Analysis of Algorithms
CS 477/677

Instructor: Monica Nicolescu

Lecture 15

Dynamic Programming

- An algorithm design technique used for **optimization problems**
  - Find a solution with the **optimal value** (minimum or maximum)
  - A set of **choices** must be made to get an optimal solution
  - There may be multiple solutions that return the optimal value: we want to find one of them

Dynamic Programming

- Similar to divide and conquer, but with one key difference
  - Subproblems are **not independent:** subproblems share subsubproblems
- Divide and conquer
  - Partition the problem into **independent** subproblems
  - Solve the subproblems recursively
  - Combine the solutions to solve the original problem

Dynamic Programming

- Applicable when subproblems are **not independent**
  - Subproblems share subsubproblems

E.g.: Fibonacci numbers:

- Recurrence: $F(n) = F(n-1) + F(n-2)$
- Boundary conditions: $F(1) = 0$, $F(2) = 1$
- Compute: $F(5) = 3$, $F(3) = 1$, $F(4) = 2$

  - A divide and conquer approach would repeatedly solve the common subproblems
  - Dynamic programming solves every subproblem just once and stores the answer in a table

Dynamic Programming Algorithm

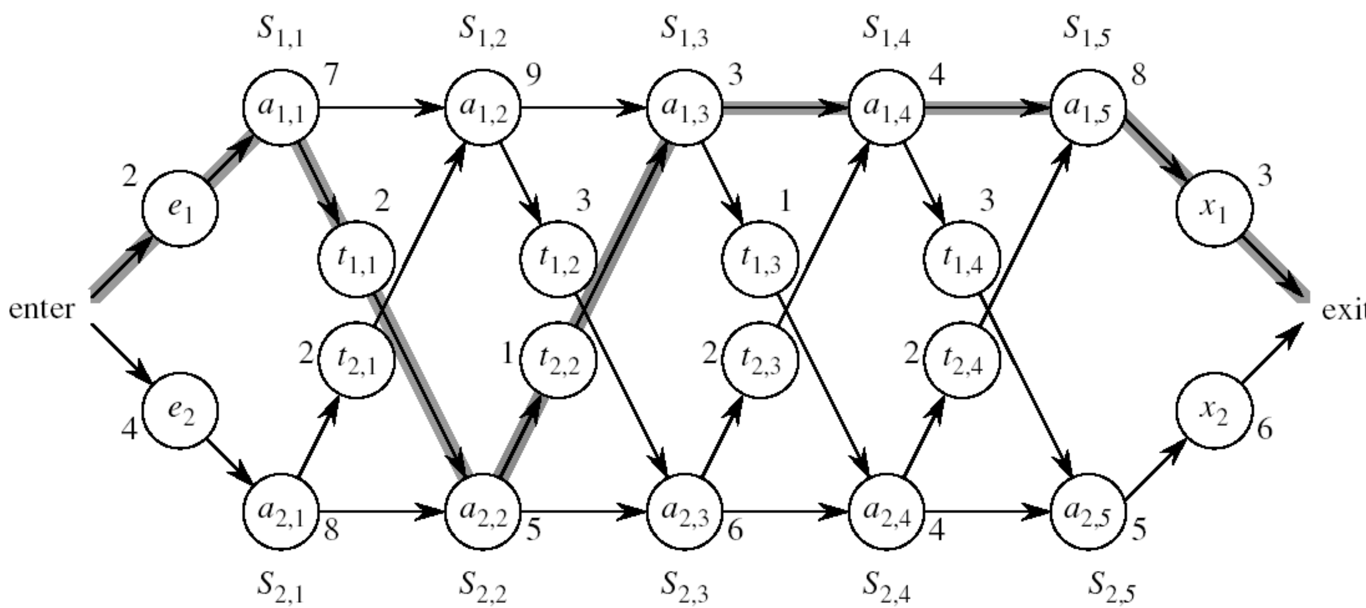
1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution in a bottom-up fashion
4. Construct an optimal solution from computed information

Elements of Dynamic Programming

- Optimal Substructure
  - An optimal solution to a problem contains within it an optimal solution to subproblems
  - Optimal solution to the entire problem is built in a bottom-up manner from optimal solutions to subproblems
- Overlapping Subproblems
  - If a recursive algorithm revisits the same subproblems again and again $\Rightarrow$ the problem has overlapping subproblems

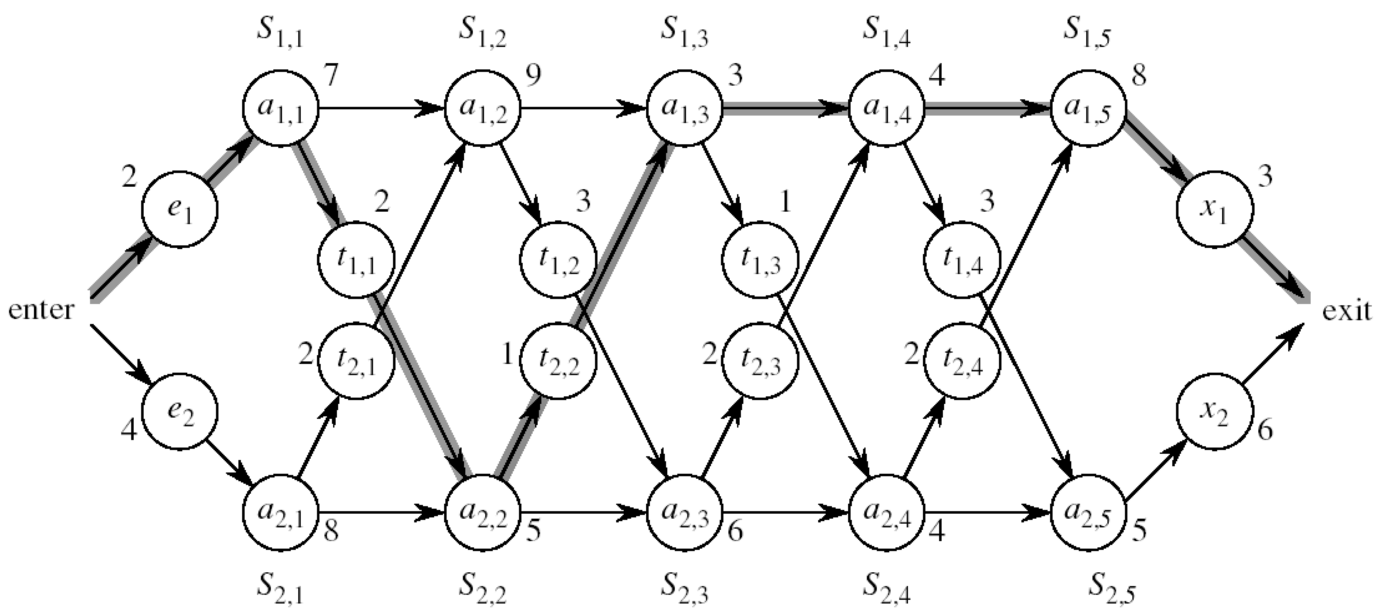
Assembly Line Scheduling

- Automobile factory with two assembly lines
  - Each line has n stations: $S_{1,1}, \ldots, S_{1,n}$ and $S_{2,1}, \ldots, S_{2,n}$
  - Corresponding stations $S_{1,j}$ and $S_{2,j}$ perform the same function but can take different amounts of time $a_{1,j}$ and $a_{2,j}$
  - Times to enter are e_1 and e_2 and times to exit are x_1 and x_2



Assembly Line

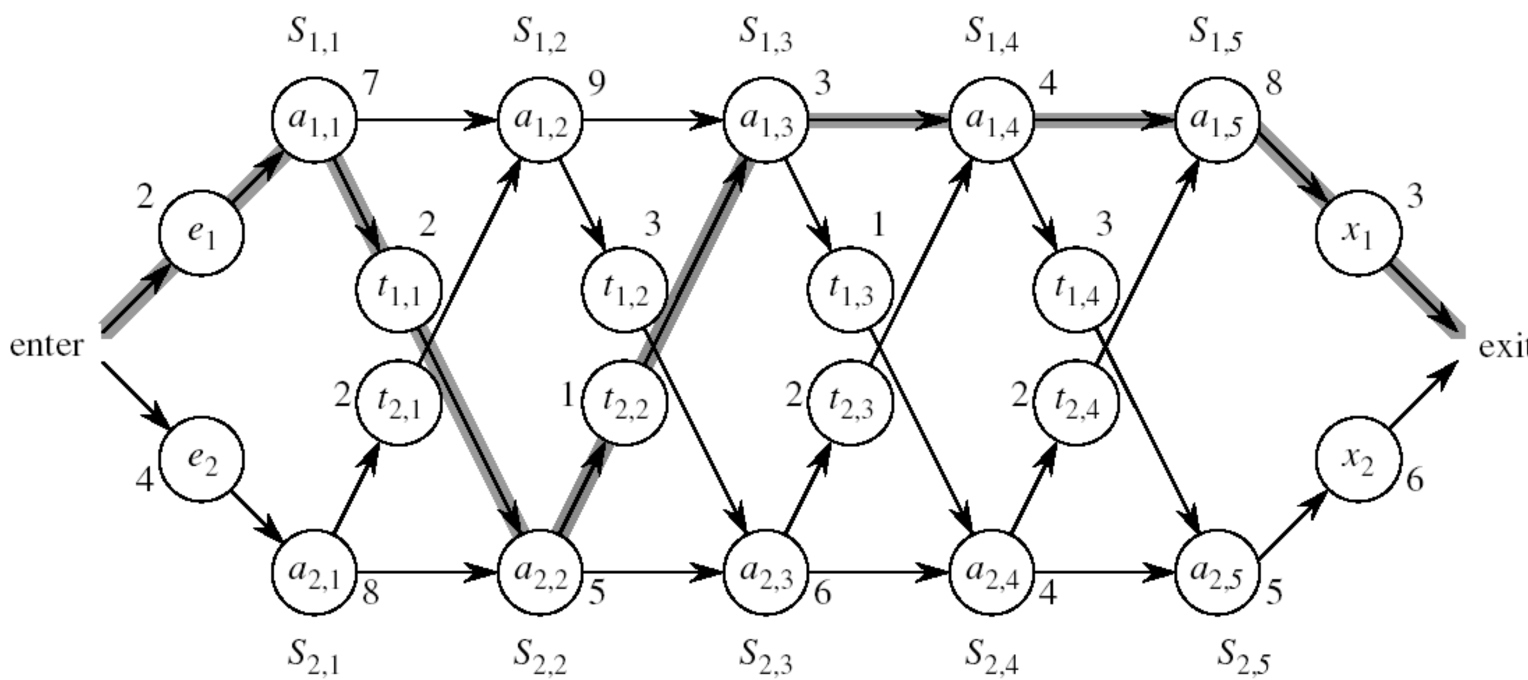
- After going through a station, the car can either:
  - stay on same line at no cost, or
  - transfer to other line: cost after $S_{i,j}$ is $t_{i,j}$, $i = 1, 2, j = 1, . . . , n-1$



Assembly Line Scheduling

- Problem:

  What stations should be chosen from line 1 and what from line 2 in order to minimize the total time through the factory for one car?



One Solution

- Brute force
  - Enumerate all possibilities of selecting stations
  - Compute how long it takes in each case and choose the best one

1	2	3	4				n
1	0	0	1	- - -	- - -	- - -	1

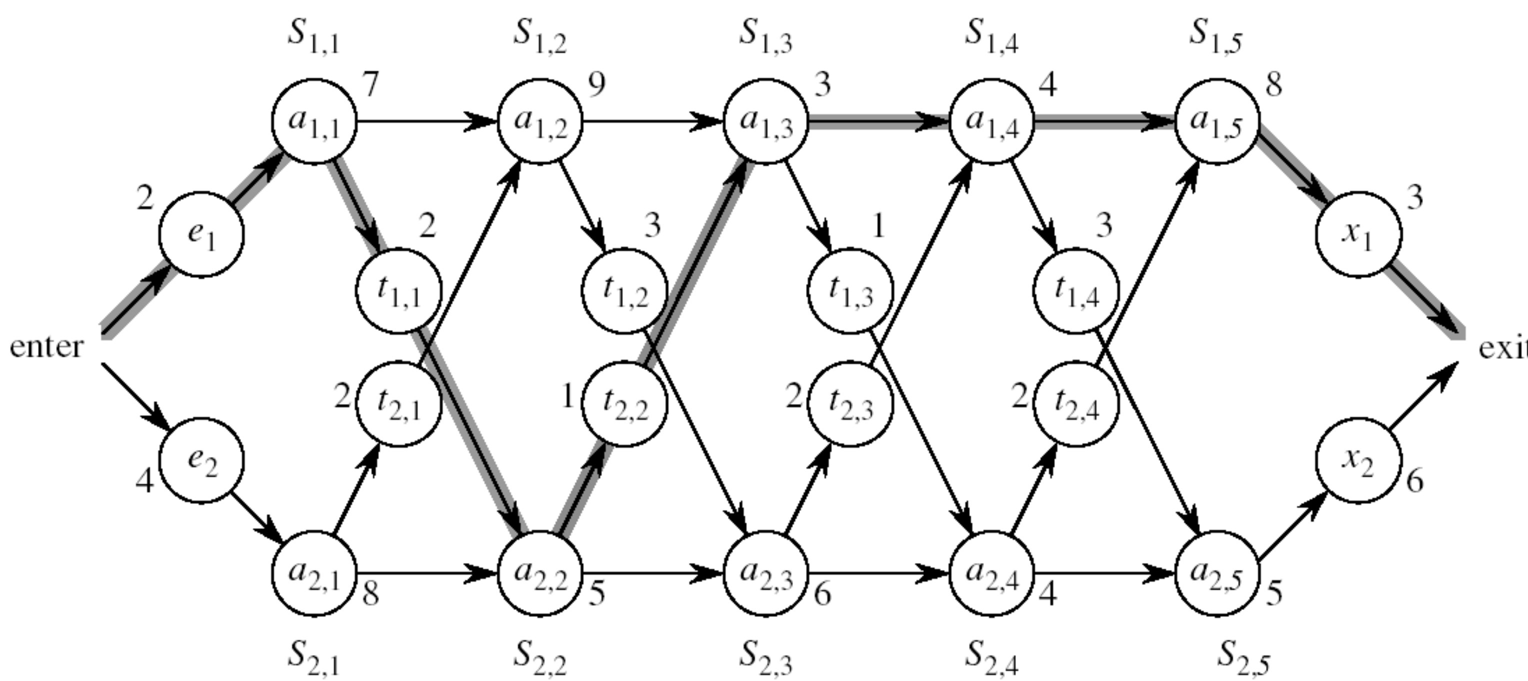
0 if choosing line 2
at step j (= 3)

1 if choosing line 1
at step j (= n)

  - There are 2^n possible ways to choose stations
  - Infeasible when n is large

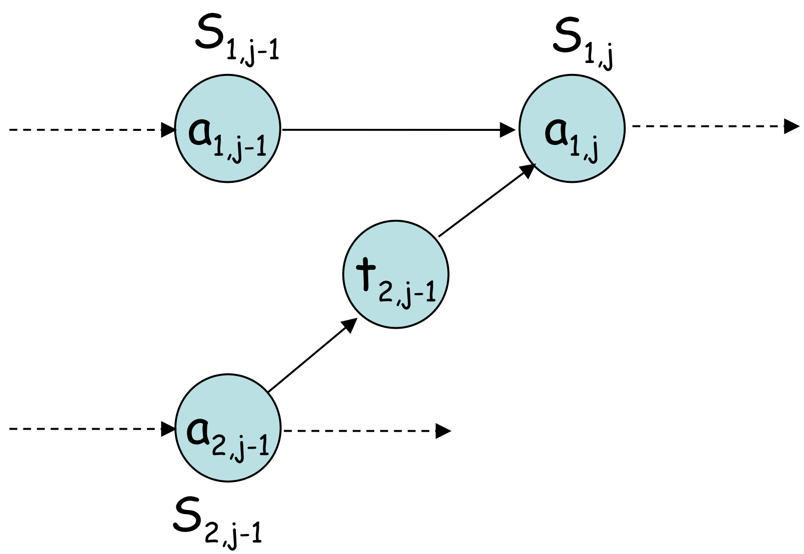
1. Structure of the Optimal Solution

- How do we compute the minimum time of going through the station?



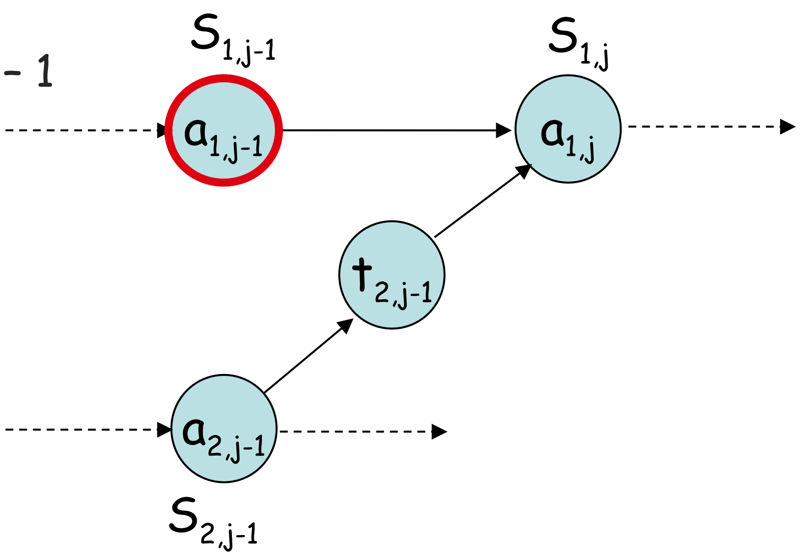
1. Structure of the Optimal Solution

- Let's consider all possible ways to get from the starting point through station $S_{1,j}$
  - We have two choices of how to get to $S_{1,j}$:
    - Through $S_{1,j-1}$, then directly to $S_{1,j}$
    - Through $S_{2,j-1}$, then transfer over to $S_{1,j}$



1. Structure of the Optimal Solution

- Suppose that **the fastest way through $S_{1,j}$** is **through $S_{1,j-1}$**
  - We must have taken the fastest way from entry through $S_{1,j-1}$
  - If there were a faster way through $S_{1,j-1}$, we would use it instead
- Similarly for $S_{2,j-1}$



Optimal Substructure

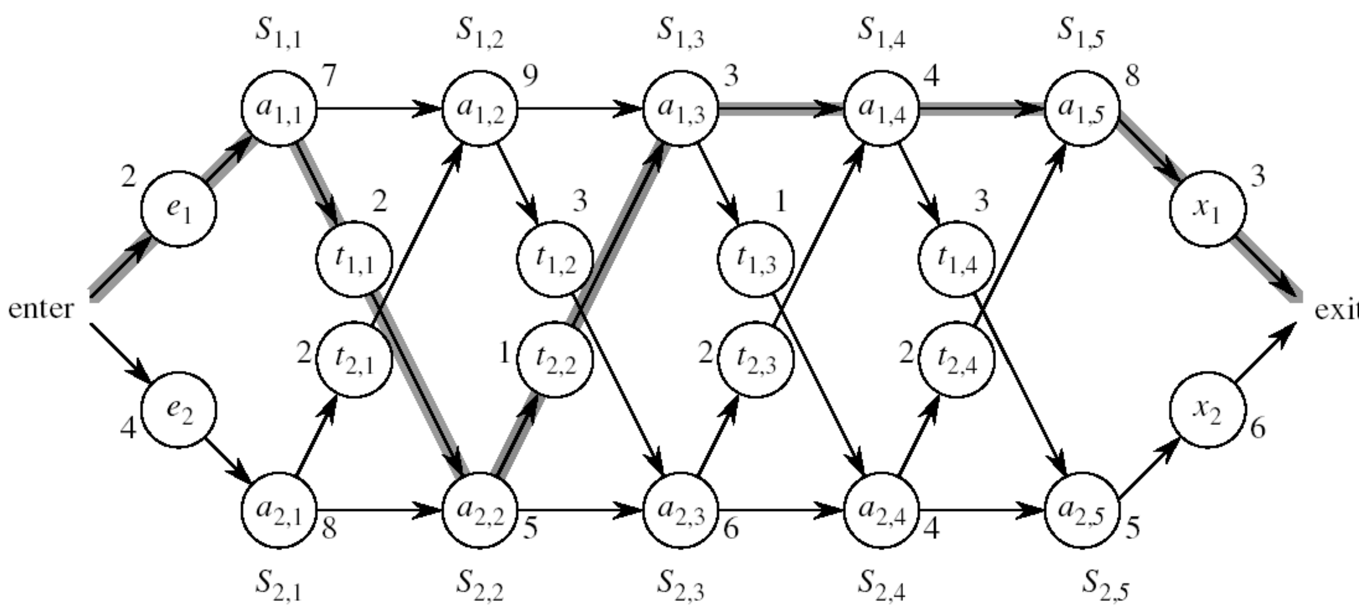
- **Generalization**: an optimal solution to the problem *find the fastest way through* $S_{1,\,j}$ contains within it an optimal solution to subproblems: *find the fastest way through* $S_{1,\,j-1}$ *or* $S_{2,\,j-1}$.

- This is referred to as the **optimal substructure** property

- We use this property to construct an optimal solution to a problem from optimal solutions to subproblems

2. A Recursive Solution

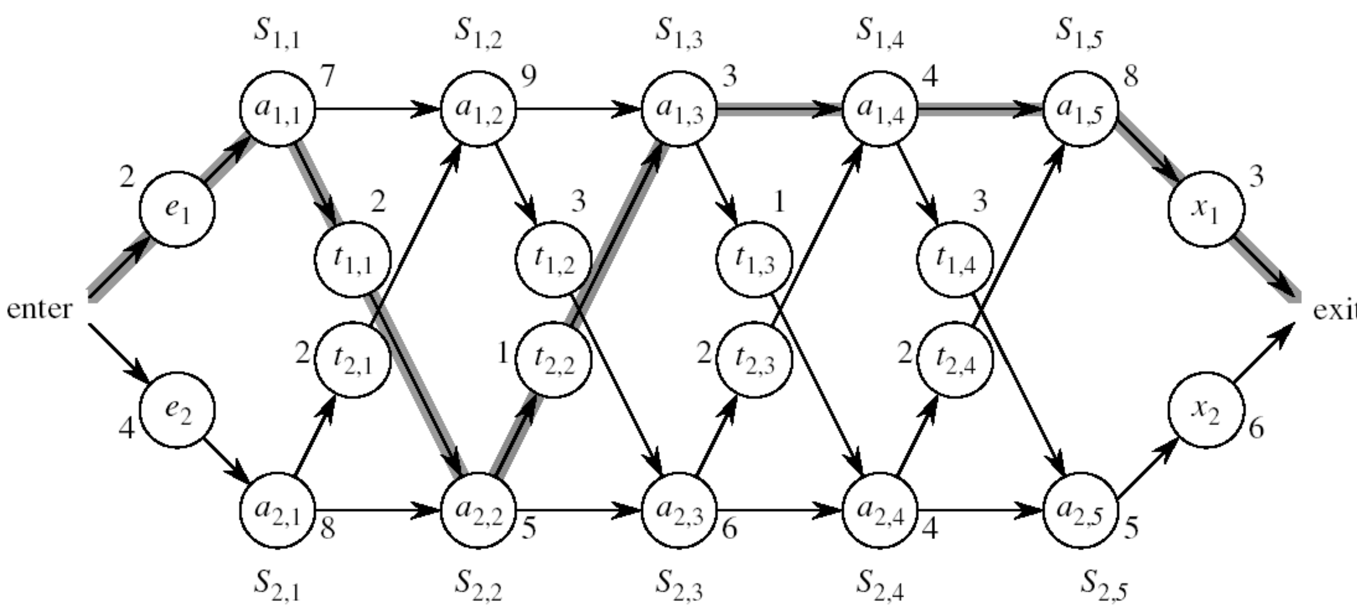
- Define the value of an optimal solution in terms of the optimal solution to subproblems
- Assembly line subproblems
  - Finding the fastest way through each station j on each line i ($i = 1,2, \; j = 1, 2, \ldots, n$)



2. A Recursive Solution

- f^* = the fastest time to get through the entire factory
- $f_i[j]$ = the fastest time to get from the starting point through station $S_{i,j}$

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

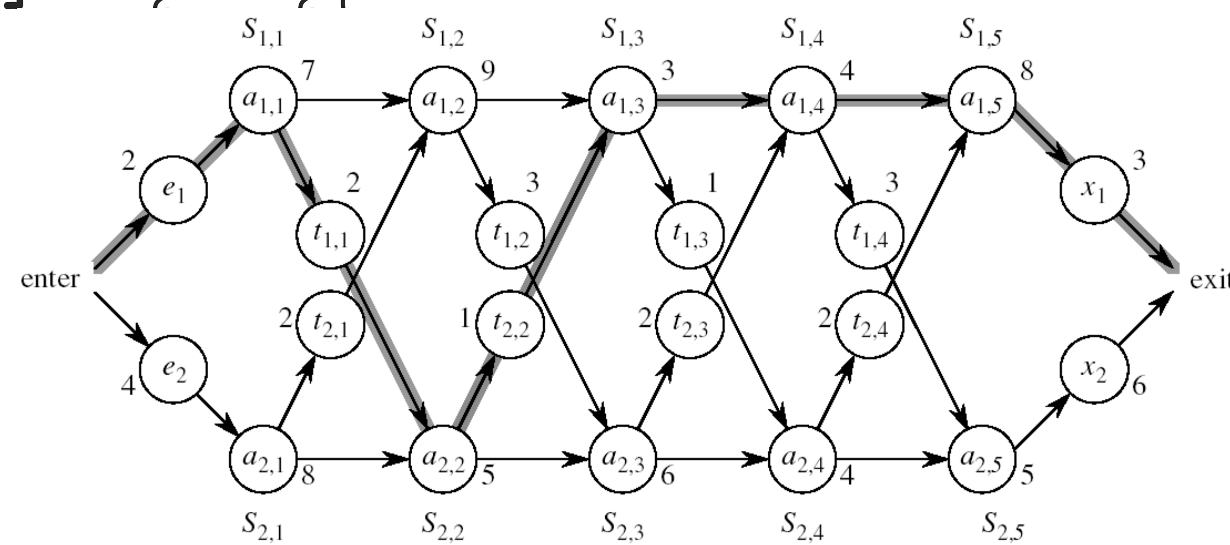


2. A Recursive Solution

- $f_i[j]$ = the fastest time to get from the starting point through station $S_{i,j}$
- j = 1 (getting through station 1)

  $f_1[1] = e_1 + a_{1,1}$

  $f_2[1] = e_2 + a_{2,1}$



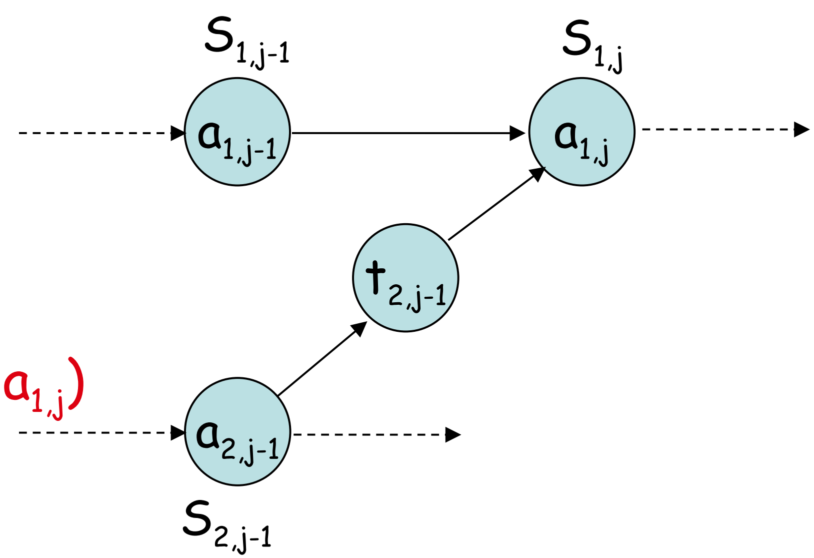
2. A Recursive Solution

- Compute $f_i[j]$ for $j = 2, 3, \ldots, n$, and $i = 1, 2$
- Fastest way through $S_{1,j}$ is either:
  - the way through $S_{1,j-1}$ then directly through $S_{1,j}$, or

    $$f_1[j-1] + a_{1,j}$$

  - the way through $S_{2,j-1}$, transfer from line 2 to line 1, then through $S_{1,j}$

    $$f_2[j-1] + t_{2,j-1} + a_{1,j}$$



$$f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$$

2. A Recursive Solution

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

$$f_2[j] = \begin{cases} e_2 + a_{2,1} & \text{if } j = 1 \\ \min(f_2[j-1] + a_{2,j}\ , f_1[j-1] + t_{1,j-1} + a_{2,j}) & \text{if } j \geq 2 \end{cases}$$

3. Computing the Optimal Value

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

$$f_1[j] = \min\left(f_1[j-1] + a_{1,j},\ f_2[j-1] + t_{2,j-1} + a_{1,j}\right)$$

	1	2	3	4	5
$f_1[j]$	$f_1(1)$	$f_1(2)$	$f_1(3)$	$f_1(4)$	$f_1(5)$
$f_2[j]$	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$	$f_2(5)$

4 times 2 times

- Solving top-down would result in exponential running time

3. Computing the Optimal Value

- For $j \geq 2$, each value $f_i[j]$ depends only on the values of $f_1[j - 1]$ and $f_2[j - 1]$
- Compute the values of $f_i[j]$
  - in increasing order of j

increasing j

	1	2	3	4	5
$f_1[j]$					
$f_2[j]$					

- Bottom-up approach
  - First find optimal solutions to subproblems
  - Find an optimal solution to the problem from the subproblems

4. Construct the Optimal Solution

- We need the information about which line has been used at each station:
  - $l_i[j]$ – the line number $(1, 2)$ whose station $(j - 1)$ has been used to get in fastest time through $S_{i,j}$, $j = 2, 3, \ldots, n$
  - l^* – the line number $(1, 2)$ whose station n has been used to get in fastest time through the exit point

increasing j

	2	3	4	5
$l_1[j]$				
$l_2[j]$				

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

$f_1[1] \leftarrow e_1 + a_{1,1}$

$f_2[1] \leftarrow e_2 + a_{2,1}$

Compute initial values of f_1 and f_2

for $j \leftarrow 2$ **to** n

    do if $f_1[j-1] + a_{1,j} \le f_2[j-1] + t_{2,j-1} + a_{1,j}$

        then $f_1[j] \leftarrow f_1[j-1] + a_{1,j}$

            $l_1[j] \leftarrow 1$

        else $f_1[j] \leftarrow f_2[j-1] + t_{2,j-1} + a_{1,j}$

            $l_1[j] \leftarrow 2$

Compute the values of $f_1[j]$ and $l_1[j]$

      if $f_2[j-1] + a_{2,j} \le f_1[j-1] + t_{1,j-1} + a_{2,j}$

        then $f_2[j] \leftarrow f_2[j-1] + a_{2,j}$

            $l_2[j] \leftarrow 2$

        else $f_2[j] \leftarrow f_1[j-1] + t_{1,j-1} + a_{2,j}$

            $l_2[j] \leftarrow 1$

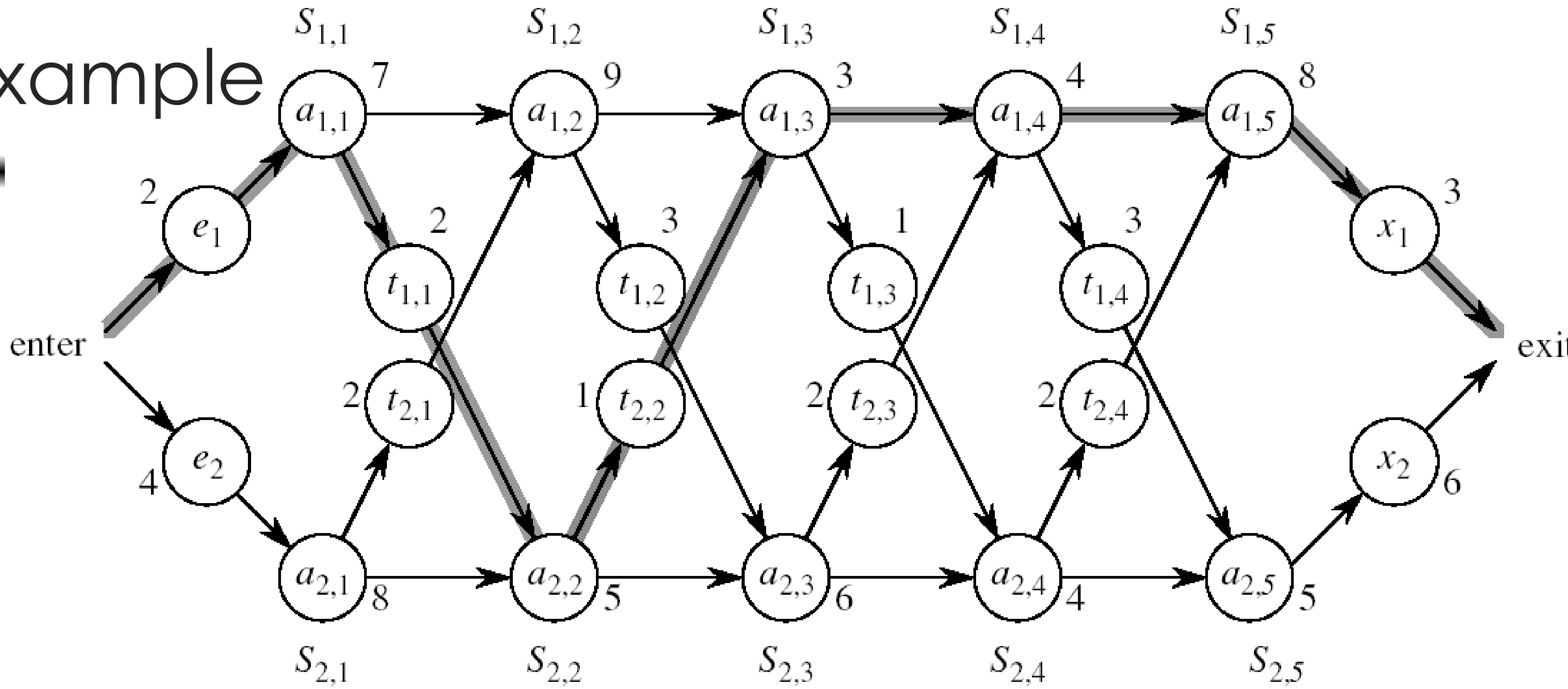
Compute the values of $f_2[j]$ and $l_2[j]$

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

```
if f_1[n] + x_1 ≤ f_2[n] + x_2
    then f* = f_1[n] + x_1
         l* = 1
    else f* = f_2[n] + x_2
         l* = 2
```

Compute the values of the fastest time through the entire factory

Example



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

	1	2	3	4	5
$f_1[j]$	9	18[1]	20[2]	24[1]	32[1]
$f_2[j]$	12	16[1]	22[2]	25[1]	30[2]

$f^* = 35$[1]

4. Construct an Optimal Solution

Alg.: PRINT-STATIONS(l, n)

$i \leftarrow l^*$

print "line " i ", station " n

for $j \leftarrow n$ **downto** 2

  do $i \leftarrow l_i[j]$

  print "line " i ", station " $j - 1$

line 1, station 5

line 1, station 4

line 1, station 3

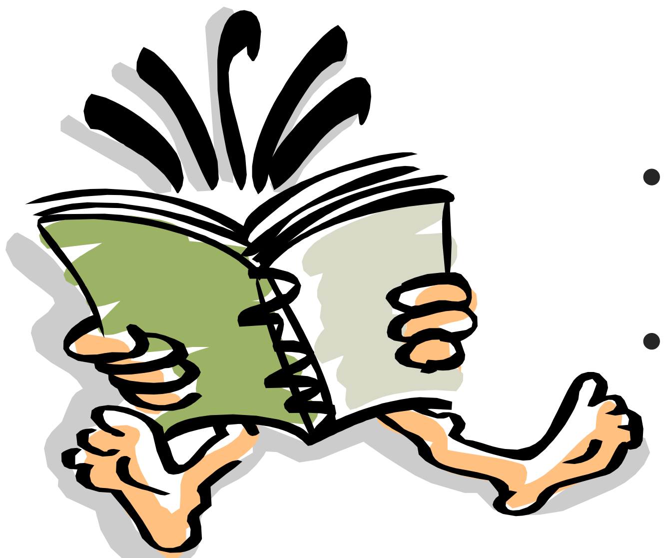
line 2, station 2

line 1, station 1

	1	2	3	4	5
$f_1[j]$ $l_1[j]$	9	18[1]	20[2]	24[1]	32[1]
$f_2[j]$ $l_2[j]$	12	16[1]	22[2]	25[1]	30[2]

$l^* = 1$

Readings



- For this lecture
  - Chapter 14
- Coming next
  - Chapter 14