Escritura del problema de binario invertido

Jorge Luis Esposito Albornoz[1] Juan Sebastián Herrera Guaitero[1]

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

18 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

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

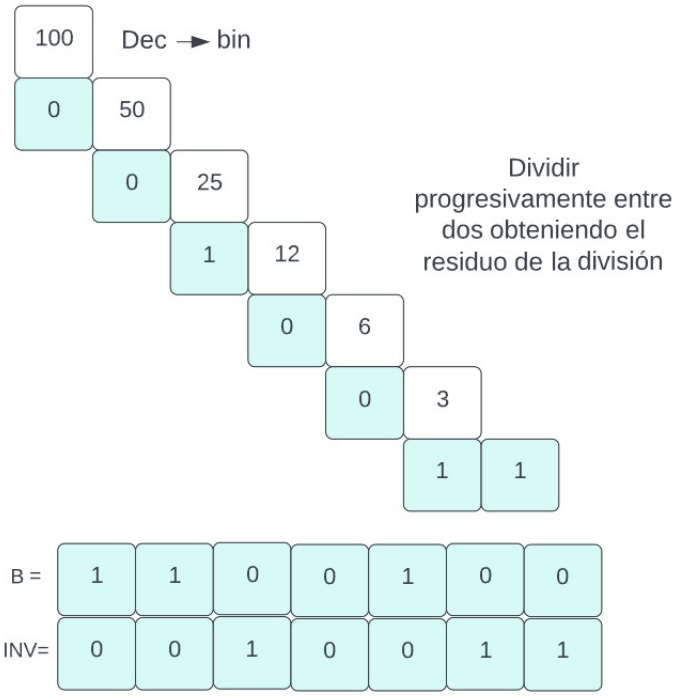

Figura 1: Idea solución iterativo

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($log(n)$) debido que el programa se ejecutará únicamente mientras $\frac{n_i}{2} > 0$.

2.2.2. Invariante

Después de cada iteración controlada por $n/2 > 0$

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

2.3. Idea de solución DV

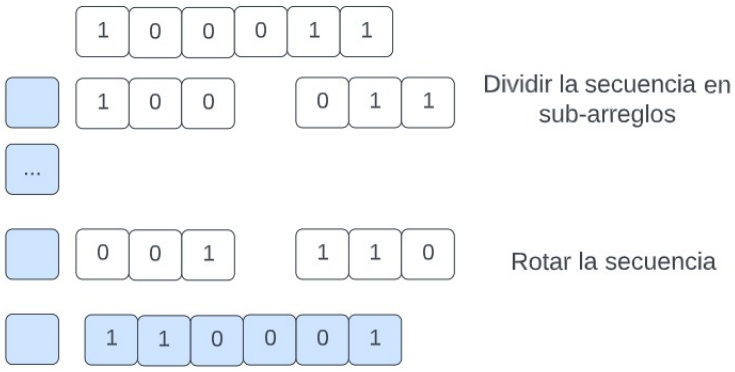

Figura 2: Idea solución DV

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) ∨ NEXT(first) = last then        ▷ NEXT siguiente posición del iterador
3:          return
4:          middle ← first
5:      end if
6:      dis ← DISTANCE(first, last)/2    ▷ DISTANCE calcula el número de elementos entre dos iteradores
7:      ADVANCE(middle, dis)                                ▷ 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 $O(nlogn)$.

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.

2.5. Análisis experimental

En esta sección se presentará el análisis experimental de la solución iterativa y por divide y vencerás del problema binario invertido.

2.5.1. Protocolo

1. Definir la arquitectura del computador 32 bits o 64 bits, para saber cual es el mayor número que soporta como entero.
2. Definir un rango entre 1 a 32 o 64 dependiendo la arquitectura.
3. Correr el algoritmo al menos 3 veces para promediar los tiempos generados.
4. Generación de la gráfica para la comparación de los algoritmos.

2.5.2. Procedimiento

1. Identificación la arquitectura del computador(64 bits).
2. Ejecución del programa entre 1 a 64 en 3 ocasiones.
3. Se obtuvieron como resultados 64 datos en cada ejecución del programa.
4. Finalmente, se elaboró una gráfica con los resultados obtenidos promediados (Figura 3).

2.5.3. Resultado

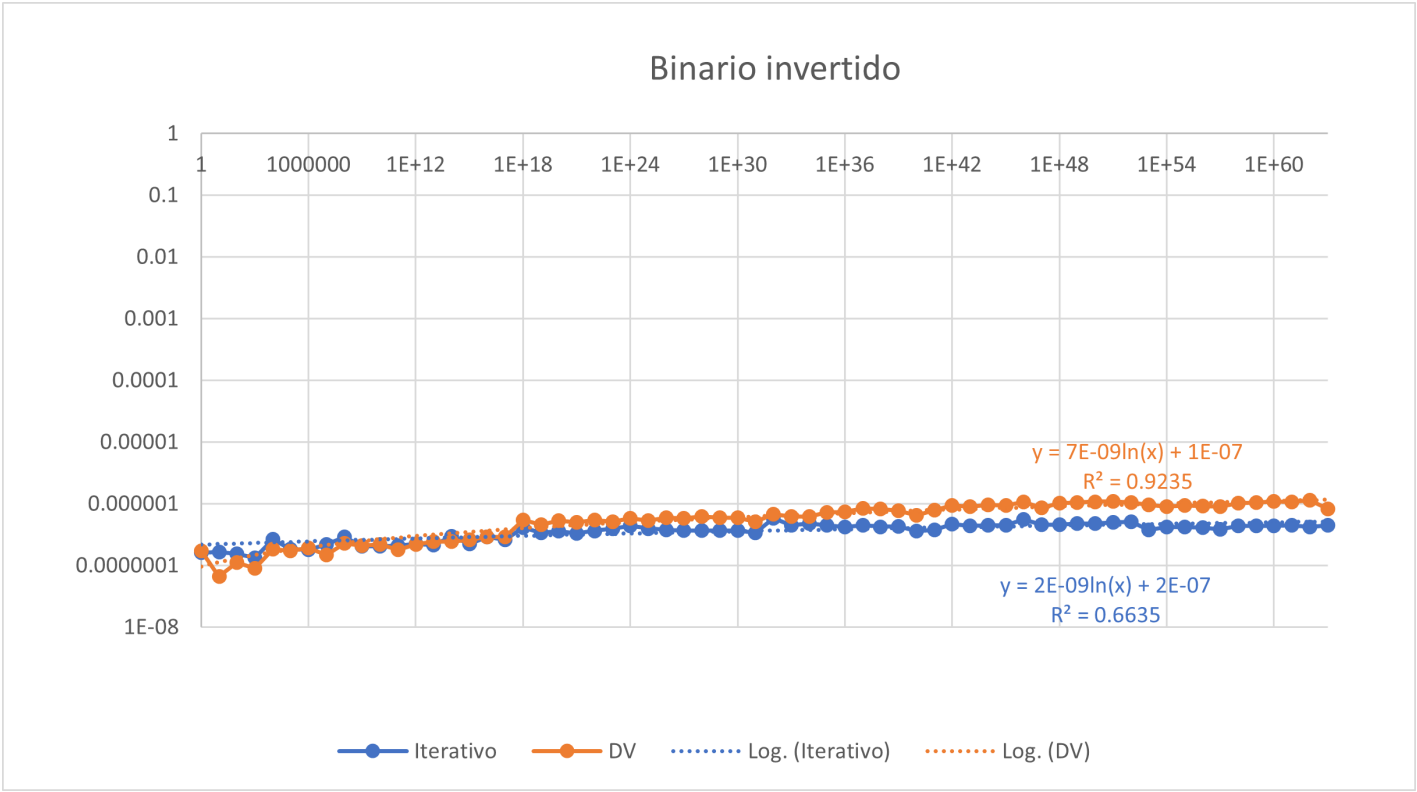

Figura 3: Resultados experimentación

2.5.4. Análisis

Como se puede observar en la gráfica presentada, el algoritmo iterativo representa una carga menor con respecto al algoritmo divide y vencerás que implementamos, estos resultados eran de esperarse considerando la complejidad teórica de ambos algoritmos. En primer lugar, el algoritmo iterativo tiene una complejidad O($log(n)$), esto debido a que solo realiza divisiones y asigna al vector correspondiente lo cual seria similar en términos operacionales a simplemente hallar la cantidad de dígitos de un número binario. Por otro lado, el algoritmo divide y vencerás que aplicamos toma pasos adicionales como lo son el generar primero la secuencia binaria y luego invertirla recursivamente, que acaban a la larga haciéndolo más lento.

De igual modo, teniendo en cuenta que R^2 para el algoritmo de Divide y Vencerás es más cercano a 1, de muestra que la regresión logarítmica aplicada a la gráfica concuerda con una tendencia logarítmica, sin embargo, para la implementación iterativa aunque R^2 esta más alejado del uno, utilizando IA entrenamos un dataset y al predecir el comportamiento de la gráfica se logro evidenciar que R^2 siempre estaba más cerca del 1. Lo que nos da a entender que ambas gráficas siguen su tendencia logarítmica que representa su complejidad.

2.6. Conclusiones

1. El algoritmo iterativo se comporta mejor que el algoritmo divide y vencerás, $O(log(n))$ vs $O(nlog(n))$.
2. Pueden haber algoritmos divide y vencerás que se comporten mejor de acuerdo a esta problemática.
3. A nivel general y considerando la potencia de los equipos actuales, la opción iterativa puede ser hoy en día las más adecuada.