String Matching usando Transformada Discreta de Fourier

Notación

Considerando el problema de matching exacto de cadenas de caracteres, sean:

- t: Un texto
- p: Un patrón a buscar dentro del texto
- $t[n]$: Una representación numérica del n-ésimo caracter de t
- $p[n]$: Una representación numérica del n-ésimo caracter de p
- N: La longitud del texto t
- M: La longitud del patrón p
- $((x))_a$: La operación x mod a

Planteo Formal

El problema se puede escribir como:

$$\sum_{k=0}^{M-1} |\, t\,[k+i] \;-\; p\,[k]\,| = 0 \;\Leftrightarrow\; \textit{Hay un match que comienza en la posición } i.$$

O bien, por conveniencia, calculamos la convergencia cuadrática (notar que no estamos elevando ambos miembros al cuadrado, sino cada término):

$$\sum_{k=0}^{M-1} (t[k+i]-p[k])^2 = 0 \;\Leftrightarrow\; \textit{Hay un match que comienza en la posición } i.$$

Desarrollando cada binomio de la sumatoria:

$$\sum_{k=0}^{M-1} \left(t[k+i]^2 - 2t[k+i]p[k] + p[k]^2 \right) = 0$$

Multiplicando punto a punto por el arreglo $t[i+k]$:

$$\sum_{k=0}^{M-1} \left(t[k+i]^3 - 2t[k+i]^2 p[k] + t[k+i]\, p[k]^2 \right) = 0$$

Y haciendo lo propio por el arreglo $p[k]$:

$$\sum_{k=0}^{M-1} \left(t[k+i]^3 p[k] - 2t[k+i]^2 p[k]^2 + t[k+i] p[k]^3 \right) = 0$$

Extendiendo con ceros el arreglo *p*, puedo cambiar los límites de la sumatoria:

$$\sum_{k=0}^{N-1}\left(t[((k+i))_N]^3 p[((k))_N] - 2t[((k+i))_N]^2 p[((k))_N]^2 + t[((k+i))_N]p[((k))_N]^3\right) = 0$$

Por asociatividad de la suma:

$$\sum_{k=0}^{N-1} t[((k+i))_N]^3 p[((k))_N] - 2\sum_{k=0}^{N-1} t[((k+i))_N]^2 p[((k))_N]^2 + \sum_{k=0}^{N-1} t[((k+i))_N]p[((k))_N]^3 = 0$$

Aplicando la definición de convolución circular:

$$t[i]^3 \odot p[M-1-i] - 2t[i]^2 \odot p[M-1-i]^2 + t[i] \odot p[M-1-i]^3 = 0$$

Sean:

- $T_i[k]$: La DFT de $t[n]^i$.
- $P_i[k]$: La DFT de $p[n]^i$.

Y las siguientes propiedades de la transformada discreta de Fourier:

- $DFT_N\{ x[((n-a))_N] \} = e^{-j(2\pi/N)ak} X[k]$
- $DFT_N\{ x^*[((-n))_N] \} = X^*[k]$

Puedo transformar ambos miembros de la ecuación como:

$$T_3[k]\, P_1^*[k]\, e^{j(2\pi/N)(M-1)k} - 2\, T_2[k]\, P_2^*[k]\, e^{j(2\pi/N)(M-1)k} + T_1[k]\, P_3^*[k]\, e^{j(2\pi/N)(M-1)k} = 0$$

Aplicando los productos puntuales en cada término:

$$R_1[k]\, e^{j(2\pi/N)(M-1)k} - 2\, R_2[k]\, e^{j(2\pi/N)(M-1)k} + R_3[k]\, e^{j(2\pi/N)(M-1)k} = 0$$

Y aplicando la transformada inversa:

$$r_1[((i+M-1))_N] - 2\, r_2[((i+M-1))_N] + r_3[((i+M-1))_N] = 0$$

Finalmente, si se cumple esta ecuación, hay un match que comienza en la posición i de t.

Además, se observan las siguientes igualdades:

$$r_1[((i+M-1))_N] = t[i]^3 \odot p[M-1-i]$$

$$r_2[((i+M-1))_N] = t[i]^2 \odot p[M-1-i]^2$$

$$r_3[((i+M-1))_N] = t[i] \odot p[M-1-i]^3$$

Algoritmo

A partir del planteo anterior y utilizando como primitiva una implementación de la transformada rápida de Fourier (con orden algorítmico O(N*log(N))), podemos escribir el siguiente algoritmo:

```
# Sea la Transformada Rápida de Fourier (Fast Fourier Transform):
Arreglo<N> FFT(Arreglo<N>)

# Se instancian los arreglos
t_1, t_2, t_3, p_1, p_2, p_3 = Arreglos de largo N inicializado con ceros

# Representación numérica del texto
for i in 0..N-1:
    t_1[i] := Representación numérica del i-ésimo caracter del texto
    t_2[i] := t_1[i] * t_1[i]
    t_3[i] := t_2[i] * t_1[i]

# Representación numérica del patrón invirtiendo el dominio
for i in 0..M-1:
    p_1[M-i-1] := Representación i-ésimo caracter del pattern
    p_2[M-i-1] := p_1[M-i-1] * p_1[M-i-1]
    p_3[M-i-1] := p_2[M-i-1] * p_1[M-i-1]

# Cálculo de las transformadas discretas de Fourier del texto
T_1 := FFT(t_1)
T_2 := FFT(t_2)
T_3 := FFT(t_3)

# Cálculo de las transformadas discretas de Fourier del patrón
P_1 := FFT(p_1)
P_2 := FFT(p_2)
P_3 := FFT(p_3)

# Productos punto a punto de las transformadas
R_1 := T_3 * P_1
R_2 := T_2 * P_2
R_3 := T_1 * P_3

# Cálculo de las transformadas inversas de los productos
r_1 := IFFT(R_1)
r_2 := IFFT(R_2)
r_3 := IFFT(R_3)

for i in 0..N-M:
    if r_1[M-1-i] - 2 * r_2[M-1-i] + r_3[M-1-i] == 0:
        Mostrar match en la posición i
```

Orden del Algoritmo

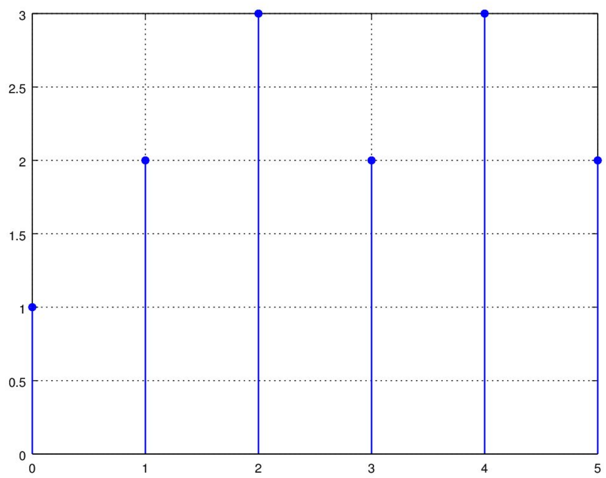
Como la ***FFT*** tiene orden ***O(N*log(N))***, y el resto de las operaciones del algoritmo son de orden lineal, se conservará el orden de la ***FFT***.

Ejemplo gráfico

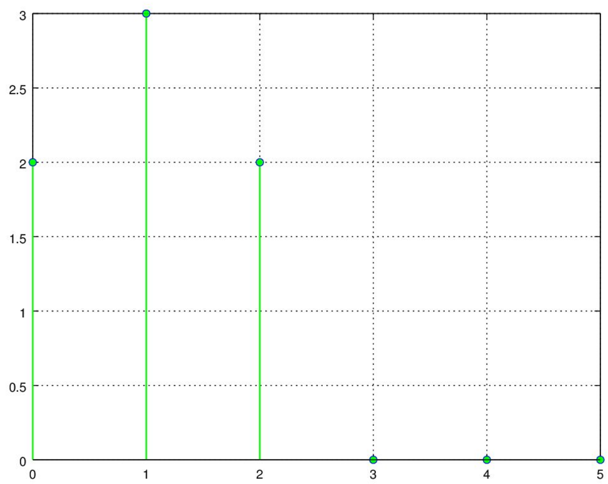
Buscaremos la subcadena `‘ANA’` dentro de la cadena `‘BANANA’`.
Tenemos entonces:

- `t = {1, 2, 3, 2, 3, 2}`
- `p = {2, 3, 2, 0, 0, 0}`
- `El patrón invertido será también p’ = {2, 3, 2, 0, 0, 0}`
- `N = 6`
- `M = 3`

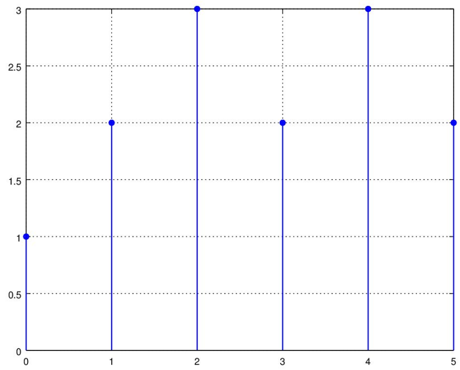
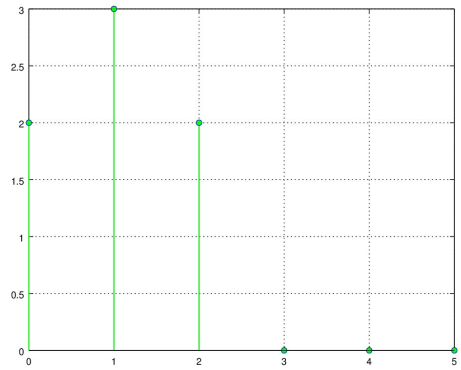
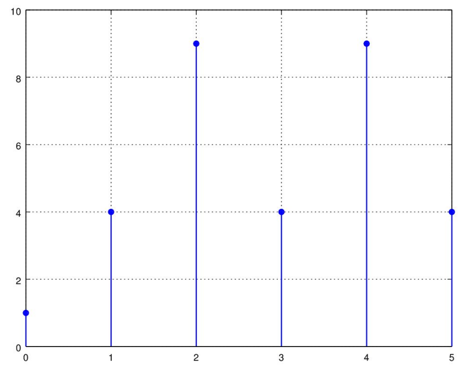
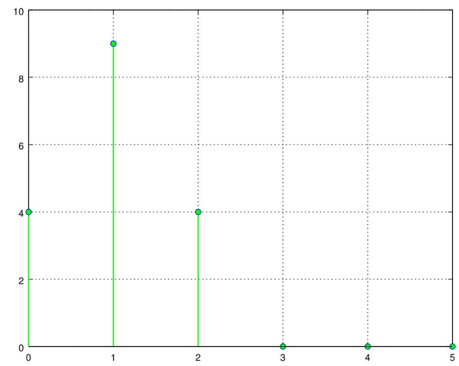
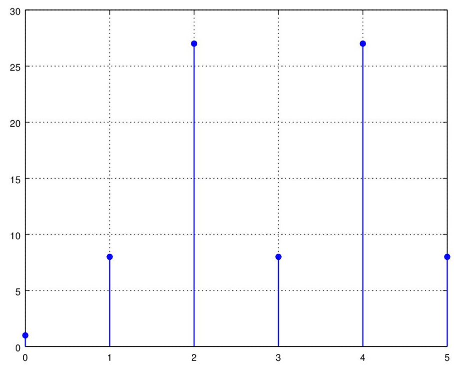
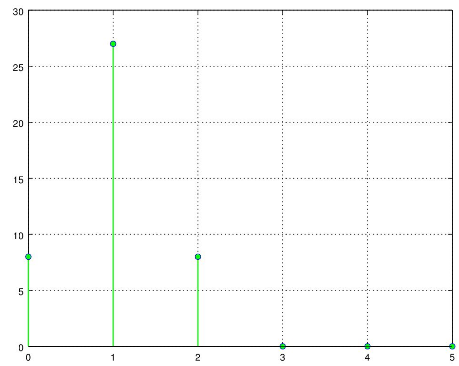
El texto representado como señal se verá como:



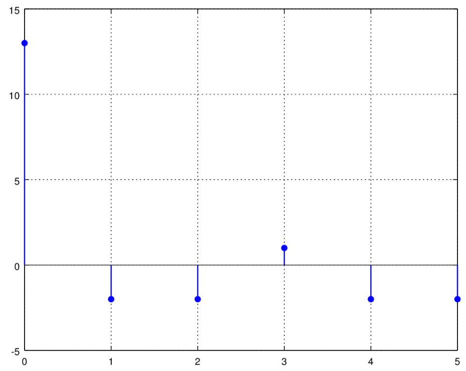
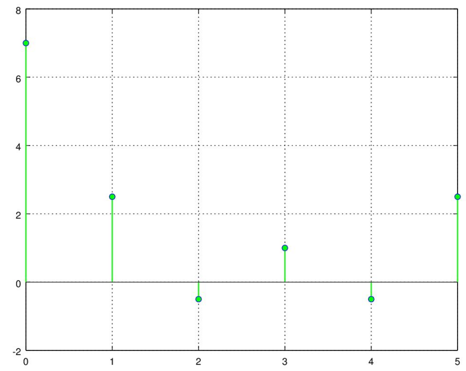
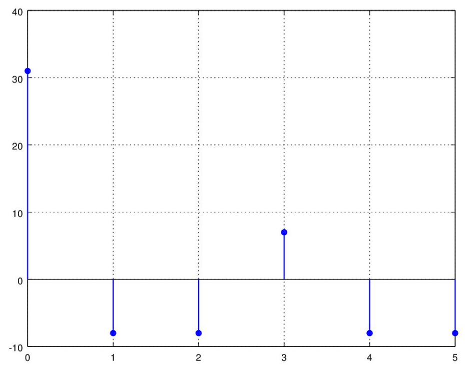
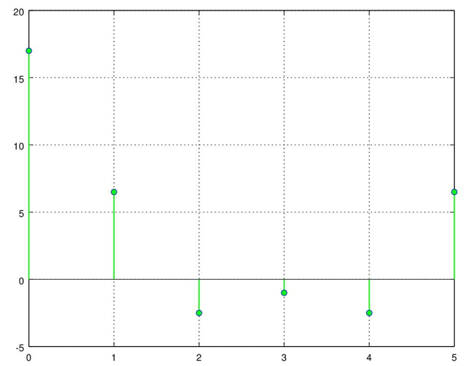
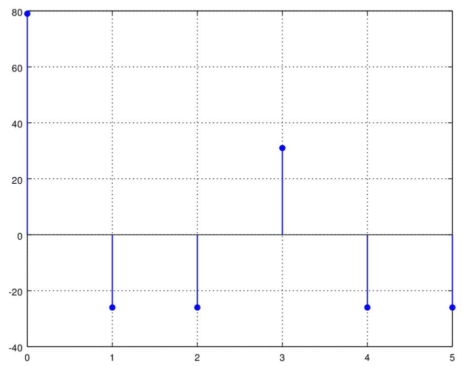
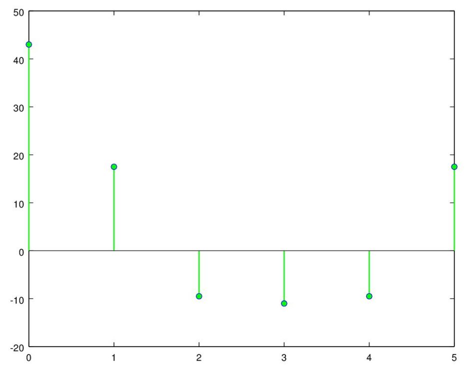
Y el patrón (ya invertido) como:



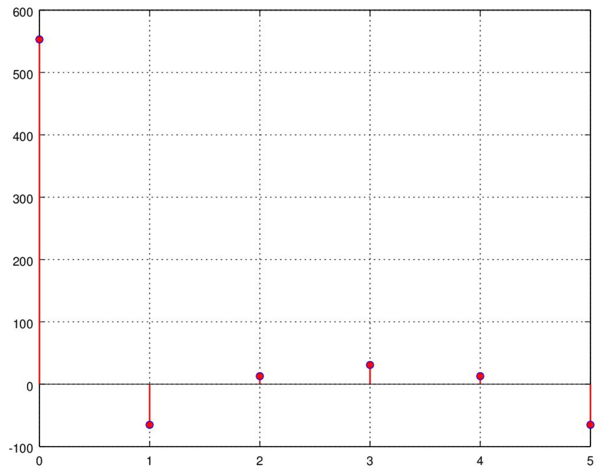
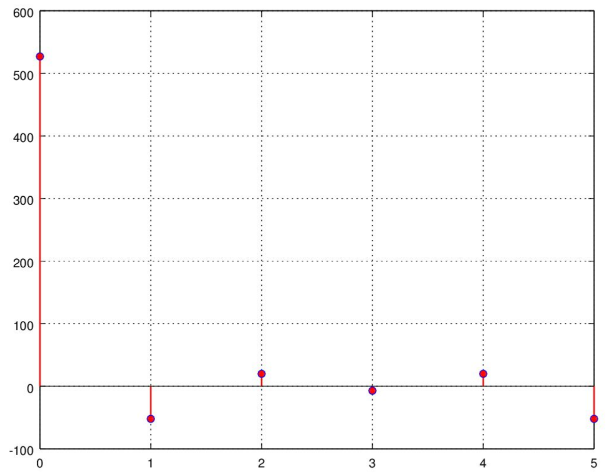
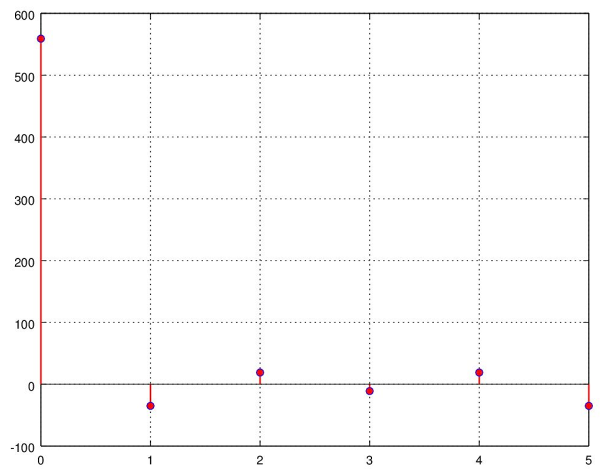
Las señales elevadas punto a punto al cuadrado y al cubo:

	Texto	Patrón
x		
x^2		
x^3		

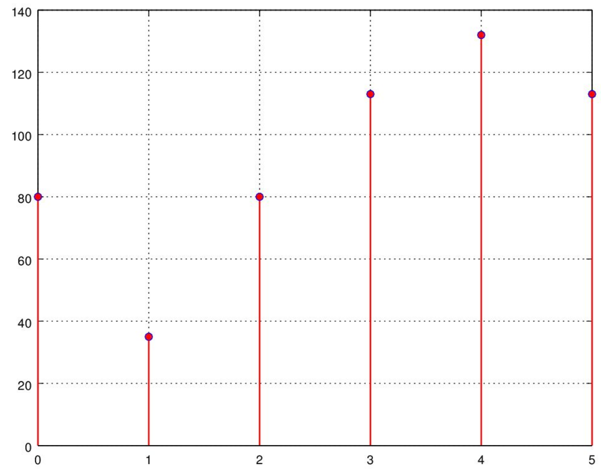
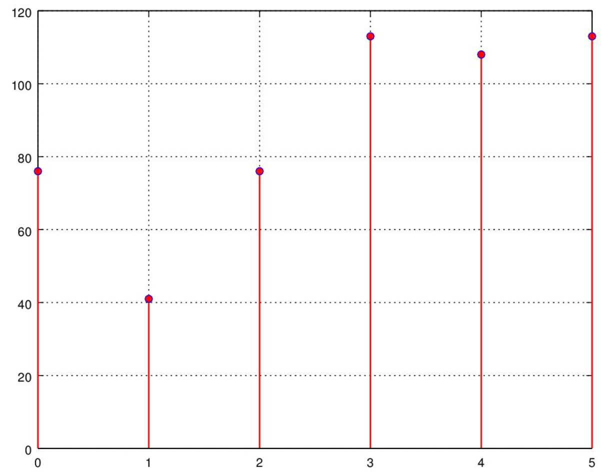
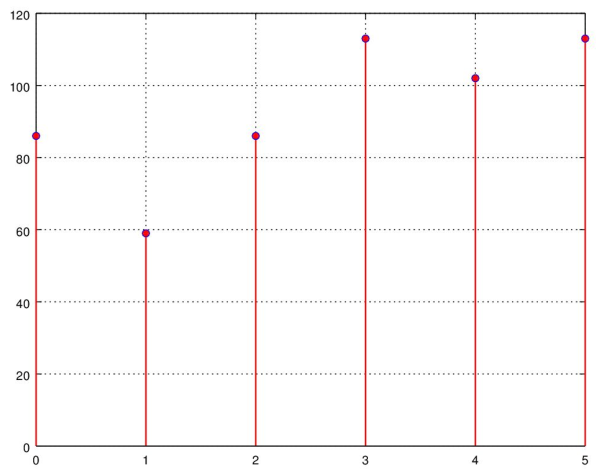
Sus transformadas de Fourier discretas (tomo parte real para graficar en dos dimesiones):

	Texto	Patrón
X_1		
X_2		
X_3		

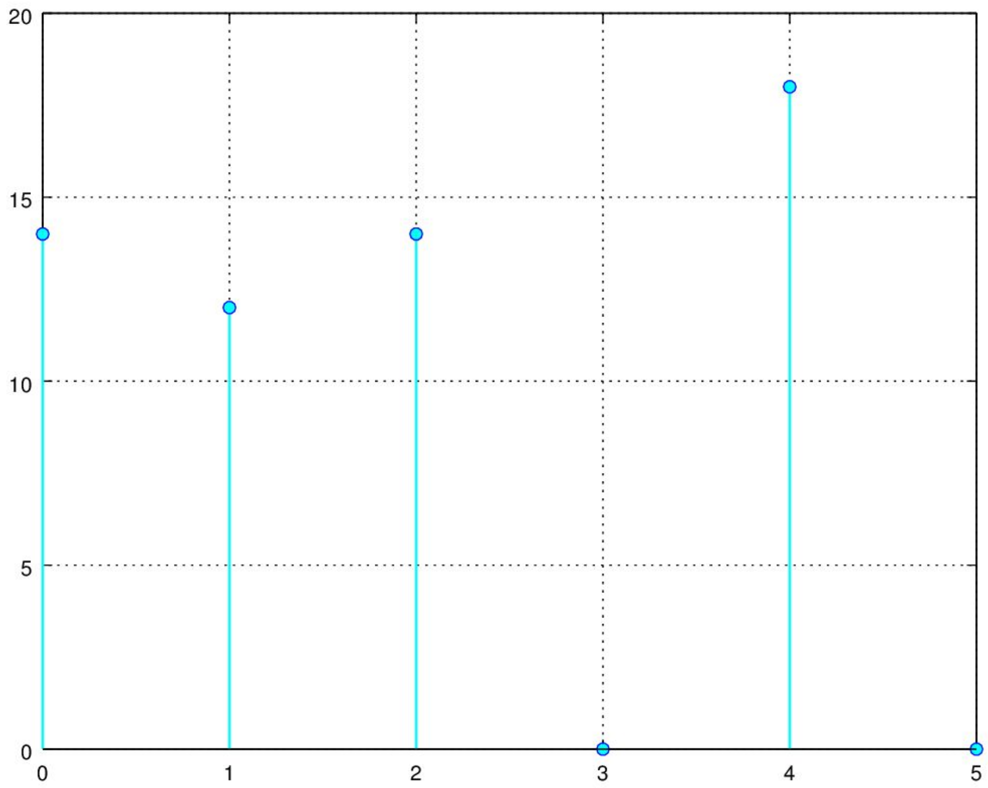
Haciendo el producto punto a punto de las señales:

$R_1 = T_3P_1$	
$R_2 = T_2P_2$	
$R_3 = T_1P_3$	

Aplicando la transformada discreta inversa de Fourier:

$r_1 = \mathrm{IDFT}(R_1)$	
$r_2 = \mathrm{IDFT}(R_2)$	
$r_3 = \mathrm{IDFT}(R_3)$	

Y por último, aplicando la combinación lineal $r_1[i]\ -\ 2\ r_2[i]\ +\ r_3[i]$ punto a punto:



Vemos que hay ceros en $i=3$ e $i=5$, lo cual quiere decir que los matches terminan en las posiciones 3 y 5 del texto (empiezan en $n=3+M-1=1$ y $n=5+M-1=3$).

Adaptaciones del Algoritmo

- Si buscamos sub-cadenas, sólo tenemos en cuenta las posiciones entre $M-1$ y $N-1$ de la señal resultante, las anteriores son útiles para buscar sub-cadenas en el texto visto de manera circular (encontrar un match desde la posición -2 hasta la 3, por ejemplo).
- Si queremos buscar con "*don't cares*" (cualquier caracter), simplemente tenemos que poner un cero en la representación del caracter correspondiente en la representación numérica del patrón.