# IIC2283 — Diseño y Análisis de Algoritmos
# **Ayudantía 10**

## Inversos multiplicativos

De la identidad de Bézout se sabe que dados $a, b \in \mathbb{N}$, existen $s, t \in \mathbb{Z}$ tales que:

$$\text{MCD}(a, b) = s \cdot a + t \cdot b$$

**Parte 1** Desarrolle un algoritmo que dados $a, b \in \mathbb{N}$, determine los valores $s, t, \text{MCD}(a, b)$ mencionados en la identidad.

**Idea de solución** Se utiliza una extensión del algoritmo de Euclides, que genere los tres valores buscados de forma simultánea.

**function** EMCD$(a, b)$
  **if** a=0 **and** b=0 **then**
    ERROR
  **else if** a=0 **then**
    **return** $(b, 0, 1)$
  **else if** b=0 **then**
    **return** $(a, 1, 0)$
  **else if** $a \leq b$ **then**
    $(g', s', t') \leftarrow$EMCD$(a, b \mod a)$
    **return** $(g', s' + \lfloor b/a \rfloor \cdot t', t')$
  **else**
    $(g', s', t') \leftarrow$EMCD$(b, a \mod b)$
    **return** $(g', t', s' - \lfloor a/b \rfloor \cdot t')$
  **end if**
**end function**

En el tercer caso, los valores retornados se obtienen desde:

$$g' = a \cdot s' + (b \mod a) \cdot t' = a \cdot s' + (b - \lfloor b/a \rfloor \cdot a) \cdot t'$$

y el último es análogo.

El algoritmo funciona en tiempo proporcional al de Euclides, y por lo tanto pertenece a $O(n)$.

**Parte 2** Describa un algoritmo que dados $a$ y $n$, determine si existe $b \in \mathbb{Z}_n$ tal que $a \cdot b = 1$, y encuentre el valor en ese caso.

**Solución** Se utiliza el algoritmo anterior. Un elemento $a$ es invertible módulo $n$ si y solo si $\text{MCD}(a, n) = 1$. Luego, se evalúa EMCD$(a, n)$ para obtener $\text{MCD}(a, b), s, t$ tales que:

$$\text{MCD}(a, b) = s \cdot a + t \cdot n,$$

y finalmente se retorna $(s \mod n)$ si $\text{MCD}(a, b) = 1$, y $\perp$ en otro caso.

## Pertenencia a conjuntos

**Parte 1** Dados enteros $\{d_1, \ldots, d_N\}$, demuestre que existen enteros $x_1, \ldots, x_N$ tales que:

$$\text{MCD}(d_1, \ldots, d_N) = d_1 \cdot x_1 + \cdots + d_N \cdot x_N$$

**Solución** Por inducción, utilizando que $\text{MCD}(d_1, d_2, \ldots, d_N)$ es igual a $\text{MCD}(d_1, \text{MCD}(d_2, \ldots, d_N))$. Cuando $N = 2$, el resultado es directo de la identidad de Bézout. Si la propiedad se cumple hasta $N$, entonces para $N + 1$ se tiene que

$$\text{MCD}(d_1, \ldots, d_{N+1}) = \text{MCD}(d_1, \text{MCD}(d_2, \ldots, d_{N+1})),$$

donde por hipótesis de inducción existen $x_2, \ldots, x_{N+1}$ tales que:

$$\text{MCD}(d_2, \ldots, d_{N+1}) = d_2 \cdot x_2, \ldots, x_{N+1} \cdot x_{N+1}.$$

Por otra parte, de la identidad se tiene que existen $s, t$ tales que:

$$\text{MCD}(d_1, \ldots, d_{N+1}) = s \cdot d_1 + t \cdot \text{MCD}(d_2, \ldots, d_{N+1})$$

y por lo tanto que:

$$\begin{aligned} \text{MCD}(d_1, \ldots, d_{N+1}) &= s \cdot d_1 + t(d_2 \cdot x_2, \ldots, d_{N+1} \cdot x_{N+1}) \\ &= s \cdot d_1 + td_2 \cdot x_2, \ldots, td_{N+1} \cdot x_{N+1} \end{aligned}$$

obteniendo el resultado.

**Parte 2** Sea $J \subseteq \mathbb{Z}$ tal que para todo $x \in J$, y para todo $z \in \mathbb{Z}$ se cumple que $xz \in J$, y además que si $a, b \in J$, entonces $a + b \in J$. Demuestre que existe $a \in \mathbb{Z}$ tal que:

$$J = \{at | t \in \mathbb{Z}\}$$

**Solución** Se fija un conjunto $J$ con las condiciones descritas. Luego, si el conjunto es distinto de $\{0\}$, entonces contiene algún elemento $a \neq 0$, y por lo tanto algún elemento de $\{a, -a\}$ es positivo. Lo anterior prueba que el conjunto de elementos positivos de $J$ no es vacío, y entonces por principio de buen orden, existe un menor $d > 0$ tal que $d \in J$. A partir de ese elemento, se define el conjunto:

$$(d) = \{dt | t \in \mathbb{Z}\} \subseteq J$$

Dado un elemento arbitrario $b \in J$, entonces por algoritmo de división, existen $q, r$ tales que $b = q \cdot d + r$, con $0 \leq r < d$. Luego, como $b, d \in J$, entonces $r = b - q \cdot d \in J$, y entonces por minimalidad de $d$ se concluye que $r = 0$, y que $b \in (d)$.

**Parte 3** Dados enteros positivos $\{d_1, \ldots, d_N\}$, se define el conjunto:

$$(d_1, \ldots, d_N) \doteq \{d_1 \cdot x_1 + \cdots + d_N \cdot x_N | x_1 \cdots x_N \in \mathbb{Z}\}$$

Desarrolle y analice un algoritmo que dado un entero $X$, y enteros $\{d_1, \ldots, d_N\}$, determine si $X$ pertenece o no a $(d_1, \ldots, d_N)$.

**Idea de solución** Por la parte 1, el conjunto $(d_1, \ldots, d_N)$ contiene a $\text{MCD}(d_1, \ldots, d_N)$, y además como ese elemento divide a todos los elementos del conjunto, entonces es el menor elemento positivo del conjunto. Además, se tiene que $(d_1, \ldots, d_N)$ cumple las condiciones de la parte 2, y por lo tanto que sus elementos son exactamente los múltiplos de $\text{MCD}(d_1, \ldots, d_N)$.

Utilizando lo anterior, se obtiene el siguiente algoritmo:

- Calcular $g = \text{MCD}(d_1, \ldots, d_N)$.
- Verificar si $g|X$.

# Elemento mayoritario

Dada una lista de enteros $D = \{d_1, \ldots, d_N\}$, decimos que $M$ es un elemento mayoritario de $D$ si: (1) $M$ pertenece a $D$, y (2) el valor $M$ está repetido más de $N/2$ veces en $D$.

**Parte 1** Describa y analice un algoritmo determinístico que dada una lista de enteros, retorne su elemento mayoritario en caso de que exista, o indique que no existe en caso contrario. El algoritmo solo puede comparar igualdad entre elementos.

**Solución** Dividir para conquistar. Si la lista tiene largo 1, se retorna SI con el único valor de la lista. Si tiene largo $N$, se divide en dos partes que se resuelven recursivamente, y luego se consideran las 4 combinaciones:

- En las dos partes se retorna que no hay un elemento mayoritario: en este caso se retorna falso, por que no puede haber un elemento mayoritario que no lo sea en alguna de las partes.
- En una parte se retorna un valor, y en la otra no: se verifica ese valor, es decir se cuenta la cantidad de ocurrencias en el arreglo completo, y se retorna SI cuando la cantidad sea mayor a $N/2$.
- Si ambas partes entregan un valor, se verifican ambos.

El tiempo del algoritmo está dado por $T(n) = 2T(n/2) + 2n \in \Theta(n \log n)$.

**Parte 2** Desarrolle un algoritmo aleatorizado para el mismo problema, y compare su eficiencia con el algoritmo determinístico.

**Idea de solución** Se elige un índice entre 1 y $n$ de forma aleatoria, y se cuenta la cantidad de ocurrencias del valor en esa posición. Se retorna ese elemento si aparece más de $N/2$ veces, y se retorna que no hay un elemento mayoritario en otro caso.

Si la lista no tiene un elemento mayoritario, entonces el algoritmo entrega el resultado correcto con probabilidad 1. Por otra parte, la probabilidad de que retorne que no hay un elemento mayoritario cuando si lo hay es menor a $1/2$, por que el elemento mayoritario aparece al menos $N/2$ veces, y por lo tanto la probabilidad de elegir otro valor es menor a $1/2$.

El algoritmo funciona en tiempo lineal, y por lo tanto es más eficiente y simple que el algoritmo determinístico.

# Multiplicación de matrices

Desarrolle y analice un algoritmo que dadas 3 matrices $A, B, C$ de $n \times n$, determine si $AB = C$. El algoritmo debe funcionar en tiempo $o(n^{2,2})$, y debe retornar SI con probabilidad 1 cuando $AB = C$, y con probabilidad menor a $10^{-100}$ si $AB \neq C$.

**Idea de solución** Se genera un vector aleatorio $x$, eligiendo $n$ valores de $\{0, 1\}$ de forma uniforme e independiente. Luego se verifica si $Cx = A(Bx)$. Esto funciona en tiempo $O(n^2)$, ya que multiplicar una matriz cuadrada por un vector de la forma usual toma tiempo cuadrático, y el algoritmo realiza esa operación 3 veces.

Si $AB = C$, entonces los vectores obtenidos son iguales, por lo que el algoritmo entrega la respuesta correcta con probabilidad 1.

Por otra parte, si $AB \neq C$, entonces se evalúa la probabilidad de que $Cx = A(Bx)$. Para eso reescribimos la expresión como $(C - AB)x = 0$, y definimos $D = C - AB$. Como $AB \neq C$, entonces $D \neq 0$.

Si consideramos $x$ como un vector de variables, y definimos $Dx = r$, entonces por definición tenemos que:

$$r_i = \sum_{j=1}^{n} (d_{ij} x_j),$$

es decir un polinomio de grado 1 con $n$ variables. Luego, como $D \neq 0$, entonces para algún $\ell$ el polinomio que define $r_\ell$ no es nulo, y por lo tanto se puede utilizar el lema de Schwartz-Zippel para demostrar que la probabilidad de que $r_\ell = 0$ es menor o igual a $1/2$. Luego, la probabilidad de que el vector completo sea 0 también es menor o igual a $1/2$.

Si repetimos el algoritmo $k$ veces, la probabilidad de error se reduce a $1/2^k$, por lo que necesitamos que:

$$\begin{aligned} 2^{-k} &< 10^{-100} \\ k &> 100 \log_2(10) \\ k &\geq 333 \end{aligned}$$