# Álgebra Computacional

Álvaro García Tenorio Miguel Pascual Domínguez

28 de octubre de 2018

**Resumen**

# Índice general

# Capítulo 1

# Algoritmos

## 1.1. Algoritmos de euclides

### 1.1.1. Algoritmo de euclides simple

Sea $A$ un dominio euclídeo con función de grado $\varphi$.

Consideramos dos elementos $a, b \in A$, siendo $b \neq 0$. Nuestro objetivo es calcular el máximo común divisor de $a$ y $b$.

Para ello, dividimos $a$ entre $b$, obteniendo dos elementos $q_1, r_1 \in A$ tales que $a = bq_1 + r_1$, siendo, o bien $r_1 = 0$, o bien $r_1 \neq 0$ y $\varphi(r_1) < \varphi(b)$.

Es claro que si $r_1 = 0$, $b$ es divisor de $a$, y por tanto $b = \text{mcd}(a, b)$, con lo que habríamos terminado. En caso contrario, como por el lema de Euclides se cumple que $\text{mcd}(a, b) = \text{mcd}(b, r_1)$, podemos repetir el proceso con $b$ y $r_1$, obteniendo dos nuevos elementos $q_2, r_2 \in A$, cumpliendo hipótesis análogas a $q_1$ y $r_1$.

Nótese que este proceso solo puede ser repetido un número finito de veces, ya que, cada vez que repetimos el proceso, la función de grado del resto de la división decrece estrictamente.

A este método de cálculo del máximo común divisor se le conoce como ***algoritmo de Euclides***, y es precisamente el algoritmo que implementa el método `gcd` de la clase `EuclideanDomain`.

### 1.1.2. Algoritmo de euclides extendido

Basándonos en el algoritmo de Euclides, podemos, además de calcular el máximo común divisor de dos números, extraer una **identidad de Bézout** que los relacione. Si bien esta extensión del algoritmo de Euclides sólo es válida para dominios euclídeos con unidad, como veremos a continuación.

Partimos de un dominio euclídeo con unidad $A$, considerando dos elementos $a, b \in A$, siendo $b$ no nulo.

Procedemos, como en el algoritmo de Euclides, dividiendo $a$ y $b$, obteniendo $q_1, r_1 \in A$ cumpliendo las hipótesis habituales que garantizan que nuestro procedimiento es finito.

Despejando $r_1$ obtenemos que $r_1 = a - q_1 b$. Es decir, tenemos una "pseudo–identidad de Bézout". Definimos $\alpha_1 := 1$, $\beta_1 := -q_1$ para verlo más claro, $r_1 = \alpha_1 a + \beta_1 b$.

Usando el lema de Euclides, como en el algoritmo anterior, tenemos que $\text{mcd}(a, b) = \text{mcd}(b, r_1)$, por lo que procedemos a dividir $b$ entre $r_1$, siempre y cuando $r_1 \neq 0$, al final veremos que el procedimiento es válido para todos los casos.

De la división de $b$ y $r_1$ obtenemos la igualdad $r_2 = b - r_1 q_2$, sustituyendo $r_1$ por la pseudo–identidad de Bézout, obtenemos, tras reordenar, una nueva pseudo–identidad de

bezout, esta vez para $r_2$, esta es $r_2 = \alpha_2 a + \beta_2 b$, siendo $\alpha_2 = -\alpha_1 q_2$ y $\beta_2 = 1 - \beta_1 q_2$.

Si repetimos el proceso una vez más, obtendremos una pseudo–identidad para $r_3$, siendo esta $r_3 = \alpha_3 a + \beta_3 b$, con $\alpha_3 = \alpha_1 - \alpha_2 q_3$ y $\beta_3 = \beta_1 - \beta_2 q_3$.

Por inducción no es complicado comprobar que la pseudo–identidad para $r_n$ tendrá por coeficientes $\alpha_n = \alpha_{n-2} - \alpha_{n-1} q_n$ y $\beta_n = \beta_{n-2} - \beta_{n-1} q_n$. Además, si definimos $\alpha_{-1} := 1$, $\alpha_0 = 0$, $\beta_{-1} = 0$ y $\beta_0 = 1$, esta fórmula es válida para todo $n \in \mathbb{N}$.

De esta manera, cuando llegamos a una iteración del procedimiento en la cual $r_l = 0$, es decir, $r_{l-1}$ es el máximo común divisor, para obtener la identidad de Bézout basta recuperar los coeficientes $\alpha_{l-1}$ y $\beta_{l-2}$.

Esto puede ir haciéndose sobre la marcha, con una implementación muy similar a la del cálculo de términos de la sucesión de Fibonacci. A este algoritmo se le conoce como ***algoritmo de Euclides extendido***, y es el algoritmo programado en el método `bezout` de la clase `EuclideanUnitDomain`.