

Escritura del problema de binario invertido

Jorge Luis Esposito Albornoz¹ Juan Sebastián Herrera Guaitero¹

¹Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
{jesposito,jsebastianherrera}@javeriana.edu.co

16 de agosto de 2022

Resumen

En este documento se presenta la formalización del problema de conversión de un número natural a su representación binaria inversa. **Palabras clave:** binaria, inverso.

Índice

1. Formalización del problema	1
1.1. Definición del problema del “Binario invertido”	1
2. Algoritmos de solución	2
2.1. Idea de solución iterativa	2
2.2. Método iterativo	2
2.2.1. Análisis de complejidad	2
2.2.2. Invariante	2
2.3. Idea de solución DV	3
2.4. Método Divide y vencerás	3
2.4.1. Análisis de complejidad	3
2.4.2. Invariante	3

1. Formalización del problema

El problema de la representación binaria inversa es uno que a simple vista puede resultar sencilla, si bien este solo consiste en convertir un número de su base 10 a su base 2, la dificultad particular radica en los algoritmos que se implementen para llegar a dicha solución. En esta ocasión por medio de la teoría y experimentación se busca determinar si para este caso particular conviene más usar un algoritmo iterativo o utilizar uno del modelo dividir y vencer.

1.1. Definición del problema del “Binario invertido”

Así, el problema de la representación binaria inversa se define a partir de:

1. un número natural

Una secuencia que representa dicho número en su base 2 inversa

■ Entradas:

- $\langle x \in \mathbb{N} \rangle \mid 0 \leq x$.

■ Salidas:

- $S' = \langle s_i = 0 \vee 1 \rangle$.

2. Algoritmos de solución

2.1. Idea de solución iterativa



Figura 1: Idea solución iterativo

2.2. Método iterativo

La idea de este algoritmo es utilizar el módulo 2 para ir calculando cada posición del número binario.

Algoritmo 1 Representación binaria invertida Iterativa.

Require: $\langle x \in \mathbb{N} \rangle \mid 0 \leq x$.

Ensure: S' con los valores de la representación binaria inversa.

```

1: procedure BINARY( $n$ )
2:   while  $n/2 > 0$  do
3:      $vec.pushback(n \% 2)$ 
4:      $n = n/2$ 
5:   end while
6:    $vec.pushback(n \% 2)$ 
7:   return  $vec$ 
8: end procedure

```

2.2.1. Análisis de complejidad

Este algoritmo tiene una complejidad $O(n)$ debido que se recorre todo el arreglo.

2.2.2. Invariante

Después de cada iteración controlada por el contador i , se guarda el elemento de mayor coincidencia.

1. Inicio: $n = 0$, $n = 1$.
2. Iteración: Por cada operación $\%2$ se guarda en una posición del vector.
3. Terminación: Se imprime la secuencia generada que contiene el inverso del binario.

2.3. Idea de solución DV

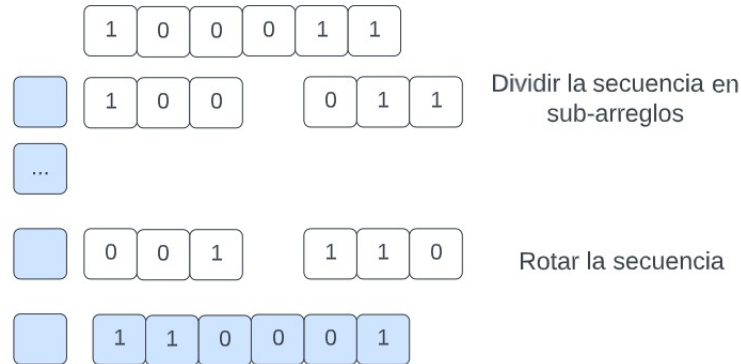


Figura 2: Idea solución DV

2.4. Método Divide y vencerás

Este algoritmo a partir de dos iteradores que representa el inicio y el final de una cadena de texto, se busca dividir el problema e ir rotando la cadena para obtener el inverso.

Algoritmo 2 Representación binaria inversa Divide y Vencerás.

```

1: procedure REVERSEROTATE(first, last)
2:   if (first = last)  $\vee$  NEXT(first) = last then                                 $\triangleright$  NEXT siguiente posición del iterador
3:     return
4:     middle  $\leftarrow$  first
5:   end if
6:   dis  $\leftarrow$  DISTANCE(first, last)/2       $\triangleright$  DISTANCE calcula el número de elementos entre dos iteradores
7:   ADVANCE(middle, dis)                       $\triangleright$  ADVANCE avanza el iterador
8:   REVERSEROTATE(first, middle)
9:   REVERSEROTATE(middle, last)
10:  ROTATE(first, middle, last)
11: end procedure

```

2.4.1. Análisis de complejidad

Ecuación de recurrencia: $T(n) = 2T(n/2) + O(n)$ esta ecuación es similar a la del algoritmo merge sort, por lo cual sabemos que tiene una complejidad $O(n \log n)$.

2.4.2. Invariante

1. Inicio: $n = 0$, $n = 1$.
2. Iteración: Se invierte cada sub arreglo y proceden a unirse obteniendo así la secuencia binaria invertida.
3. Terminación: Se imprime la secuencia generada que contiene el inverso del binario.