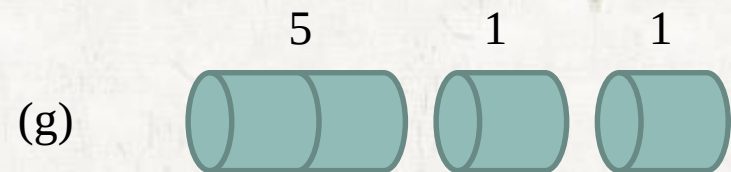
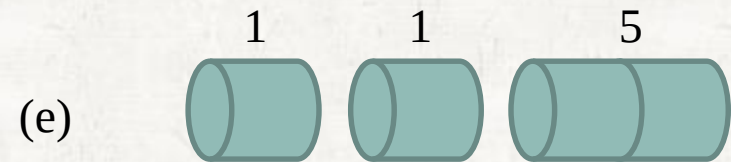
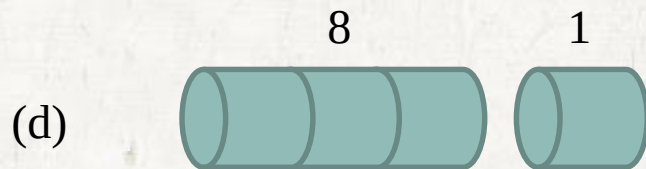
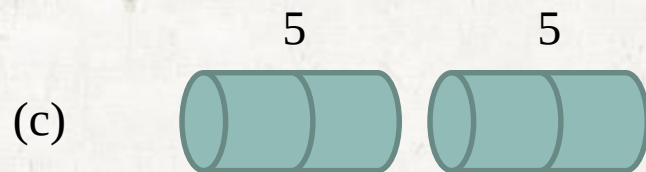
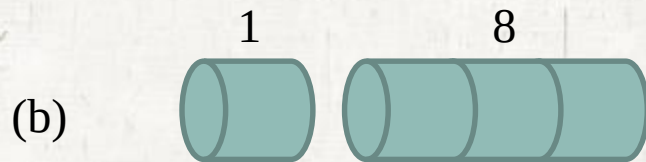
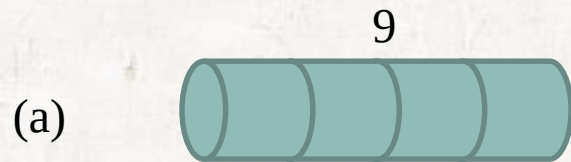
Rod cutting

- The **rod-cutting problem**: Given a rod of length n inches and a table of prices p_i for $i = 1, 2, \dots, n$, determine the maximum revenue r_n obtainable by cutting up the rod and selling the pieces.

length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30



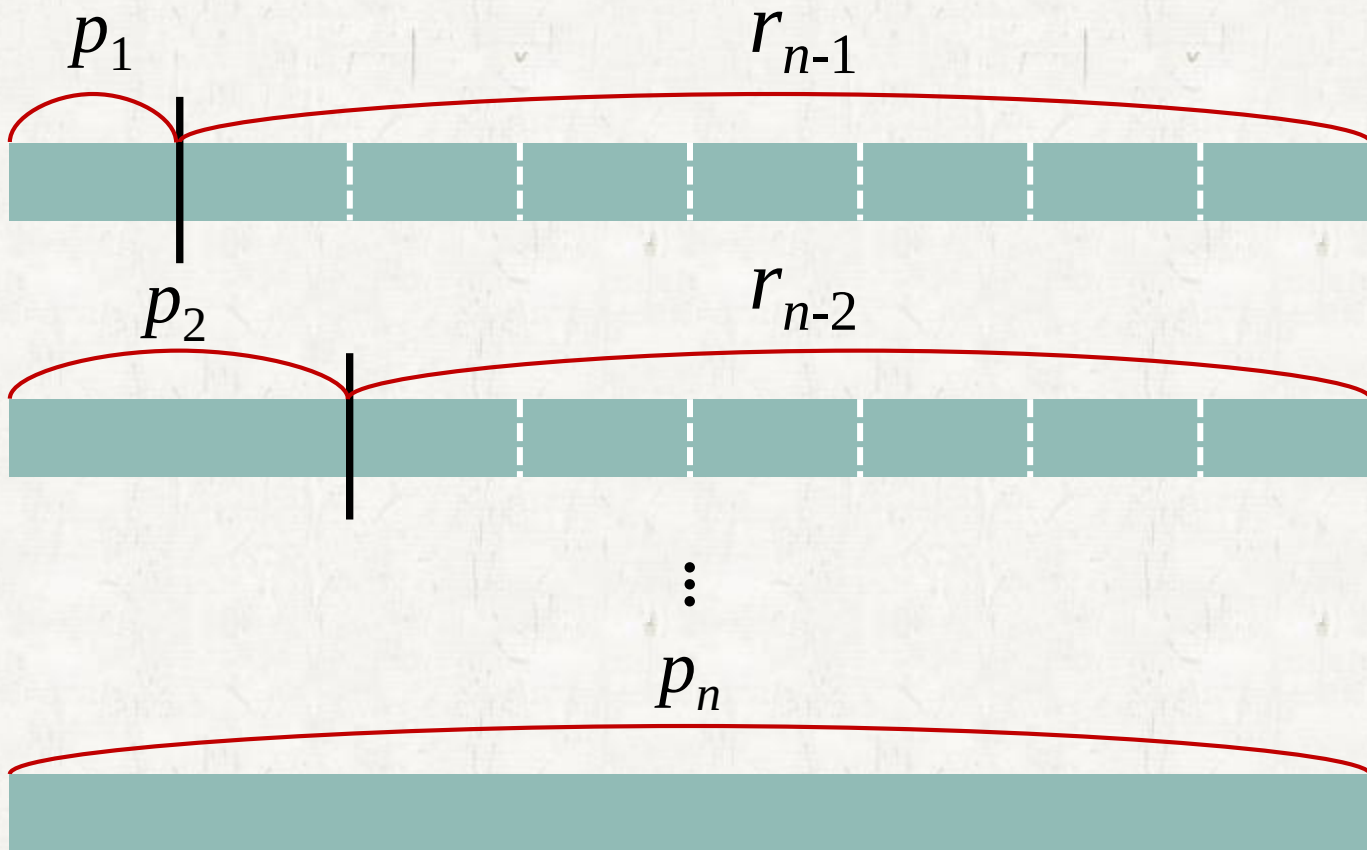
Rod cutting



length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30

Rod cutting

$$r_n = \max_{1 \leq i \leq n} (p_i + r_{n-i})$$

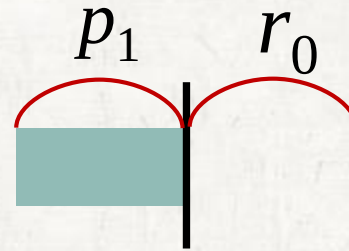


Rod cutting

i	0	1	2	3	4	5	6	7	8	9	10
$p[i]$	0	1	5	8	9	10	17	17	20	24	30
$r[i]$	0	1									

$$r[0] = 0$$

$$r[1] = p[1] + r[0] = 1 + 0 = 1$$



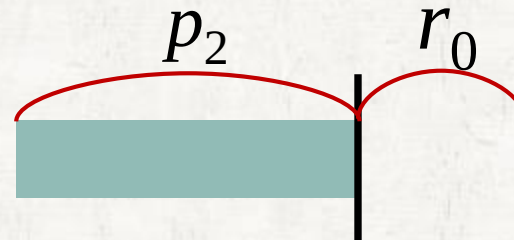
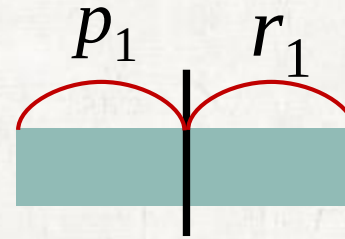
Rod cutting

i	0	1	2	3	4	5	6	7	8	9	10
$p[i]$	0	1	5	8	9	10	17	17	20	24	30
$r[i]$	0	1	5								

$$r[0] = 0$$

$$r[1] = p[1] + r[0] = 1 + 0 = 1$$

$$r[2] = \begin{cases} p[1] + r[1] = 1 + 1 = 2 \\ p[2] + r[0] = 5 + 0 = \mathbf{5} \end{cases}$$



Rod cutting

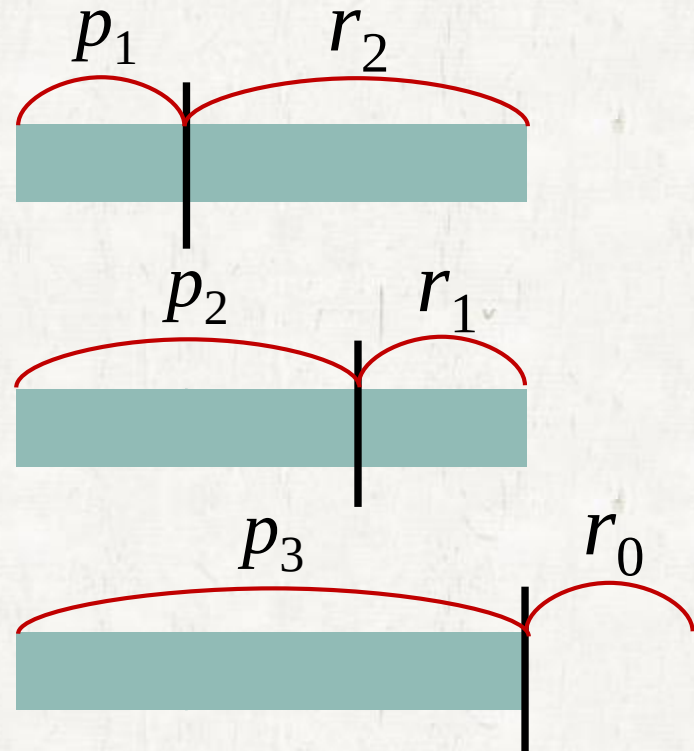
i	0	1	2	3	4	5	6	7	8	9	10
$p[i]$	0	1	5	8	9	10	17	17	20	24	30
$r[i]$	0	1	5	8							

$$r[0] = 0$$

$$r[1] = p[1] + r[0] = 1 + 0 = 1$$

$$r[2] = \begin{cases} p[1] + r[1] = 1 + 1 = 2 \\ p[2] + r[0] = 5 + 0 = 5 \end{cases}$$

$$r[3] = \begin{cases} p[1] + r[2] = 1 + 5 = 6 \\ p[2] + r[1] = 5 + 1 = 6 \\ p[3] + r[0] = 8 + 0 = 8 \end{cases}$$



Rod cutting

i	0	1	2	3	4	5	6	7	8	9	10
$p[i]$	0	1	5	8	9	10	17	17	20	24	30
$r[i]$	0	1	5	8	10	13	17	18	22	25	30

$$r[0] = 0$$

$$r[1] = p[1] + r[0] = 1 + 0 = \mathbf{1}$$

$$\vdots$$

$$r[2] = \begin{cases} p[1] + r[1] = 1 + 1 = 2 \\ p[2] + r[0] = 5 + 0 = \mathbf{5} \end{cases}$$

$$r[3] = \begin{cases} p[1] + r[2] = 1 + 5 = 6 \\ p[2] + r[1] = 5 + 1 = 6 \\ p[3] + r[0] = 8 + 0 = \mathbf{8} \end{cases}$$

$$r[10] = \begin{cases} p[1] + r[9] = 1 + 25 = 26 \\ p[2] + r[8] = 5 + 22 = 27 \\ \vdots \\ p[10] + r[0] = 30 + 0 = \mathbf{30} \end{cases}$$

Rod cutting

i	0	1	2	3	4	5	6	7	8	9	10
$p[i]$	0	1	5	8	9	10	17	17	20	24	30
$r[i]$	0	1	5	8	10	13	17	18	22	25	30
$s[i]$	0	1	2	3	2	2	6	1	2	3	10

Rod cutting

EXTENDED-BOTTOM-UP-CUT-ROD (p, n)

```
1  let  $r[0 .. n]$  and  $s[0 .. n]$  be new arrays
2   $r[0] = 0$ 
3  for  $j = 1$  to  $n$ 
4       $q = -\infty$ 
5      for  $i = 1$  to  $j$ 
6          if  $q < p[i] + r[j - i]$ 
7               $q = p[i] + r[j - i]$ 
8               $s[j] = i$ 
9       $r[j] = q$ 
10 return  $r$  and  $s$ 
```

Rod cutting

PRINT-CUT-ROD-SOLUTION(p, n)

```
1  ( $r, s$ ) = EXTENDED-BOTTOM-UP-CUT-ROD ( $p, n$ )  
2  while  $n > 0$   
3    print  $s[n]$   
4     $n = n - s[n]$ 
```

Rod cutting

i	0	1	2	3	4	5	6	7	8	9	10
$r[i]$	0	1	5	8	10	13	17	18	22	25	30
$s[i]$	0	1	2	3	2	2	6	1	2	3	10

Rod cutting

- Space consumption

- $\Theta(n)$

- Running time

- $\Theta(n^2)$

- $1 + 2 + 3 + 4 + \dots + n = n(n+1)/2$

Rod cutting

- Table r can be reduced like table s in assembly-line scheduling?

Contents

Problems	Space	Time
Assembly-line scheduling	$\Theta(n)$	$\Theta(n)$
Rod cutting	$\Theta(n)$	$\Theta(n^2)$
Longest common subsequence		
Matrix-chain multiplication		

Longest common subsequence

• Definition

- *Character*
- *String* (or *sequence*): A list of characters
 - ex> strings over $\{0,1\}$: Binary strings
 - ex> strings over $\{A,C,G,T\}$: DNA sequences

Longest common subsequence

- **Substring**
 - CBD is a substring of $ABCBDAB$
- **Subsequence**
 - $BCDB$ is a subsequence of $ABCBDAB$
- **Common subsequence**
 - BCA is a common subsequence of
 $X=ABCBDAB$ and $Y=BD CABA$

Longest common subsequence

- **Longest common subsequence (LCS)**

- $BCBA$ is a longest common subsequence of X and Y

$X = A \textcolor{blue}{B} \textcolor{blue}{C} \textcolor{blue}{B} D \textcolor{blue}{A} B$
 / | \ |
 $Y = \textcolor{blue}{B} D \textcolor{blue}{C} A \textcolor{blue}{B} A$

Longest common subsequence

• Brute force approach

- Enumerate all subsequences of X and check each subsequence if it is also a subsequence of Y and find the longest one.
- Infeasible!
 - The number of subsequences of X is 2^m .

Longest common subsequence

- **Dynamic programming**

- The *ith prefix* X_i of X is $X_i = x_1x_2 \dots x_i$

- If $X = ABCBDAB$

- $X_4 = ABCB$

- $X_0 =$

Longest common subsequence

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1} .

$X = A B C B D A B$
 / / / /
 $Y = B D C A B$

Longest common subsequence

2. If $x_m \neq y_n$, Z is an LCS of X_{m-1} and Y or an LCS of X and Y_{n-1} .

$X = A B C B D A \textcolor{blue}{B}$
 / \ /
 $Y = B D C A A$

$X = A B C B D A B$
 / | /
 $Y = B D C A \textcolor{blue}{A}$

Longest common subsequence

- $c[i][j]$: The length of an LCS of the sequences X_i and Y_j .

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

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		(B)	D	(C)	A	(B)	(A)
i	x_i		0	0	0	0	0	0	0
0	x_i		0	0	0	0	0	0	0
1	A		0	↑	↑	↑	↖	←	↖
2	(B)		0	↖	←	←	↑	↖	←
3	(C)		0	↑	↑	↖	←	↑	↑
4	(B)		0	↖	↑	↑	↑	↖	←
5	D		0	↑	↖	↑	↑	↑	↑
6	(A)		0	↑	↑	↑	↖	↑	↖
7	B		0	↖	↑	↑	↑	↖	↑

Longest common subsequence

LCS-LENGTH (X, Y)

```
1   $m = X.length$ 
2   $n = Y.length$ 
3  let  $b[1 .. m][1 .. n]$  and  $c[0 .. m][0 .. n]$  be new tables
4  for  $i = 1$  to  $m$ 
5       $c[i][0] = 0$ 
6  for  $j = 0$  to  $n$ 
7       $c[0][j] = 0$ 
8  for  $i = 1$  to  $m$ 
9      for  $j = 1$  to  $n$ 
10         if  $x_i == y_j$ 
11              $c[i][j] = c[i - 1][j - 1] + 1$ 
12              $b[i][j] = "\nwarrow"$ 
13         elseif  $c[i - 1][j] \geq c[i][j - 1]$ 
14              $c[i][j] = c[i - 1][j]$ 
15              $b[i][j] = "\uparrow"$ 
16         else  $c[i][j] = c[i][j - 1]$ 
17              $b[i][j] = "\leftarrow"$ 
18  return  $c$  and  $b$ 
```

Longest common subsequence

```
PRINT-LCS ( $b, X, i, j$ )
1  if  $i == 0$  or  $j == 0$ 
2    return
3  if  $b[i][j] == "\nwarrow"$ 
4    PRINT-LCS ( $b, X, i - 1, j - 1$ )
5    print  $x_i$ 
6  elseif  $b[i][j] == "\uparrow"$ 
7    PRINT-LCS ( $b, X, i - 1, j$ )
8  else PRINT-LCS ( $b, X, i, j - 1$ )
```


Longest common subsequence

Multiple LCSs

		j	0	1	2	3	4	5	6
		y_j		(B)	D	(C)	A	(B)	(A)
i	x_i								
0			0	0	0	0	0	0	0
1	A		0	← ↑ 0	← ↑ 0	← ↑ 0	↖ 1	← 1	↖ 1
2	(B)		0	↖ 1	← 1	← 1	← ↑ 1	↖ 2	← 2
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

Longest common subsequence

Multiple LCSs

		j	0	1	2	3	4	5	6
		y_j		(B)	D	(C)	A	(B)	(A)
i	x_i								
0			0	0	0	0	0	0	0
1	A		0	← ↑	← ↑	← ↑	↖	←	↖
2	(B)		0	↖	←	←	← ↑	↖	←
3	(C)		0	↑	← ↑	↖	←	← ↑	← ↑
4	(B)		0	↖	← ↑	↑	← ↑	↖	←
5	D		0	↑	↖	← ↑	← ↑	↑	← ↑
6	(A)		0	↑	↑	← ↑	↖	← ↑	↖
7	B		0	↖	↑	← ↑	↑	↖	← ↑

Longest common subsequence

Multiple LCSs

		j	0	1	2	3	4	5	6
		y_j		(B)	D	(C)	(A)	(B)	A
i	x_i								
0	x_i		0	0	0	0	0	0	0
1	A		0	0 ↗	0 ↗	0 ↗	1 ↖	1 ↖	1 ↖
2	(B)		0	1 ↖	1 ↖	1 ↖	1 ↗	2 ↖	2 ↖
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 ↖

Longest common subsequence

Multiple LCSs

		j	0	1	2	3	4	5	6
		y_j		(B)	(D)	C	(A)	(B)	A
i	x_i								
0			0	0	0	0	0	0	0
1	A		0	← ↑	← ↑	← ↑	↖	←	↖
2	B		0	↖	←	←	← ↑	↖	←
3	C		0	↑	← ↑	↖	←	← ↑	← ↑
4	(B)		0	↖	← ↑	↑	← ↑	↖	←
5	(D)		0	↑	↖	← ↑	← ↑	↑	← ↑
6	(A)		0	↑	↑	← ↑	↖	← ↑	↖
7	(B)		0	↖	↑	← ↑	↑	↖	← ↑

Longest common subsequence

- Space: $\Theta(mn)$
- Time: $\Theta(mn)$
- Space reduction: $\Theta(\min(m,n))$ (LCS length only)

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i		0	0	0	0	0	0	0
0	x_i		0	0	0	0	0	0	0
1	A								
2	B								
3	C								
4	B								
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i		0	0	0	0	0	0	0
1	A		0	0	0	0	1	1	1
2	B								
3	C								
4	B								
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A		0	0	0	0	1	1	1
2	B								
3	C								
4	B								
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A	0	0	0	0	1	1	1	
2	B	0	1	1	1	1	2	2	
3	C								
4	B								
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B	0	1	1	1	1	2	2	
3	C								
4	B								
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B	0	1	1	1	1	2	2	
3	C	0	1	1	2	2	2	2	
4	B								
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C	0	1	1	2	2	2	2	
4	B								
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C	0	1	1	2	2	2	2	
4	B	0	1	1	2	2	3	3	
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C								
4	B		0	1	1	2	2	3	3
5	D								
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C								
4	B	0	1	1	2	2	3	3	
5	D	0	1	2	2	2	3	3	
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C								
4	B								
5	D		0	1	2	2	2	3	3
6	A								
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C								
4	B								
5	D	0	1	2	2	2	3	3	
6	A	0	1	2	2	3	3	4	
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C								
4	B								
5	D								
6	A		0	1	2	2	3	3	4
7	B								

Longest common subsequence

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0	x_i								
1	A								
2	B								
3	C								
4	B								
5	D								
6	A		0	1	2	2	3	3	4
7	B		0	1	2	2	3	4	4

Longest common subsequence

		j	0	1	2	3	4	5	6	...	100,000
			y_j	B	D	A	C	B	A	...	A
i	x_i										
0	x_i									...	
1	A									...	
2	B									...	
3	C									...	
4	B									...	
5	A									...	

Longest common subsequence

		j	0	1	2	3	4	5
		y_j		A	B	C	B	A
i	x_i							
0	x_i							
1	B							
2	D							
3	A							
4	C							
5	B							
6	A							
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
100,000	A							

Longest common subsequence

- Space reduction: $\Theta(\min(m,n))$ (LCS length only)

Contents

Problems	Space	Time
Assembly-line scheduling	$\Theta(n)$	$\Theta(n)$
Rod cutting	$\Theta(n)$	$\Theta(n^2)$
Longest common subsequence	$\Theta(mn)$ $\Theta(n^2)$	$\Theta(mn)$ $\Theta(n^2)$
Matrix-chain multiplication		

Matrix-chain multiplication

- **Multiplying two matrices A and B**

- A ($p \times q$) and B ($r \times s$) can be multiplied only if $q = r$.

$$\begin{matrix} \left[\begin{array}{ccc} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{array} \right] & \times & \left[\begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \\ \bullet & \bullet \end{array} \right] & = & \left[\begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} \right] \\ (A \ 2 \times 3) & & (B \ 3 \times 2) & & (C \ 2 \times 2) \end{matrix}$$

- If A is a $p \times q$ matrix and B is a $q \times r$ matrix, the resulting matrix is a $p \times r$ matrix.
- **The number of scalar multiplications is pqr .**
- pr elements are computed and each element needs q scalar multiplications.

Matrix-chain multiplication

- The order of multiplications

$$(A_1 \cdot A_2) \cdot A_3 = A_1 \cdot (A_2 \cdot A_3)$$

- It does not change *the value of computation*.
- However, it changes *the amount of computation*.

Matrix-chain multiplication

- Computing $A_1 A_2 A_3$ where $A_1: 10 \times 100$ $A_2: 100 \times 5$ $A_3: 5 \times 50$
 - $(A_1 A_2) A_3$
 - $(A_1 A_2) = 10 * 100 * 5 = 5000$, $(10 \times 5) A_3 = 10 * 5 * 50 = 2500$
 $\Rightarrow 5000 + 2500 = \mathbf{7,500}$
 - $A_1 (A_2 A_3)$
 - $(A_2 A_3) = 100 * 5 * 50 = 25000$, $A_1(100 \times 50) = 10 * 100 * 50 = 50000$
 $\Rightarrow 25000 + 50000 = \mathbf{75,000}$
- Computing $(A_1 A_2) A_3$ is **10** times faster.

Matrix-chain multiplication

Matrix-chain multiplication problem

- Given a chain $A_1, A_2, \dots, A_n$ of n matrices, where matrix A_i has dimension $p_{i-1} \times p_i$, find the order of matrix multiplications minimizing the scalar multiplications to compute the product.
- That is, to fully parenthesize the product of matrices minimizing scalar
- $A_1: p_0 \times p_1, A_2: p_1 \times p_2, A_3: p_2 \times p_3 \dots$

Matrix-chain multiplication

- The product $A_1 A_2 A_3 A_4$ can be fully parenthesized in five distinct ways.

$$A_1(A_2(A_3 A_4)), \quad A_1((A_2 A_3) A_4),$$

$$(A_1 A_2)(A_3 A_4),$$

$$(A_1(A_2 A_3))A_4, \quad ((A_1 A_2) A_3) A_4.$$

Matrix-chain multiplication

- **The Brute-force approach is inefficient.**

- The number of parenthesizations of a product of n matrices, denoted by $P(n)$, is as follows.

$$P(n) = \begin{cases} 1 & \text{if } n=1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2 \end{cases}$$

- The number of enumerated parenthesizations is $\Omega(4^n/n^{3/2})$.

Matrix-chain multiplication

- Dynamic programming

- Optimal substructure

- $m[i][j]$: The minimum number of scalar multiplications for computing $A_i A_{i+1} \dots A_j$.

$$m[i][j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k < j} \{m[i][k] + m[k+1][j] + p_{i-1} p_k p_j\} & \text{if } i < j \end{cases}$$

- matrix $A_i : p_{i-1} \times p_i$
- computing $A_{i..k} A_{k+1..j}$ takes $p_{i-1} p_k p_j$ scalar multiplications.
- $s[i][j]$ stores the optimal k for tracing the optimal solution.

Matrix-chain multiplication

$i \backslash j$	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

m

$i \backslash j$	2	3	4	5	6
1	1	1	3	3	3
2		2	3	3	3
3			3	3	3
4				4	5
5					5

s

$$m[2][5] = \min \begin{cases} m[2][2] + m[3][5] + p[1]p[2]p[5] = 0 + 2500 + 35 \cdot 15 \cdot 20 = 13000, \\ m[2][3] + m[4][5] + p[1]p[3]p[5] = 2625 + 1000 + 35 \cdot 5 \cdot 20 = 7125, \\ m[2][4] + m[5][5] + p[1]p[4]p[5] = 4375 + 0 + 35 \cdot 10 \cdot 20 = 11375 \end{cases}$$

$$i=2, j=5, i \leq k < j$$

$$(A_2)(A_3 A_4 A_5)$$

$$(A_2 A_3)(A_4 A_5)$$

$$(A_2 A_3 A_4)(A_5)$$

	0	30
1		35
2		15
3		5
4		10
5		20
6		25

p

Matrix-chain multiplication

MATRIX-CHAIN-ORDER (p)

```
1   $n = p.length - 1$ 
2  let  $m[1 .. n][1 .. n]$  and  $s[1 .. n - 1][2 .. n]$  be new tables
3  for  $i = 1$  to  $n$ 
4       $m[i][i] = 0$ 
5  for  $l = 2$  to  $n$                 //  $l$  is the chain length
6      for  $i = 1$  to  $n - l + 1$ 
7           $j = i + l - 1$ 
8           $m[i][j] = \infty$ 
9          for  $k = i$  to  $j - 1$ 
10              $q = m[i][k] + m[k + 1][j] + p[i-1]p[k]p[j]$ 
11             if  $q < m[i][j]$ 
12                  $m[i][j] = q$ 
13                  $s[i][j] = k$ 
14  return  $m$  and  $s$ 
```

Matrix-chain multiplication

PRINT-OPTIMAL-PARENS (s, i, j)

```
1  if  $i == j$ 
2    print " $A_i$ "
3  else print "("
4    PRINT-OPTIMAL-PARENS( $s, i, s[i][j]$ )
5    PRINT-OPTIMAL-PARENS( $s, s[i][j] + 1, j$ )
6    print ")"
```

Matrix-chain multiplication

- **Space consumption**

- $\Theta(n^2)$ space to store m and s tables.

- **Running time**

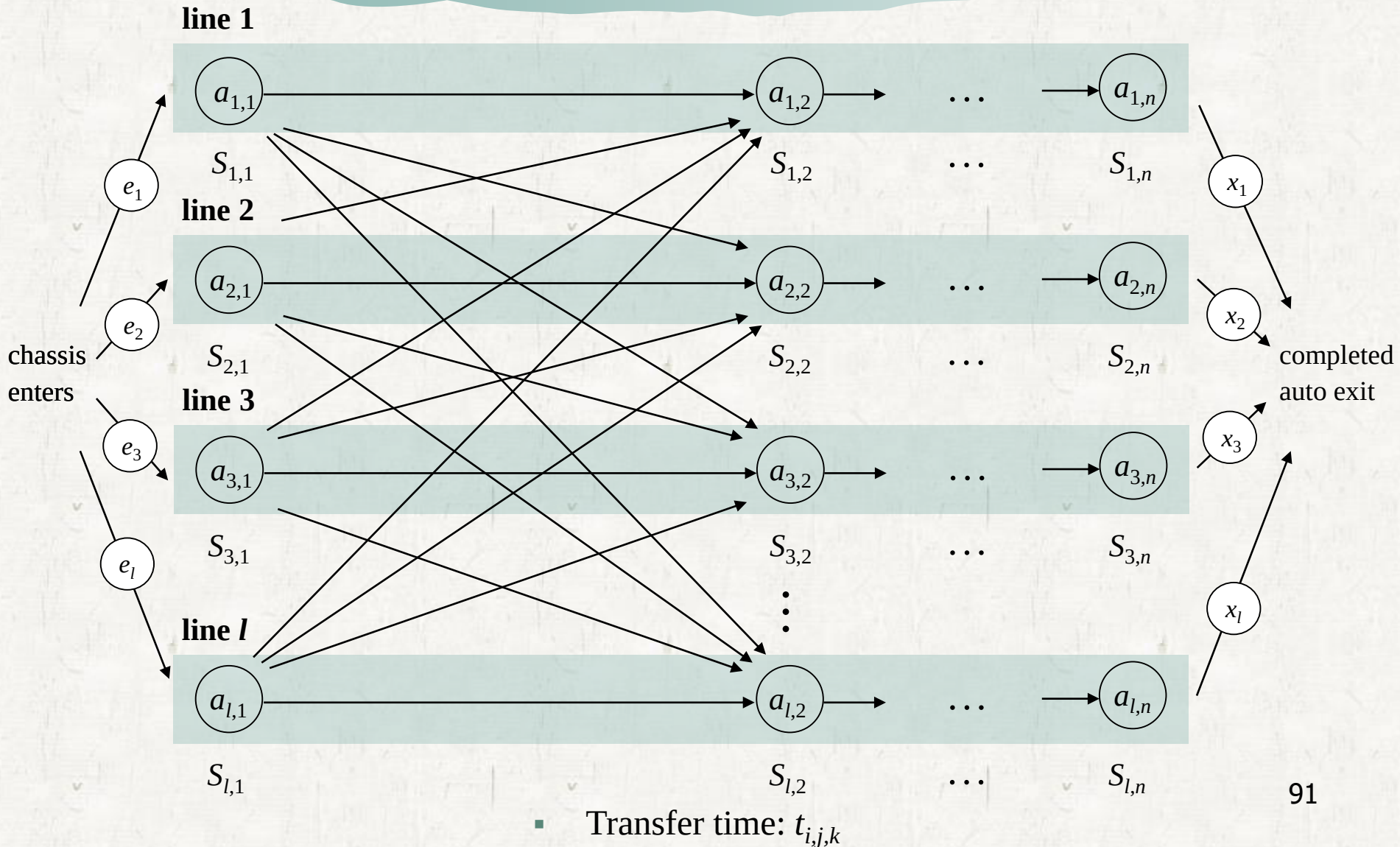
- $\Theta(n^3)$

$$\begin{aligned} & 1 \cdot n + 1 \cdot n-1 + 2 \cdot n-2 + \cdots + (n-1) \cdot 1 \\ &= 1 \cdot n + \sum_{k=1}^{n-1} k(n-k) \\ &= 1 \cdot n + \sum_{k=1}^{n-1} kn - \sum_{k=1}^{n-1} k^2 \\ &= n + n^2(n-1)/2 - n(n-1)(2n-1)/6 \\ &= (n^3 + 5n)/6 \\ &= \Theta(n^3) \end{aligned}$$

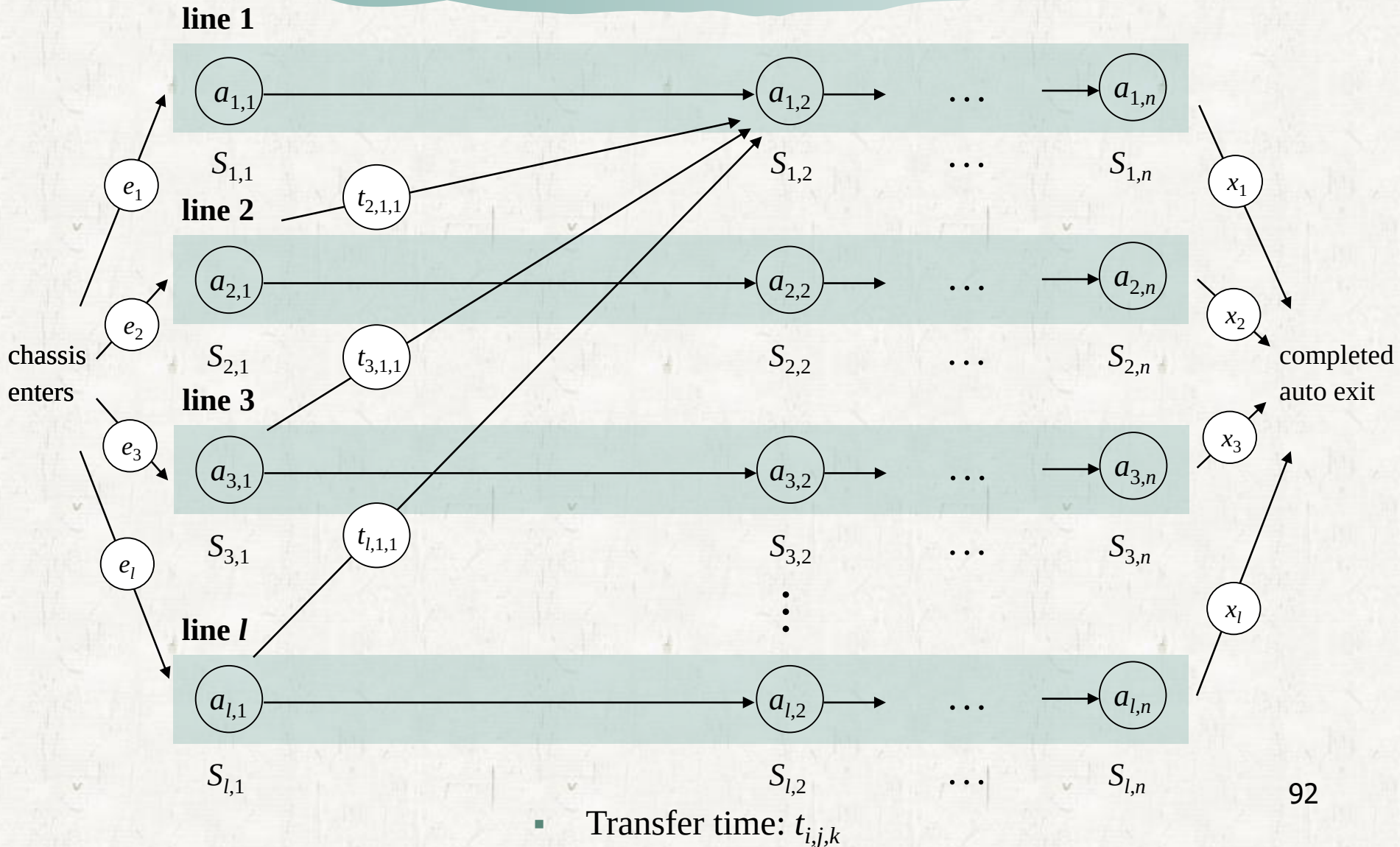
Conclusion

Problems	Space	Time
Assembly-line scheduling	$\Theta(n)$	$\Theta(n)$
Rod cutting	$\Theta(n)$	$\Theta(n^2)$
Longest common subsequence	$\Theta(n^2)$	$\Theta(n^2)$
Matrix-chain multiplication	$\Theta(n^2)$	$\Theta(n^3)$

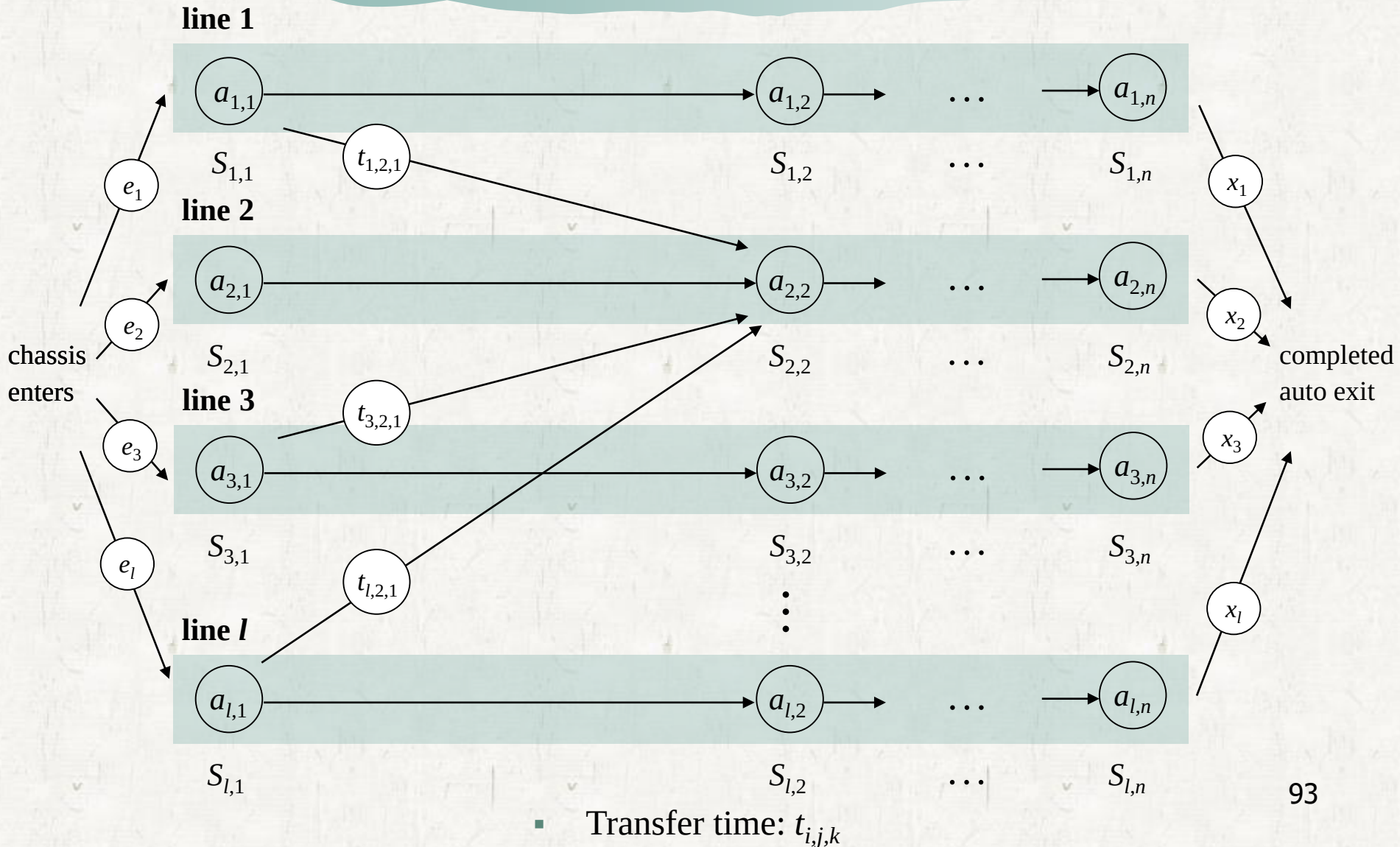
Assembly-line scheduling (L lines)



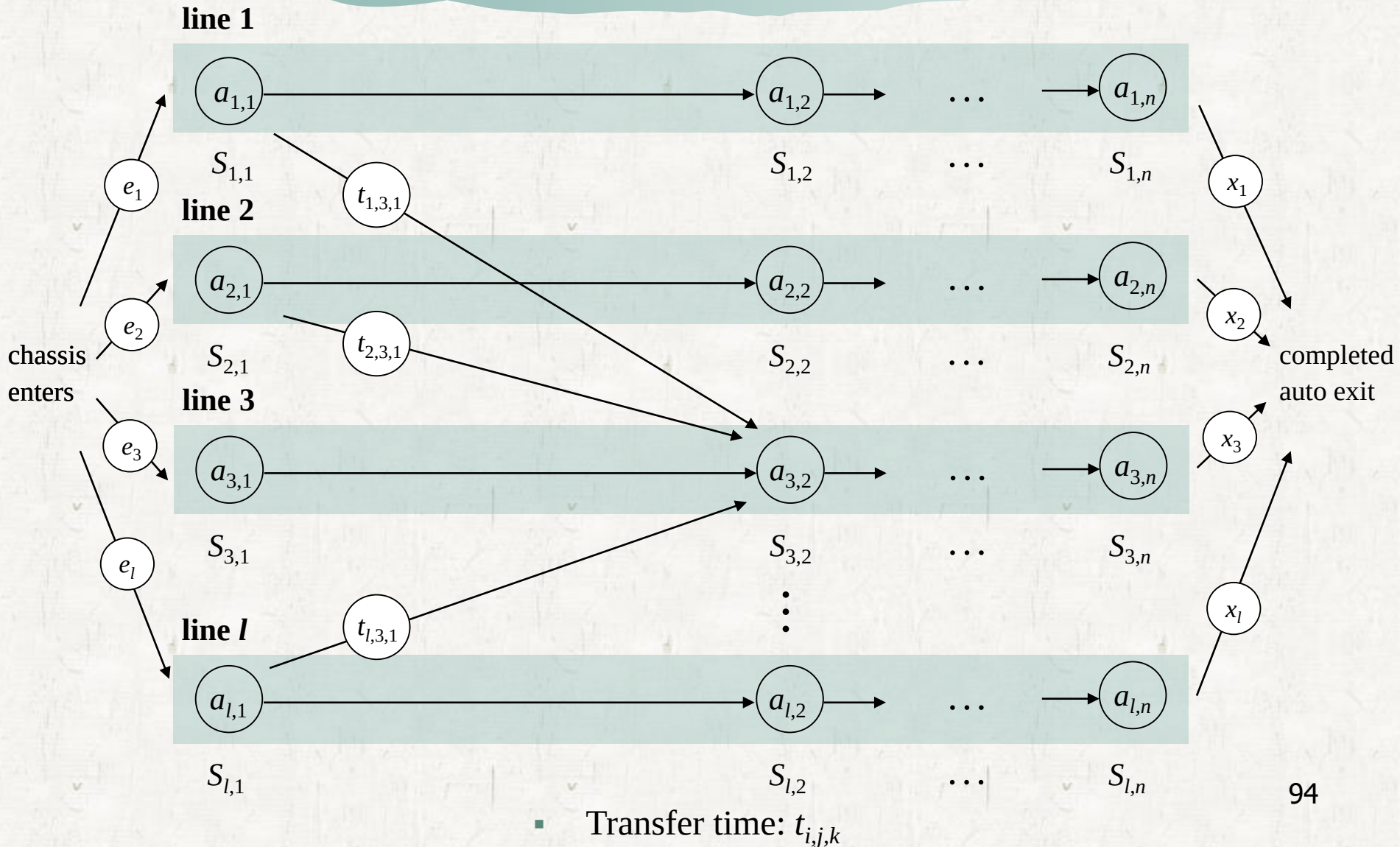
Assembly-line scheduling (L lines)



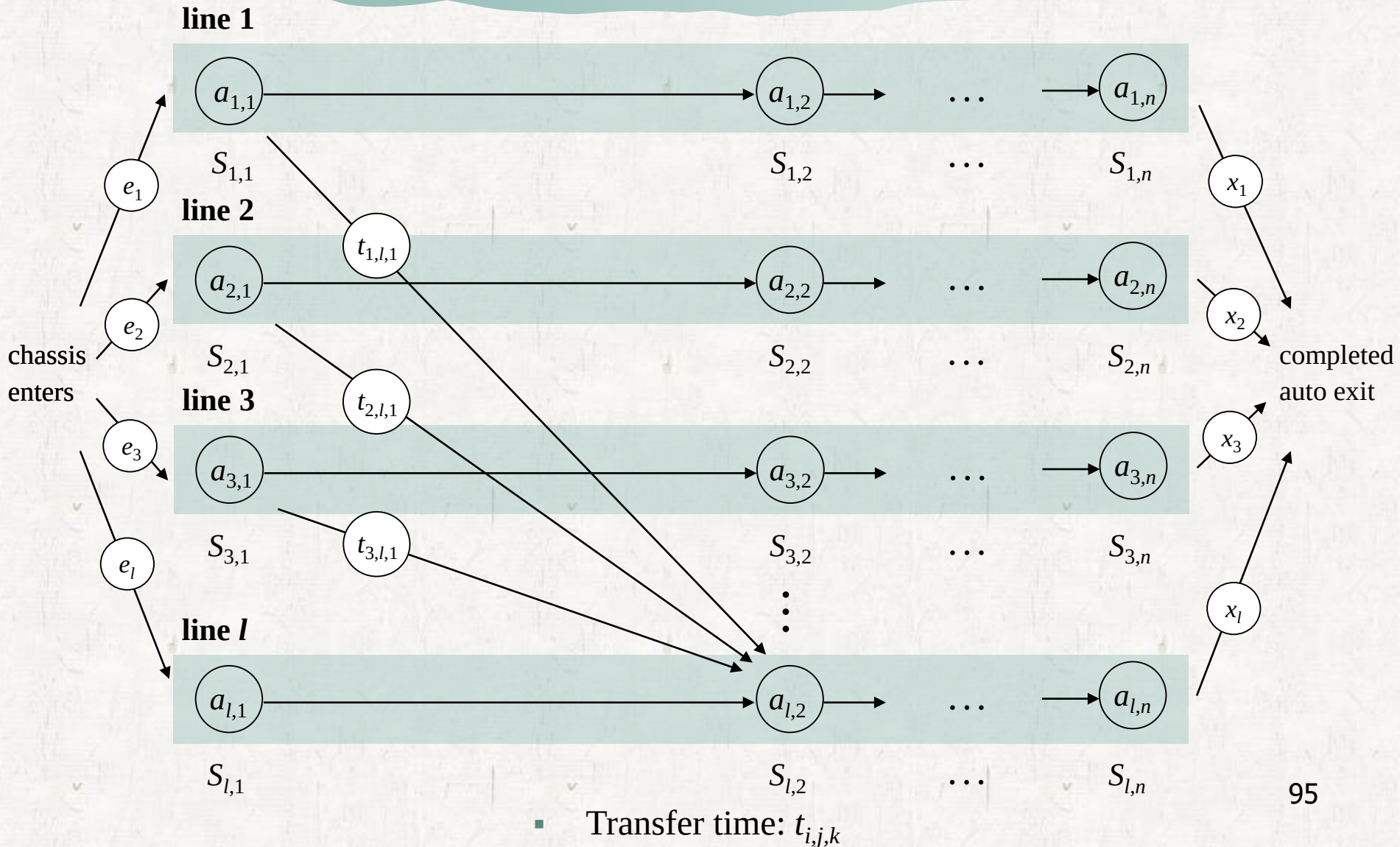
Assembly-line scheduling (L lines)



Assembly-line scheduling (L lines)



Assembly-line scheduling (L lines)



Assembly-line scheduling (L lines, same transfer costs)

