TÉCNICO LISBOA

Análise e Síntese de Algoritmos

Programação dinâmica

CLRS Cap. 15

Prof. Pedro T. Monteiro

IST - Universidade de Lisboa

2024/2025

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica [CLRS, Cap.15]
  - Algoritmos greedy [CLRS, Cap.16]
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes [CLRS, Cap.23]
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais
  - Emparelhamento de Cadeias de Caracteres [CLRS, Cap.32]
  - Complexidade Computacional [CLRS, Cap.34]

Resumo

Maior Sub-Sequência Comum

Realizar trocos

Maior Palíndromo

Multiplicação de cadeias de matrizes

Ex 3 - Maior Sub-Sequência Comum

Definição

- Dada uma sequência $X = \langle x_1, \ldots, x_n \rangle$, uma sequência $Z = \langle z_1, \ldots, z_k \rangle$ é uma sub-sequência de X se existe uma sequência estritamente crescente $\langle i_1, \ldots, i_k \rangle$ tal que para todo o $j = 1, \ldots, k, x_{i_j} = z_j$

- Dadas as sequências $X = \langle x_1, \ldots, x_n \rangle$ e $Y = \langle y_1, \ldots, y_m \rangle$, Z é uma sub-sequência comum se Z é sub-sequência de X e de Y
  Obs: $X_i = \langle x_1, \ldots, x_i \rangle$

Objectivo
Encontrar sub-sequência comum de maior comprimento (LCS) entre duas sequências X e Y

Exemplo

- Um caso concreto:
  - $X = \langle \mathbf{abefcghd} \rangle$
  - $Y = \langle e\mathbf{agbcfdh} \rangle$
  - $Z = \langle \mathbf{abcd} \rangle$ é sub-sequência comum de X e Y
- Uma solução exaustiva é impraticável:
  - Considerar inclusão (ou não) de cada caracter de X e de Y
  - Total de sub-sequências em X: 2^n
  - Total de sub-sequências em Y: 2^m
  - Total de casos a analisar: 2^{n+m}
  - Impraticável para valores elevados de n e m

Exemplo

- Alguns resultados formais:
  - Sejam $X = \langle x_1, \ldots, x_n \rangle$ e $Y = \langle y_1, \ldots, y_m \rangle$ duas sequências, e seja $Z = \langle z_1, \ldots, z_k \rangle$ uma LCS de X e Y
  - Se $x_n = y_m$, então $z_k = x_n = y_m$ e Z_{k-1} é LCS de X_{n-1} e Y_{m-1}
  - Se $x_n \neq y_m$, então:
    - se $z_k \neq x_n$ implica que Z é LCS de X_{n-1} e Y
    - se $z_k \neq y_m$ implica que Z é LCS de X e Y_{m-1}
- Abordagem:
  - Se $x_n = y_m$, encontrar LCS W de X_{n-1} e Y_{m-1}
    - Adicionar $x_n = y_m$ a W permite obter Z
  - Se $x_n \neq y_m$, encontrar LCSs W_1 e W_2 para: X_{n-1} e Y, e para X e Y_{m-1}
    - Escolher a maior LCS $max(W_1, W_2)$

Formulação

- $c[i,j]$: comprimento da LCS para as sequências X_i e Y_j
- Formulação:

$$c[i,j] = \begin{cases} 0 & \text{se } i = 0 \text{ ou } j = 0 \\ c[i-1,j-1]+1 & \text{se } i,j > 0 \text{ e } x_i = y_j \\ max(c[i,j-1], c[i-1,j]) & \text{se } i,j > 0 \text{ e } x_i \neq y_j \end{cases}$$

- Tempo de execução: $O(n\,m)$

Exemplo

	-	a	b	e	f	c	g	h	d
-									
e									
a									
g									
b									
c									
f									
d									
h									

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0								
a	0								
g	0								
b	0								
c	0								
f	0								
d	0								
h	0								

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0								
g	0								
b	0								
c	0								
f	0								
d	0								
h	0								

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0								
b	0								
c	0								
f	0								
d	0								
h	0								

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0	1							
b	0								
c	0								
f	0								
d	0								
h	0								

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0	1							
b	0	1							
c	0								
f	0								
d	0								
h	0								

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0	1							
b	0	1							
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1							
g	0	1							
b	0	1							
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1	1						
g	0	1							
b	0	1							
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1	1						
g	0	1	1						
b	0	1	2						
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1	1						
g	0	1	1						
b	0	1	2						
c	0	1	2						
f	0	1	2						
d	0	1	2						
h	0	1	2						

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1					
a	0	1	1	1					
g	0	1	1	1					
b	0	1	2	2					
c	0	1	2	2					
f	0	1	2	2					
d	0	1	2	2					
h	0	1	2	2					

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1				
a	0	1	1	1	1				
g	0	1	1	1	1				
b	0	1	2	2	2				
c	0	1	2	2	2				
f	0	1	2	2					
d	0	1	2	2					
h	0	1	2	2					

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1				
a	0	1	1	1	1				
g	0	1	1	1	1				
b	0	1	2	2	2				
c	0	1	2	2	2				
f	0	1	2	2	3				
d	0	1	2	2					
h	0	1	2	2					

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1	1	1	1	1
a	0	1	1	1	1	1	1	1	1
g	0	1	1	1	1	1	2	2	2
b	0	1	2	2	2	2	2	2	2
c	0	1	2	2	2	3	3	3	3
f	0	1	2	2	3	3	3	3	3
d	0	1	2	2	3	3	3	3	4
h	0	1	2	2	3	3	3	4	4

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1	1	1	1	1
a	0	1	1	1	1	1	1	1	1
g	0	1	1	1	1	1	2	2	2
b	0	1	2	2	2	2	2	2	2
c	0	1	2	2	2	3	3	3	3
f	0	1	2	2	3	3	3	3	3
d	0	1	2	2	3	3	3	3	4
h	0	1	2	2	3	3	3	4	4

LCS: abch (de tamanho 4)

Ex 3 - Maior Sub-Sequência Comum

Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1	1	1	1	1
a	0	1	1	1	1	1	1	1	1
g	0	1	1	1	1	1	2	2	2
b	0	1	2	2	2	2	2	2	2
c	0	1	2	2	2	3	3	3	3
f	0	1	2	2	3	3	3	3	3
d	0	1	2	2	3	3	3	3	4
h	0	1	2	2	3	3	3	4	4

LCS: abcd (de tamanho 4)

Ex 4 - Realizar trocos

Definição

- Dado um conjunto de moedas, denominadas $1, \ldots, n$, com valores $d_1, \ldots, d_n$, calcular o menor número de moedas cuja soma de valores é T
  - Número ilimitado de moedas de cada denominação
- Solução greedy pode não funcionar:
  - $d_1 = 1; d_2 = 5; d_3 = 20; d_4 = 25$
  - Troco de 40?!
- Solução baseada em programação dinâmica

Ex 4 - Realizar trocos

Formulação

- $c[i,j]$: Menor número de moedas necessárias para pagar j unidades, $0 \leq j \leq T$, utilizando apenas moedas com denominação entre 1 e i, $1 \leq i \leq n$

- Objectivo é calcular $c[n, T]$
- Formulação:

$$c[i,j] = \begin{cases} 0 & \text{se } i > 0 \text{ e } j = 0 \\ +\infty & \text{se } i = 0 \text{ ou } j < 0 \\ \min(c[i-1,j], c[i,j-d_i]+1) & \text{caso contrário} \end{cases}$$

- Tempo de execução: $O(n\,T)$

Ex 5 - Maior Sub-Sequência Palíndromo

Definição

- Dada uma sequência $X = \langle x_1, \ldots, x_n \rangle$, calcular a maior sub-sequência $Z = \langle z_1, \ldots, z_k \rangle$ de X tal que Z seja um palíndromo

Exemplo

- $X = \langle \mathbf{abcfdbca} \rangle$
- $Z = \langle \mathbf{abdba} \rangle$

Formulação

- l[i,j]: tamanho da maior sub-sequência palíndromo de X (entre os índices i e j, tal que $j \geq i$)

$$l[i,j] = \begin{cases} 1 & \text{i = j} \\ l[i+1,j-1]+2 & \text{se } X_i = X_j \\ \max(l[i,j-1], l[i+1,j]) & \text{caso contrário} \end{cases}$$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \ j	1	2	3	4	5	6	7	8
1	1							
2		1						
3			1					
4				1				
5					1			
6						1		
7							1	
8								1

$l[i,j] = 1, \quad i = j$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1						
2		1						
3			1					
4				1				
5					1			
6						1		
7							1	
8								1

$max(l[i, j-1], l[i+1, j]), \quad X_i \neq X_j$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1						
2		1	1					
3			1	1				
4				1	1			
5					1	1		
6						1	1	
7							1	1
8								1

$max(l[i, j-1], l[i+1, j]), \quad X_i \neq X_j$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1	1					
2		1	1	1				
3			1	1	1			
4				1	1	1		
5					1	1	1	
6						1	1	1
7							1	1
8								1

$max(l[i, j-1], l[i+1, j]), \quad X_i \neq X_j$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1	1	1				
2		1	1	1	1			
3			1	1	1	1		
4				1	1	1	1	
5					1	1	1	1
6						1	1	1
7							1	1
8								1

$max(l[i, j-1], l[i+1, j]), \quad X_i \neq X_j$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1	1	1	1			
2		1	1	1	1	3		
3			1	1	1	1	3	
4				1	1	1	1	1
5					1	1	1	1
6						1	1	1
7							1	1
8								1

$l[i+1, j-1]+2, \quad X_i = X_j$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1	1	1	1	3		
2		1	1	1	1	3	3	
3			1	1	1	1	3	3
4				1	1	1	1	1
5					1	1	1	1
6						1	1	1
7							1	1
8								1

$max(l[i, j-1], l[i+1, j]), \quad X_i \neq X_j$

Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1	1	1	1	3	3	
2		1	1	1	1	3	3	3
3			1	1	1	1	3	3
4				1	1	1	1	1
5					1	1	1	1
6						1	1	1
7							1	1
8								1

$max(l[i, j-1], l[i+1, j]), \quad X_i \neq X_j$

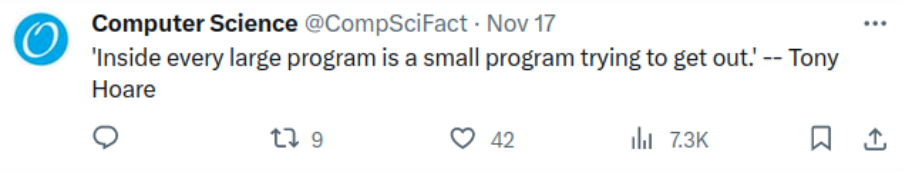
Ex 5 - Maior Sub-Sequência Palíndromo

Exemplo

i	1	2	3	4	5	6	7	8
X_i	a	b	c	f	d	b	c	a

i \j	1	2	3	4	5	6	7	8
1	1	1	1	1	1	3	3	5
2		1	1	1	1	3	3	3
3			1	1	1	1	3	3
4				1	1	1	1	1
5					1	1	1	1
6						1	1	1
7							1	1
8								1

$l[i+1, j-1]+2, \quad X_i = X_j$



Ex 6 - Multiplicação de cadeias de matrizes

TÉCNICO LISBOA

Definição

- $A_1, A_2, \ldots, A_n$ tal que A_i tem dimensões $(l_i \times c_i)$
- **Objectivo:** colocar parêntesis na cadeia de produtos de matrizes $A_1 \times A_2 \times \ldots \times A_n$, tal que o número de multiplicações escalares é minimizado

Observações

- Tempo para multiplicar as n matrizes é dominado pelo tempo para realizar as multiplicações escalares necessárias
  - Para multiplicar duas matrizes $(r \times s)$ e $(s \times t)$, o número de multiplicações escalares é: $r \times s \times t$
- Número de produtos depende do modo como os produtos de matrizes são organizados
  - Colocação de parêntesis define organização da multiplicação de matrizes

Ex 6 - Multiplicação de cadeias de matrizes

TÉCNICO LISBOA

Exemplo

$A(13 \times 5)$; $B(5 \times 89)$; $C(89 \times 3)$; $D(3 \times 34)$

- $(((A \times B) \times C) \times D)$:
  - $13 \times 5 \times 89 + 13 \times 89 \times 3 + 13 \times 3 \times 34 = 10.582$ produtos
- $((A \times B) \times (C \times D))$:
  - $13 \times 5 \times 89 + 89 \times 3 \times 34 + 13 \times 89 \times 34 = 54.201$ produtos
- $((A \times (B \times C)) \times D)$:
  - $5 \times 89 \times 3 + 13 \times 5 \times 3 + 13 \times 3 \times 34 = 2.856$ produtos !
- $(A \times ((B \times C) \times D))$:
  - $5 \times 89 \times 3 + 5 \times 3 \times 34 + 13 \times 5 \times 34 = 4.055$ produtos
- $(A \times (B \times (C \times D)))$:
  - $89 \times 3 \times 34 + 5 \times 89 \times 34 + 13 \times 5 \times 34 = 26.418$ produtos

Ex 6 - Multiplicação de cadeias de matrizes

TÉCNICO LISBOA

Observação

- Número de colocações possíveis de parêntesis cresce exponencialmente com número de matrizes:

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

$$P(n) = C(n-1)$$
$$C(n) = \frac{C_n^{2n}}{n+1} = \Omega(4^n/n^{3/2})$$

Características da Solução Óptima

Seja $A_{1\ldots n}$ solução com colocação óptima de parêntesis

- Admitir solução óptima com parêntesis em k, $A_{1\ldots k}A_{k+1\ldots n}$
- Facto:
  - Colocação de parêntesis para $A_{1\ldots k}$ é também óptima
- Porquê?
  - Caso contrário seria possível encontrar uma melhor colocação de parêntesis para $A_{1\ldots k}$ e portanto para $A_{1\ldots n}$
- Conclusão:
  - Solução óptima para o problema da colocação de parêntesis é composta por soluções óptimas para os seus sub-problemas

Solução Recursiva

- m[i,j]: menor número de multiplicações escalares necessário para calcular multiplicação cadeias matrizes $A_{i\ldots j}$
- Solução óptima para $A_{1\ldots n}$ é $m[1,n]$
- $i = j$: $m[i,j] = 0$
- $i < j$:
  - Admitir que solução óptima coloca parêntesis em k:
    - $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j$
  - Mas qual é o valor de k?
    - certamente k tem valor entre i e $j-1$
    - considerar todos os valores de k possíveis
- s[i,j]: define colocação óptima de parêntesis entre i e j

Solução Recursiva

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

$$s[i,j] = k \text{ sse } m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j$$

Cálculo dos valores de $m[i,j]$

- Número de sub-problemas distintos:
  - 1 para cada $1 \le i \le j \le n$
  - número de problemas: $\Theta(n^2)$
- Problema:
  - solução recursiva requer tempo exponencial
  - resolução repetida dos mesmos subproblemas
- Solução:
  - solução construtiva (bottom-up)
  - tempo de execução: $O(n^3)$

Memorização (Memoization)

- Permite obter tempo de execução das soluções dos sub-problemas, mas utilizando abordagem recursiva (top-down)
  - É necessário memorizar resultados de sub-problemas já resolvidos
- Exemplo: caminhos mais curtos num DAG, com DFS
- Exemplo: cálculo das combinações
  - Não calcular todo o triângulo de Pascal
  - Calcular apenas as entradas necessárias
  - Calcular cada entrada apenas 1 vez

Dúvidas?