Aula Prática 4

ASA 2021/2022

Q1 (T2 20/21): Uma sequência diz-se um *palíndromo* se é simétrica, isto é, se permanece igual quando lida de trás para diante; por exemplo, são palíndromos as sequências: *a*, *aa*, *abbba* e *abbaabba*. Pretende-se desenvolver um algoritmo que, dada uma sequência de caracteres arbitrária, retorne o tamanho do maior palíndromo que esta contém. Por exemplo, dada a sequência *abbaabbabaabc*, o algoritmo deve retornar 8, que corresponde ao tamanho do palíndromo *abbaabba*.

a. Seja $x[1..n]$ a string de texto dada como input e $B(i,j)$ o valor Booleano que indica se a cadeia de caracteres $x[i..j]$ forma um palíndromo. Defina $B(i,j)$ recursivamente completando os campos em baixo:

$$B(i,j) = \begin{cases} \textbf{true} & \text{se } j < i \\ \boxed{} & \text{se } j = i \\ \boxed{} & \text{c.c.} \end{cases}$$

Admite-se, para simplificar a formulação, que $B(i,j) = \textbf{true}$ quando $j < i$.

b. Complete o template de código em baixo que calcula o tamanho do maior palíndromo contido no array dado como input, $x[1..n]$. Para obter a cotação máxima, o algoritmo deve retornar o valor pretendido assim que encontra o palíndromo de tamanho máximo, não devendo de efectuar o preenchimento completo da matriz $B[1..n,1..n]$.

`BiggestPalindromeSize`$(x[1..n])$
let $B[1..n,1..n]$ be a new matrix of size $n \times n$ with all cells initialised to true

c. Determine a complexidade assimptótica do algoritmo proposto na alínea anterior.

Solução:

a. $B(i,j) = \begin{cases} \textbf{true} & \text{se } j \leq i \\ B(i+1, j-1) \;\wedge\; (x[i] == x[j]) & \text{c.c.} \end{cases}$

b.

$\texttt{BiggestPalindromeSize}(x[1..n])$
  let $B[1..n, 1..n]$ be a new matrix of size $n \times n$ with all cells initialised to true
  let $not\_found = 0$
  for $s = 1$ **to** $n-1$ **do**
    let $found = false$
    for $i = 1$ **to** $n-s$ **do**
      $B[i, i+s] := B[i+1, (i+s)-1]$ && $(x[i] == x[i+s])$
      $found := found \,||\, B[i, i+s]$
    endfor
    if($\textbf{not}\ found$){
      $not\_found := not\_found + 1$
      if($not\_found == 2$) **return** s
    } **else** $not\_found := 0$
  endfor
  if($not\_found == 1$) **return** $n-1$
  else return n

c. Complexidade: $O(n^2)$. No pior caso o algoritmo terá de preencher a metade diagonal superior da matriz B.

Q2 (R2 20/21): Dada uma sequência de inteiros positivos $\langle x_1, ..., x_n \rangle$, pretende desenvolver-se um algoritmo que determina o maior valor suceptível de ser obtido a partir da expressão $x_1/x_2/x_3/.../x_n$, determinando a ordem pela qual as divisões devem ser efectuadas. Por exemplo, dada a sequência $\langle 16, 8, 4, 2 \rangle$, a parentização que resulta no maior valor final é: $(16/((8/4)/2)) = 16$.

a. Seja $M[i, j]$ o maior valor que é possível obter a partir da expressão $x_i/x_{i+1}/.../x_j$ e $m[i, j]$ o menor valor. Por exemplo, dada a sequência $\langle 16, 8, 4, 2 \rangle$, $M[1, 4] = 16$ e $m[1, 4] = 0.25$. Admitindo que a sequência dada como input é $\langle x_1, ..., x_n \rangle$, defina $M[i, j]$ e $m[i, j]$ recursivamente completando os campos em baixo:

$$M(i,j) = \begin{cases} \boxed{} & \text{se } i = j \\ \boxed{} & \text{se } j > i \end{cases}$$

$$m(i,j) = \begin{cases} \boxed{} & \text{se } j = i \\ \boxed{} & \text{se } j > i \end{cases}$$

b. Complete o template de código em baixo que, dada uma sequência de inteiros $\langle x_1, ..., x_n \rangle$, calcula $m[1, n]$ e $M[1, n]$.

`GreatestValue`$(x[1..n])$
  let $M[1..n, 1..n]$ be a new matrix of size $n \times n$
  let $m[1..n, 1..n]$ be a new matrix of size $n \times n$
  for $i = 1$ **to** n **do**
    $M[i, i] := \boxed{}$
    $m[i, i] := \boxed{}$
  endfor
  for $s = 1$ **to** $n-1$ **do**
    for $i = 1$ **to** $n-s$ **do**

$$\boxed{}$$

    endfor
  endfor
  return $M[1, n]$

c. Determine a complexidade assimptótica do algoritmo proposto na alínea anterior.

Solução:

a.

$$M(i,j) = \begin{cases} x[i] & \text{se } i = j \\ \mathbf{max}\{M[i,k]/m[k+1,j] \mid i \leq k < j\} & \text{se } j > i \end{cases}$$

$$m(i,j) = \begin{cases} x[i] & \text{se } j = i \\ \mathbf{min}\{m[i,k]/M[k+1,j] \mid i \leq k < j\} & \text{se } j > i \end{cases}$$

b.

```
GreatestValue(x[1..n])
  let M[1..n,1..n] be a new matrix of size n × n
  let m[1..n,1..n] be a new matrix of size n × n
  for i = 1 to n do
    M[i,i] := x[i]
    m[i,i] := x[i]
  endfor
  for s = 1 to n−1 do
    for i = 1 to n−s do
      let j = i+s
      let M[i,j] = −∞
      let m[i,j] = +∞
      for k = i to j−1 do
        M[i,j] := max(M[i,j],M[i,k]/m[k+1,j])
        m[i,j] := min(m[i,j],m[i,k]/M[k+1,j])
      endfor
    endfor
  endfor
  return M[1,n]
```

c. Complexidade: $O(n^3)$. O algoritmo tem de preencher a metade diagonal superior das matrizes $M[1..n,1..n]$ e $m[1..n,1..n]$, sendo que para cada posição da matriz o algoritmo pode percorrer s posições. Formalmente:

$$\begin{aligned}
&\textstyle\sum_{s=1}^{n-1}\sum_{i=1}^{n-s}\sum_{k=i}^{j-1} O(1) \\
&= \textstyle\sum_{s=1}^{n-1}\sum_{i=1}^{n-s}\sum_{k=i}^{i+s-1} O(1) \\
&= \textstyle\sum_{s=1}^{n-1}\sum_{i=1}^{n-s} O(1).(i+s-1-i+1) \\
&= \textstyle\sum_{s=1}^{n-1}\sum_{i=1}^{n-s} O(s) \\
&= \textstyle\sum_{s=1}^{n-1} O(s).(n-s) \\
&= O(\textstyle\sum_{s=1}^{n-1} n.s - s^3) \\
&\leq O(n.\textstyle\sum_{s=1}^{n-1} s) \\
&\leq O(n^3)
\end{aligned}$$

Q3 (EE 20/21): Dadas duas sequências de caracteres $\vec{X} = \langle X_1, ..., X_n \rangle$ e $\vec{Z} = \langle Z_1, ..., Z_k \rangle$, $\vec{Z}$ diz-se uma *subsequência contígua* de $\vec{X}$ se existir um inteiro $0 \leq i < n$ tal que: $X_{i+1} = Z_1$, $X_{i+2} = Z_2$, ..., $X_{i+k} = Z_k$. Por exemplo, a sequência de caracteres *abb* é uma subsequência contígua de *ababb* (basta escolher o deslocamento $i = 2$).

Dadas duas sequências de caracteres $\vec{X} = \langle X_1, \dots, X_n \rangle$ e $\vec{Y} = \langle Y_1, \dots, Y_m \rangle$, pretende desenvolver-se um algoritmo que determine o tamanho da sua maior subsequência contígua comum.

a. Dadas duas sequências de caracteres $\vec{X} = \langle X_1, ..., X_n \rangle$ e $\vec{Y} = \langle Y_1, \dots, Y_m \rangle$, seja $B(i, j)$ o tamanho do maior sufixo comum entre $\langle X_1, ..., X_i \rangle$ e $\langle Y_1, ..., Y_j \rangle$. Por exemplo, para $\vec{X} =$ *abaabb* e $\vec{Y} =$ *abbbbb*, temos que $B(3,3) = 0$ e $B(6,3) = 3$. Defina $B(i, j)$ recursivamente completando os campos em baixo:

$$B(i,j) = \begin{cases} 0 & \text{se } i = 0 \lor j = 0 \\ \boxed{} & \text{se } \boxed{} \\ \boxed{} & \text{c.c.} \end{cases}$$

Admite-se, para simplificar a formulação, que $B(i, j) = 0$ quando $i = 0$ ou $j = 0$.

b. Complete o template de código em baixo que, dadas duas sequências de caracteres $\langle X_1, \dots, X_n \rangle$ e $\langle Y_1, \dots, Y_m \rangle$, calcula o tamanho da sua maior subsequência contígua comum.

```
LongestContiguousCommonSubstring(x[1..n], y[1..m])
  let B[0..n, 0..m] be a new matrix of size (n+1) × (m+1)
  B[0,0] := [        ]
  for i = 1 to n do
    B[i,0] := [        ]
  endfor
  for j = 1 to m do
    B[0,j] := [        ]
  endfor
  let max = 0
  for i = 1 to n do
    for j = 1 to m do
      [                                        ]
    endfor
  endfor
  return max
```

c. Determine a complexidade assimptótica do algoritmo proposto na alínea anterior.

Solução:

a.

$$B(i,j) = \begin{cases} 0 & \text{se } i = 0 \lor j = 0 \\ B(i-1, j-1) + 1 & \text{se } X_i = Y_j \\ 0 & \text{c.c.} \end{cases}$$

b.

$\texttt{LongestContiguousCommonSubstring}(x[1..n], y[1..m])$
  let $B[0..n, 0..m]$ be a new matrix of size $(n+1) \times (m+1)$
  $B[0,0] := 0$
  for $i = 1$ **to** n **do**
    $B[i,0] := 0$
  endfor
  for $j = 1$ **to** m **do**
    $B[0,j] := 0$
  endfor
  let $max = 0$
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** m **do**
      if $x[i] == y[j]$ **then**
        $B[i,j] := B[i-1, j-1] + 1$
        $max := \mathbf{max}(B[i,j], max)$
      else $B[i,j] := 0$
    endfor
  endfor
  return max

c. Complexidade: $O(n^2)$. O algoritmo tem de preencher toda a matriz $B[0..n, 0..m]$. O preenchimento de cada célula faz-se em tempo constante, $O(1)$.

Q4 (T2 08/09 II.2) Considere o problema de determinar a colocação óptima de parêntesis, que permite reduzir o número de operações na multiplicação de matrizes. Como sabe, o número de operações mínimo para efectuar a multiplicação $A_i\,A_{i+1}\,\ldots\,A_j$ é dado por:

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_kp_j\} & \text{if } i < j \end{cases}$$

Considerando as matrizes A, B, C e D com as seguintes dimensões:

Matriz	Dimensão
A	2×5
B	5×3
C	3×1
D	1×2

Indique qual a colocação óptima de parêntesis para o produto $ABCD$. Para o efeito deverá escrever a expressão do produto $ABCD$, colocando os parêntesis na posição correcta. Adicionalmente, indique os valores de $m[1,2]$, $m[1,4]$, $m[1,3]$ e $m[2,4]$.

Solução:

Expressão	$m[1,2]$	$m[1,4]$	$m[1,3]$	$m[2,4]$
$(A \times (B \times C)) \times D$	30	29	25	25

Q5 (R2 08/09 II.2) Considere o problema da identificação da maior subsequência comum (LCS) entre duas sequências, S e T. Admita que, numa formulação do problema em termos de programação dinâmica, o comprimento da maior subsequência comum entre os prefixos $S_i = \langle s_1, s_2, \ldots, s_i \rangle$ e $T_j = \langle t_1, t_2, \ldots, t_j \rangle$ é definido por:

$$c[i,j] = \begin{cases} 0 & \text{se } i = 0 \vee j = 0 \\ c[i-1,j-1]+1 & \text{se } i,j > 0 \wedge s_i = t_j \\ \max(c[i-1,j], c[i,j-1]) & \text{se } i,j > 0 \wedge s_i \neq t_j \end{cases}$$

Dadas as sequências $S = ABCBCDBBDCABCDB$ e $T = ABBACBDCCDBACD$, indique qual a LCS, bem como os seguintes valores: $c[0,10]$, $c[4,6]$, $c[5,12]$, $c[9,13]$, $c[10,10]$, $c[14,14]$ e $c[15,14]$.

Solução:

LCS	$c[0,10]$	$c[4,6]$	$c[5,12]$	$c[9,13]$	$c[10,10]$	$c[14,14]$	$c[15,14]$
ABCBCDBACD	0	4	5	7	7	10	10