LCS (Longest Common Subsequence) (Cormen, Cap. 15.4)

- **Entrada:** Duas strings $x[1 \ldots m]$ e $y[1 \ldots n]$
- **Objetivo:** Obter $LCS(x, y)$: a maior string s comum à x e y.

Exemplo: $S_3 \;=\; LCS(S_1, S_2)$

- $S_1 \;=\; ACCG\underline{GTCG}AG\underline{T}G\underline{CG}CG\underline{GAAGCCGGCCG}C\underline{AA}$
- $S_2 \;=\; \underline{GTCGT}T\underline{CGGAA}T\underline{GCCG}TT\underline{GC}T\underline{C}T\underline{G}TA\underline{AA}$
- $S_3 \;=\; LCS(S_1, S_2) \;=\; GTCGTCGGAAGCCGGCCGAA$

1. Propriedade da Subestrutura Ótima

- Em linguagem popular, pergunta-se se "*pedaços da solução ótima são soluções ótimas de pedaços do problema*".
- Exemplo: $S_3' = GTCGTCGGAAGCCGGCCG$ é solução ótima de $(LCS(S_1', S_2'))$:
- $S_1' \;=\; ACCG\underline{GTCG}AG\underline{T}G\underline{CG}CG\underline{GAAGCCGGCCG}C$
- $S_2' \;=\; \underline{GTCGT}T\underline{CGGAA}T\underline{GCCG}TT\underline{GC}T\underline{C}T\underline{G}TA$

2. Equação de Recorrência - Algoritmo recursivo simples

- Seja $\ell cs(i, j) \;=\; |LCS(x[1 \ldots i], y[1 \ldots j])|$
- $\ell cs(0, j) = 0;$ $\qquad\qquad \ell cs(i, 0) = 0$
- Comparar x_i e y_j: se igual, entra na LCS. Senão, x_i ou $y_j \notin$ LCS.
- $$\ell cs(i,j) \;=\; \begin{cases} \ell cs(i-1, j-1) \;+\; 1, \;\text{se } x_i = y_j \\ \max\{\ell cs(i-1, j),\; \ell cs(i, j-1)\}, \;\text{se } x_i \neq y_j \end{cases}$$

$LCS\text{-}rec(x, y, i, j)$

```
se (i = 0) ou (j = 0) então retorne 0
se (x_i = y_j) então
        retorne LCS-rec(x, y, i − 1, j − 1) + 1
senão retorne max{LCS-rec(x, y, i − 1, j), LCS-rec(x, y, i, j − 1)}
```

2. Equação de Recorrência - Algoritmo recursivo simples

Sobreposição de Subproblemas

- **Chamada inicial**: $LCS\text{-}rec(x, y, m, n)$
- **Tempo Pior caso**: $[x = AA \dots A$ e $y = BB \dots B]$
- Exponencial $\Omega(2^{\min\{m,n\}})$
- **Indução**: $T(m, n) \geq T(m-1, n) + T(m, n-1) \geq$
  $2 \cdot T(m-1, n-1) \;\geq\; 2 \cdot 2^{\min\{m,n\}-1} \;=\; 2^{\min\{m,n\}}$
- **Superposição**: Instâncias $(m\text{-}1, n)$ e $(m\text{-}1, n)$ chamam $(m\text{-}1, n\text{-}1)$
- **Tempo Melhor caso**: Polinomial - Linear $\Theta(\min\{m, n\})$ [x=y]

$LCS\text{-}rec(x, y, i, j)$

```
se (i = 0) ou (j = 0) então retorne 0
se (x_i = y_j) então
        retorne LCS-rec(x, y, i − 1, j − 1) + 1
senão retorne max{LCS-rec(x, y, i − 1, j), LCS-rec(x, y, i, j − 1)}
```

2b. Memoização (Alg. Recursivo + memória) - Top Down

LCS-memo$(x, y, m, n, \ell cs)$

para $i \leftarrow 0$ **até** m: $\ell cs[i, 0] \leftarrow 0$
para $j \leftarrow 0$ **até** n: $\ell cs[0, j] \leftarrow 0$
para $i \leftarrow 1$ **até** m:
    para $j \leftarrow 1$ **até** n:
        $\ell cs[i, j] \leftarrow -1$
retorne *LCS-rec-memo*$(x, y, m, n, \ell cs)$

LCS-rec-memo$(x, y, i, j, \ell cs)$

se $(\ell cs[i, j] \geq 0)$ **então retorne** $\ell cs[i, j]$
se $(x_i = y_j)$ **então**
    $\ell cs[i, j] \leftarrow$ *LCS-rec-memo*$(x, y, i-1, j-1, \ell cs) + 1$
senão
    $\ell cs[i, j] \leftarrow \max\{$ *LCS-rec-memo*$(x, y, i-1, j, \ell cs)$,
        LCS-rec-memo$(x, y, i, j-1, \ell cs)$ $\}$
retorne $\ell cs[i, j]$

3. Algoritmo p/ Valor Ótimo (Bottom-up, não recursivo)

LCS-PD(x, y, m, n)

Criar matriz $\ell cs[0 \ldots m,\ 0 \ldots n]$
para $i \leftarrow 0$ **até** m: $\ell cs[i, 0] \leftarrow 0$
para $j \leftarrow 0$ **até** n: $\ell cs[0, j] \leftarrow 0$
para $i \leftarrow 1$ **até** m:
    para $j \leftarrow 1$ **até** n:
        se $(x_i = y_j)$ **então**
            $\ell cs[i, j] \leftarrow \ell cs[i-1, j-1] + 1$
        senão
            $\ell cs[i, j] \leftarrow \max\{\ \ell cs[i-1, j],\ \ell cs[i, j-1]\ \}$
retorne $\ell cs[m, n]$

3. Algoritmo p/ Valor Ótimo (Bottom-up, não recursivo)

.	-	B	D	C	A	B	A
-	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	1	1	1	1	2	2
C	0	1	1	2	2	2	2
B	0	1	1	2	2	3	3
D	0	1	2	2	2	3	3
A	0	1	2	2	3	3	4
B	0	1	2	2	3	4	4

Todas as soluções ótimas possíveis:

- B C B A B C A B B D A B

3. Algoritmo p/ Valor Ótimo (Bottom-up, não recursivo)

.	-	B	D	C	A	B	A
-	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	1	1	1	1	2	2
C	0	1	1	2	2	2	2
B	0	1	1	2	2	3	3
D	0	1	2	2	2	3	3
A	0	1	2	2	3	3	4
B	0	1	2	2	3	4	4

Todas as soluções ótimas possíveis:

- B C B A   B C A B   B D A B

3. Algoritmo p/ Valor Ótimo (Bottom-up, não recursivo)

.	-	B	D	C	A	B	A
-	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	1	1	1	1	2	2
C	0	1	1	2	2	2	2
B	0	1	1	2	2	3	3
D	0	1	2	2	2	3	3
A	0	1	2	2	3	3	4
B	0	1	2	2	3	4	4

Todas as soluções ótimas possíveis:

- B C B A   B C A B   B D A B

3. Algoritmo p/ Valor Ótimo (Bottom-up, não recursivo)

.	-	B	D	C	A	B	A
-	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	1	1	1	1	2	2
C	0	1	1	2	2	2	2
B	0	1	1	2	2	3	3
D	0	1	2	2	2	3	3
A	0	1	2	2	3	3	4
B	0	1	2	2	3	4	4

Todas as soluções ótimas possíveis:

- B C B A   B C A B   **B D A B**

4. Algoritmo p/ obter uma Solução Ótima (Bottom-up)

LCS-PD(x, y, m, n)

Criar matriz $\ell cs[0 \ldots m,\ 0 \ldots n]$
para $i \leftarrow 0$ **até** m: $\ell cs[i, 0] \leftarrow 0$; $R[i, 0] \leftarrow$ "$\uparrow$"
para $j \leftarrow 0$ **até** n: $\ell cs[0, j] \leftarrow 0$; $R[0, j] \leftarrow$ "$\leftarrow$"
para $i \leftarrow 1$ **até** m:
  para $j \leftarrow 1$ **até** n:
    se $(x_i = y_j)$ **então**
      $\ell cs[i, j] \leftarrow \ell cs[i-1, j-1] + 1$; $R[i, j] \leftarrow$ "$\nwarrow$"
    senão se $(\ell cs[i-1, j] \geq \ell cs[i, j-1])$ **então**
      $\ell cs[i, j] \leftarrow \ell cs[i-1, j]$; $R[i, j] \leftarrow$ "$\uparrow$"
    senão
      $\ell cs[i, j] \leftarrow \ell cs[i, j-1]$; $R[i, j] \leftarrow$ "$\leftarrow$"
retorne $\ell cs[m, n]$ **e** R

Tempo $\Theta(m \cdot n)$

4b. Algoritmo p/ escrever a Solução Ótima (recursivo)

Print-LCS(x, y, i, j, R)

se $i = 0$ **ou** $j = 0$ **então retorne**

se $R[i, j] =$"$\nwarrow$" **então**

  Print-LCS$(x, y, i-1, j-1, R)$; **print** x_i

se $R[i, j] =$"$\leftarrow$" **então**

  Print-LCS$(x, y, i, j-1, R)$

se $R[i, j] =$"$\uparrow$" **então**

  Print-LCS$(x, y, i-1, j, R)$

Tempo $\Theta(m + n)$

4b. Algoritmo p/ escrever a Solução Ótima (recursivo2)

Print-LCS2(x, y, i, j, R, LCS)

se $i = 0$ **ou** $j = 0$ **então print** LCS; **retorne**
se $R[i, j] =$ "↖" **então**
    $LCS \leftarrow x_i + LCS$
    Print-LCS2$(x, y, i-1, j-1, R, LCS)$
se $R[i, j] =$ "←" **então**
    Print-LCS2$(x, y, i, j-1, R, LCS)$
se $R[i, j] =$ "↑" **então**
    Print-LCS2$(x, y, i-1, j, R, LCS)$

Tempo $\Theta(m + n)$

4b. Algoritmo p/ escrever a Solução Ótima (não-recursivo)

Print-LCS$(x, y, m, n, R, \ell cs)$

$LCS \leftarrow \emptyset$; $k \leftarrow \ell cs[m, n]$; $i \leftarrow m$; $j \leftarrow n$
enquanto $(i > 0)$ **e** $(j > 0)$ **faça**
  se $R[i, j] =$ "$\nwarrow$" **então**
    $LCS[k] \leftarrow x_i$; $k \leftarrow k - 1$; $i \leftarrow i - 1$; $j \leftarrow j - 1$
  senão se $R[i, j] =$ "$\leftarrow$" **então**
    $j \leftarrow j - 1$
  senão
    $i \leftarrow i - 1$
print LCS

Tempo $\Theta(m + n)$

Número exponencial de LCS's

Exemplo:

- Letras $a_1, \ldots, a_n$ e $b_1, \ldots, b_n$ todas diferentes entre si
- $x \; = \; a_1 b_1 a_2 b_2 a_3 b_3$
- $y \; = \; b_1 a_1 b_2 a_2 b_3 a_3$
- $LCS(x, y) \; = \; a_1 a_2 a_3$
- $LCS(x, y) \; = \; a_1 a_2 b_3$
- $LCS(x, y) \; = \; a_1 b_2 a_3$
- $LCS(x, y) \; = \; a_1 b_2 b_3$
- $LCS(x, y) \; = \; b_1 a_2 a_3$
- $LCS(x, y) \; = \; b_1 a_2 b_3$
- $LCS(x, y) \; = \; b_1 b_2 a_3$
- $LCS(x, y) \; = \; b_1 b_2 b_3$

Número de LCS's $= 2^n$, onde $|x| = |y| = 2n$.