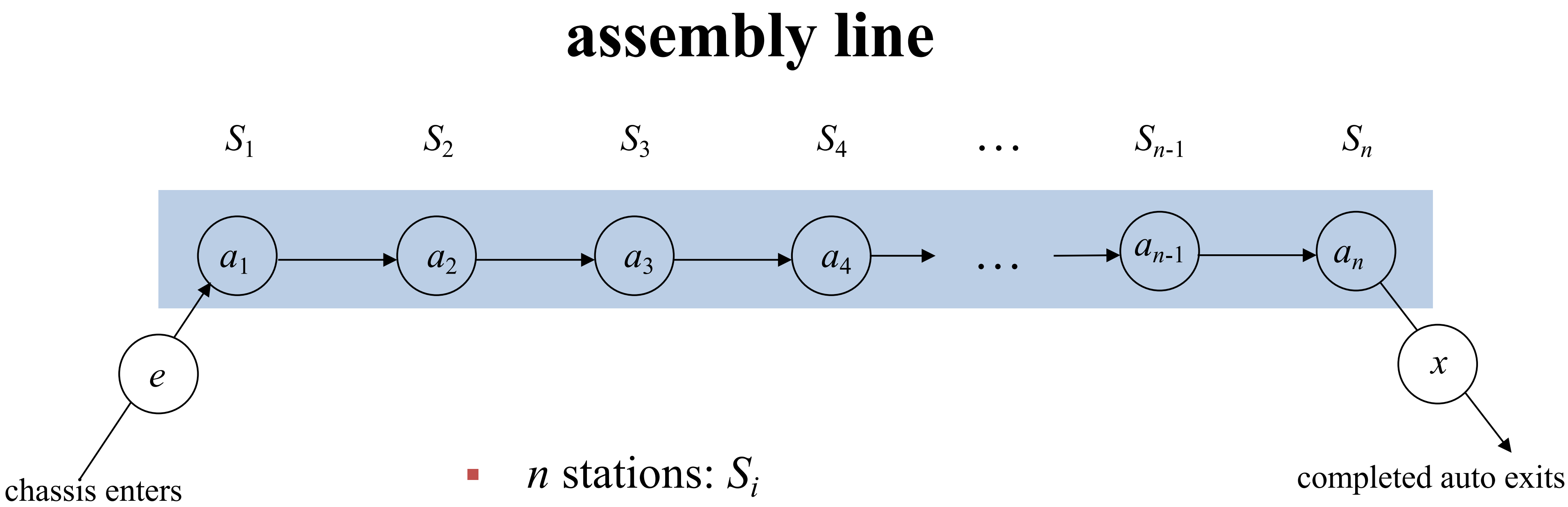
Dynamic Programming

Slides from Heejin Park

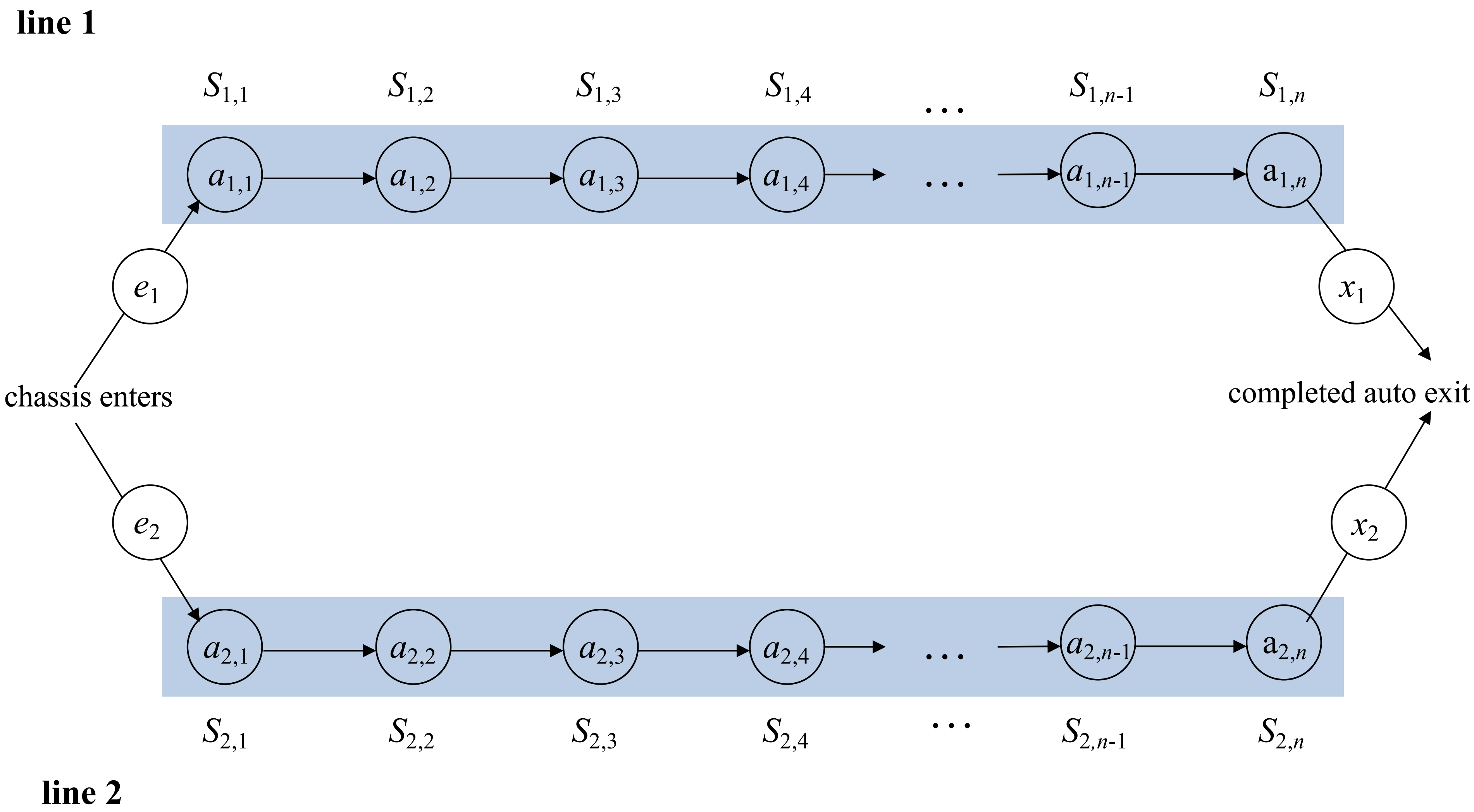
Contents

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

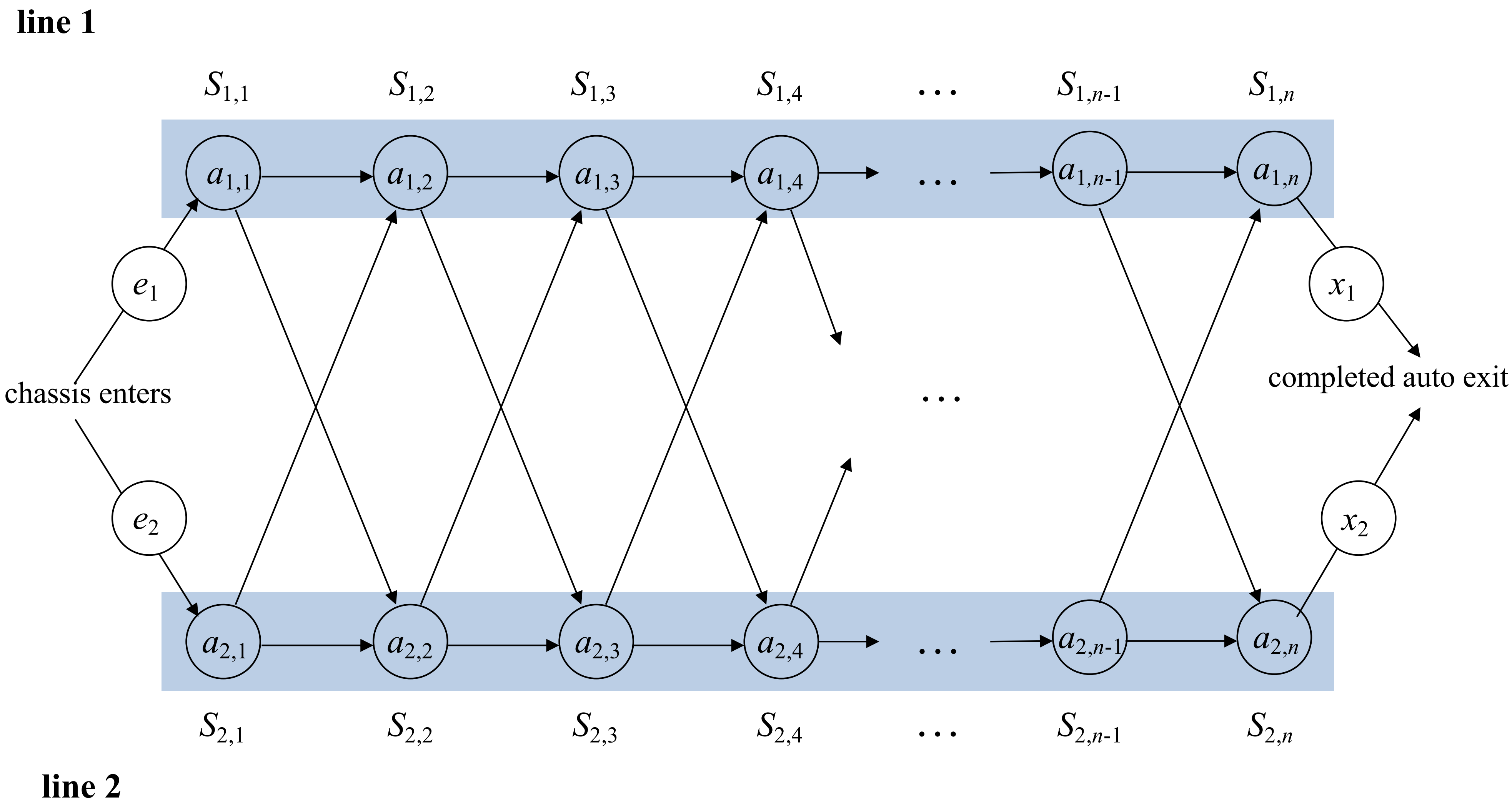


- n stations: S_i
- Assembly time in the ith station: a_i
- Entry time : e
- Exit time : x

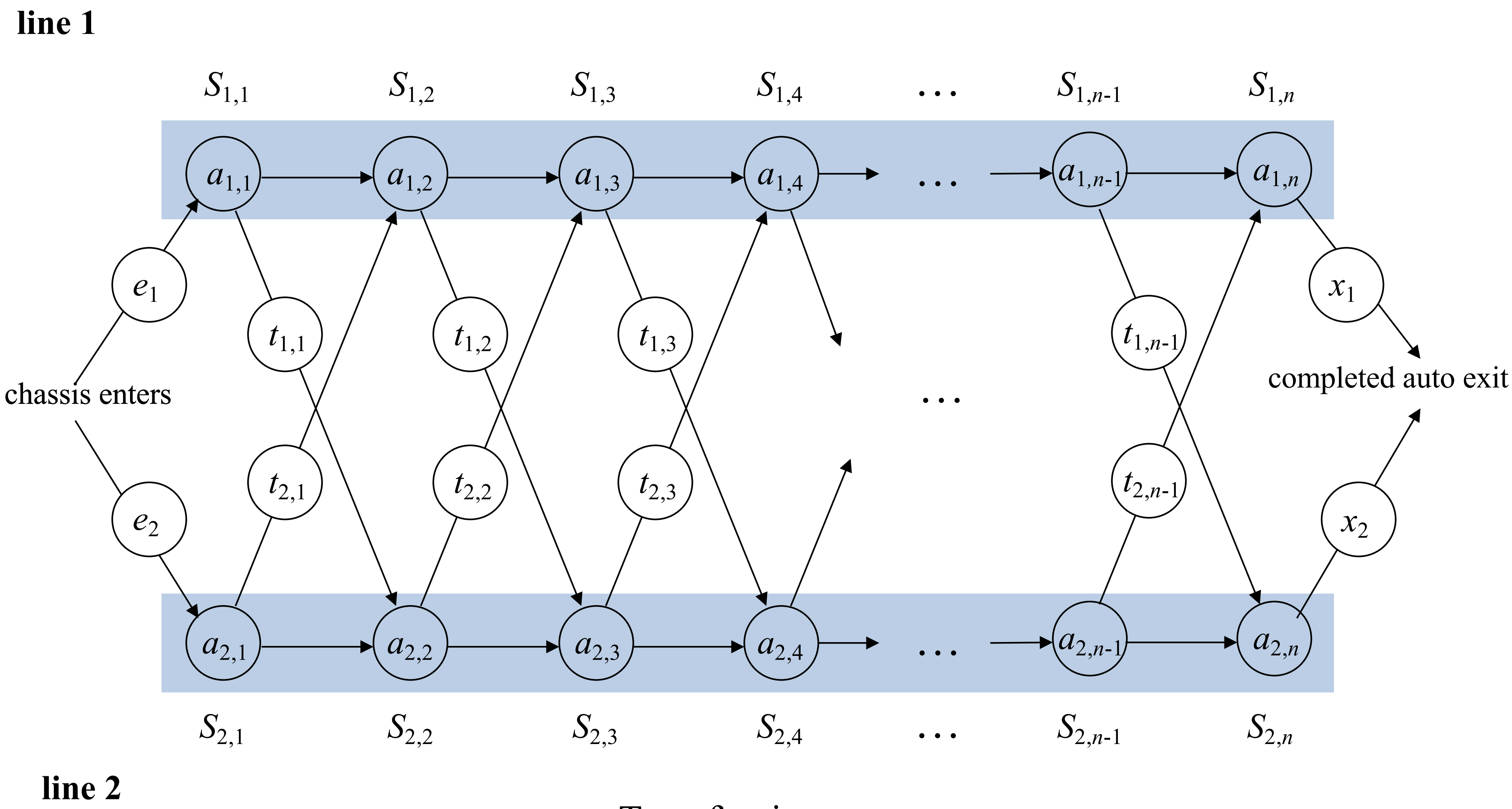
- **The problem:** Determine the fastest assembly time


line 1
$S_{1,1}$
$S_{1,2}$
$S_{1,3}$
$S_{1,4}$
…
$S_{1,n-1}$
$S_{1,n}$
$a_{1,1}$
$a_{1,2}$
$a_{1,3}$
$a_{1,4}$
…
$a_{1,n-1}$
$a_{1,n}$
e_1
x_1
chassis enters
completed auto exit
e_2
x_2
$a_{2,1}$
$a_{2,2}$
$a_{2,3}$
$a_{2,4}$
…
$a_{2,n-1}$
$a_{2,n}$
$S_{2,1}$
$S_{2,2}$
$S_{2,3}$
$S_{2,4}$
…
$S_{2,n-1}$
$S_{2,n}$
line 2

Assembly-line scheduling



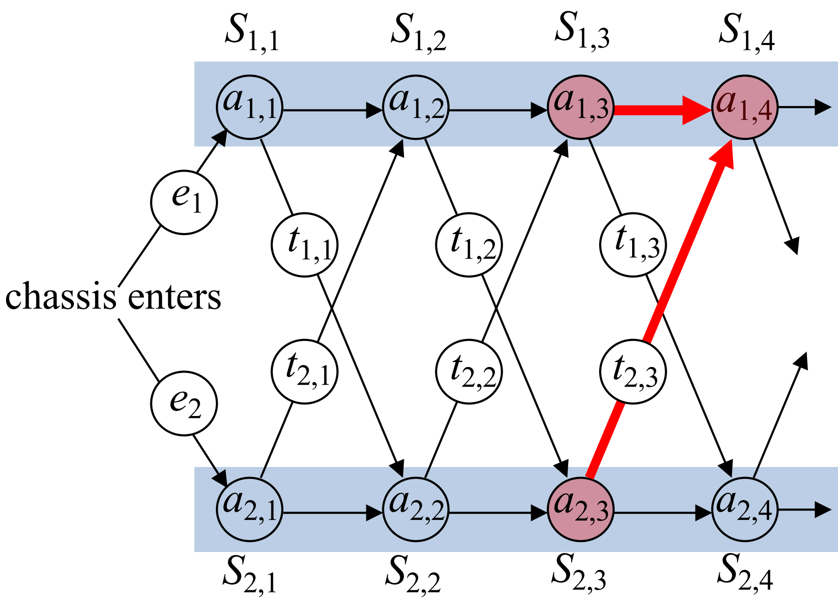
Assembly-line scheduling



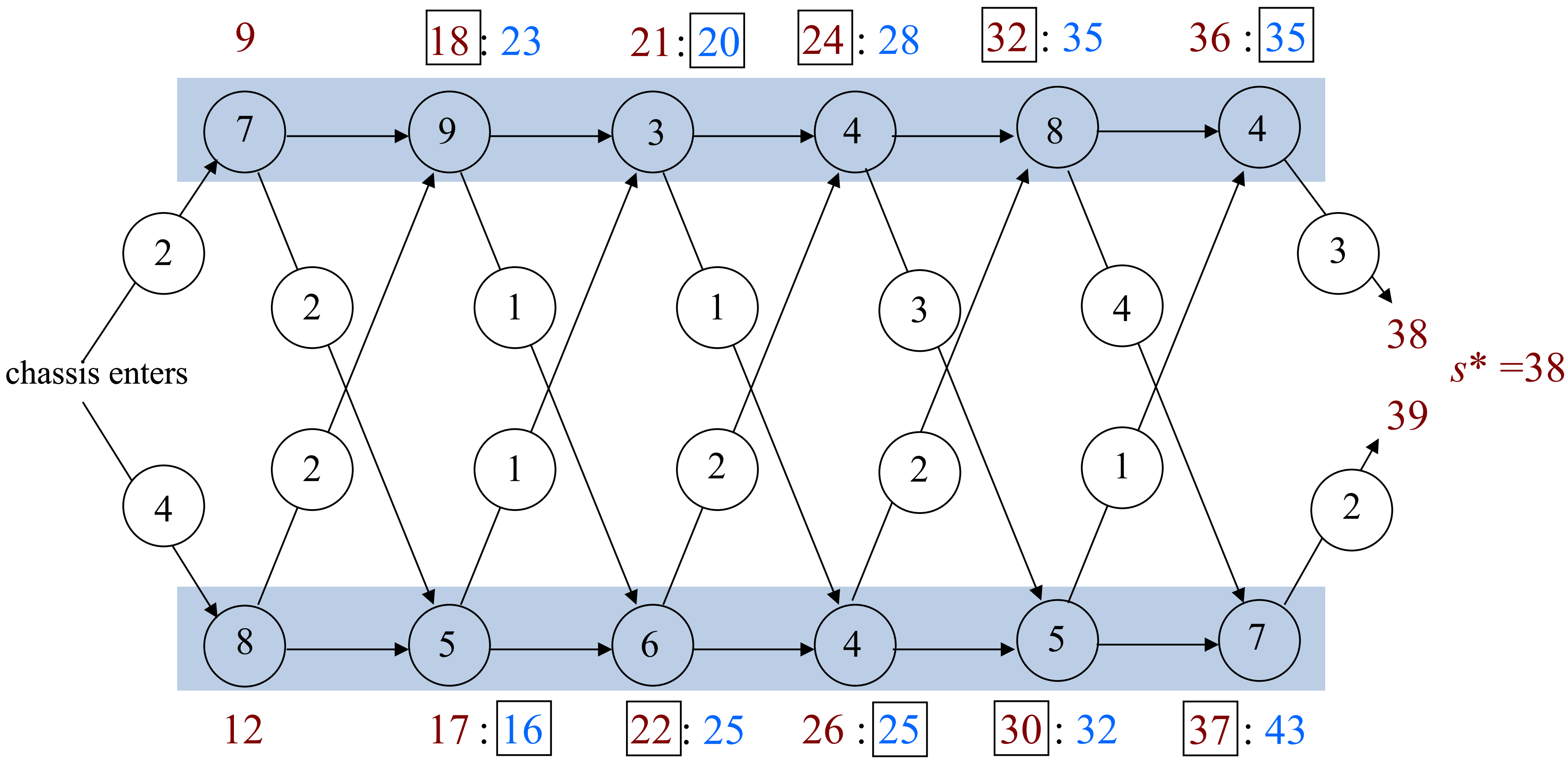
- Transfer time: $t_{i,j}$

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

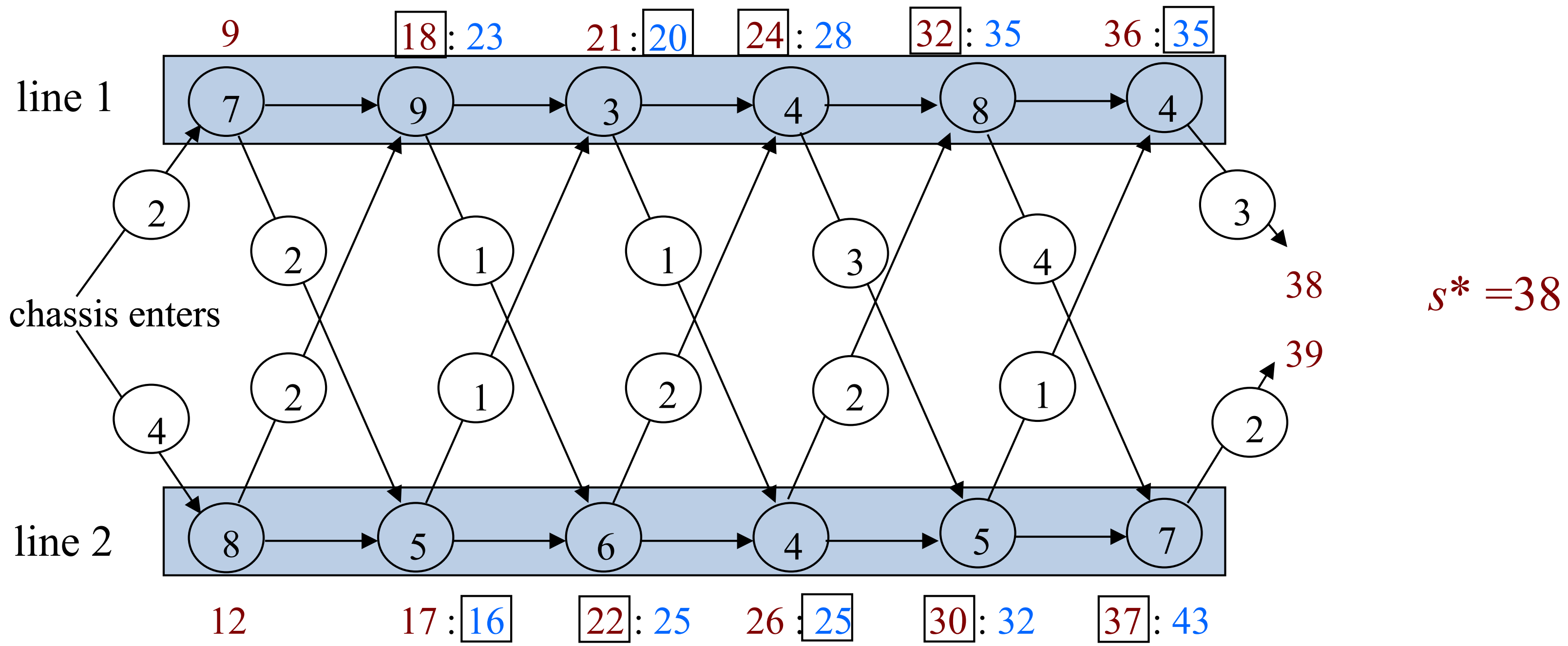
The fastest way to $S_{i,j}$ goes through $S_{1,j\text{-}1}$ or $S_{2,j\text{-}1}$.



- *Dynamic programming*



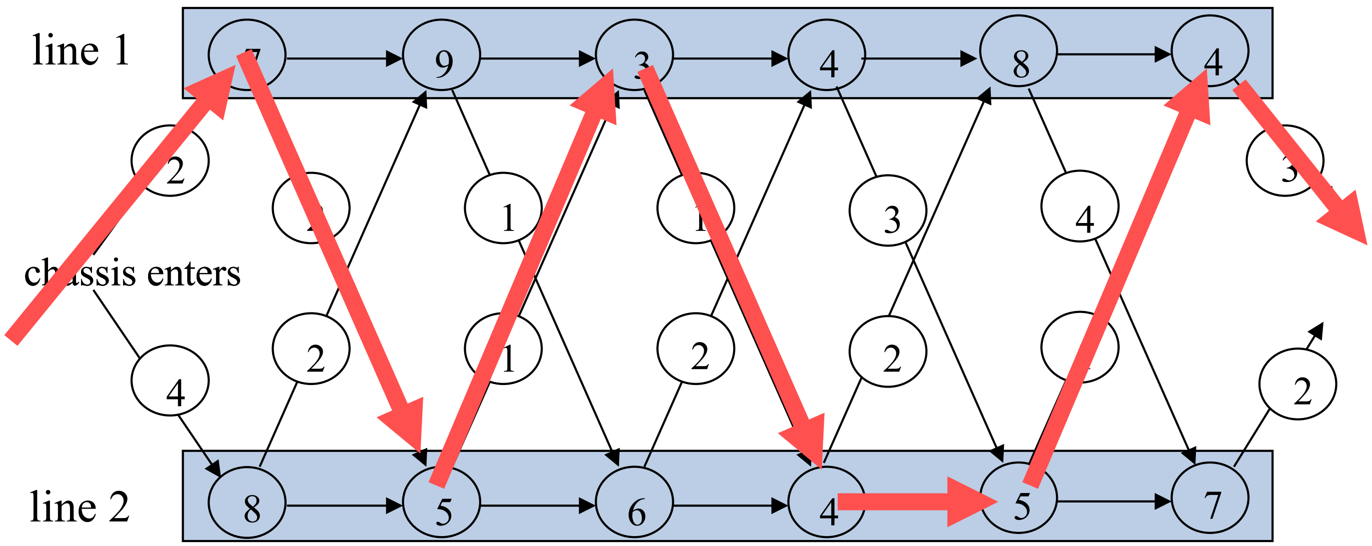
Assembly-line scheduling



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

$s^* = 38$

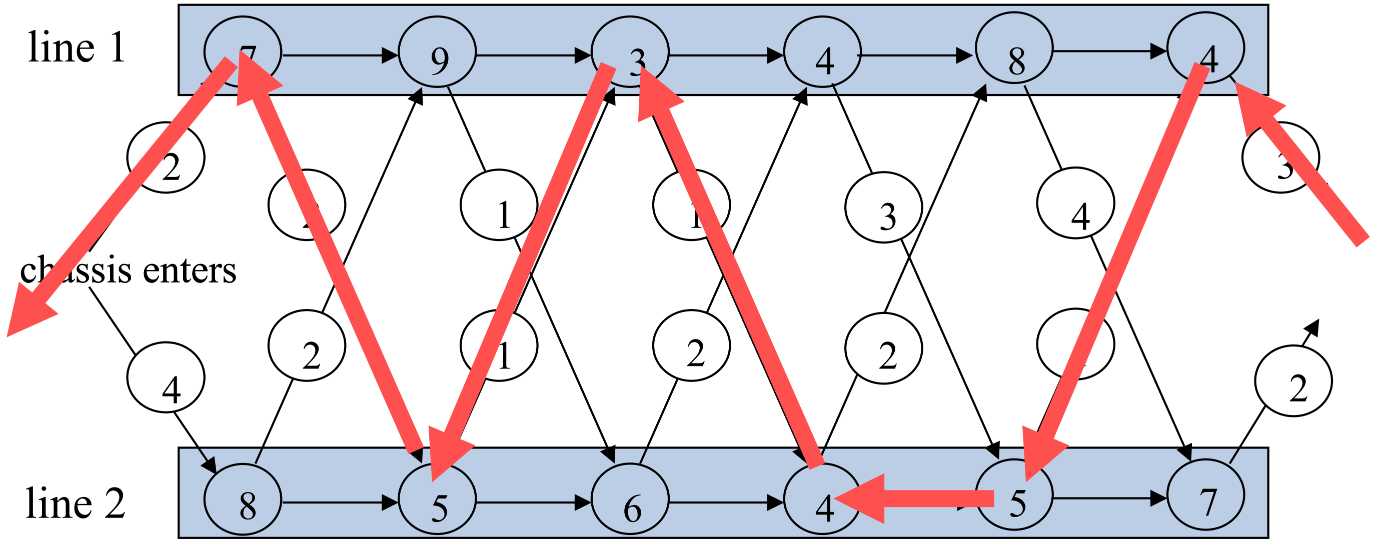
- ***Fastest time***



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***

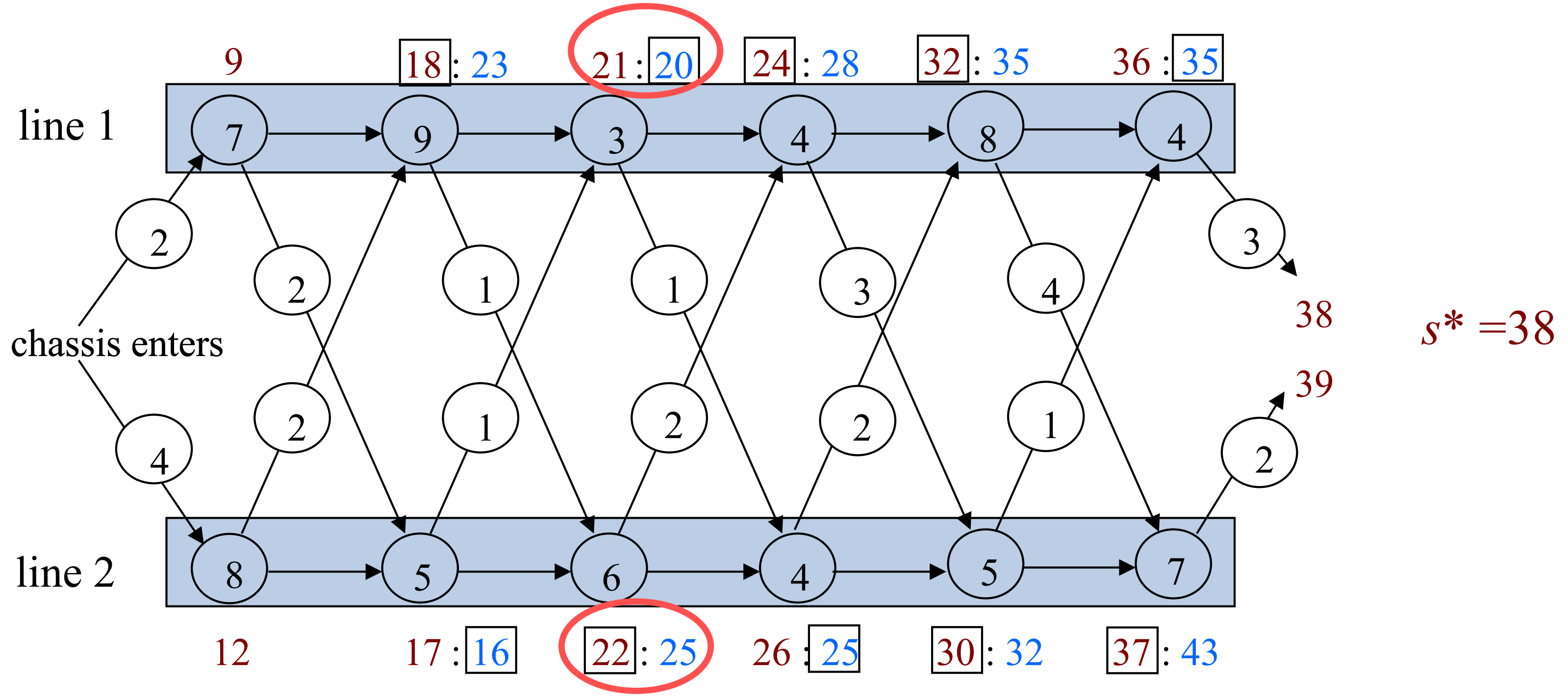


S	1	2	3	4	5	6
1	9	18	20	24	32	35
2	12	16	22	25	30	37

$s^* = 38$

- ***Fastest way***

Assembly-line scheduling



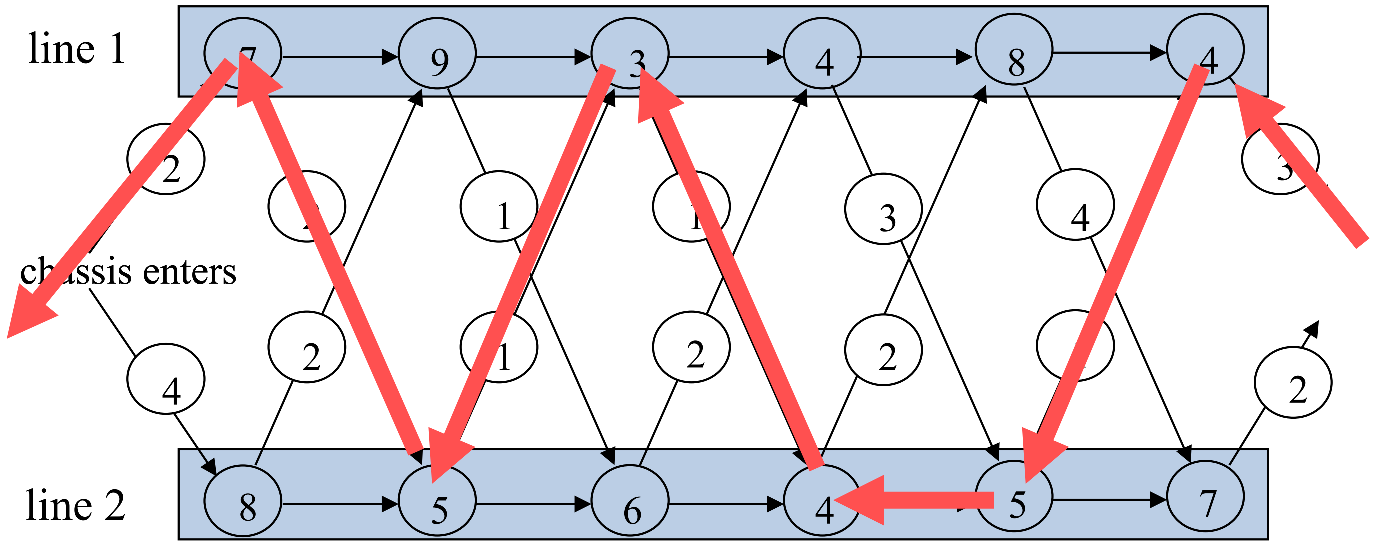
S	1	2	3	4	5	6
1	9	18	20	24	32	35
2	12	16	22	25	30	37

$s^* = 38$

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)
s[1][1] = e[1] + a[1][1]
s[2][1] = e[2] + a[2][1]
for j = 2 to n
   if s[1][j−1] ≤ s[2][j−1] + t[2][j−1]
      s[1][j] = s[1] [j−1] + a[1][j]
      l[1][j] = 1
   else s[1][j] = s[2][j−1] + t[2][j−1] + a[1][j]
      l[1][j] = 2
   if s[2][j−1] ≤ s[1][j−1] + t[1][j−1]
      s[2][j] = s[2][j−1] + a[2][j]
      l[2][j] = 2
   else s[2][j] = s[1][j−1] + t[1][j−1] + a[2][j]
      l[2][j] = 1
if s[1][n] + x[1] ≤ s[2][n] + x[2]
   s* = s[1][n] + x[1]
   l* = 1
else s* = s[2][n] + x[2]
   l* = 2
```

```
PRINT-STATIONS (l, l*, n)
i = l*
print "line " i ", station" n
for j = n downto 2
    i = l[i][j]
    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

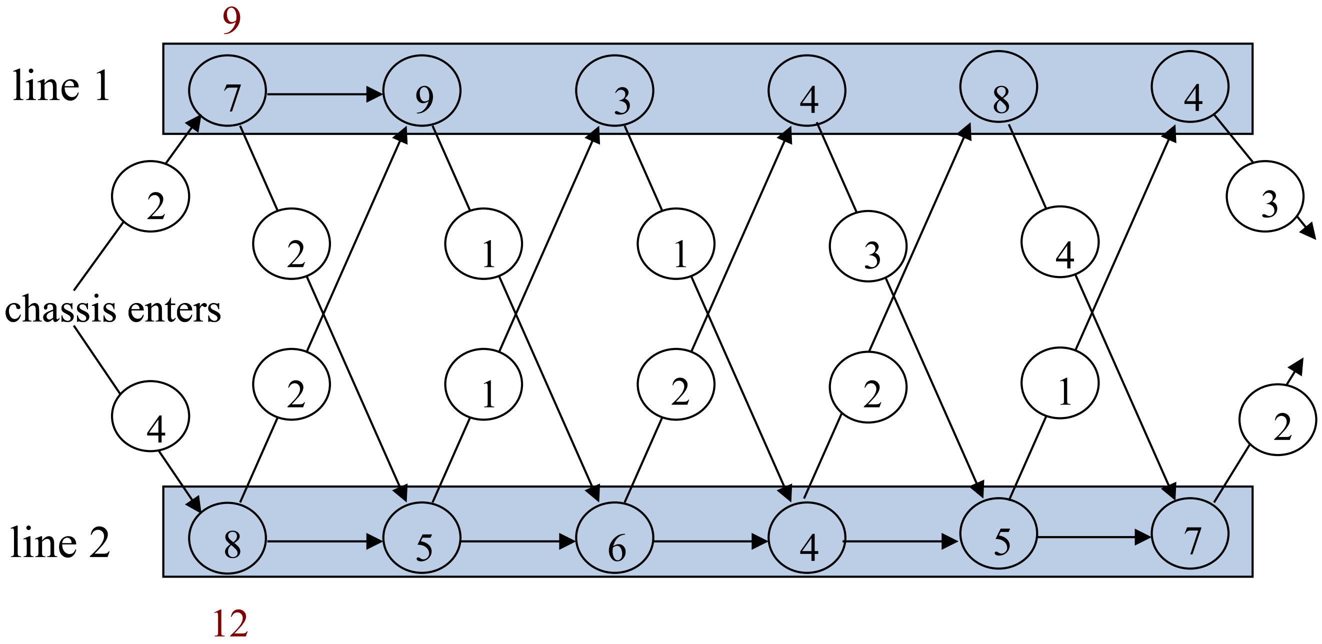
$l^* = 1$

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

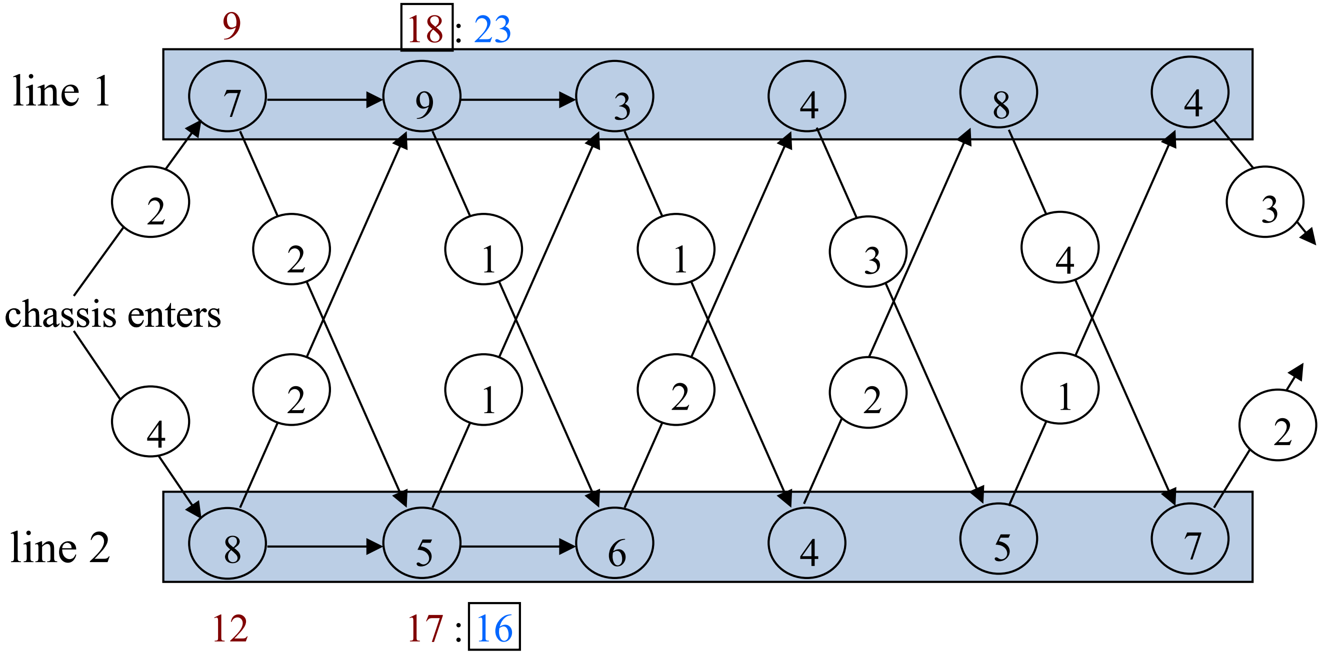
- **Running time**
  - Computing each element requires $\Theta(1)$ time.
  - $\Theta(n)$ time in total

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

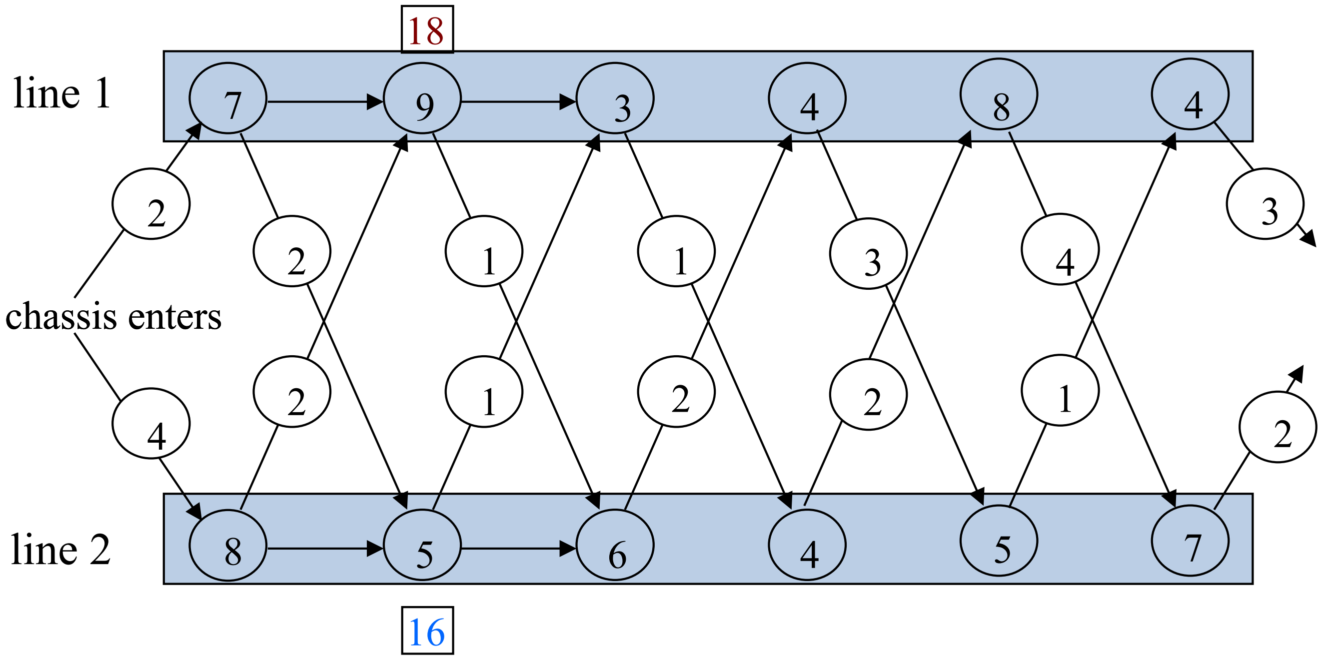
Assembly-line scheduling



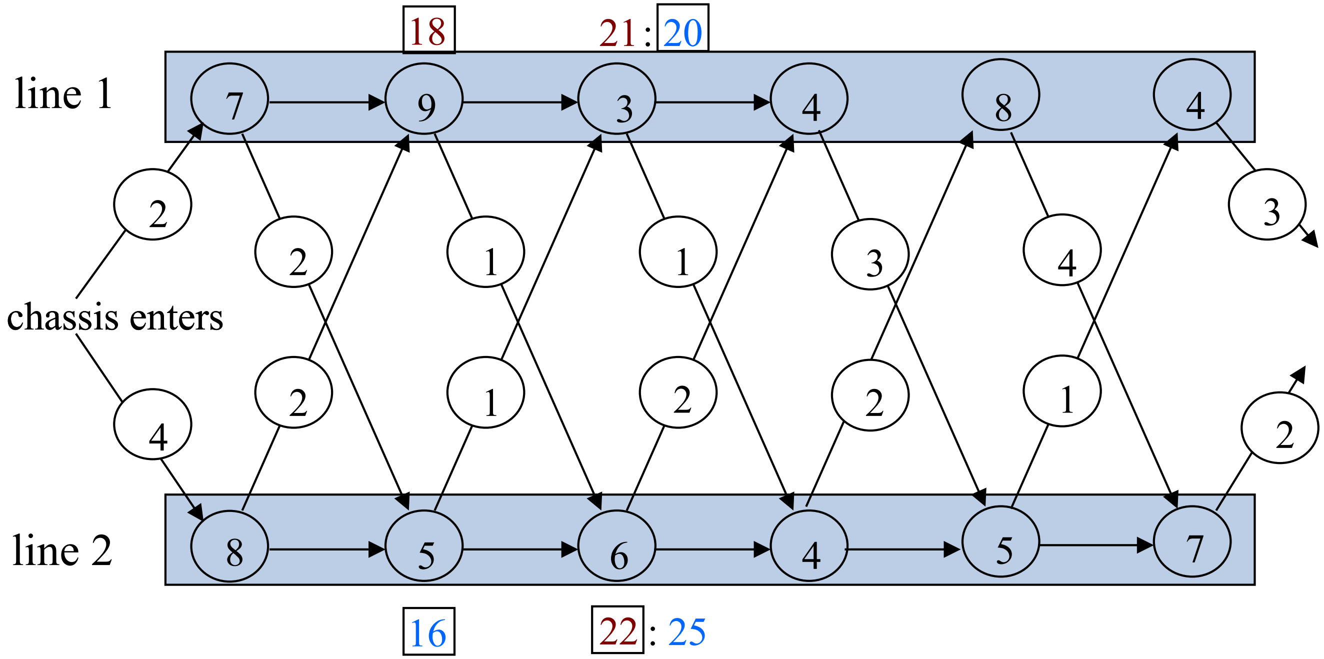
S	1	2	3	4	5	6
1	9					
2	12					



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

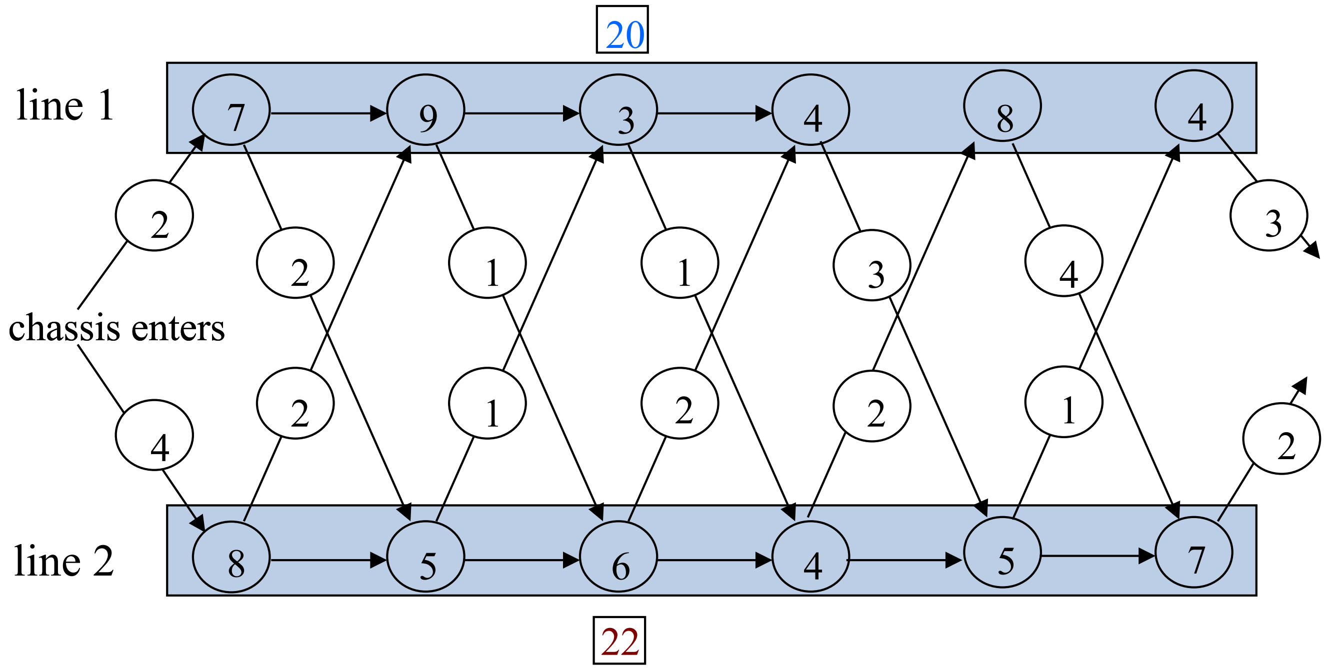


S	1	2	3	4	5	6
1		18				
2		16				

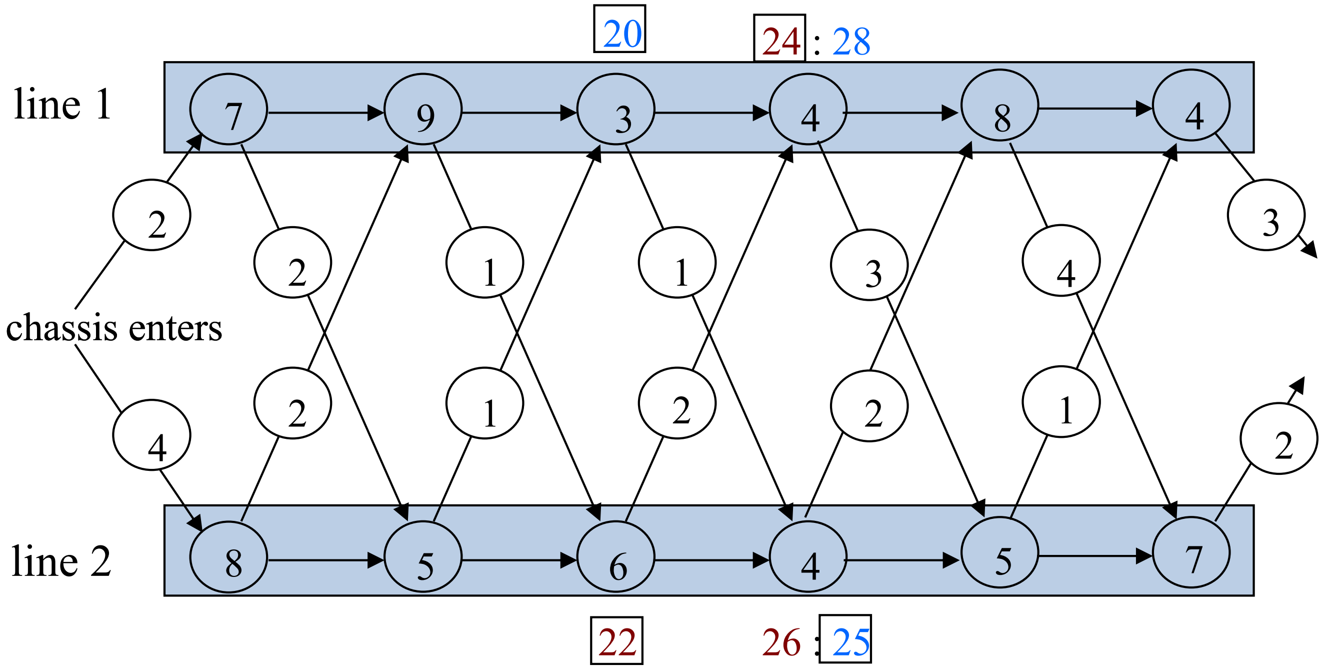


S	1	2	3	4	5	6
1		18	20			
2		16	22			

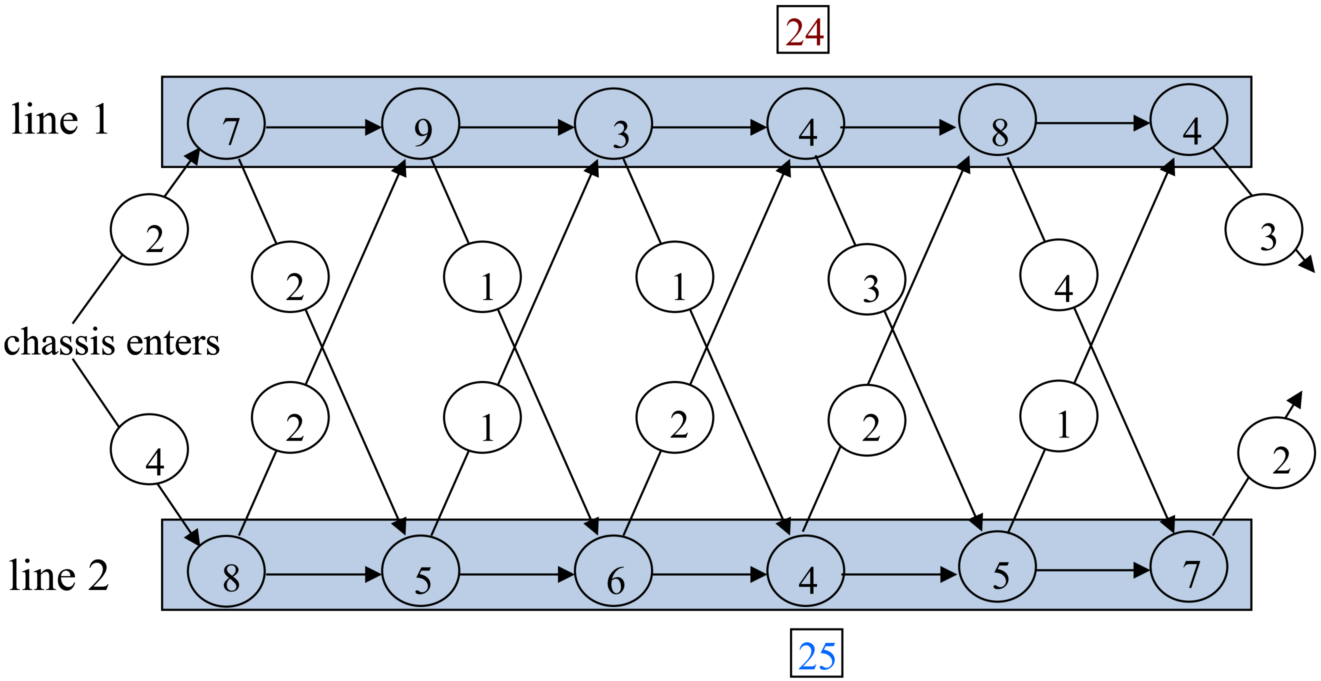
Assembly-line scheduling



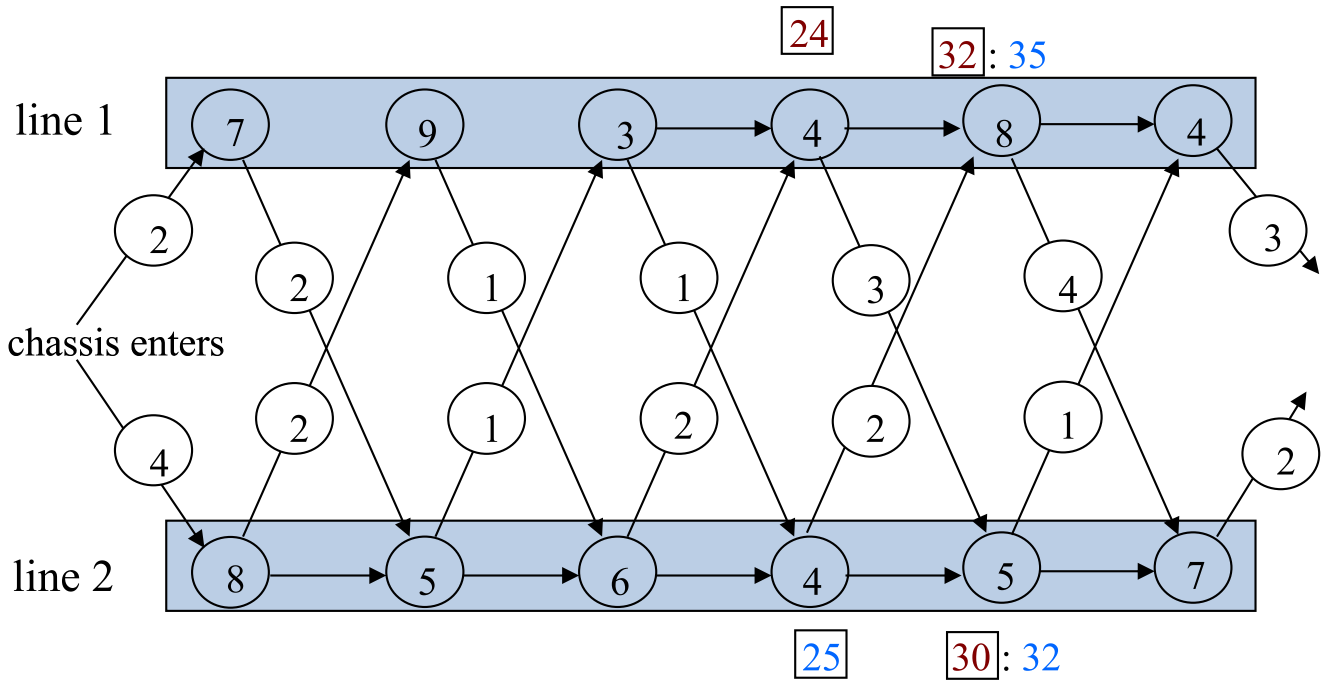
S	1	2	3	4	5	6
1			20			
2			22			



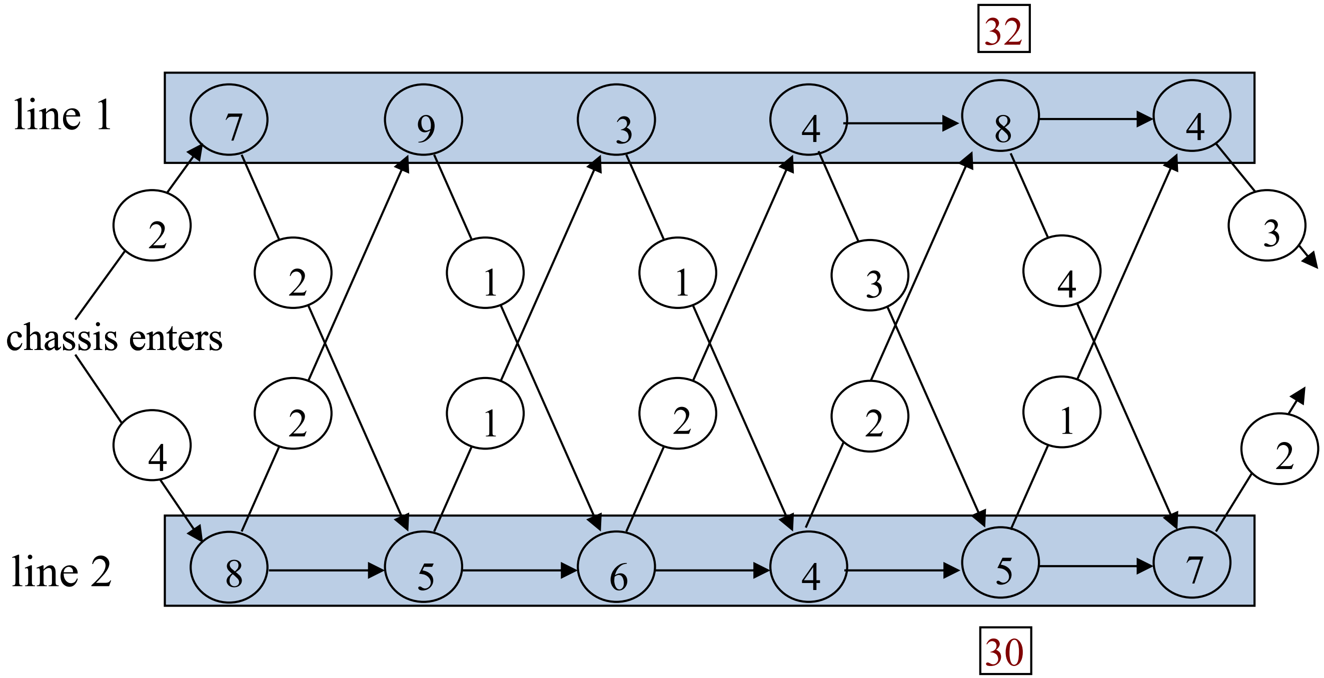
S	1	2	3	4	5	6
1			20	24		
2			22	25		



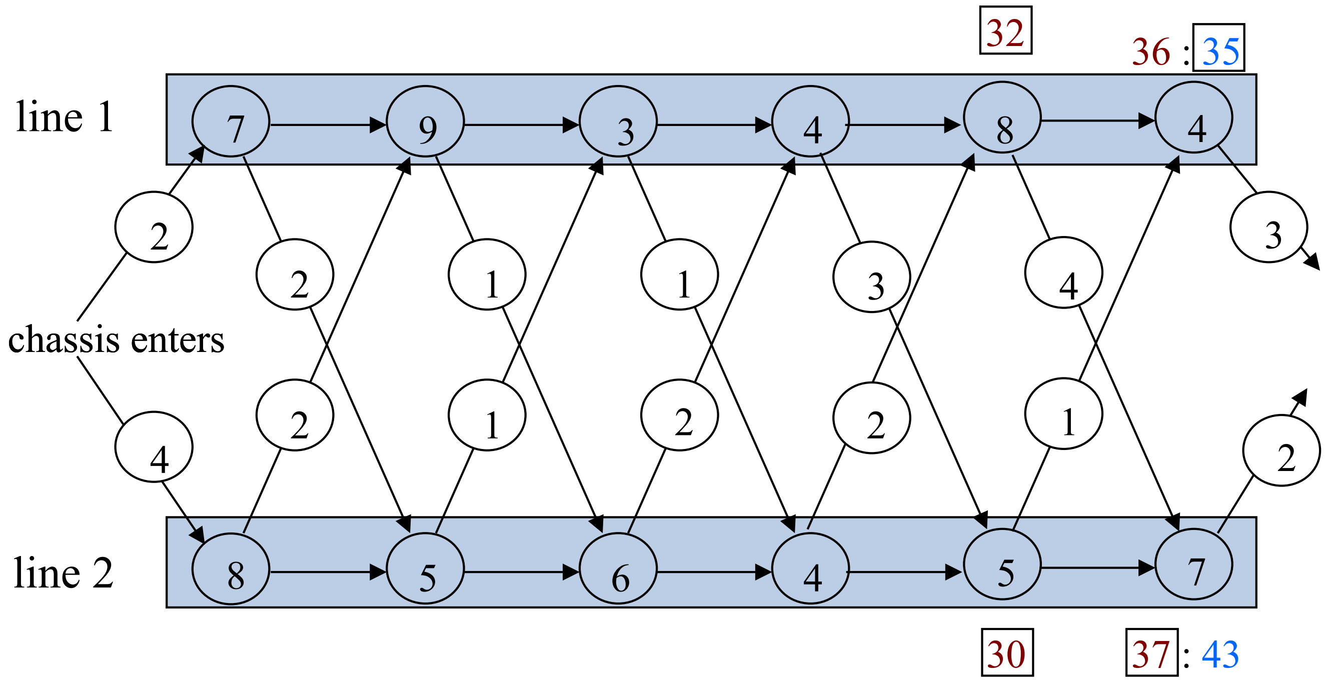
S	1	2	3	4	5	6
1				24		
2				25		



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



S	1	2	3	4	5	6
1					32	
2					30	



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

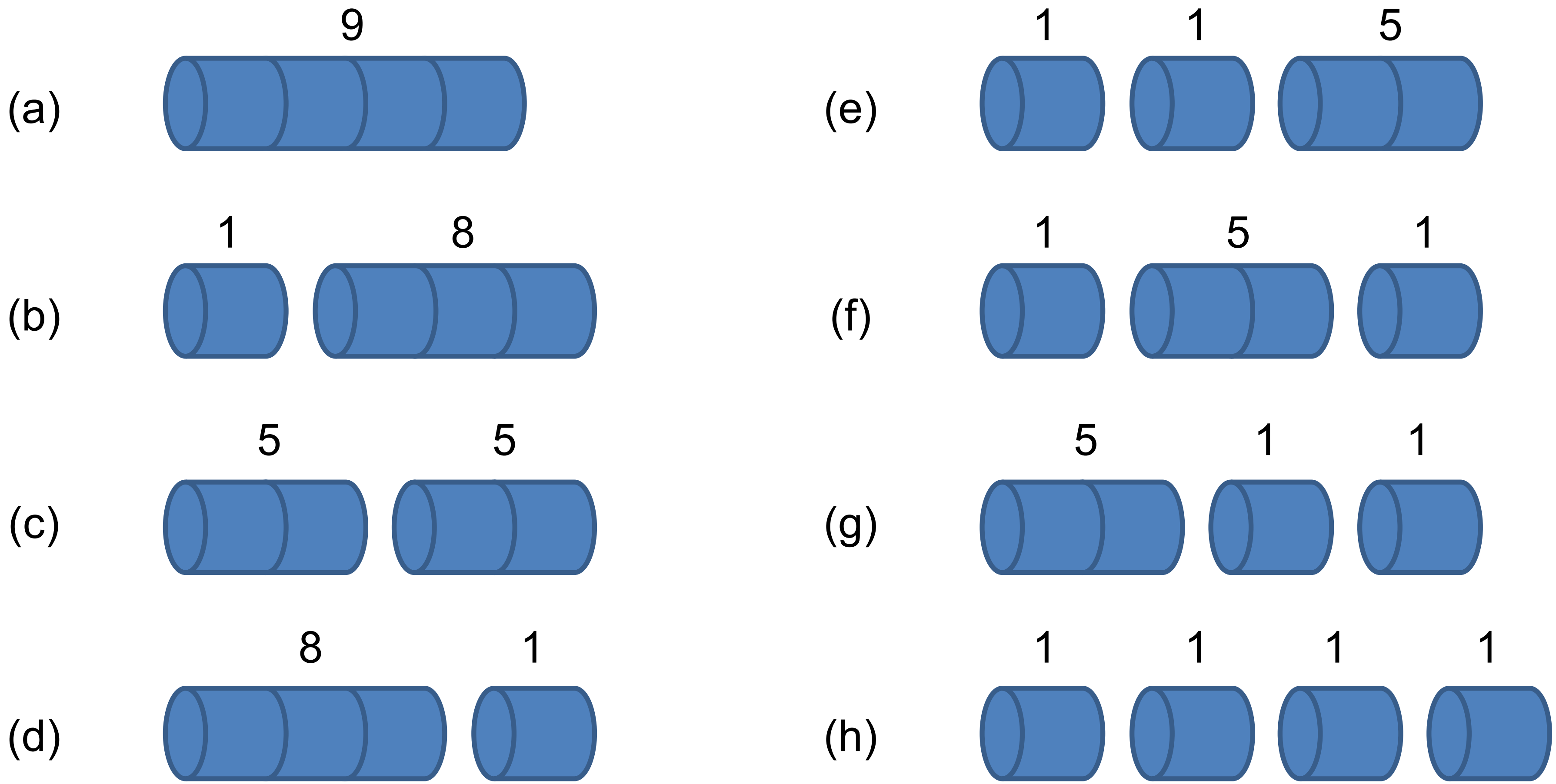
- **Fastest time only**
  - $\Theta(1)$ space
    - Table $\boldsymbol{s}$
      - $2n$ elements → 4 elements

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

Rod cutting

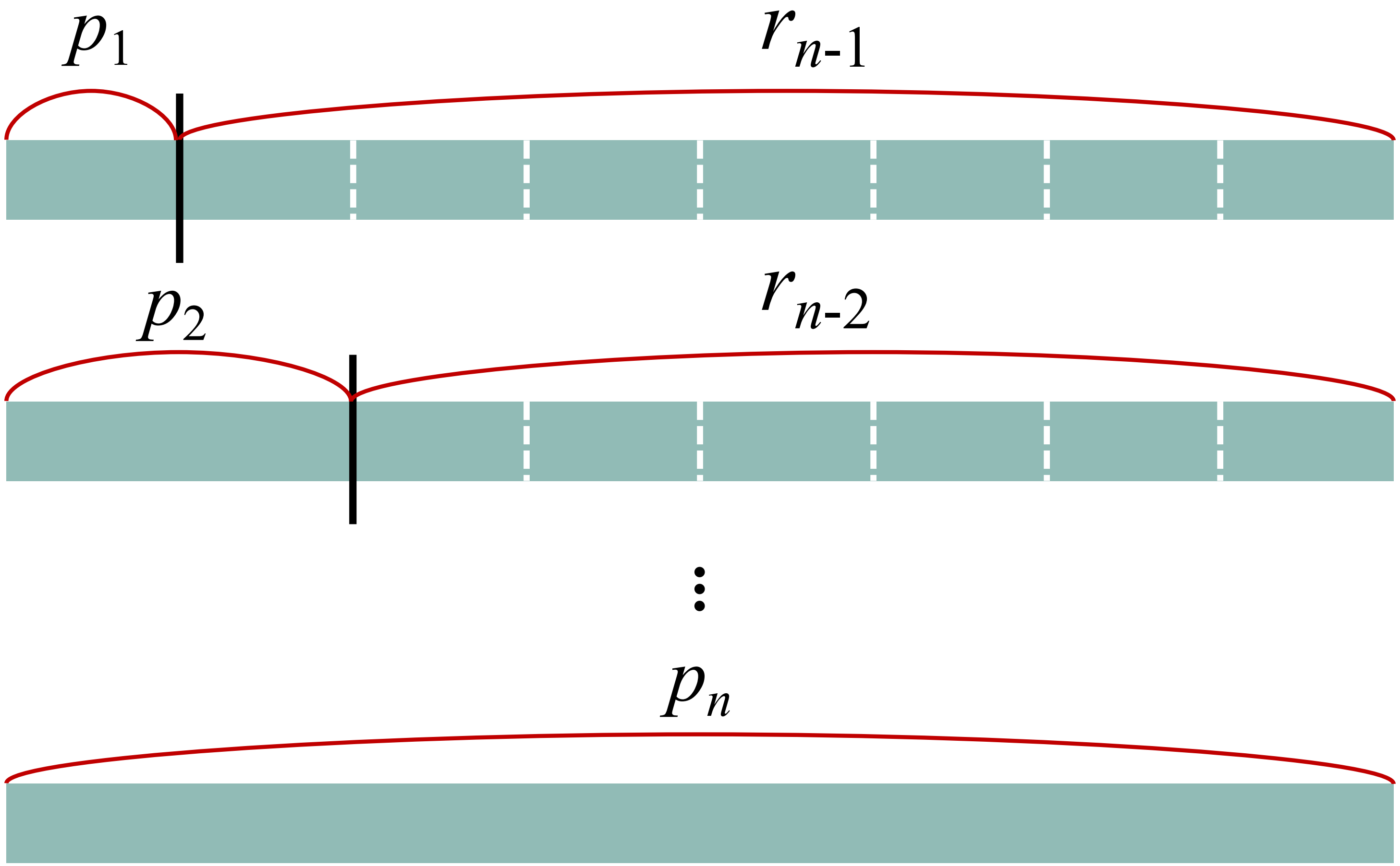
- The *rod-cutting problem*: Given a rod of length n inches and a table of prices p_i for i = 1, 2, … n, determine the maximum revenue r_n obtainable by cutting up the rod and selling the pieces.

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



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

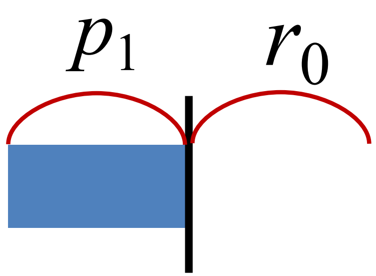
$$r_n = \max_{1 \le i \le n} (p_i + r_{n-i})$$



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**									

$r[0] = 0$

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

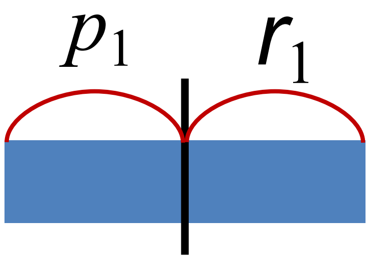


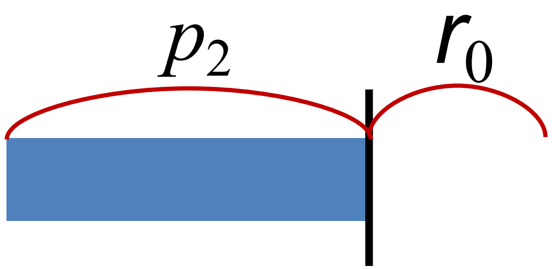
i	0	1	2	3	4	5	6	7	8	9	10
$p[i]$	0	1	5	8	9	10	17	17	20	24	30
$r[i]$	0	1	**5**								

$$r[0] = 0$$

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

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





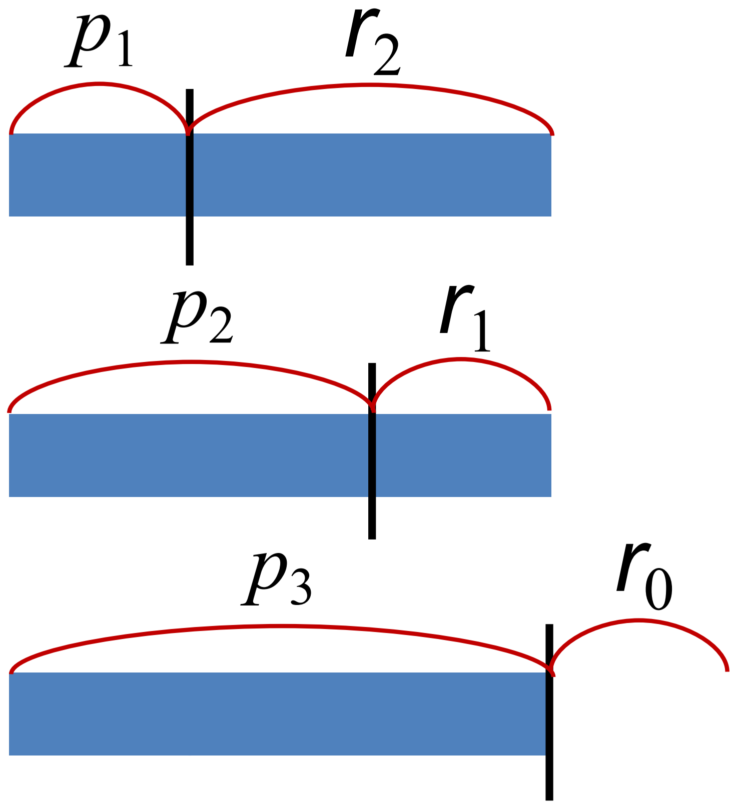
i	0	1	2	3	4	5	6	7	8	9	10
$p[i]$	0	1	5	8	9	10	17	17	20	24	30
$r[i]$	0	1	5	**8**							

$r[0] = 0$

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

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

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



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

$r[0] = 0$

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

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

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

$\vdots$

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

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

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

```
let r[0 .. n] and s[0 .. n] be new arrays
r[0] = 0
for j = 1 to n
    q = −∞
    for i = 1 to j
        if q < p[i] + r[j − i]
            q = p[i] + r[j − i]
            s[j] = i
    r[j] = q
return r and s
```

Rod cutting

```
PRINT-CUT-ROD-SOLUTION(p, n)
(r, s) = EXTENDED-BOTTOM-UP-CUT-ROD (p, n)
while n > 0
    print s[n]
    n = n – s[n]
```

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

- Space consumption
  - $\Theta(n)$

- Running time
  - $\Theta(n^2)$
  - $1 + 2 + 3 + 4 + \cdots + n = n(n+1)/2$

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

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

- **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

- ***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 $Y=BDCABA$

- ***Longest common subsequence*** **(LCS)**
  - $BCBA$ is the longest common subsequence of X and Y

$$X = A\ B\ C\ B\ D\ A\ B$$

$$Y = B\ D\ C\ A\ B\ A$$

- **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.

- **Dynamic programming**
  - **The *i*th *prefix* X_i of X** is $X_i = x_1x_2\ldots x_i$
  - If $X = ABCBDAB$
    - $X_4 = ABCB$
    - $X_0 =$

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

$$X = A\ B\ C\ B\ D\ A\ B$$

$$Y = B\ D\ C\ A\ B$$

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

$X = A\ B\ C\ B\ D\ A\ B$

$Y = B\ D\ C\ A\ A$

$X = A\ B\ C\ B\ D\ A\ B$

$Y = B\ D\ C\ A\ A$

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

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

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

```
LCS-LENGTH (X, Y)
    m = X.length
    n = Y.length
    let b[1 .. m][1 .. n] and c[0 .. m][0 .. n] be new tables
    for i = 1 to m
        c[i][0] = 0
    for j = 0 to n
        c[0][j] = 0
    for i = 1 to m
        for j = 1 to n
            if x_i == y_j
                c[i][j] = c[i − 1][j − 1] + 1
                b[i][j] = "↖"
            elseif c[i − 1][j] ≥ c[i][j − 1]
                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
```

Longest common subsequence

```
PRINT-LCS (b, X, i, j)
    if i == 0 or j == 0
        return
    if b[i][j] == "↖"
        PRINT-LCS (b, X, i − 1, j − 1)
        print x_i
    elseif b[i][j] == "↑"
        PRINT-LCS (b, X, i − 1, j)
    else PRINT-LCS (b, X, i, j − 1)
```

- Multiple LCSs

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

- Multiple LCSs

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

- Multiple LCSs

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

- Multiple LCSs

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

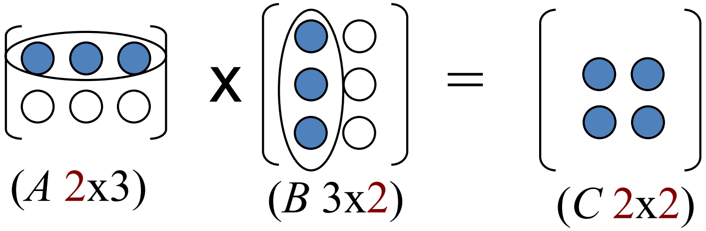
	j	0	1	2	3	4	5	6	⋯	100,000
i		y_j	B	D	A	C	B	A	⋯	A
0	x_i								⋯	
1	A								⋯	
2	B								⋯	
3	C								⋯	
4	B								⋯	
5	A								⋯	

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

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

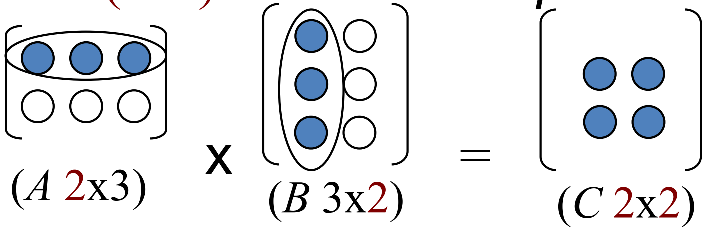
Problems	Space	Time
Assembly-line scheduling	$\Theta(n)$	$\Theta(n)$
Rod cutting	$\Theta(n)$	$\Theta(n^2)$
Longest common subsequence	$\Theta(mn)$ $\Theta(n^2)$	$\Theta(mn)$ $\Theta(n^2)$
Matrix-chain multiplication		

- **Multiplying two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$**
  - A $(p \times q)$ and B $(r \times s)$ *can be multiplied only if* $q = r$.



Matrix-chain multiplication

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



  - 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.

- **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.***

- Computing $A_1A_2A_3$ where A_1: 10×100 A_2: 100×5 A_3 : 5×50

  - $(A_1\ A_2)\ A_3$
    - $(A_1\ A_2) = 10*100*5 = 5000$, $(10 \times 5)\ A_3 = 10*5*50 = 2500$

      $\Rightarrow 5000 + 2500 =$ **7,500**

  - $A_1\ (A_2\ A_3)$
    - $(A_2\ A_3) = 100*5*50 = 25000$, $A_1(100 \times 50) = 10*100*50 = 50000$

      $\Rightarrow 25000 + 50000 =$ **75,000**

  - **Computing $(A_1\ A_2)\ A_3$ is 10 times faster.**

- **Matrix-chain multiplication problem**
  - Given a chain $A_1, A_2, ..., A_n$ 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: $p_2\times p_3$ ...

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

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

  $(A_1 A_2)(A_3 A_4)$,

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

- **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 $\Omega(4^n/n^{3/2})$.

- **Dynamic programming**
- **Optimal substructure**
  - $m[i][j]$: The minimum number of scalar multiplications for computing $A_i A_{i+1} \ldots A_{j.}$

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

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

$i \backslash j$	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

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

p

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

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

$(A_2)(A_3 A_4 A_5)$

$(A_2 A_3)(A_4 A_5)$

$(A_2 A_3 A_4)(A_5)$

Matrix-chain multiplication

```
MATRIX-CHAIN-ORDER (p)
n = p.length − 1
let m[1 .. n][1 .. n] and s[1 .. n − 1][2 .. n] be new tables
for i = 1 to n
    m[i][i] = 0
for l = 2 to n                          // l is the chain length
    for i = 1 to n − l + 1
        j = i + l − 1
        m[i][j] = ∞
        for k = i to j − 1
            q = m[i][k] + m[k + 1][j] + p[i−1]p[k]p[j]
            if q < m[i][j]
                m[i][j] = q
                s[i][j] = k
return m and s
```

Matrix-chain multiplication

```
PRINT-OPTIMAL-PARENS (s, i, j)
if i == j
    print "A_i"
else print "("
    PRINT-OPTIMAL-PARENS(s, i, s[i][j])
    PRINT-OPTIMAL-PARENS(s, s[i][j] + 1, j)
    print ")"
```

- **Space consumption**
  - $\Theta(n^2)$ space to store m and s tables.
- **Running time**
  - $\Theta(n^3)$

$1 \cdot n + 1 \cdot n\text{-}1 + 2 \cdot n\text{-}2 + \cdots + (n - 1) \cdot 1$

$= 1 \cdot n + \sum_{k=1}^{n-1} k(n-k)$

$= 1 \cdot n + \sum_{k=1}^{n-1} kn - \sum_{k=1}^{n-1} k^2$

$= n + n^2(n-1)/2 - n(n-1)(2n-1)/6$

$= (n^3 + 5n)/6$

$= \Theta(n^3)$

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	$\Theta(n^2)$	$\Theta(n^3)$