Escritura de los Algoritmos Binario Inverso

Pablo Santander Álvarez, Steban Vanegas Camacho

Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
santander_pjaveriana.edu.co, steban_vanegasc@javeriana.edu.co

18 de agosto de 2022

Resumen

En este documento se presenta la descripcion de dos algoritmos que solucionan el caso de: dado un número natural, calcular y retornar su representación binaria inversa. Además se presentan analisis experimentales para la complejidad de los algoritmos. **Palabras clave:** binario, secuencia, inverso, algoritmo, formalización, experimentación, complejidad.

Índice

1. Introducción

La identificación del número binario inverso de un número natural es un problema que puede resolverse mediante una cantidad amplia de técnicas. En el presente documento plasmaremos dos algoritmos que cumplen este proposito, siendo uno de ellos de naturaleza iterativa, y el otro recursiva. El objetivo es mostrar: la formalización del problema (sección 2), la escritura formal de dos algoritmos (sección 3) y un análisis experimental de la complejidad de ambos (sección **??**).

2. Formalización del problema

Para el problema de *identificación del número binario inverso* de un número $n|n \in \mathbb{N}$. Podemos partir de que $n = \frac{n}{2}|n \in \mathbb{N}$, donde n es el resultado entero de la división de n entre 2. Además, si posterior a la división

$n\,\%2 = 0$, entonces su representación binaria es un 0, de lo contrario es un 1. Esta técnica se usa para ir dividiendo nuestro n a la vez que obtenemos su representación binaria luego de cada división. Entonces, si $n >= 1 | n \in \mathbb{N}$, es posible determinar el número binario inverso de cualquier n que cumpla esas condiciones, resultando entonces en un $n' | n' \in \mathbb{N}$ siendo n' el número natural que resulta del número binario inverso de n.

2.1. Definición del problema de Binario Inverso

Así, el problema de identificacion del número binario inverso de un número natural se define a partir de:

1. Un número n

Produciendo un número n' cuya representación binaria sea la inversa de la de un número n.

- Entradas:
  - $n >= 1| \in \mathbb{N}$.
- Salidas:
  - $n' \neq n| \in \mathbb{N}$.

3. Algoritmos de solución

3.1. Binario Inverso Iterativo

La idea de este algoritmo es: Mediante un primer ciclo, se divide el número natural entre 2 por cada iteracion, donde a su vez se valida si la division entera resultante es modulo de dos, esto se hace para determinar si es par o impar y poder ir asignando una representacion binaria a cada resultado. Esta representación binaria se almacena en una secuencia empezando desde el final, asegurando entonces que por cada iteracion se va almacenando el número binario de manera invertida. Finalmente, se realiza un ciclo adicional para convertir la secuencia binaria invertida obtenida anteriormente en un número decimal.

Algoritmo 1 Binario Inverso Iterativo.

Require: $n >= 1| \in \mathbb{N}$
Ensure: $n' \neq n| \in \mathbb{N}$

```
1: procedure BinarioInverso(n)
2:     vector ← v
3:     i ← 1
4:     decimal ← 0
5:     while n >= 1 do
6:         if n%2 = 0 then
7:             InsertarAlFinal(v, 0)
8:         else
9:             InsertarAlFinal(v, 1)
10:        end if
11:        n ← n/2
12:    end while
13:    for i ← 1 to |v| do
14:        decimal ← (decimal ≪ 1) + v[i]
15:    end for
16: end procedure
```

3.1.1. Análisis de complejidad

Por inspección de código: El algoritmo posee dos ciclos los cuales no estan anidados, el peor de los casos realmente seria recibir un número natural n que sea grande, esto debido a que nuestro primer ciclo divide este número entre dos por cada iteración, y claramente entre más grande sea el número, más se demorara. Podemos establecer entonces que el primer ciclo tiene una notacion $O(\log_2(n))$. El segundo ciclo debe recorrer la totalidad de la secuencia, asi que su mejor o peor caso es el mismo, por ende, su notacion es $O(|v|)$. En conclusion, la complejidad algoritmica de la totalidad del algoritmo es $O(\log_2(n)+|V|)$ lo cual es equivalente a $O(n)$.

3.1.2. Invariante

Después de cada iteración en el primer ciclo, el número n es dividido entre dos, donde posteriormente se valida si es par o impar y se almacena su representacion binaria en una secuencia iniciando desde el final, la secuencia invertida es posteriormente transformada en un número decimal.

1. Inicio: Un número $n >= 1$, otro número $n' = 0$ y una secuencia v vacia.
2. Iteración: $n = n/2$, la nueva iteración dividira el elemento n entre dos y llevará la representacion binaria $1 \vee 0$ a la última posicion de la secuencia v donde $v[-1] = 1 \vee 0$
3. Iteración 2: v siendo una secuencia que almacena el número binario invertido, por cada iteración i, $n' = (n' \ll 1) + v[i]$, donde n' se le asigna el valor de operación de cambio bits hacia la izquierda.
4. Terminación: $n' \neq n$, al finalizar el último ciclo, se obtiene un n' que representa el valor decimal del número binario invertido.

3.2. Binario Inverso Recursivo

La idea de este algoritmo es: Mediante 2 funciones recursivas lograr que un número natural dado, se obtenga una representación binaria inversa del mismo. La primer función llamada "DecToBin", se encarga de recursivamente convertir el número natural a una cadena binaria, dividiendo el número entre 2 por cada recursión para saber su módulo, a esto le sumamos 10 y lo multiplicamos por la recursión del número decimal dividido en 2 ya que esto nos permitirá representar cada dígito en binario. Posteriormente se recurre a la

siguiente función recursiva llamada Reversal", se encargará de recibir la cadena de la representación binaria, preguntar si la cadena es de tamaño 1, ya que si es así, la cadena invertida será igual. en caso contrario se irá asignando cada dígito del arreglo n en w de forma recursiva, de tal forma que cada dígito se asigna del último puesto al primero del arreglo de w, siendo w la secuencia binaria invertida final.

Algoritmo 2 Binario Inverso Recursivo.

Require: $n >= 1| \in \mathbb{N}$
Ensure: $n' \neq n| \in \mathbb{N}$

```
1: procedure DecToBin(n)
2:     if n == 0 then
3:         return 0;
4:     else
5:         return (n%2+10*DecToBin(n/2));
6:     end if
7: end procedure
8: procedure Reversal(n)
9:     if —n— = 1 then
10:        [w] ← [n];
11:    else
12:        w[← n_|n|, Reversal(n(1 : |n| − 1)) ]
13:    end if
14:    Return [w];
15: end procedure
16: procedure Main(n)
17:    DecToBin(n);
18:    Reversal(n);
19: end procedure
```

3.2.1. Análisis de complejidad

Por inspección de código: El algoritmo no posee ciclos, el peor de los casos realmente seria recibir un número natural n que sea grande, esto hará que el programa ocupe más recursiones de cada función para completar las tareas de convertir a binario y luego invertir la cadena, Podemos establecer entonces que el programa tiene una notación $O(\log(2n))$, ya que en ambas funciones tiene que recorrer a plenitud los arreglos, En conclusion, la complejidad algoritmica de la totalidad del algoritmo es $O(\log_2(2n) + |V|)$ lo cual es equivalente a $O(2n)$.

3.2.2. Invariante

En cada recursión de la primer función, se obtendrá el módulo del número n, se le sumará 10 y se multiplicará por la recursión dividiendo el número n entre dos, para formar la cadena de su representación binaria. En cada recursión de la segunda función se asignará a un vector de forma recursiva desde su primer posición (0), hasta la última (—n—) la cadena obtenida de la primer función, de forma que el primer elemento del vector final será el último del obtenido luego de la primer función, y el segundo será el penúltimo del obtenido en la primer función y así sucesivamente hasta completar —n—.

1. Inicio: Un número $n >= 1$, otro número $n' = 0$ y una secuencia w vacia.

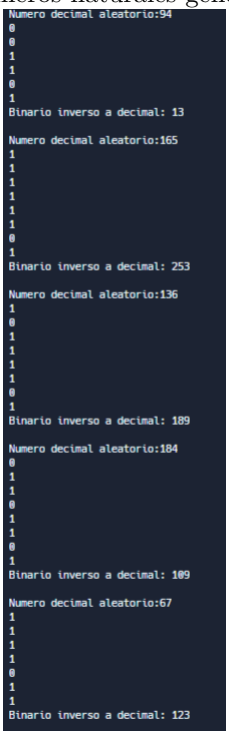
2. Recursión: $n = n/2$, la nueva iteración dividira el elemento n entre dos y llevará la representacion binaria $1 \vee 0$ a la última posicion de la secuencia v donde $v[-1] = 1 \vee 0$

3. Recursión 2: w siendo una secuencia que almacena el número binario invertido, por cada recursión i, $n' = (n' \ll 1) + v[i]$, donde n' se le asigna el valor de operación de cambio binario hacia la izquierda.

4. Terminación: $n' \neq n$, al finalizar la última recursión, se obtiene un n' que representa el valor decimal del número binario invertido.

4. Análisis experimental

En esta sección, se presenta el comportamiento de los algoritmos (binario inverso) de manera iterativa y "divide y vencerás". Para analizar el comportamiento del algoritmo iterativo se realizaron un total de 20 experimentos, utilizando números naturales aleatorios entre 25 y 200, cada experimento fue comprobado de manera manual, a razón de comprobar el correcto funcionamiento del algoritmo. Es importante resaltar que la experimentación con estos algoritmos los algoritmos no busca explicar y demostrar la velocidad con la que el algoritmo procesa un número natural y obtiene su binario inverso, es más bien la busqueda de la precisión del algoritmo y si este logra reproducir todas las respuestas correctas dentro de un rango de números naturales generados aleatoriamente.

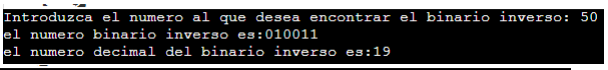


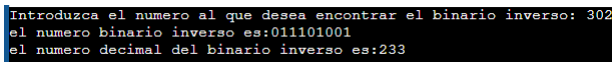
En los 20 experimentos realizados con el algoritmo iterativo se obtuvieron los resultados esperados, no hubo caso alguno que se saliera del margen de lo considerado como "respuesta correcta" que no es más que que el resultado existoso del proceso del algoritmo.

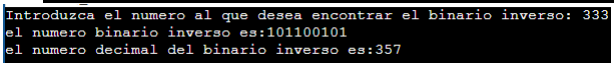
Ejemplos de pruebas experimentales:

1. Entrada: Número natural 170
2. Binario Inverso: 0010001
3. Salida: Número natural inverso 17

Para analizar el comportamiento del algoritmo "Divide y vencerás" se realizaron un total de 20 experimentos igualmente, utilizando 20 números naturales generados aleatoriamente por un programa externo, cada experimento fue comprobado de manera manual, a razón de comprobar el correcto funcionamiento del algoritmo.







En los 20 experimentos realizados con el algoritmo "Divide y vencerás"se obtuvieron los resultados esperados al igual que en el algoritmo iterativo, no hubo caso alguno que se saliera del margen de lo considerado como "respuesta correcta" que no es más que que el resultado existoso del proceso del algoritmo.

5. Conclusiones

1. Los números decimales inversos, en algunos casos fueron mayores y en otros menores al número natural de entrada, por lo tanto no se puede establecer una norma de orden parcial $a > b$ ó $b < a$ puntualmente, pues no se logró identificar una constante o forma de identificar cuándo el inverso podrá ser mayor o menor al natural original, sin embargo en los experimentos tampoco nos topamos con que el inverso de un número fuera igual al natural original.

2. A pesar de que por ambos algoritmos se llega a la respuesta exacta, la forma de aplicar ambos algoritmos resultó muy diferente, logramos converger en que el algoritmo iterativo se logra de manera muy intuitiva, contrario al algoritmo "Divide y vencerás" para el cual hay que llegar un paso más allá de la intuición, por lo tanto, llega a ser más complejo de implementar. Sin embargo, se logra que por simple inspección de código el algoritmo "Divide y vencerás" logra ser significativamente más compacto que el algoritmo iterativo.