# Algoritmo Manacher - Palíndromo más largo

Pedro O. Pérez M., PhD.

Análisis y diseño de algoritmos avanzados
Tecnológico de Monterrey

*pperezm@tec.mx*

09-2022

# Contenido

# Palíndromo más largo

Un palíndromo es una cadena que se lee igual de izquierda a derecha que de derecha a izquierda. En algunas aplicaciones es necesario encontrar las subcadenas que son palíndromos, dentro de un string dado, o el palíndromo más grande que exista como subcadena de un cadena dada. El problema del palíndromo más largo y se define así:

- Dada una cadena $S$, se desea encontrar la subcadena de la misma que forma un palíndromo y que es el más largo que se pueda encontrar, es decir, el que está formado con el mayor número de caracteres.

# Algoritmo de fuerza bruta

La forma más simple de resolver el problema del palíndromo más largo de una cadena $S$ es usar fuerza bruta: se recorren todas las posiciones, $i$, de la cadena, y para cada una de ellas se trata de expandir hacia la izquierda y a la derecha hasta encontrar la subcadena más grande, lo cual se debe hacer tanto para cadenas de longitud impar como para longitud par. Complejidad $O(n^2)$.

**Procedure 1** LONGEST_PALINDROME

**Input:** S : String

```
maxLong ← 1
startAt ← 0
for i ← 1 to S.length do
    /* LONGITUD IMPAR */
    NEXT_SLIDE
    /* LONGITUD PAR */
    NEXT_SLIDE
end for
return PAIR(startAt, maxLong)
```

```
/* LONGITUD IMPAR */
j ← 1
while (i − j) ≥ 1 and (i + j) ≤ n and S[i − j] = S[i + j] do
  j ← j + 1
end while
if (2 ∗ j) − 1 > maxLong then
  maxLong ← (2 ∗ j) + 1
  startAt ← i − j + 1
end if
```

```
/* LONGITUD PAR */
j ← 1
while (i − j − 1) ≥ 1 and (i + j) ≤ n and S[i − j − 1] = S[i + j] do
  j ← j + 1
end while
if (2 ∗ j) > maxLong then
  maxLong ← 2 ∗ j
  startAt ← i − j
end if
```

# Algoritmo de Manacher

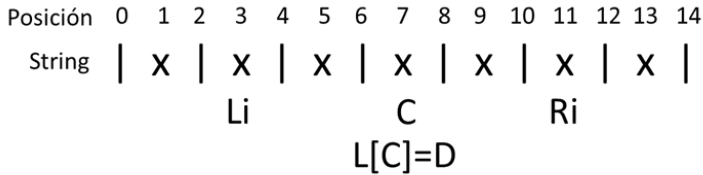
- Este algoritmo, propuesto por Glenn K. Manacher en 1975, logra resolver el problema del palíndromo más largo de una cadena $S$, en una forma muy eficiente, con una complejidad O(n), proporcional a la longitud n de la cadena.
- Al igual que el algoritmo a fuerza bruta, éste se basa en buscar un palíndromo centrado en $i$, con algunas diferencias:
  - En el algoritmo a fuerza bruta, se considera si la cadena es longitud par o impar. Por otra parte, el algoritmo de Manacher se basa en una nueva cadena, $T$, a la que se le agregan posiciones intermedias, una al inicio y otra al final de la cadena, lo que garantiza que todo el proceso se realizará sobre una cadena de longitud impar.
  - Los valores calculados se almacenan en un arreglo, $L$, que tiene la longitud de la nueva cadena.
  - Se utilizan los valores previamente calculados (a la izquierda de $i$) que están almacenados en $L$, para calcular los valores que siguen (a la derecha de $i$).

- El algoritmo inicial con la construcción de la nueva cadena, $T$, agregando un carácter especial en medio de cada uno de los caracteres de la cadena original, además de una al inicio y otro al final.
- El carácter que se agrega puede ser cualquiera que no pertenezca al alfabeto de la cadena analizada.
- Por ejemplo,
  - Si tenemos la cadena $S = baab$, de longitud 4, al agregar el carácter "|", la nueva cadena sería $T = |b|a|a|b|$, la cual tiene longitud de 9.
  - Si tenemos la cadena $S = bacab$, de longitud 5, al agregar el carácter "|", la nueva cadena sería $T = |b|a|c|a|b|$, la cual tiene longitud de 11.

Al finalizar el algoritmo, el valor en la posición $i$ del arreglo $L$ indica la longitud, en número de caracteres, que tiene el palíndromo más grande centrado en $i$, medido a la derecha o a la izquierda de $i$.

| Posición | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| String | \| | c | \| | a | \| | c | \| | b | \| | c | \| | a | \| | c | \| |
| L | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 7 | 0 | 1 | 0 | 3 | 0 | 1 | 0 |

Manacher se dio cuenta que la propiedad de simetría de un palíndromo podría ayudar en el cálculo de L aprovechando los valores que ya se habían calculado anteriormente. Sin embargo, las condiciones que se deben cumplir para poder disminuir la cantidad de cálculos no son triviales.




Notación:

- **C**: La última posición calculada de $L$. Todos los valores en posiciones menores o iguales en $C$ en $L$ se conocen.
- $R_i$: Posición a la derecha de $C$ a la cuál se le quiere calcular su valor $L$.
- $L_i$: Posición a la izquierda de $C$ que está separada exactamente el mismo número de caracteres de $C$ que $R_i$. Esto implica que $R_i - \mathrm{C} = \mathrm{C} - L_i$.

**Procedure 2** MANACHER

**Input:** S : String

$Q \leftarrow EXPAND(S)$
$P \leftarrow Array(Q.length)$
$center \leftarrow 1, right \leftarrow 1$
**for** $i \leftarrow 2$ **to** $Q.length$ **do**
  $iMirror \leftarrow center - (i - center)$
  **if** $right > i$ **then**
    $P[i] \leftarrow min(right - i, P[iMirror])$
  **end if**
  **while** $Q[i + 1 + P[i]] = Q[i - 1 - P[i]]$ **do**
    $P[i] \leftarrow P[i] + 1$
  **end while**
  **if** $i + P[i] > right$ **then**
    $center \leftarrow i$
    $right \leftarrow i + P[i]$
  **end if**
**end for**

```
maxPalindrome ← 0
centerIndex ← 0
for i ← 2 to Q.length do
  if maxPalindrome < P[i] then
    maxPalindrome ← P[i]
    centerIndex ← i
  end if
end for
return PAIR(((centerIndex − 1 − maxPlaindrome)/2), maxPalindrome)
```