Representacion binaria

Aarón C.d.C

December 2, 2023

1 Cualquier entero puede representarse como una combinacion de potencias de 2

1.1 Demostracion:

1.1.1 1. Para un entero n, n es par si y solo si $n = 2k$ para algun entero k

Por el teorema del cociente-residuo, $n = pk + r$ para algunos enteros p, k, r con $0 \leq r < n$. Dado que n es par, 2 divide a n. Lo que implica que $n = 2k$ y $r = 0$. Esto es verdadero para todo entero n y k ya que, por induccion, el siguiente par es:

$n + 1 = 2(k + 1)$

Que tambien es par.

1.1.2 2. Para un entero n, n es impar si y solo si $n = 2k + 1$ para algun entero k

Si n es impar, entonces 2 no divide a n, y por tanto $n = 2k + r$ con $0 \leq r < n$. No obstante, dado que estamos dividiendo por 2, los unicos restos posibles son 0 o 1. Y como 2 no divide a n, el resto r no puede ser igual a 0. Por tanto:

$n = 2k + 1$

Esto es cierto para todo entero n. Por induccion, el siguiente entero impar es:

$n + 1 = 2(k + 1) + 1$

que es impar.

1.1.3 3. Si n es un entero, n solo puede ser par o impar

Un entero, o es divisible entre 2, o no es divisible entre 2. Por tanto, cualquier numero se puede representar como $n = 2k + r$ con $r \in [0, 1]$.

1.1.4 4. La potencia $2^0 = 1$

Supongamos que a es un numero real cualquiera. El numero a se puede representar tambien como $a = 1 \cdot a$, ya que cualquier numero contiene un multiplo 1 implicito. Una potencia de a, tal que a^n para algun entero $n \geq 0$, es un producto de a donde los factores forman un conjunto. Por ejemplo:

$a^3 = a \cdot a \cdot a$

En este caso, los factores de a^3 forman un multiconjunto con tres elementos a. Cuando el exponente es 0, este multiconjunto esta vacio ($\emptyset$). Es decir, no contiene ningun factor a. Pero recordemos que

$a = 1 \cdot a$. Es decir, contiene un factor 1 implicito. Y por tanto, el único factor es 1, de modo que $a^0 = 1$. Y cuando $a = 2$, $2^0 = 1$ (que es lo que se queria demostrar).

1.1.5 5. La suma de n numeros pares es par

Sea a_i un entero par tal que $a_i = 2k_i$ para algun entero k_i, la suma es:

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} 2k_i = 2\left(\sum_{i=1}^{n} k_i\right)$$

Que por (1) es par.

1.1.6 6. La suma de un entero par y un entero impar es impar

Sean $n = 2k$ y $m = 2s + 1$, su suma es

$n + m = 2k + 2s + 1 = 2(k + s) + 1$

Que por (2) es impar.

1.1.7 7. El producto de dos enteros pares es par

Sean $n = 2k$ y $m = 2s$, su producto es

$nm = 2k \cdot 2s = 4ks = 2(2ks)$

Que por (1) es par.

1.1.8 8. Cualquier combinacion de potencias de 2 es par, siempre que no exista ningun termino 2^0

Sea $n = \sum_{i=0}^{m-1} 2^{m-i}$ para algun entero $m > 1$, por (5) podemos concluir que n es par si y solo si 2^{m-i} es par. Dado que 2^{m-i} es una potencia de 2 (que es par), por (5) podemos concluir que 2^{m-i} es par. Y asi, n es par.

Si añadimos 2^0 a la suma, por (4) $2^0 = 1$, y asi:

$$n = \left(\sum_{i=0}^{m-1} 2^{m-i}\right) + 1$$

que es impar. Sea N el conjunto de m elementos 2^i tal que su suma sea n, o mas formalmente:

$$N = \{2^i (i \in \mathbb{Z} \geq 0) \mid \sum_{i=0}^{m} 2^i = n\}$$

si $2^0 \notin N$, entonces por (5) concluimos que la suma de todos los elementos en N es par. Y si $2^0 \in N$, por (5) y por (6) concluimos que la suma de todos los elementos en N es impar.

1.1.9 9. Cualquier entero $n > 0$ se puede representar como una combinacion de potencias de dos

Por (3), un entero solo puede ser par o impar. Por (8), podemos representar un entero $n > 0$ par como una combinacion de potencias de dos tal que el ultimo elemento de la suma sea mayor o

igual a 2^1, y podemos representar un entero n impar como una combinacion de potencias de dos tal que el ultimo elemento de la suma sea igual a 2^0. Por tanto, por (3), cualquier entero puede representarse como una combinacion de potencias de dos.

Para $n = 0$, su representacion solo puede ser 0.

---

2 Representacion de un numero entero decimal positivo en binario

Sea n un numero entero mayor que 0, y $N = \{2^i(i \in \mathbb{Z} \geq 0) \mid \sum_{i=0}^{m} 2^i = n\}$. Y se define $b(i)$ como:

$$b(i) = \begin{cases} 1 & si\ i \in N \\ 0 & si\ i \notin N \end{cases}$$

Entonces la representacion binaria de n se define como el multiconjunto $B = [\forall i \in (card(N) \geq i \geq 0) \wedge i \in \mathbb{Z} \mid b(i)]$.

El multiconjunto B contiene todos los bits que representan a n. Por ejemplo, para el numero 30, tenemos que $N = \{2^4, 2^3, 2^2, 2^1\}$. Y su representacion binaria es $B = [1, 1, 1, 1, 0]$, o expresado como un digito, $n = b11110$ (donde "b" se usa para denotar que el numero es una representacion binaria y no decimal). Naturalmente, B se puede tambien definir como un conjunto normal, tal que:

$B = \{\forall i \in (card(N) \geq i \geq 0) \wedge i \in \mathbb{Z} \mid (i, b(i))\}$

O mas compacto como una funcion, sea $S = [card(N), 0] \in \mathbb{Z}$ y $V = \{\forall i \in V \mid b(i)\}$, se define B como $B : S \to V$.

---

3 Numero de bits requeridos para representar un numero entero n

Buscamos un numero i tal que $n \leq 2^i$. Como n es un entero pero $log_2 i$ es un numero real, buscamos un entero m tal que $m \leq log_2 i < m + 1$, y hacemos que $i = m$ de modo que $n \leq 2^m$ y $m \in \mathbb{Z}$.

Por ejemplo, supongamos que $n = 15$. Buscamos un entero m tal que $15 \leq 2^m$ y $m \leq log_2 i < m+1$. Supongamos que $i = 16$, entonces $log_2 16 = 4$ y $m \leq 4 < m + 1$. Como 4 es un entero, $m = 4$. Y así, $15 \leq 2^4 = 16$. Por tanto, el numero de bits requerido para representar a 15 es $m = 4$. Esta formula se simplifica como

$m = \lfloor log_2 i \rfloor + 1$ para algun entero $i > 0$.

---

4 Convertir un numero binario positivo a decimal

Sea B el multiconjunto de todos los bits (unos y ceros) que forman la representacion binaria del numero entero n, siendo $n > 0$. Entonces el valor decimal (D) de n es:

$$D = \sum_{i=0}^{card(B)-1} B_i \times 2^i$$

Ejemplo: Sea $n = b11110$, $B = [1,1,1,1,0]$ y $D = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 = 2 + 4 + 8 + 16 = 30$

Los siguientes programas representan algoritmos en Python para calcular el numero de bits requerido para representar un entero, su combinacion de potencias de dos, asi como convertir su combinacion de potencias de dos en un entero decimal:

[1]:
```
from math import log
```

[2]:
```
# Esto es un alias para log(x,2). Devuelve el logaritmo en base 2 de x.
log2 = lambda x: log(x,2)

print(log2(8)) # Esperado: 3.0
```

```
3.0
```

[3]:
```
# Devuelve el número de bits requerido para
# representar el número entero x.
def rbits(x: int)->int:
    # Los valores negativos se representan
    # mediante complemento a 2. No implementado.
    if x < 0: return None
    # Si x = 0, entonces necesitamos 1 bit para representar el número.
    if x == 0: return 1

    # rbits = piso(log2(x))
    # Convertir un número decimal a entero (int) es
    # igual a la función piso.
    return int(log2(x))+1

print(rbits(15)) # Valor esperado: 4
```

```
4
```

[4]:
```
# Devuelve una composicion de potencias de dos, como una
# lista de los exponentes.
def bincomb(x: int)->list:
    res = [] # Lista con los exponentes
    rb = rbits(x) # rb es el número requerido de bits
    c = 0 # Acumulador
    d = 0 # Variable temporal para computar 2^(rb-i)

    # Si el numero es 0, no tiene potencias de 2.
    # Se devuelve una lista vacía.
    if x == 0: return []
```

```
    # Si x == 2^(rb) entonces solo tiene rb.
    if x == 2**rb: return [rb]

    # Iterar de 0 a rb (bits requeridos), rb incluido (El intervalo es [0, rb])
    for i in range(0,rb+1):
        # d = 2^(rb-i). Cuando i = 0, d = 2^rb. Cuando i = rb, d = 2^0 = 1.
        d = 2**(rb - i)
        # Es c (accumulator) + d (2^(rb-i)) <= el numero (x)? Si es asi,␣
↪2^(rb-i) esta en la combinacion.
        # If c + d > x then this power of 2 doesn't fit, it's not in the sum of␣
↪powers of 2.
        if c + d <= x:
            # Anyadir (rb-i) a la lista.
            res.append(rb-i)
            # Sumar d al acumulador.
            c += d
    # Devolver la lista.
    return res

print(bincomb(15)) # Esperado [3,2,1,0]
print(bincomb(256)) # Esperado [8]
print ([2**i for i in bincomb(30)]) # Esperado [16, 8, 4, 2]
print (sum([2**i for i in bincomb(30)])) # Esperado 30
```

```
[3, 2, 1, 0]
[8]
[16, 8, 4, 2]
30
```

[5]:
```
# Devuelve una cadena de texto con la representacion binaria de x.
def binrepr(x: int)->str:
    # Lista con las potencias de 2 en la combinacion de x
    f = bincomb(x)
    # Salida
    out = ""

    # 0 <= i < rbits(x)
    for i in range(0,rbits(x)):
        # Anyade un '1' a la cadena si i esta en la lista de exponentes.
        # En caso contrario, anyade un '0'.
        out += "1" if i in f else "0"
    # El texto con los bits esta invertido, ya que hemos iterado de 0 a rbits(x)
    # dado que esto es Python, es mas facil hacerlo asi y luego invertir el␣
↪texto
    # usando out[::-1]. El resto es b'{bits}'
    return f"b'{out[::-1]}'"
```

```
print(binrepr(131)) # b'1000011'
print(binrepr(16)) # b'10000'
```

```
b'10000011'
b'10000'
```

4.1 Bibliografia y recursos

1. Matematicas discretas con aplicaciones (Susanna S. Epp)
2. Wikipedia Binary Number
3. Wikipedia Multiset
4. Wikipedia Cardinality

[ ]: