Ejercicio de examen de Programación 3

15 de julio de 2022

Ejercicio 2 (40 puntos)

Un *palíndromo* es una cadena de caracteres que se lee igual de izquierda a derecha que de derecha a izquierda; por ejemplo, *anilina*.

Dada una cadena de caracteres, $x = x_1, x_2, \ldots, x_n$, una *subsecuencia* de x es una cadena de la forma $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$, con $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. En palabras, una subsecuencia de x está formada por la concatenación de caracteres de x, no necesariamente contiguos pero preservando su orden relativo. Por ejemplo, la cadena *ABCDDABE* contiene como subsecuencias a *ABA* y también a *BDB* (entre otras), que son palíndromos de largo 3.

Dada una cadena de caracteres no vacía, x, queremos obtener el largo de las subsecuencias más largas de x que son palíndromos.

Se pide:

(a) Defina una relación de recurrencia que permita calcular el largo máximo entre todas las subsecuencias de x que son palíndromos. Justifique su respuesta explicando la procedencia de cada término.
Sugerencia: Considere una función de la forma $OPT(i, j)$, que resuelve el problema para $x_i, \ldots, x_j$.

(b) Dé un algoritmo eficiente **iterativo**, usando la técnica de Programación Dinámica, para calcular el largo máximo entre todas las subsecuencias de x que son palíndromos.

Solución:

(a) Para $1 \leq i \leq j \leq n$ definimos $OPT(i, j)$ como el máximo largo entre todas las subsecuencias de $x_i, \ldots, x_j$ que son palíndromos.

Las cadenas de largo 1 son en sí mismas palíndromos, por lo cual la mayor subsecuencia que es un palíndromo tiene largo 1, como establece (1). Por otra parte, la subsecuencia más larga de una cadena vacía tiene largo cero, de donde se desprende (2). Estas dos ecuaciones definen el paso base de la recurrencia.

Consideremos ahora una cadena $S = x_i, \ldots, x_j$ con al menos dos caracteres, y sea $P = x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ una subsecuencia de largo máximo entre todas las subsecuencias de S que son palíndromos.

Si $x_i = x_j$, afirmamos que existe una subsecuencia de S, P', que es un palíndromo del mismo largo que P e incluye tanto a x_i como a x_j. En efecto, en P tenemos o bien $i_1 = i$, o bien $i_k = j$, porque en caso contrario $x_i P x_j$ sería una subsecuencia de S que además es un palíndromo de largo mayor que P. Si $i_1 = i$, obtenemos P' reemplazando i_k por j, y si $i_k = j$, obtenemos P' reemplazando i_1 por i.

La subcadena $x_{i_2}, \ldots, x_{i_{k-1}}$ de P' es una subsecuencia de $x_{i+1}, \ldots, x_{j-1}$, que es de largo máximo entre todas sus subsecuencias que son palíndromos, de donde surge el segundo término de (3), al cual se suman dos unidades correspondientes a los caracteres x_i y x_j que forman los extremos de P'.

Si en cambio $x_i \neq x_j$, tenemos o bien $i_1 > i$, o bien $i_k < j$, porque en caso contrario P no sería un palíndromo. Si $i_1 > i$, entonces P es una subsecuencia de $x_{i+1}, \ldots, x_j$ cuyo largo es como máximo $OPT(i+1, j)$. Si $i_k < j$, entonces P es una subsecuencia de $x_i, \ldots, x_{j-1}$ cuyo largo es como máximo $OPT(i, j-1)$. Como P es de largo máximo, su largo debe coincidir con el máximo de estos dos valores, como expresa la ecuación (4).

$$OPT(i,i) = 1, \quad 1 \leq i \leq n, \tag{1}$$

$$OPT(i,j) = 0, \quad 1 \leq j < i \leq n, \tag{2}$$

$$OPT(i,j) = 2 + OPT(i+1, j-1), \quad 1 \leq i < j \leq n, x_i = x_j, \tag{3}$$

$$OPT(i,j) = \text{máx}\left\{OPT(i, j-1), OPT(i+1, j)\right\}, \quad 1 \leq i < j \leq n, x_i \neq x_j. \tag{4}$$

```
1  Algorithm MaximoPalindromo
       /* Incialización según paso base definido en (1) y (2)                    */
2      for i = 1 to n do
3          OPT [i, i] = 1
           /* Incialización según (2). Alternativamente es suficiente definir
              OPT[i, i − 1] = 0 para i > 1.                                       */
4          for j = 1 to i − 1 do
5              OPT [i, j] = 0
6          end
7      end
8      for i = n − 1 downto 1 do
9          for j = i + 1 to n do
               /* Se aplica (3) o (4) según corresponda                           */
10             if x_i = x_j then
11                 OPT [i,j] = 2 + OPT [i + 1, j − 1]
12             else
13                 OPT [i,j] = máx{OPT[i + 1, j], OPT[i, j − 1]}
14             end
15         end
16     end
17     return OPT [1, n]
18 end
```

Figura 1: Algoritmo para encontrar el largo maximo de una subsecuencia palindromica.

(b)