Taller dividir y vencer

Jose Fernando Zuluaga, Nicolas Daniel Vargas

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

16 de agosto de 2022

Resumen

En este documento se presenta una solución al problema de ordenamiento de datos, mediante el algoritmo TimSort, se hará la descripción y formalización del mismo. **Palabras clave:** ordenamiento, algoritmo, formalización, experimentación, complejidad.

Índice

1. Introducción

Los algoritmos de ordenamiento de datos son muy útiles en una cantidad considerable de algoritmos que requieren orden en los datos que serán procesados. En este documento se presentan tres de ellos, con el objetivo de mostrar: la formalización del problema (sección 2), la escritura formal de tres algoritmos (sección 3) y un análisis experimental de la complejidad de cada uno de ellos (sección **??**).

2. Formalización del problema

El problema recibe incialmente un numero $n \in \mathbb{N}$, se busca y opera su representación binaria. Esto con el fin, de hallar su representación binaria inversa, como requisito para el desarrollo del taller se pide dos algoritmos diferentes, el primero iterativo, y un segundo recursivo utilizando la tecnica dividir y vencer.

2.1. Definición del problema del "numero binario inverso"

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

1. un numero n tal que $n \in \mathbb{N}$ y donde $-2,147,483,647 < n < 2,147,483,647$, restricción para el usuario

2. una secuencia de numeros S que contiene la representación binaria de n

producir un nuevo numero n' cuyo valor es la representación decimal de la secuencia inversa de S

- Entradas:
  - $n \in \mathbb{N}$
- Salidas:
  - $n' \in \mathbb{N}$

3. Algoritmos de solución

3.1. Iterativa

El algoritmo iterativo maneja un arreglo de numeros enteros, en el cual almacena la representación binaria inversa del numero recibido. Se manejan dos ciclos, el primer ciclo tiene como función calcular la representación binaria inversa del numero, en este se guarda en la posicion i del arreglo mecionado el reciduo de la de division $n/2$, acto seguido se divide el numero n entre dos, esto se repetira mientras $n \neq 0$. El segundo ciclo se utiliza para la transformación de la representación binaria inversa obtenida a un numero decimal y se hara uso de dos variables extra que son num,el cual es el numero decimal deseado(n') y exp que sera el exponente de la ubicacion,en este ciclo se recorre el arreglo S' verificando cuales valores son 1, si lo es entonces se eleva 2 a la exp y eso se le suma a la variable num. Finalmente se muestra en pantalla el numero obtenido n'.

Algoritmo 1 Version Iterativa

Require: $S = \langle S_i = 1|0 \rangle$, $n \in \mathbb{N}$
Ensure: n será convertido a expresión binaria, S contendrá la expresión binaria de n

```
1:  procedure ToBin(n, S)
2:      for i ← 0 to n <> 0 do
3:          s_i ← nMOD 2
4:          n ← n/2
5:      end for
6:  end procedure
7:
8:  procedure ToDec(n, e, S)
9:      for i ← |S| − 1 to i = 0 do
10:         if S_i = 1 then
11:             n ← n + 2^e
12:             e + +
13:         end if
14:         i − −
15:     end for
16: end procedure
```

3.1.1. Análisis de complejidad

Por inspección de código: hay dos ciclos *para-todo* no anidados que, en el peor de los casos, recorren todo la secuencia de datos; entonces, este algoritmo es $O(|S|)$.

3.2. Recursivo usando dividir y vencer

Para iniciar este algoritmo convierte el numero decimal ingresado n en binario haciendo uso de una funcion recursiva que guarda en una lista S la forma binaria inversa del numero n, esto se guarda de forma

inversa ya que asi es la forma en la que se calcula el numero binario de n pero ya que lo que se nos pide es la representació binaria de este entonce se implementa un funcion que recibira la posicion del inicio del arreglo ini y la posicion final del arreglo tam, el objetivo de esta funcion es comparar la posicion inicial y final de la lista, si estas dos son iguales entonces se tomara el caso base en donde se guarda el valor en una lista auxiliar S' que llamaremos numeros nuevos, de lo contrario se parte la lista S en dos partes con ayuda de un pivote q, luego de esto la funcion se llama a si misma dos veces, una enviando solo la mitad derecha de la lista S y la otra la mitad izquierda, es decir, en el caso del lado derecho el inicio ini de la lista que se envia sera $q+1$ y la posicion final tam sera el mismo tam original, mientras que el lado izquierdo comprendera desde ini hasta q, siempre se envia primero la parte derecha de la lista para asi guardar la lista desde el ultimo elemento hasta el primero. Finalmente se nos pide convertir el numero binario inverso , que se encuentra dentro de la lista S', en su representación decimal, para esto haremos uso de la función "desbinarización"la cual recibe como parametros el inicio (ini) y final (tam) de la lista, un entero que llamaremos "respuesta" y un exponente que llamaremos "exp",estos ultimos dos empiezan en cero y se pasan como parametros para poder manjarlo dentro de la funcion la cual al igual que la funcion anterior parte la lista en dos con ayuda de un pivote q, no obstante, en el caso base despues de comparar el ini y tam reviza si el valor que se tiene llamemoslo v es igual a 1, si es asi entonces entonces a "respuesta" se le suma 2^{exp} y se aumenta exp en 1, si en caso contrario v no es 1 entonces solo se aumenta exp en 1. Al igual que la función anteriormente descrita, "desbinarización" reviza primero la parte derecha del arreglo que recibe

3.2.1. Análisis de complejidad

Como podemos ver en el codigo de la funcion reverse se hacen dos llamados recursivos y siempre se parten los datos en dos partes, las partes del algoritmo que no son recursivas tampoco son iterativos, por ende, si hacemos uso de la formula del teorema maestro obtenemos una complejidad de

$$T(n) = 2T(n/2) + O(1)$$

$$1 = n^{log_2(2-E)} \rightarrow E = 1$$
$$1 = n^{log_2(2)} log_2^k n \rightarrow$$
$$1 = n^{log_2(2+E)} \rightarrow E = -1$$

Como podemos ver el caso mas optimo es el primero donde $E = 1$ por ende podemos concluir que este caso tiene una complejidad

$$\theta(n^{log_2 2})$$
$$\theta(n)$$

4. Análisis Experimental

Para la parte experimental de nuestro taller, se realizo una tabla con ciertos datos a tener en escenarios de prueba como lo es:

- **Numero ingresado**: Representa el numero ingresado por el usuario
- **Inverso Deseado**: Representa el numero teórico, el numero que se debería de recibir como resultado
- **Inverso obtenido:** Representa el numero experimental, el numero recibido como resultado por el algoritmo
- **Tiempo:** Representa el tiempo que tardo el algortimo durante su ejecución
- **Tendencia:** Representa la tendencia del algortimo durante su ejecución

Algoritmo 2 Versión Dividir y vencer

Require: $n \in \mathbb{N}$

Ensure: n será tranformado a su expresión binaria, S contendra la expresión binaria de n, Z contendrá la expresión binaria inversa de n

```
1: procedure ToBin(n)
2:     if n <> 0 then
3:         S.push_back(n MOD 2)
4:         ToBin(n/2)
5:     end if
6: end procedure
7:
1: procedure Reverse(i, f)
2:     if i = f then
3:         q ← S_i
4:         Z.push_back(q)
5:     end if
6:     if i <> f then
7:         p ← (i + f)/2
8:         Reverse(p + 1, f)
9:         Reverse(i, p)
10:    end if
11: end procedure
12:
1: procedure ToDec(n, e, i, f)
2:     if i = f then
3:         if Z_f = 1 then
4:             n ← n + 2^e
5:         end if
6:         e ← e + 1
7:         if Z_f <> 1 then
8:             q ← (f + i)/2
9:             ToDec(n, e, q + 1, f)
10:            ToDec(n, e, i, q)
11:        end if
12:    end if
13: end procedure
```

Como observamos de los resultados, la tendencia es constante su cambio es poco notable, a diferencia del tiempo, que es tomado en microsegundos, es realmente muy poco en tiempo de ejecución, teniendo en cuenta las propiedades del entorno de ejecución. Concluimos que la complejidad efectivamente, como fue mencionado anteriormente, es

$$\theta(n)$$

Numero ingresado	Inverso Deseado	Inverso obtenido	Tiempo (microsegundos)	Tendencia
4	1	1	0	268,10945
5	5	5	0	268,10945
7	7	7	0	268,10945
10	5	5	0	268,10944
28	7	7	512	268,10942
56	7	7	508	268,10938
138	81	81	511	268,10926
345	309	309	0	268,10897
865	539	539	507	268,10824
1134	945	945	0	268,10786
1864	151	151	510	268,10683
3486	1947	1947	512	268,10455
5316	1125	1125	0	268,10198
9165	11505	11505	0	268,09656
10614	7077	7077	0	268,09453
35685	42705	42705	510	268,05926
45963	53709	53709	0	268,0448
74965	87625	87625	511	268,00401
123456	1167	1167	1069	267,93579
563412	176529	176529	0	267,31693
864315	901323	901323	511	266,89366
1000000	9263	9263	513	266,70279
1365789	1514853	1514853	0	266,18825
5632587	6891477	6891477	0	260,18631
10658623	16565573	16565573	507	253,11637
185654321	147042445	147042445	0	6,9564734

Figura 1: Tabla de datos y resultados.

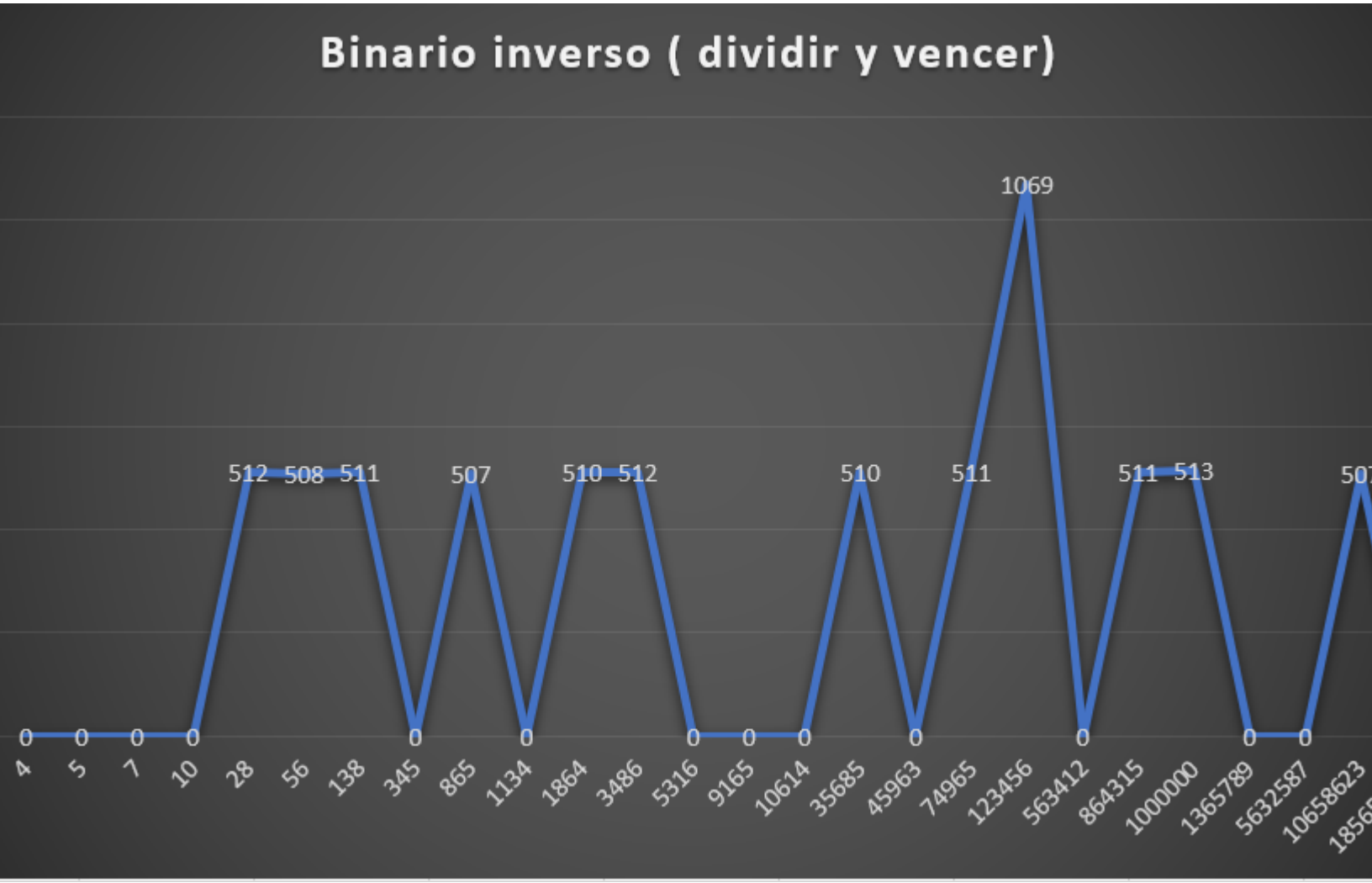

Figura 2: Grafica resultado.