Projeto e Análise de Algoritmos

Prof. Dr. Jonas Lopes de Vilas Boas

Divisão e conquista

Divisão e conquista

- É criada uma solução para uma pequena instância de um problema.
- A instância do problema é dividida em duas ou mais instâncias menores.
- Cada instância menor é resolvida usando o algoritmo definido.
- As soluções das instâncias menores são combinadas para produzir uma solução da instância original.
- Implementada por uma chamada recursiva.

- O método da divisão e conquista produz um algoritmo eficiente se a fase de divisão e a fase da combinação forem suficientemente rápidos.

Quick sort

```
DIVIDE (A, p, r)    ▷ p ≤ r
x := A[r]    ▷ pivô
i := p−1
para j := p até r−1
    se A[j] ≤ x
        i := i+1
        troque A[i] com A[j]
troque A[i+1] com A[r]
devolva i + 1
```

```
QUICKSORT (A, p, r)    ▷ p ≤ r+1
se p ≤ r
    q := DIVIDE (A, p, r)
    QUICKSORT (A, p, q−1)
    QUICKSORT (A, q+1, r)
```

Programação Dinâmica

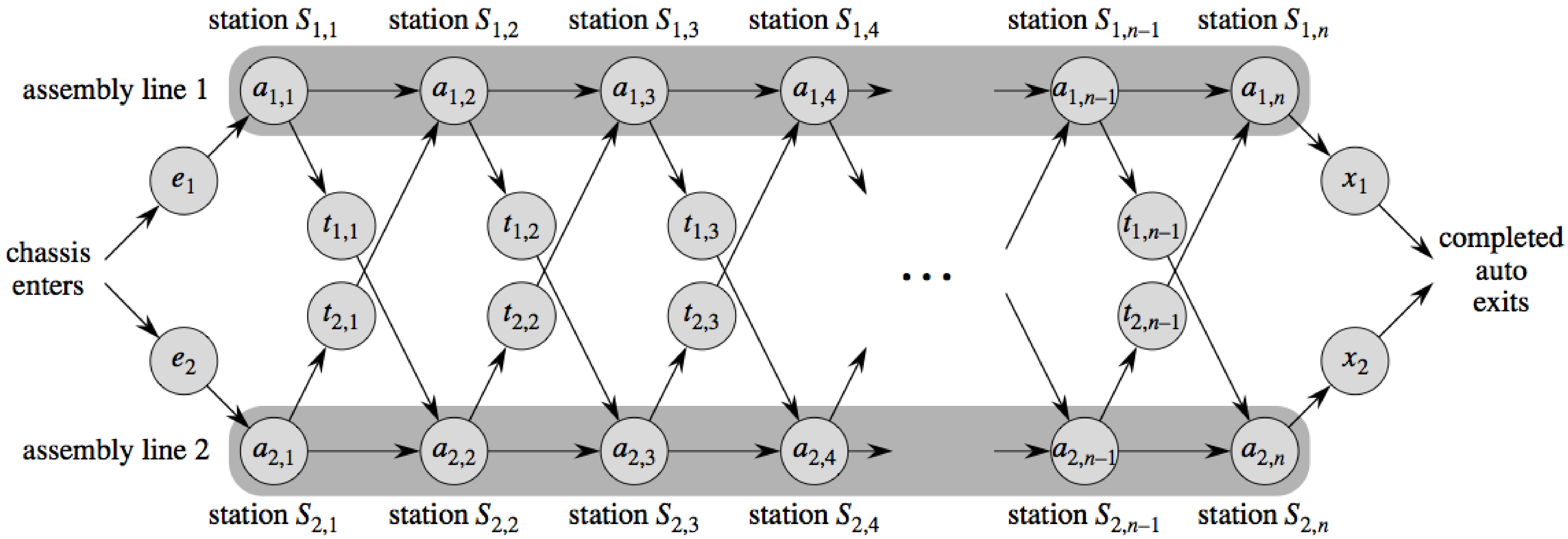
Programação Dinâmica

- Considera sistematicamente todas as decisões possíveis e **sempre** seleciona aquela que prova ser a melhor.
- Armazenando as consequências de todas as possíveis decisões até o momento e usando esta informação de forma sistemática, **a quantidade total de trabalho é minimizada**.
- Usada para problemas combinatórios (de otimização).

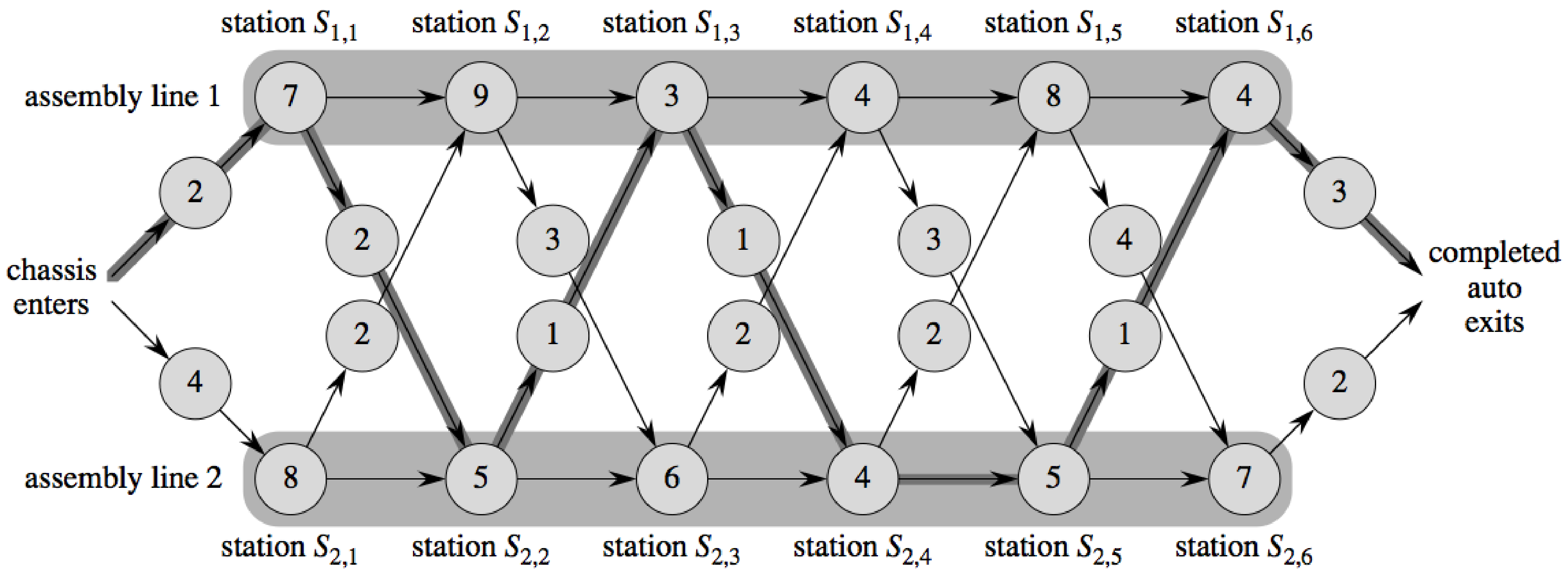
Passos

1. Garantir a propriedade de **subestrutura ótima**;
2. Obter uma **recursão**;
3. Fazer um algoritmo bottom-up para calcular o **valor ótimo**;
4. Obter a **solução ótima**.

Problema da linha de montagem



Problema da linha de montagem



Problema da linha de montagem

$a_{i,j}$: tempo de processamento da máquina **j** da linha **i**

$t_{i,j}$: tempo pra ir da máquina **j** da linha **i** para a outra linha

e_i: tempo para entrar na linha **i**

x_i: tempo para sair da linha **i**

Vamos definir também:

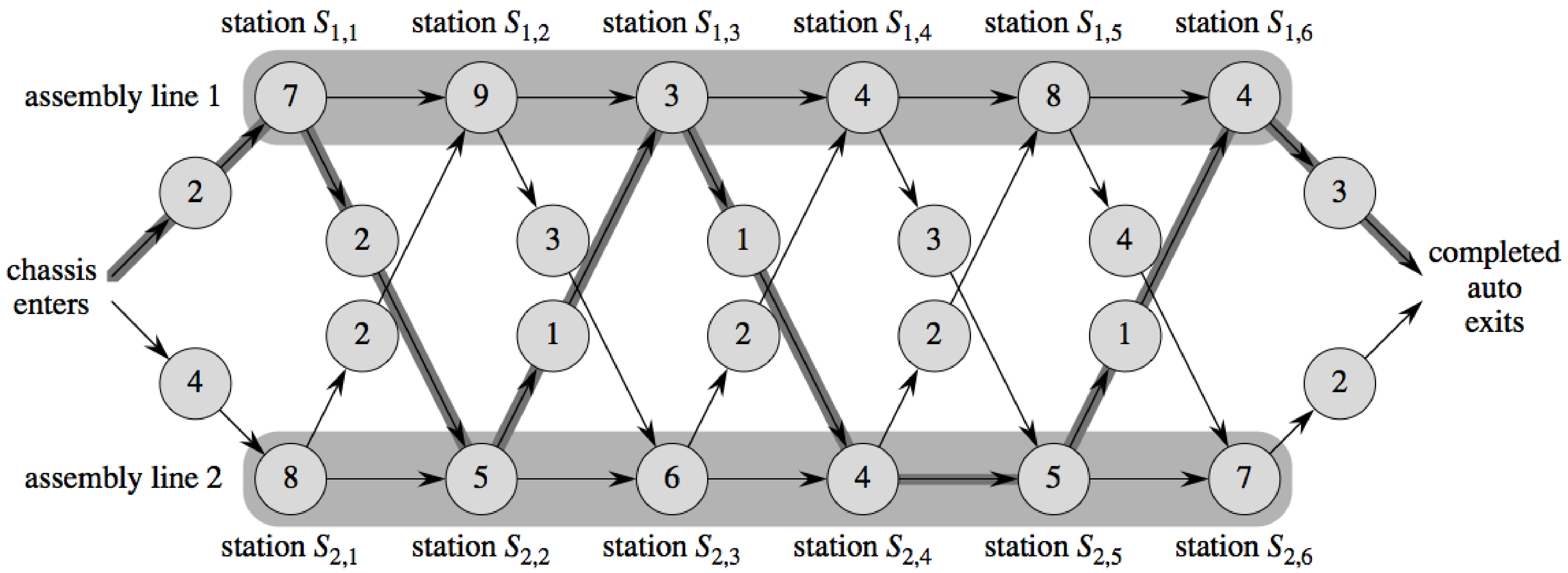
$f_i(j)$: menor tempo para levar um chassi desde a entrada até a estação $S_{i,j}$

f*: menor tempo total

$l_i(j)$: linha cuja estação **j-1** é usada como o caminho mais rápido através da estação $S_{i,j}$

l*: linha cuja estação **n** é usada como o caminho mais rápido através de toda a fábrica

Problema da linha de montagem



j	1	2	3	4	5	6
$f_1[j]$	9	18	20	24	32	35
$f_2[j]$	12	16	22	25	30	37

$f^* = 38$

j	2	3	4	5	6
$l_1[j]$	1	2	1	1	2
$l_2[j]$	1	2	1	2	2

$l^* = 1$

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

Algoritmo

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

$f_1[1] \leftarrow e_1 + a_{1,1}$
$f_2[1] \leftarrow e_2 + a_{2,1}$

Primeira estação

for $j \leftarrow 2$ to n
  do if $f_1[j-1] + a_{1,j} \leq f_2[j-1] + t_{2,j-1} + a_{1,j}$
    then $f_1[j] \leftarrow f_1[j-1] + a_{1,j}$
      $l_1[j] \leftarrow 1$
    else $f_1[j] \leftarrow f_2[j-1] + t_{2,j-1} + a_{1,j}$
      $l_1[j] \leftarrow 2$
  if $f_2[j-1] + a_{2,j} \leq f_1[j-1] + t_{1,j-1} + a_{2,j}$
    then $f_2[j] \leftarrow f_2[j-1] + a_{2,j}$
      $l_2[j] \leftarrow 2$
    else $f_2[j] \leftarrow f_1[j-1] + t_{1,j-1} + a_{2,j}$
      $l_2[j] \leftarrow 1$

Demais estações

if $f_1[n] + x_1 \leq f_2[n] + x_2$
  then $f^* = f_1[n] + x_1$
    $l^* = 1$
  else $f^* = f_2[n] + x_2$
    $l^* = 2$

Saída da linha

Recuperando o caminho ótimo

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 ” $j - 1$