

Algoritmos em Sequências

Notas de aula da disciplina
TE: Técnicas de Construção de Algoritmos

Fabiano de Souza Oliveira
(fabiano.oliveira@ime.uerj.br)

Paulo Eustáquio Duarte Pinto
(pauloedp@ime.uerj.br)

setembro/2020

Algoritmos complementares em Sequências:

- Subsequência crescente máxima
- Subsequência consecutiva de soma máxima
- Subsequência máxima comum
- Distância de Edição

Subsequência Crescente Máxima (SCM)

Dada uma sequência S de inteiros, $\langle s_1, s_2, \dots, s_n \rangle$, determinar a **subsequência crescente** de tamanho máximo.

Exemplo: $S = \langle 10, 2, 15, 3, 20, 4, 20 \rangle$

A **SCM** é **$\langle 2, 3, 4, 20 \rangle$**

Subsequência Crescente Máxima (SCM)

```
função SCM-t(S[], n: Inteiro)
    //retorna o comprimento da subsequência crescente
    //máxima que termina em S[n]
    m ← 1
    para i ← 1 até n-1 faça
        se S[i] < S[n] então
            m ← máx {m, SCM-t(S,i)+1}
    retornar m
```

```
função SCM(S[], n: Inteiro)
    //retorna o comprimento da subsequência crescente
    //máxima contida em S[1..n]
    m ← 0
    para i ← 1 até n faça
        m ← máx {m, SCML-t(S,i)}
    retornar m
```

Subsequência Crescente Máxima (SCM)

```
var T[1..100000]: Inteiro ← -1
```

```
função SCM-t(S[], n: Inteiro)
    //retorna o comprimento da subsequência crescente
    //máxima que termina em S[n]
    se T[n]=-1 então
        m ← 1
        para i ← 1 até n-1 faça
            se S[i] < S[n] então
                m ← máx {m, SCM-t(S,i)+1}
        T[n] ← m
    retornar T[n]
```

```
função SCM(S[], n: Inteiro)
    //retorna o comprimento da subsequência crescente
    //máxima contida em S[1..n]
    m ← 0
    para i ← 1 até n faça
        m ← máx {m, SCM-t(S,i)}
    retornar m
```

Tempo:
 $\Theta(n^2)$

Subsequência crescente máxima (SCM):

Solução: manter, para cada tamanho de subsequência crescente, qual o menor elemento possível que termina essa subsequência, à medida que se varre linearmente a sequência original.

Exemplo: $S = \langle 10, 2, 15, 3, 20, 4, 20 \rangle$

$i = 1$	SCM de tamanho 1 termina em 10.
$i = 2$	SCM de tamanho 1 termina em 2.
$i = 3$	SCM de tamanho 2 termina em 15.
$i = 4$	SCM de tamanho 2 termina em 3.
$i = 5$	SCM de tamanho 3 termina em 20.
$i = 6$	SCM de tamanho 3 termina em 4.
$i = 7$	SCM de tamanho 4 termina em 20.

Subsequência crescente máxima (SCM):

SCM:

$k \leftarrow 1; T[1] \leftarrow S[1]; O[1] \leftarrow 1$

para $i \leftarrow 2$ **até** n **faça**

se $S[i] > T[k]$ **então**

$k \leftarrow k+1; T[k] \leftarrow S[i]; O[i] \leftarrow k$

senão

$j \leftarrow \text{BuscaBinária}(T, 1, k, S[i])$

 // $T[j-1] < S[i] \leq T[j]$, para algum $1 \leq j \leq k$

$T[j] \leftarrow S[i]; O[i] \leftarrow j$

Tempo:
 $O(n \log n)$

k = tamanho da SCM

$T[j]$ = menor elemento que termina uma SC de tamanho j

$O[i]$ = tamanho da SCM terminada em $S[i]$

Subsequência crescente máxima (SCM):

```
função BuscaBinária (T[], ini, fim, x):  
    se ini > fim então  
        retornar ini  
    senão  
        m ← (ini + fim) div 2  
        se T[m] < x então  
            retornar BuscaBinária (T, m+1, fim, x)  
        senão  
            retornar BuscaBinária (T, ini, m-1, x)
```


Exemplo: $S = \langle 10, 2, 15, 8, 20, 16, 10, 5, 20 \rangle$
 $O = \langle 1, 1, 2, 2, 3, 3, 3, 2, 4 \rangle$

$i \setminus k$	s_i	1	2	3	4
1	10	10	-	-	-
2	2	2	-	-	-
3	15	2	15	-	-
4	8	2	8	-	-
5	20	2	8	20	-
6	16	2	8	16	-
7	10	2	8	10	-
8	5	2	5	10	-
9	20	2	5	10	20

Soluções:

10	15	16	20
2	15	16	20
2	8	16	20
2	8	10	20

Subsequência crescente máxima (SCM):

Para determinar uma SCM:

```
Obter(S[], O[], n, k, ref R[]):  
  m ← ∞; i ← n; j ← k  
  enquanto j > 0 faça  
    se O[i] = j e S[i] < m então  
      R[j] ← S[i]  
      j ← j-1; m ← S[i]  
  i ← i - 1
```

Para determinar todas as SCMs, usa-se
Backtracking

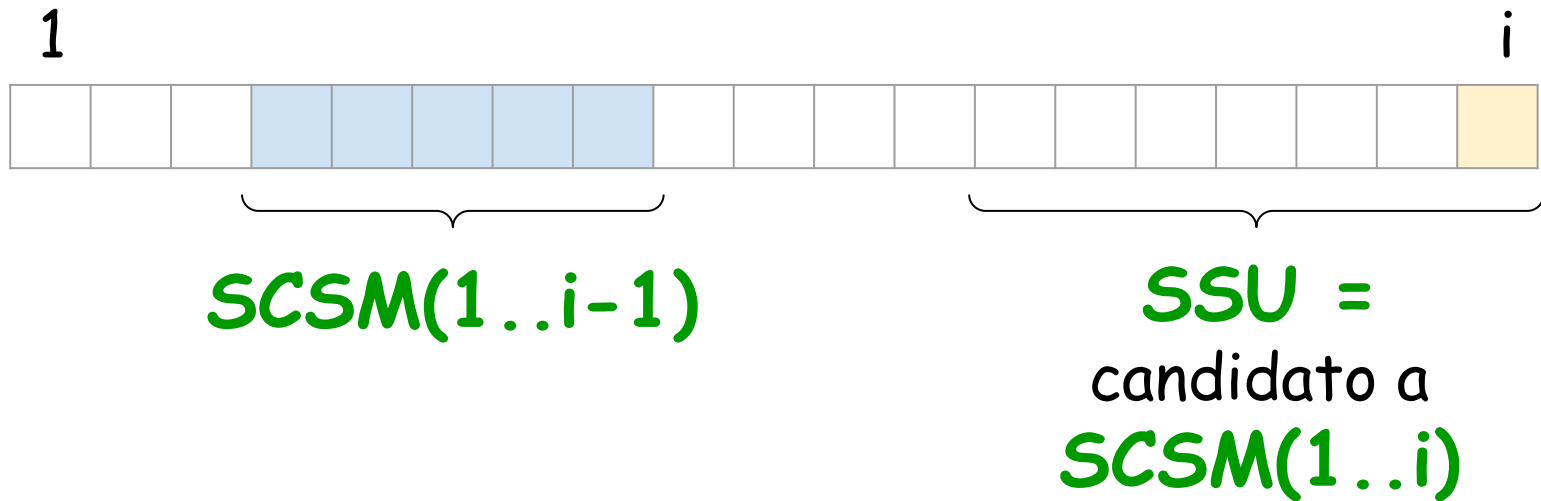
Subsequência Consecutiva de Soma Máxima (SCSM)

Dada uma sequência S de inteiros, $\langle s_1, s_2, \dots, s_n \rangle$, determinar a subsequência consecutiva de soma máxima.

Exemplo: $S = \langle 6, 2, -7, 12, -15, 8, -2, 3, -5, 10, -1 \rangle$

A SCSM é $\langle 8, -2, 3, -5, 10 \rangle$, com soma = 14

Subsequência Consecutiva de Soma Máxima (SCSM)



Subsequência Consecutiva de Soma Máxima (SCSM)

O problema é resolvido varrendo a sequência e guardando a subsequência de soma máxima (SCSM) e a subsequência sufixo não negativa (SSU)

Exemplo: $S = \langle 6, 2, -7, 12, -15, 8, -2, 3, -5, 10, -1 \rangle$

$i = 1, 2$ SCSM = $\langle 6, 2 \rangle$ e SSU = $\langle 6, 2 \rangle$.

$i = 3$ SCSM = $\langle 6, 2 \rangle$ e SSU = $\langle 6, 2, -7 \rangle$.

$i = 4$ SCSM = SSU = $\langle 6, 2, -7, 12 \rangle$, com soma 13.

$i = 5$ SCSM = $\langle 6, 2, -7, 12 \rangle$ e SSU = $\langle \rangle$.

$i = 6$ a 9 SCSM = $\langle 6, 2, -7, 12 \rangle$ e SSU = $\langle 8, -2, 3, -5 \rangle$.

$i = 10$ SCSM = SSU = $\langle 8, -2, 3, -5, 10 \rangle$, com soma 14.

$i = 11$ SCSM = $\langle 8, -2, 3, -5, 10 \rangle$, SSU = SCSM U $\langle -1 \rangle$

Subsequência Consecutiva de Soma Máxima (SCSM)

SCSM: ¹

$sm, im, fm \leftarrow 0, 0, 0$

$ss, is, fs \leftarrow 0, 0, 0$

para $i \leftarrow 1$ até n faça

se $ss + x[i] \geq 0$ então

$ss \leftarrow ss + x[i]$

$fs \leftarrow i$

se $is = 0$ então

$is \leftarrow i$

se $ss > sm$ então

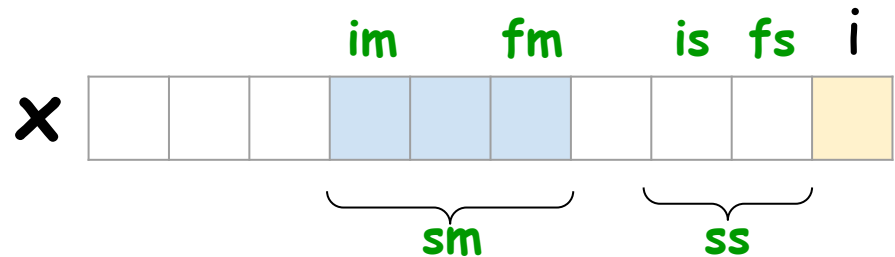
$sm, im, fm \leftarrow ss, is, fs$

senão

$ss, is, fs \leftarrow 0, 0, 0$

// sm é SCSM de início im e fim fm

// ss é SSU de início is e fim fs



Subsequência Consecutiva de Soma Máxima (SCSM)

Exemplo: $S = \langle 6, 2, -7, 12, -15, 8, -2, 3, -5, 10, -1 \rangle$

i	s_i	is	fs	ss	im	fm	sm
1	6	1	1	6	1	1	6
2	2	1	2	8	1	2	8
3	-7	1	3	1	1	2	8
4	12	1	4	13	1	4	13
5	-15	0	0	0	1	4	13
6	8	6	6	8	1	4	13
7	-2	6	7	6	1	4	13
8	3	6	8	9	1	4	13
9	-5	6	9	4	1	4	13
10	10	6	10	14	6	10	14
11	-1	6	11	13	6	10	14

Subsequência máxima comum (SMC):

Dadas as sequências $A \langle a_1 a_2 \dots a_n \rangle$ e $B \langle b_1 b_2 \dots b_m \rangle$ determinar a subsequência máxima comum, S .

Ex: $A = \text{OPACO}$ $B = \text{PARADO}$, $S = \text{PAO}$

Solução (DC):

Considerando $A_i \langle a_1 a_2 \dots a_i \rangle$ e $B_j \langle b_1 b_2 \dots b_j \rangle$ e a SMC $S_k \langle s_1 s_2 \dots s_k \rangle$

se $a_i = b_j$, então

$S_k = S_{k-1} + a_i$ e S_{k-1} é SMC entre A_{i-1} e B_{j-1}

senão

S_k é SMC ou entre $(A_{i-1}$ e $B_j)$ ou entre $(A_i$ e $B_{j-1})$

Subsequência máxima comum (SMC):

O problema é resolvido tabelando os tamanhos das subsequências comuns. Seja $T(i, j)$ o tamanho da SMC relativa a A_i e B_j .

Recorrência:

$$T(i, 0) = 0, 0 \leq i \leq n.$$

$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1) + 1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Subsequência máxima comum (SMC):

$$T(i, 0) = 0, 0 \leq i \leq n.$$


$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1) + 1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Exemplo: achar a SMC entre **OPACO** e **PARADO**

			P	A	R	A	D	O
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	1



Subsequência máxima comum (SMC):

$$T(i, 0) = 0, 0 \leq i \leq n.$$

$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1) + 1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Exemplo: achar a SMC entre **OPACO** e **PARADO**

			P	A	R	A	D	O
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	1
P	2	0	1	1	1	1	1	1

Subsequência máxima comum (SMC):

$$T(i, 0) = 0, 0 \leq i \leq n.$$

$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1)+1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Exemplo: achar a SMC entre **OPACO** e **PARADO**

			P	A	R	A	D	O
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	1
P	2	0	1	1	1	1	1	1
A	3	0	1	2	2	2	2	2

Subsequência máxima comum (SMC):

$$T(i, 0) = 0, 0 \leq i \leq n.$$

$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1)+1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Exemplo: achar a SMC entre **OPACO** e **PARADO**

			P	A	R	A	D	O
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	1
P	2	0	1	1	1	1	1	1
A	3	0	1	2	2	2	2	2
C	4	0	1	2	2	2	2	2

Subsequência máxima comum (SMC):

$$T(i, 0) = 0, 0 \leq i \leq n.$$


$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1) + 1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Exemplo: achar a SMC entre **OPACO** e **PARADO**

			P	A	R	A	D	O
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	1
P	2	0	1	1	1	1	1	1
A	3	0	1	2	2	2	2	2
C	4	0	1	2	2	2	2	2
O	5	0	1	2	2	2	2	3



Subsequência máxima comum (SMC):

$$T(i, 0) = 0, 0 \leq i \leq n.$$

$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1) + 1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Algoritmo:

SCM(n, m):

para $i \leftarrow 0$ até n incl.:

$T[i, 0] \leftarrow 0$

para $j \leftarrow 0$ até m incl.:

$T[0, j] \leftarrow 0$

para $i \leftarrow 1$ até n incl.:

para $j \leftarrow 1$ até m incl.:

se $(a_i = b_j)$:

$T[i, j] \leftarrow T[i-1, j-1] + 1$

senão:

$T[i, j] \leftarrow \max(T[i-1, j], T[i, j-1])$

Subsequência máxima comum (SMC):

$$T(i, 0) = 0, 0 \leq i \leq n.$$

$$T(0, j) = 0, 0 \leq j \leq m.$$

$$T(i, j) = T(i-1, j-1)+1, \text{ se } i, j > 0 \text{ e } a_i = b_j.$$

$$T(i, j) = \max(T(i-1, j), T(i, j-1)) \text{ se } i, j > 0 \text{ e } a_i \neq b_j.$$

Para gerar uma SMC, usa-se DC:

Gerar (i, j, k):

se (k > 0):

se ($a_i = b_j$):

$$s_k \leftarrow a_i$$

Gerar (i-1, j-1, k-1)

senão se ($T[i-1, j] = k$):

Gerar (i-1, j, k)

senão:

Gerar (i, j-1, k)

Para gerar todas as SMC,
usa-se Backtracking

Gerar(n, m, T[n, m])

escrever S

Subsequência máxima comum (SMC):

```

Gerar (i, j, k):
  se (k > 0):
    se (ai = bj):
      sk ← ai
      Gerar (i-1, j-1, k-1)
    senão se (T[i-1, j] = k):
      Gerar (i-1, j, k)
  senão:
    Gerar (i, j-1, k)
  
```

Ex: mostrar a SMC entre **OPACO** e **PARADO**

			P	A	R	A	D	O
		0	1	2	3	4	5	6
	0	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	1
P	2	0	1	1	1	1	1	1
A	3	0	1	2	2	2	2	2
C	4	0	1	2	2	2	2	2
O	5	0	1	2	2	2	2	3

P A O

Exercício 11: Qual o tamanho da maior subsequência comum aos strings:
ABRACADABRA e BABADECRABA?

Exercício 12: Qual o tamanho da maior subsequência crescente em:
26 3 15 29 60 20 4 8 19 16 24 32 2 18 35

Exercício 13: Qual a soma da subsequência consecutiva de maior soma em:
-1 8 -7 6 15 -23 9 -3 -5 16 -5 8 -9 12 -6

Distância de Edição

Dados dois strings A e B, quer-se determinar a menor sequência de operações para transformar A em B.

Os tipos de operação são:

- inserção de um caracter
- deleção de um caracter
- substituição de um caracter

Ex: ERRO transforma-se em ACERTO com 3 operações:

- | | |
|---------------------|--------|
| - inserção do A | AERRO |
| - inserção do C | ACERRO |
| - substituição do R | ACERTO |

Distância de Edição - Formulação DC

Dados os substrings A_i (primeiros i caracteres) e B_j ,

$D(i, j)$ = distância de edição entre A_i e B_j =

$D(i-1, j-1)$, se $a_i = b_j$

$1 + \min(D(i-1, j), D(i, j-1), D(i-1, j-1))$, se $a_i \neq b_j$

Deleção de a_i

Inserção de b_j

Substituição
de a_i por b_j

Cálculo da Distância de Edição

	0	1 A	2 C	3 E	4 R	5 T	6 O
0	0	1	2	3	4	5	6
1 E	1	1	2	2	3	4	5

The diagram illustrates the calculation of the edit distance between the sequences "E" and "ACERTOS". The table shows the edit distance for each prefix of "ACERTOS" (columns) against the prefixes of "E" (rows). Red arrows indicate the path of minimum edits:

- From (0,0) to (1,1): Edit 'A' (1 edit).
- From (1,1) to (2,2): Edit 'C' (2 edits).
- From (2,2) to (3,2): Edit 'E' (2 edits).
- From (3,2) to (4,3): Edit 'R' (3 edits).
- From (4,3) to (5,4): Edit 'T' (4 edits).
- From (5,4) to (6,5): Edit 'O' (5 edits).

Cálculo da Distância de Edição

	0	1 A	2 C	3 E	4 R	5 T	6 O
0	0	1	2	3	4	5	6
1 E	1	1	2	2	3	4	5
2 R	2	2	2	3	2	3	4

Cálculo da Distância de Edição

	0	1 A	2 C	3 E	4 R	5 T	6 O
0	0	1	2	3	4	5	6
1 E	1	1	2	2	3	4	5
2 R	2	2	2	3	2	3	4
3 R	3	3	3	3	3	3	4

Cálculo da Distância de Edição

	0	1 A	2 C	3 E	4 R	5 T	6 O
0	0	1	2	3	4	5	6
1 E	1	1	2	2	3	4	5
2 R	2	2	2	3	2	3	4
3 R	3	3	3	3	3	3	4
4 O	4	4	4	4	4	4	3

Distância de Edição

$$D(i, j) = \begin{cases} D(i-1, j-1) = 0, & \text{se } a_i = b_j \\ \min(D(i-1, j), D(i, j-1), D(i-1, j-1)) + 1, & \text{se } a_i \neq b_j \end{cases}$$

DistânciaEdição():

#Dados: Strings A e B, com $|A| = n$ e $|B| = m$

para $i \leftarrow 0$ até n incl.:

$D[i, 0] \leftarrow i$

para $j \leftarrow 0$ até m incl.:

$D[0, j] \leftarrow j$

para $i \leftarrow 1$ até n incl.:

para $j \leftarrow 1$ até m incl.:

se $(A[i] = B[j])$:

$D[i, j] \leftarrow D[i-1, j-1]$

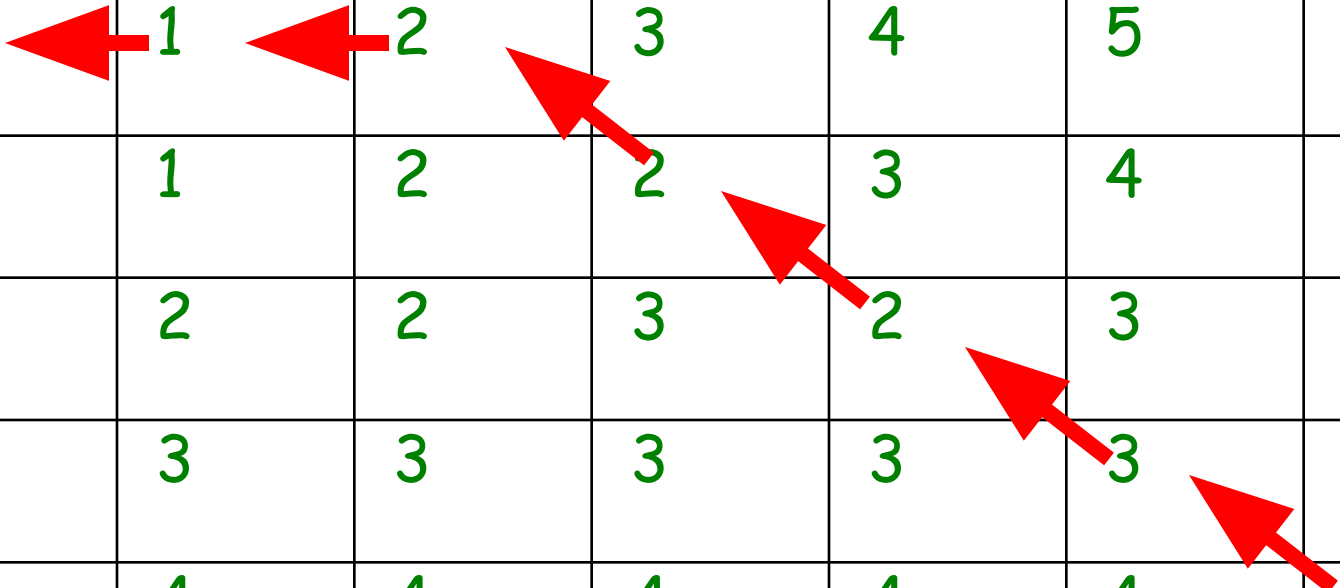
senão:

$D[i, j] \leftarrow 1 + \min(D[i-1, j-1], D[i-1, j], D[i, j-1])$

Complexidade: $O(n.m)$

Distância de Edição - Apresentando a transformação

	0	1 <i>A</i>	2 <i>C</i>	3 <i>E</i>	4 <i>R</i>	5 <i>T</i>	6 <i>O</i>
0	0	1	2	3	4	5	6
1 <i>E</i>	1	1	2	2	3	4	5
2 <i>R</i>	2	2	2	3	2	3	4
3 <i>R</i>	3	3	3	3	3	3	4
4 <i>O</i>	4	4	4	4	4	4	3



Distância de Edição - Apresentando uma transformação

Transf(i,j):

#Dados Matriz de distâncias D, Strings A e B

se ((i>0) ou (j>0)):

se (A[i] = B[j]):

Transf(i-1,j-1); escrever 'Manteve' A[i];

senão se ((j > 0) e D[i,j] = D[i,j-1]+1):

Transf(i,j-1); escrever 'Inseriu' B[j];

senão se ((i > 0) e D[i,j] = D[i-1,j]+1):

Transf(i-1,j); escrever 'Deletou' A[i];

senão:

Transf(i-1,j-1)

escrever 'Transformou' A[i] 'em' B[j]

Transf(n,m)

Para gerar todas as transformações usa-se Backtracking

Distância de Edição - Apresentando a transformação

	0	1 M	2 A	3 R	4 R	5 O	6 M
0	0	1	2	3	4	5	6
1 A	1	1	1	2	3	4	5
2 M	2	1	2	2	3	4	4
3 O	3	2	2	3	3	3	4
4 R	4	3	3	2	3	4	4
5 A	5	4	3	3	3	4	5

FIM