Projeto e Análise de Algoritmos

Aula 08 – Maior Subsequência Comum (LCS)

Edirlei Soares de Lima
<edirlei@iprj.uerj.br>

Problema

- **Subsequência:** sequência de caracteres não necessariamente contínuos, retirados de uma cadeia em ordem.
  - Exemplo: **AAAG** é subsequência da cadeia CG**A**T**AA**TT**G**AGA

- **Problema:** Dadas duas sequências $X = \{x_1, x_2, \ldots, x_m\}$ e $Y = \{y_1, y_2, \ldots, y_n\}$, encontrar uma subsequência de maior tamanho (LCS(X, Y))

Problema

- **Problema:** Dadas duas sequências $X = \{x_1, x_2, \ldots, x_m\}$ e $Y = \{y_1, y_2, \ldots, y_n\}$, encontrar uma subsequência de maior tamanho (LCS(X, Y))

- **Exemplo:**
  - X = {A, B, C, B, D, A, B}
  - Y = {B, D, C, A, B, A}
  - LCS(X, Y) = {B, C, B, A} ou {B, D, A, B}

Problema

- **Exemplo de Aplicação:** análise do DNA de dois ou mais organismos distintos.
  - Um DNA é composto por uma sequência de moléculas, chamadas de bases:
    - Adenina (A); Timina (T); Citosina (C); Guanina (G).
  - Exemplo: ACGGGTAGTCGCAA
  - Computacionalmente, um DNA pode ser visto como um vetor de caracteres, com o alfabeto {A, T, G, C}

Problema

- Dado os DNAs de dois organismos:
  - S_1 = ACCGTGGAAAAGGTTAAGGCCAGGATTTAACCGCGGGC
  - S_2 = ACCGCGGTTTAATCCGGATAGGTTGAAATGGTTGAAAC
- É possível questionar:
  - Quão semelhantes são estes dois organismos?
  - Estes organismos são da mesma espécie?
  - Um destes organismos é ancestral do outro organismo?
- As respostas vão depender de semelhanças.

Como Resolver?

- **Problema:** Dadas duas sequências $X = \{x_1, x_2, \ldots, x_m\}$ e $Y = \{y_1, y_2, \ldots, y_n\}$, encontrar uma subsequência de maior tamanho (LCS(X, Y))

- **Exemplo:**
  - X = {A, B, C, B, D, A, B}
  - Y = {B, D, C, A, B, A}

  - LCS(X, Y) = {B, C, B, A} ou {B, D, A, B}

LCS – Força Bruta

- **Algoritmo:** Enumera-se todas as possíveis subsequências de X. Checa-se se cada uma destas subsequências também é subsequências de Y, guardando a de maior comprimento.

- Complexidade?
  - Existem 2^m subsequências diferentes de X;
  - Tempo linear para verificar se a subsequência de X está em Y;
  - $O(2^m n)$ = **$O(2^m)$ → INEFICIENTE!**

LCS – Força Bruta

- Outras soluções?

LCS – Programação Dinâmica

- **Solução:** Programação Dinâmica!

- **Ideia:** quebrar o problema em subproblemas, com estrutura semelhante ao problema geral. Armazena-se a solução para cada um desses subproblemas para utiliza-las para a solução geral.

LCS – Programação Dinâmica

- Dada uma sequência $X = \{x_1, x_2, \ldots, x_m\}$, define-se o i-ésimo prefixo de X, para $i = 0, 1, 2, \ldots, m$ como $X_i = \{x_1, x_2, \ldots, x_i\}$

- Por exemplo, dada a sequência X = {A, B, C, B,D, A, B}
  - $X_0 = \{\}$
  - $X_3 = \{A, B, C\}$
  - $X_4 = \{A, B, C, B\}$

LCS – Programação Dinâmica

- Uma LCS(X, Y) pode ser obtida recurivamente da seguinte forma:
  - Se $x_m = y_n$ deve-se procurar uma LCS(X_m-1, Y_n-1) e depois acrescentar x_m ou y_n.
  - Se $x_m \neq y_n$, então deve-se resolver 2 subproblemas:
    - Encontrar uma LCS(X_m-1, Y_n);
    - Encontrar uma LCS(X_m, Y_n-1);
    - Sendo a LCS mais longa destas duas.

LCS – Programação Dinâmica

- Defina *c*(*i*, *j*) como o tamanho da LCS(X_i, Y_j):

$$c(i,j) = \begin{cases} 0 & se\ i = 0\ ou\ j = 0 \\ C(i-1, j-1) + 1 & se\ i = j \\ \max\{c(i-1,j), c(i,j-1)\} & se\ i \neq j \end{cases}$$

LCS – Programação Dinâmica

- Ideia do Algoritmo:
  - Considere duas sequências como entrada, $X = \{x_1, x_2, \ldots, x_m\}$ e $Y = \{y_1, y_2, \ldots, y_n\}$;
  - Os valore $c(i, j)$ são armazenados em uma tabela $c[0..m, 0..n]$, onde os valores são computados linha a linha, esquerda para direita;
  - Também há uma tabela $b[1..m, 1..n]$ tal que são armazenados as entradas para a escolha da solução ótima dos subproblemas quando está se computando $c(i , j)$.

LCS – Exemplo

- X=BDCABA e Y=ABCBDAB

	Y_j	A	B	C	B	D	A	B
X_i								
B								
D								
C								
A								
B								
A								

LCS – Exemplo

- X=BDCABA e Y=ABCBDAB

	Y_j	A	B	C	B	D	A	B
X_i	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
D	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
B	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	4

LCS – Algoritmo

```
1. LCS(X, m, Y, n)
2.   para i ← 0 até m faça
3.     c[i,0] ← 0
4.   para i ← 0 até n faça
5.     c[0,i] ← 0
6.   para i ← 1 até m faça
7.     para j ← 1 até n faça
8.       se Xi == Yi então
9.         c[i,j] = c[i – 1, j – 1] + 1
10.        b[i,j] = "↖"
11.      senão se c[i – 1, j] ≥ c[i, j – 1] então
12.        c[i,j] = c[i – 1, j]
12.        b[i,j] = "↑"
14.      senão
15.        c[i,j] = c[i, j - 1]
16.        b[i,j] = "←"
17.  retorna c, b
```

O(mn)

LCS – Algoritmo

- Algoritmo para imprimir a LCS na ordem correta:

```
1. PRINT-LCS(b, X, i, j)
2.   se i == 0 ou j == 0 então
3.     retorna
4.   se b[i, j] == "↖" então
5.     PRINT-LCS(b, X, i – 1, j – 1)
6.     escreva X_i
7.   senão se b[i, j] == "↑" então
8.     PRINT-LCS(b, X, i – 1, j)
9.   senão
10.    PRINT-LCS(b, X, i, j - 1)
```

Exercícios

Lista de Exercícios 08 – Maior Subsequência Comum

http://www.inf.puc-rio.br/~elima/paa/

Leitura Complementar

- Halim e Halim. **Competitive Programming**, 3rd Edition, 2003.

- **Capítulo 6: String Processing**