

# Subsecuencia Común Más Larga

## Programación Dinámica

Sergio García Prado

10 de Noviembre de 2015

- 1 Introducción
  - Formulación del Problema
  - Definiciones
- 2 Aplicaciones
- 3 Programación Dinámica
- 4 Subsecuencia Común más larga
  - Enfoques Disponibles
  - Enfoque dinámico
- 5 Fin

# Section 1

## Introducción

## Subsection 1

### Formulación del Problema

# Formulación del Problema

Dadas dos secuencias de longitud arbitraria, nuestro objetivo es encontrar la subsecuencia común de mayor longitud entre ambas.

## Subsection 2

### Definiciones

# Secuencia

- Es una colección ordenada de elementos en la cual la repetición está permitida.
- Ejemplo: A, B, C, A, E, F

# Subsecuencia

- Es una secuencia que se obtiene a partir de otra de igual o mayor longitud mediante la supresión de algunos elementos manteniendo el orden de los restantes.
- Ejemplo: A, C, A es una subsecuencia de A, B, C, A, E, F.



## Section 2

# Aplicaciones

# Aplicaciones

- Secuenciación del ADN.
- Software de control de versiones (git).
- Comando diff.

## Section 3

# Programación Dinámica

# Programación Dinámica

- Patrón de diseño de algoritmos basado en la división del problema base en subproblemas de menor tamaño y complejidad que solo se resolverán una única vez.
- Solapamiento de Problemas.
- Subestructura Óptima.

# Solapamiento de Problemas

- Un problema tiene esta propiedad si se puede dividir en subproblemas de menor tamaño cuyos resultados se reutilizarán para resolver sucesivos subproblemas de nivel superior.
- Ejemplo: Sucesión de Fibonacci.

# Subestructura Óptima

- La solución óptima del problema contiene (o depende) de las soluciones óptimas a sus subproblemas..

## Section 4

# Subsecuencia Común más larga

## Subsection 1

### Enfoques Disponibles



# Enfoques Disponibles

- $N$  = Cantidad de secuencias.
- $n_i$  = Secuencia  $i$ -ésima.
- Boráz:  $O(2^{n_1} \sum_{i=2}^N n_i)$ .
- Programación Dinámica:  $O(N \prod_{i=1}^N n_i)$ .

## Subsection 2

### Enfoque dinámico

# Notación Utilizada

- Se expone el caso de dos secuencias.
- $X[0...m-1]$  e  $Y[0...n-1]$  secuencias de longitud  $m$  y  $n$  respectivamente.
- $L[0...m][0...n]$  matriz de tamaño  $m \times n$ , para almacenar los resultados de cada subproblema.
- Sea  $S$  una secuencia de longitud  $l$  e  $i \in (1, l)$ , entonces  $S_i$  es la correspondiente a los  $i$  primeros elementos.

Ejemplo:  $X = abcde$

$X_1 = a$ ,  $X_3 = abc$  y  $X_5 = abcde = X$

# Fórmula

$$LCS(X_i, Y_j) = \begin{cases} 0 & \text{if } i = 0 \vee j = 0 \\ LCS(X_{i-1}, Y_{j-1}) + 1 & \text{if } x_i = y_j \\ \max(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

# Funcionamiento

- Rellenar matriz  $L$  a partir de función LCS.
- Recorrer  $L$  desde  $L_{m-1n-1}$  hasta  $L_{00}$  guardando los elementos iguales.

# Ejemplo de funcionamiento 1

- $X = abcde$ ,  $m = 5$
- $Y = aert$ ,  $n = 4$

## Ejemplo de funcionamiento 2

	$\emptyset$	a	b	c	d	e
$\emptyset$						
a						
e						
r						
t						

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1				
e	0					
r	0					
t	0					

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0					
e	0					
r	0					
t	0					

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1			
e	0					
r	0					
t	0					

# Ejemplo de funcionamiento 3

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1				
r	0					
t	0					

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0					

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0					
t	0					

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0	1	1	1	1	2



# Ejemplo de funcionamiento 4

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0	1	1	1	1	2

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0	1	1	1	1	2

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0	1	1	1	1	2

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0	1	1	1	1	2

# Ejemplo de funcionamiento 5

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0	1	1	1	1	2

	$\emptyset$	a	b	c	d	e
$\emptyset$	0	0	0	0	0	0
a	0	1	1	1	1	1
e	0	1	1	1	1	2
r	0	1	1	1	1	2
t	0	1	1	1	1	2

## Ejemplo de funcionamiento 6

- Para las secuencias  $X = abcde$ ,  $Y = aert$  el resultado es:
- $LCS = ae$  de longitud 2

# Pseudocódigo

---

**Algorithm 1** lcs
 

---

```

1: function LCS( $X, Y, m, n$ )
2:    $X$  : secuencia de  $m$  elementos  $[0..m-1]$ 
3:    $Y$  : secuencia de  $n$  elementos  $[0..n-1]$ 
4:    $m$  : longitud de  $X$ 
5:    $n$  : longitud de  $Y$ 
6:
7:    $L \leftarrow \text{array}[0..m][0..n]$   $\triangleright O(1)$ 
8:
9:   for  $i \leftarrow 0, i < m + 1, i++$  do  $\triangleright O(m * n)$ 
10:    for  $j \leftarrow 0, j < n + 1, j++$  do  $\triangleright O(n)$ 
11:      if  $i = 0 \vee j = 0$  then  $\triangleright O(1)$ 
12:         $L[i][j] \leftarrow 0$   $\triangleright O(1)$ 
13:      else if  $X[i - 1] = Y[j - 1]$  then  $\triangleright O(1)$ 
14:         $L[i][j] \leftarrow L[i - 1][j - 1] + 1$   $\triangleright O(1)$ 
15:      else  $\triangleright O(1)$ 
16:         $L[i][j] \leftarrow \max(L[i - 1][j], L[i][j - 1])$   $\triangleright O(1)$ 
17:      end if
18:    end for
19:  end for
20:
21:   $\text{index} \leftarrow L[m][n]$   $\triangleright O(1)$ 
22:   $\text{LCS} \leftarrow \text{array}[0..\text{index} - 1]$   $\triangleright O(1)$ 
23:
24:   $i \leftarrow m$   $\triangleright O(1)$ 
25:   $j \leftarrow n$   $\triangleright O(1)$ 
26:  while  $i > 0 \wedge j > 0$  do  $\triangleright O(\frac{n+m}{2})$ 
27:    if  $X[i - 1] = Y[j - 1]$  then  $\triangleright O(1)$ 
28:       $\text{LCS}[\text{index} - 1] \leftarrow X[i - 1]$   $\triangleright O(1)$ 
29:       $\text{index} \leftarrow \text{index} - 1$   $\triangleright O(1)$ 
30:       $j \leftarrow j - 1$   $\triangleright O(1)$ 
31:       $i \leftarrow i - 1$   $\triangleright O(1)$ 
32:    else if  $L[i - 1][j] > L[i][j - 1]$  then  $\triangleright O(1)$ 
33:       $i \leftarrow i - 1$   $\triangleright O(1)$ 
34:    else
35:       $j \leftarrow j - 1$   $\triangleright O(1)$ 
36:    end if
37:  end while
38:
39:  return  $\text{LCS}$   $\triangleright O(1)$ 
40: end function

```

---

# Error encontrado

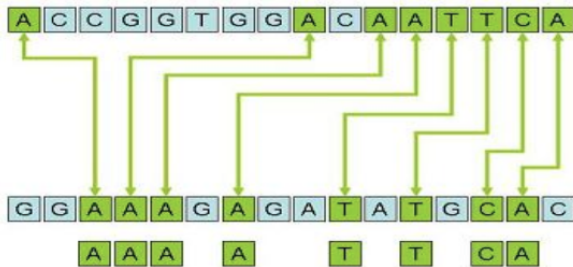


Figura 2:

# Error encontrado

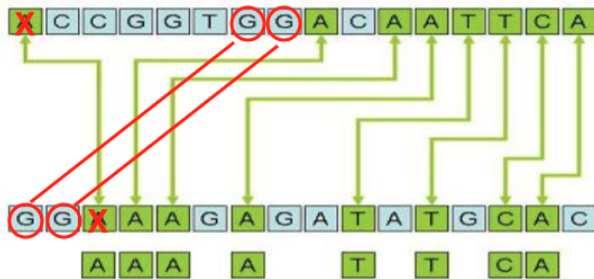


Figura 3:

## Section 5

Fin