

Algoritmos

Pedro Hokama

- [cirs] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
 - [timr] Algorithms Illuminated Series, Tim Roughgarden
 - Desmistificando Algoritmos, Thomas H. Cormen.
 - Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
 - Stanford Algorithms
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V>
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZ17A5HEK6>
 - Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
 - Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420
- Qualquer erro é de minha responsabilidade.

1 / 23

2 / 23

Problema do Alinhamento de Sequências

Sejam $X = x_1x_2 \dots x_m$ e $Y = y_1y_2 \dots y_n$ duas sequências sobre um alfabeto Σ . Deseja-se formar X' e Y' inserindo lacunas (-) em X e/ou Y até que $|X'| = |Y'| = L$, de modo a minimizar a penalidade total

$$C(X', Y') = \sum_{k=1}^L c(X'_k, Y'_k),$$

onde

$$c(a, b) = \begin{cases} 0, & \text{se } a = b \in \Sigma, \\ \beta, & \text{se } a \neq b \in \Sigma, \\ \alpha, & \text{se } a = - \text{ ou } b = -. \end{cases}$$

Considere as sequências $X = \text{ACGT}$ e $Y = \text{AGA}$, com penalidade de lacuna $\alpha = 2$ e de mismatch $\beta = 1$.

Um possível alinhamento seria:

$$\begin{array}{cccc} X' : & A & C & G & T \\ Y' : & A & G & A & - \end{array}$$

com custo total:

$$C = c(A, A) + c(C, G) + c(G, A) + c(T, -) = 0 + 1 + 1 + 2 = 4.$$

3 / 23

4 / 23

Subestrutura Ótima do Problema de Alinhamento

Entretanto um alinhamento melhor é:

$$\begin{array}{cccc} X' : & A & C & G & T \\ Y' : & A & - & G & A \end{array}$$

O custo total é:

$$C(X', Y') = c(A, A) + c(C, -) + c(G, G) + c(T, A) = 0 + 2 + 0 + 1 = 3.$$

Logo, o custo mínimo de alinhamento é 3.

Sejam $X = x_1x_2 \dots x_m$ e $Y = y_1y_2 \dots y_n$. O alinhamento ótimo de X e Y possui a seguinte propriedade:

$$D(i, j) = \text{custo mínimo de alinhar } x_1x_2 \dots x_i \text{ com } y_1y_2 \dots y_j.$$

Então, se conhecemos a solução ótima dos prefixos menores, podemos construir a solução ótima de prefixos maiores. Em particular, o alinhamento ótimo de X e Y termina com uma das três situações:

- ❶ x_i alinhado com y_j ;
- ❷ x_i alinhado com uma lacuna;
- ❸ y_j alinhado com uma lacuna.

Essa decomposição garante a **subestrutura ótima**.

5 / 23

6 / 23

Recorrência do Problema de Alinhamento

A partir da subestrutura ótima, obtemos a seguinte recorrência:

$$D(i, j) = \begin{cases} 0, & \text{se } i = j = 0, \\ i\alpha, & \text{se } j = 0, \\ j\alpha, & \text{se } i = 0, \\ \min \begin{cases} D(i-1, j-1) + (0 \text{ se } x_i = y_j, \text{ senão } \beta), \\ D(i-1, j) + \alpha, \\ D(i, j-1) + \alpha. \end{cases} & \text{caso contrário} \end{cases}$$

O valor $D(m, n)$ fornece o custo mínimo de alinhamento entre X e Y .

Algoritmo 1: Alinhamento-Seq(X, Y, α, β)

```

1   $D[0, 0] = 0$ ;
2  para  $i = 1, 2, \dots, m$  faça
3       $D[i, 0] = i \cdot \alpha$ ;
4  para  $j = 1, 2, \dots, n$  faça
5       $D[0, j] = j \cdot \alpha$ ;
6  para  $i = 1, 2, \dots, m$  faça
7      para  $j = 1, 2, \dots, n$  faça
           $D[i, j] = \min \begin{cases} D[i-1, j-1] + \begin{cases} 0, & \text{se } x_i = y_j \\ \beta, & \text{caso contrário} \end{cases}, \\ D[i-1, j] + \alpha, \\ D[i, j-1] + \alpha \end{cases};$ 
8
9  devolva  $D[m, n]$ ;
```

7 / 23

8 / 23

Multiplicação de cadeia de Matrizes

- Em diversas áreas da computação, muitos problemas requerem multiplicação de matrizes como um de seus procedimentos.
- Tratamento de imagem, Aprendizado de Máquinas, etc.
- Um problema comum é dado uma sequência de matrizes $\langle A_1, A_2, \dots, A_n \rangle$, encontrar o produto $A_1 A_2 \dots A_n$.
- A multiplicação de matrizes é uma operação associativa. O que significa que podem ser parentizadas em qualquer ordem. e. g.,

$$A_1 A_2 A_3$$

pode ser calculado

$$((A_1 A_2) A_3)$$

ou também

$$(A_1 (A_2 A_3))$$

9 / 23

$$\begin{pmatrix} 1 \cdot 7 + 2 \cdot 10 & 1 \cdot 8 + 2 \cdot 11 & 1 \cdot 9 + 2 \cdot 12 \\ 3 \cdot 7 + 4 \cdot 10 & 3 \cdot 8 + 4 \cdot 11 & 3 \cdot 9 + 4 \cdot 12 \\ 5 \cdot 7 + 6 \cdot 10 & 5 \cdot 8 + 6 \cdot 11 & 5 \cdot 9 + 6 \cdot 12 \end{pmatrix} = \begin{pmatrix} 27 & 30 & 33 \\ 61 & 68 & 75 \\ 95 & 106 & 117 \end{pmatrix}.$$

Agora, multipliquemos $A \times B$ por C :

$$(A \times B) \times C = \begin{pmatrix} 27 & 30 & 33 \\ 61 & 68 & 75 \\ 95 & 106 & 117 \end{pmatrix} \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix} =$$

$$\begin{pmatrix} 27 \cdot 13 + 30 \cdot 15 + 33 \cdot 17 & 27 \cdot 14 + 30 \cdot 16 + 33 \cdot 18 \\ 61 \cdot 13 + 68 \cdot 15 + 75 \cdot 17 & 61 \cdot 14 + 68 \cdot 16 + 75 \cdot 18 \\ 95 \cdot 13 + 106 \cdot 15 + 117 \cdot 17 & 95 \cdot 14 + 106 \cdot 16 + 117 \cdot 18 \end{pmatrix} = \begin{pmatrix} 1362 & 1452 \\ 3088 & 3392 \\ 4814 & 5132 \end{pmatrix}.$$

Quantas multiplicações foram necessárias?

11 / 23

Sejam as matrizes A , B e C definidas por

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \quad B = \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}, \quad C = \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix}.$$

Primeiro, calculemos $A \times B$:

$$A \times B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} =$$

$$\begin{pmatrix} 1 \cdot 7 + 2 \cdot 10 & 1 \cdot 8 + 2 \cdot 11 & 1 \cdot 9 + 2 \cdot 12 \\ 3 \cdot 7 + 4 \cdot 10 & 3 \cdot 8 + 4 \cdot 11 & 3 \cdot 9 + 4 \cdot 12 \\ 5 \cdot 7 + 6 \cdot 10 & 5 \cdot 8 + 6 \cdot 11 & 5 \cdot 9 + 6 \cdot 12 \end{pmatrix} = \begin{pmatrix} 27 & 30 & 33 \\ 61 & 68 & 75 \\ 95 & 106 & 117 \end{pmatrix}.$$

10 / 23

Agora iremos calcular $A \times (B \times C)$. Primeiro, calculemos $B \times C$:

$$B \times C = \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix} =$$

$$\begin{pmatrix} 7 \cdot 13 + 8 \cdot 15 + 9 \cdot 17 & 7 \cdot 14 + 8 \cdot 16 + 9 \cdot 18 \\ 10 \cdot 13 + 11 \cdot 15 + 12 \cdot 17 & 10 \cdot 14 + 11 \cdot 16 + 12 \cdot 18 \end{pmatrix} = \begin{pmatrix} 364 & 388 \\ 499 & 532 \end{pmatrix}.$$

Agora, multipliquemos A por $B \times C$:

$$A \times (B \times C) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 364 & 388 \\ 499 & 532 \end{pmatrix} =$$

$$\begin{pmatrix} 1 \cdot 364 + 2 \cdot 499 & 1 \cdot 388 + 2 \cdot 532 \\ 3 \cdot 364 + 4 \cdot 499 & 3 \cdot 388 + 4 \cdot 532 \\ 5 \cdot 364 + 6 \cdot 499 & 5 \cdot 388 + 6 \cdot 532 \end{pmatrix} = \begin{pmatrix} 1362 & 1452 \\ 3088 & 3392 \\ 4814 & 5132 \end{pmatrix}.$$

E agora? Quantas multiplicações foram necessárias?

12 / 23

- Note que é uma diferença de 50%.
- Mas em matrizes maiores, a diferença seria ainda maior.

Algoritmo 2: MatrizMultiply(A, B)

Entrada: Duas matrizes $A(p, q)$ e $B(q, r)$

Saída: o produto $A \times B$

```

1 para  $i = 1, 2, \dots, p$  faça
2   para  $j = 1, 2, \dots, r$  faça
3      $c_{ij} = 0$ ;
4     para  $k = 1, 2, \dots, q$  faça
5        $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$ ;
6 devolva  $C$ ;
```

- Qual a complexidade desse algoritmo?
- $O(pqr)^1$.

¹existem algoritmos mais eficientes, como o algoritmo de Strassen, mas esse não é o foco agora

13 / 23

- Queremos então descobrir qual a melhor ordem para fazer as multiplicações.

Problema da Multiplicação de Cadeia de Matrizes

Dada uma sequência $\langle A_1, A_2, \dots, A_n \rangle$. expresse uma parentização do produto $A_1 A_2 \dots A_n$ de forma que o número de multiplicações escalares seja mínimo.

15 / 23

- Por exemplo matrizes de dimensões $A(10, 100)$, $B(100, 5)$ e $C(5, 50)$
- Calculando $(A B) C$
- Para calcular $AB = Z(10, 5)$, fazemos $10 \cdot 100 \cdot 5 = 5000$ multiplicações.
- Para calcular ZC , fazemos $10 \cdot 5 \cdot 50 = 2500$.
- Totalizando 7500 multiplicações
- Para calcular $A (B C)$,
- Para calcular $BC = X(100, 50)$, fazemos $100 \cdot 5 \cdot 50 = 25000$
- Para calcular AX , fazemos $10 \cdot 100 \cdot 50 = 50000$.
- Totalizando 75000 multiplicações!
- a diferença é entre 7500 multiplicações e 75000 multiplicações. 10x mais rápido.
- Mas é comum termos matrizes muito maiores, e uma cadeia muito maior de matrizes.

14 / 23

- Qual o número de possíveis soluções?
- Seja $P(n)$ o número de formas de parentizar um problema com n matrizes.

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

- $\Omega(2^n)$
- Vamos então usar programação dinâmica
 - ▶ Caracterizar a estrutura de uma solução ótima
 - ▶ Definir recursivamente o valor de uma solução
 - ▶ Calcular o valor de uma solução ótima
 - ▶ Construir a solução ótima com as informações calculadas.

16 / 23

- Vamos denotar $A_{i...j}$ a matriz que resulta da multiplicação da cadeia $A_i A_{i+1} \dots A_j$.
- Essa multiplicação pode ser dividida em duas partes $A_{i...k}$ e $A_{k+1...j}$.
- Qual é o custo (quantidade de multiplicações) para obter $A_{i...j}$ considerando essa divisão?
- É o custo de obter $A_{i...k}$ mais o custo de obter $A_{k+1...j}$ mais o custo de multiplicar essas duas.
- Subestrutura ótima: uma parentização ótima de $A_i A_{i+1} \dots A_j$ inclui alguma divisão $A_i A_{i+1} \dots A_k$ e $A_{k+1} \dots A_j$.
- Necessariamente $A_i A_{i+1} \dots A_k$ deve ser parentizado de maneira ótima, senão poderíamos reduzir o custo total.

17 / 23

- Seja $m[i, j]$ o menor custo para obter $A_{i...j}$. Na solução ótima existe um k^* tal que:

$$m[i, j] = m[i, k^*] + m[k^* + 1, j] + p_{i-1} p_{k^*} p_j$$

- Obviamente não sabemos qual é o k^* mas podemos testar todos.

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

18 / 23

Algoritmo 3: Matrix-Chain-Order(p)

Entrada: A sequência de dimensões das matrizes

Saída: o custo da solução ótima

```

1 para  $i = 1, 2, \dots, n$  faça
2    $m[i, i] = 0$ ;
3 //  $l$  é o comprimento da cadeia
4 para  $l = 2, \dots, n$  faça
5   para  $i = 1, n - l + 1$  faça
6      $j = i + l - 1$ ;
7      $m[i, j] = \infty$ ;
8     para  $k = 1, \dots, j - 1$  faça
9        $q = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j$ ;
10      se  $q < m[i, j]$  então
11         $m[i, j] = q$ ;
12 devolva  $C$ ;
```

19 / 23

- Para obter a solução ótima, poderíamos reconstruí-la.
- Mas iremos adicionar uma matriz s que irá ajudar nessa tarefa.

20 / 23

Algoritmo 4: Matrix-Chain-Order(p)

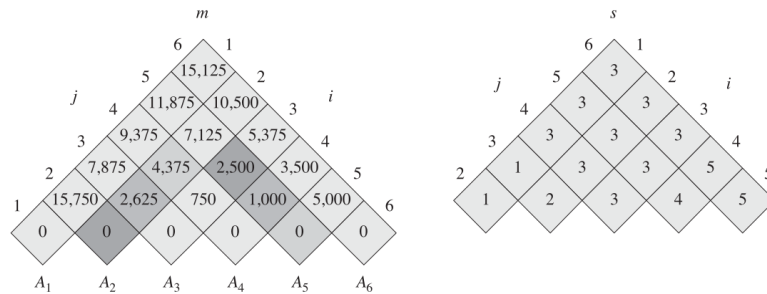
Entrada: A sequência de dimensões das matrizes

Saída: o custo da solução ótima

```
1 para  $i = 1, 2, \dots, n$  faça
2    $m[i, i] = 0$ ;
3 //  $l$  é o comprimento da cadeia
4 para  $l = 2, \dots, n$  faça
5   para  $i = 1, n - l + 1$  faça
6      $j = i + l - 1$ ;
7      $m[i, j] = \infty$ ;
8     para  $k = i, \dots, j - 1$  faça
9        $q = m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j$ ;
10      se  $q < m[i, j]$  então
11         $m[i, j] = q$ ;
12         $s[i, j] = k$ ;
13 devolva  $C$ ;
```

- Considere agora o seguinte exemplo:

matriz	A_1	A_2	A_3	A_4	A_5	A_6
dimensão	30, 35	35, 15	15, 5	5, 10	10, 20	20, 25



- Qual a complexidade desse algoritmo?