Introducción

Con qué vamos a trabajar

Anillos (conmutativos)

Terna $(A, +, \cdot)$ satisfaciendo $\forall a, b, c \in A$
Suma

- **asociativa** $a + (b + c) = (a + b) + c$
- **elemento neutro** $a + 0 = 0 + a = a$
- **elemento opuesto** $a + (-a) = (-a) + a = 0$
- **conmutativa** $a + b = b + a$

Producto

- **asociativa** $a(bc) = (ab)c$
- **elemento neutro** $a1 = 1a = a$
- **distributiva** $a(b + c) = ab + ac$, $(a + b)c = ac + bc$
- **(conmutativa)** $ab = ba$

Ejemplos de anillos

- Números Enteros: $\mathbb{Z}$

$$\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

- Enteros módulo m: $\mathbb{Z}_m$

$$\{0, 1, 2, \dots, m-1\}$$

- Polinomios: $A[x]$

$$\sum_{i=0}^{n} a_i x^i$$

Otros ejemplos

- Grafo: G

$$G = (V, E, v : E \to \mathcal{P}_2(V))$$

- Conjunto: C

$$C = \{c_1, c_2, \dots\}$$

- Lista: L

$$L = [l_1, l_2, \dots]$$

- En general cualquier estructura matemática.

TEMA 1

Formas normales y niveles de abstracción

1.1 Niveles de abstracción

Niveles de abstracción

1. *Objetos:* Elementos a tratar
2. *Formas:* Distintas y posibles representaciones de los objetos
3. *Estructura de datos:* Almacenamiento y manipulación de las formas de los objetos (en un ordenador).

Formas
Expresiones definidas mediante una gramática. A cada expresión se puede asociar un objeto.

Simplificación

Encontrar formas simples de representación es importante porque

1. permite optimizar los recursos informáticos,
2. facilita el pensamiento humano.

El concepto "simple" es ambiguo.

- Para $\mathbb{Z}$, *¿binarios, octales, decimales o hexadecimales?*
- Para $A[x]$, *¿desarrollo o factorización?*

Cero-equivalencia

¿Cómo reconocer que dos expresiones representan el mismo objeto?
¿Cómo reconocer que una expresión representa al cero?

- Es un problema bien definido, al contrario que la simplificación genérica.
- En absoluto es sencillo:

$$\text{siguienteprimo}(\left\lfloor\sqrt{1567AB6FC3286D3)_{16}}\right\rfloor)$$
$$-\text{primo}(34)*\text{primo}(50921)$$
$$-\sqrt{1525340530)_8*D55C158)_{16}}$$

1.2 Formas normales y canónicas.

Formas normales

E es un conjunto de expresiones. Utilizamos el símbolo $=$ para expresar que dos expresiones son iguales, y $\equiv$ para indicar que dos expresiones representan el mismo objeto.

Definición 1. Una *función normal* es una aplicación $f : E \to E$ calculable tal que

(I) $f(a) \equiv a$ para toda $a \in E$,

(II) $a \equiv 0 \Longrightarrow f(a) = 0$ para cualquier $a \in E$

Decimos que $a \in E$ es una *forma normal* con respecto a f si $a = f(a)$

El problema de la Cero-equivalencia tiene solución para aquellos conjuntos de expresiones sobre los que se puede definir una función normal.

Formas canónicas

Definición 2. Una *función canónica* es una función normal $f : E \to E$ que satisface la propiedad adicional

(III) $a \equiv b \Longrightarrow f(a) = f(b)$ para cualesquiera $a, b \in E$

Decimos que $a \in E$ es una *forma canónica* con respecto a f si $a = f(a)$

Teorema 3. *Las siguientes propiedades se verifican en una función canónica f:*

(a) f es idempotente, es decir, $f \circ f = f$,

(b) $f(a) = f(b)$ si y sólo si $a \equiv b$,

(c) cada expresión es equivalente a una única forma canónica.

Ejemplos: polinomios

En $\mathbb{Z}[x]$, una posible forma normal es representar cada polinomio como

$$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \quad \text{con } e_i \neq e_j \text{ cuando } i \neq j.$$

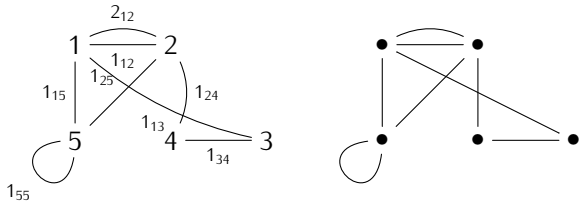
Una forma canónica puede ser representar cada polinomio como

$$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \quad \text{con } e_i < e_j \text{ cuando } i > j.$$

Ejemplos: grafos (1)

Sea $G = \{V, E\}$ un grafo no dirigido. Las siguientes son diferentes formas asociadas al mismo grafo (el objeto):

- $V = \{1, 2, 3, 4, 5\}$, $E = \{1_{12}, 2_{12}, 1_{13}, 1_{24}, 1_{34}, 1_{15}, 1_{55}, 1_{25}\}$, $f : E \to \mathcal{P}_2(V)$, $f(k_{i,j}) = \{i, j\}$.
-
$$\begin{pmatrix} 0 & 2 & 1 & 0 & 1 \\ & 0 & 0 & 1 & 1 \\ & & 0 & 1 & 0 \\ & & & 0 & 0 \\ & & & & 1 \end{pmatrix}$$



Ejemplos: grafos (y2)

Numerar los vértices y los lados incluyendo como subíndices los extremos del lado nos proporciona una forma normal de representar un grafo. Evidentemente no es una forma canónica, basta por ejemplo con reordenar los vértices para obtener otra forma del mismo grafo. Sin embargo los grafos

que juegan un papel similar al del elemento 0 de un anillo[1], pueden ser detectados con la forma anterior.

[1] siempre que consideremos que un grafo debe tener al menos un vértice

TEMA 2

Números Enteros

2.1 Representación

Bases de numeración

Teorema 4. *Sea $\beta \in \mathbb{N} \setminus \{0,1\}$. Todo número entero $d < \beta^k$ se expresa de forma única como*

$$d = (-1)^s(d_0 + d_1\beta + \cdots + d_{k-2}\beta^{k-2} + d_{k-1}\beta^{k-1}),$$

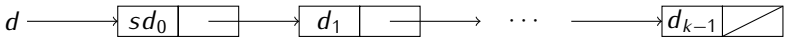
donde $s \in \{0,1\}$, $k \in \mathbb{N}$ y $0 \leq d_0, d_1, \ldots, d_{k-1} < \beta$.

- **Uso histórico** Los enteros se representan seleccionando un conjunto de β 'caracteres'
- **Uso computacional** Se selecciona una base $\beta < W$ donde W es el mayor entero que cabe en una palabra.

Estructuras de datos

Suponemos $d = (-1)^s(\sum_{i=0}^{k-1} d_i\beta^i)$.

- Punteros



- Vectores dinámicos

- Vectores fijos

La estructura de datos utilizada dependerá en gran medida del lenguaje de programación empleado.

2.2 Aritmética clásica

Suma: Algoritmo de la escuela

Suma de naturales de precisión arbitraria

procedure SumBigBig(a, b, β)
Require: $a = a_0 + a_1\beta + \cdots + a_{m-1}\beta^{m-1}$
Require: $b = b_0 + b_1\beta + \cdots + b_{n-1}\beta^{n-1}$

```
    l ← max(m, n)
    carry ← 0
    for i from 0 to l do
        temp ← a_i + b_i + carry                ▷ ¡Atención con el tamaño!
        c_i ← (temp mód β)
        carry ← (temp quo β)
    end for
    return c_0 + c_1β + ··· + c_{l−1}β^{l−1} + c_lβ^l
  end procedure
```

La complejidad es $\mathcal{O}(\max(m, n))$

Multiplicación: Algoritmo de la escuela (1)

Multiplicación de un natural grande por otro pequeño

```
  procedure ProdBigSmall(a, e, β)
Require: a = a_0 + a_1β + ··· + a_{m−1}β^{m−1}
Require: e < β
    carry ← 0
    for i from 0 to m do
        temp ← a_i e + carry                    ▷ ¡Atención con el tamaño!
        c_i ← (temp mód β)
        carry ← (temp quo β)
    end for
    return c_0 + c_1β + ··· + c_{m−1}β^{m−1} + c_mβ^m
  end procedure
```

La complejidad es $\mathcal{O}(m)$

Multiplicación: Algoritmo de la escuela (2)

Multiplicación de dos naturales grandes (versión partida)

```
  procedure ProdBigBigS(a, b, β)
Require: a = a_0 + a_1β + ··· + a_{m−1}β^{m−1}
Require: b = b_0 + b_1β + ··· + b_{n−1}β^{n−1}
    c ← 0
    for i from 0 to n − 1 do
        temp ← ProdBigSmall(a, b_i, β)
        c ← c + temp · β^i
    end for
    return c
  end procedure
```

La complejidad es $\mathcal{O}(mn)$

Multiplicación: Algoritmo de la escuela (y 3)

Multiplicación de dos naturales grandes (versión directa)

```
  procedure ProdBigBig(a, b, β)
    for i from 0 to m − 1 do c_i ← 0
    end for
    for k from 0 to n − 1 do
        carry ← 0
```

```
        for i from 0 to m − 1 do
            temp ← a_i b_k + c_{i+k} + carry
            c_{i+k} ← (temp mód β)
            carry ← (temp quo β)
        end for
        c_{k+m} ← carry
    end for
    return c = c_0 + c_1 β + ··· + c_{m+n−1} β^{m+n−1}
end procedure
```

La complejidad es $\mathcal{O}(mn)$

División: De nuevo vuelta al cole

```
  procedure DIVISION(a, b, β)
Require: a ≥ b
    r ← a_{m−1} β^{n−2} + a_{m−2} β^{n−3} + ··· + a_{m−n+1}
    for j from m − n to 0 do
        r ← r · β + a_j
        if r ≥ b then
            q ← ⌊ (r_n β + r_{n−1}) / b_{n−1} ⌋
            while r < q · b do
                q ← q − 1
            end while
            r ← r − q · b
            q_j ← q
        else
            q_j ← 0
        end if
    end for
    return (q_0 + q_1 β + ... q_{m−n} β^{m−n}) (r_0 + r_1 β + ... r_{n−1} β^{n−1})   ▷ cociente y Resto
end procedure
```

2.3 Aritmética rápida

Multiplicación rápida: Karatsuba (1)

Sea $x = x_0 + x_1\beta + \cdots + x_{n-1}\beta^{n-1}$ con n par. En base $\beta^{\frac{n}{2}}$ se representa:

$$x = x_{(0)} + x_{(1)}\beta^{\frac{n}{2}}$$

Sean a, b enteros de n dígitos en base β. El producto clásico se describe como

$$a \cdot b = a_{(0)} \cdot b_{(0)} + (a_{(1)} \cdot b_{(0)} + a_{(0)} \cdot b_{(1)}) \cdot \beta^{\frac{n}{2}} + a_{(1)} \cdot b_{(1)} \cdot \beta^n$$

La idea de Karatsuba:

$$\begin{aligned} a \cdot b = {} & a_{(0)} \cdot b_{(0)} \\ & + (a_{(0)} \cdot b_{(0)} + a_{(1)} \cdot b_{(1)} + (a_{(1)} - a_{(0)}) \cdot (b_{(0)} - b_{(1)})) \cdot \beta^{\frac{n}{2}} \\ & + a_{(1)} \cdot b_{(1)} \cdot \beta^n \end{aligned}$$

Multiplicación rápida: Karatsuba (y 2)

Útil para $n = 2^m$ dígitos.

Multiplicación rápida: Karatsuba

procedure KARATSUBA(a, b, m, β)
  if $m = 0$ **then return** $a \cdot b$
  else
    $a1 \leftarrow 2^{m-1}$ dígitos más significativos de a
    $a0 \leftarrow 2^{m-1}$ dígitos menos significativos de a
    $b1 \leftarrow 2^{m-1}$ dígitos más significativos de b
    $b0 \leftarrow 2^{m-1}$ dígitos menos significativos de b
    $t1 \leftarrow$ KARATSUBA($a1, b1, m-1, \beta$)
    $t2 \leftarrow$ KARATSUBA($a1 - a0, b0 - b1, m-1, \beta$)
    $t3 \leftarrow$ KARATSUBA($a0, b0, m-1, \beta$)
    return $t1 \cdot \beta^{2^m} + (t1 + t2 + t3) \cdot \beta^{2^{m-1}} + t3$
  end if
end procedure

Algoritmo de Euclides

Dados $a, b \in \mathbb{Z}$, el Algoritmo de Euclides Extendido proporciona d, u, v tales que $d = \gcd(a, b)$ y $d = u \cdot a + v \cdot b$

Algoritmo de Euclides Extendido

procedure EUCLIDESEXTENDIDO(a, b)
  $u_1 \leftarrow 1$, $v_1 \leftarrow 0$
  $u_2 \leftarrow 0$, $v_2 \leftarrow 1$
  while $b \neq 0$ **do**
    $(q, r) \leftarrow$ DIVISION(a, b, β)
    $a \leftarrow b$, $b \leftarrow r$
    $u \leftarrow u_1 - q \cdot u_2$, $v \leftarrow v_1 - q \cdot v_2$
    $u_1 \leftarrow u_2$, $u_2 \leftarrow u$, $v_1 \leftarrow v_2$, $v_2 \leftarrow v$
  end while
  return a
  return u_1, v_1
end procedure

Representación modular

Sea $M \in \mathbb{N} \setminus \{0, 1\}$. Los elementos de $\mathbb{Z}_M$ pueden representarse como

$$\{0, 1, \ldots, M-1\}$$

o como

$$\begin{cases} \left\{-\dfrac{M}{2}+1, \ldots, 0, \ldots, \dfrac{M}{2}\right\} & M \text{ par} \\ \left\{-\left\lfloor\dfrac{M}{2}\right\rfloor, \ldots, 0, \ldots, \left\lfloor\dfrac{M}{2}\right\rfloor\right\} & M \text{ impar} \end{cases}$$

Multiplicación modular: Teorema chino del resto

Teorema 5 (v1). *Si $m_1, \ldots, m_k$ son enteros primos relativos dos a dos, entonces el sistema de ecuaciones*

$$\begin{cases} x \equiv x_1 \pmod{m_1} \\ \quad \vdots \\ x \equiv x_k \pmod{m_k} \end{cases}$$

tiene solución única módulo $M = m_1 m_2 \ldots m_k$.

Teorema 6 (v2). *Sean $m_1, \ldots, m_k$ enteros primos relativos dos a dos y sea $M = m_1 \cdots m_k$. Entonces*

$$\mathbb{Z}_M \cong \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k}$$

Multiplicación modular: aritmética vectorial

Sean $x, y < M$**,**

$$\begin{aligned} \mathbb{Z}_M &\xrightarrow{\cong} \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k} \\ x &\longmapsto (x_1, \ldots, x_k) \\ y &\longmapsto (y_1, \ldots, y_k) \end{aligned}$$

aritmética

$$\begin{aligned} x + y &\longmapsto (x_1 + y_1, \ldots, x_k + y_k) \\ x \cdot y &\longmapsto (x_1 \cdot y_1, \ldots, x_k \cdot y_k) \end{aligned}$$

La complejidad en ambos casos es $\mathcal{O}(k)$

Multiplicación modular: Problemas (1)

- Representación: buscar suficientes m_i para representar un entero dado.
- Conversión: De base β a vector y regreso.

$m_1, \ldots, m_k$

- Primos: almacenar una lista suficientemente grande
- Utilizar elementos del tipo $2^e - 1$

$$\gcd(2^e - 1, 2^f - 1) = 2^{\gcd(e,f)} - 1$$

Multiplicación modular: Problemas (y 2)

Supongamos que trabajamos con una arquitectura de 32 bits. Una buena elección para los primos es $2147483648 = 2^{31} < p_i < 2^{32} = 4294967296$

Podemos estimar que hay aproximadamente $193635251 - 99940774 = 93694477$ usando el teorema del número primo:

$$\lim_{x \to \infty} \frac{\pi(x) \ln x}{x} = 1$$

Si $m_i = 2^{e_i} - 1$, hay menos candidatos pero más fáciles de calcular.

Conversión base $\beta \leftrightsquigarrow$ modular (1)

Dado $(x_1, \ldots, x_k) \in \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k}$, calcular el único $x \in \mathbb{Z}_M$ tal que

$$\begin{cases} x \equiv x_1 \mod m_1 \\ \quad \vdots \\ x \equiv x_k \mod m_k \end{cases}$$

Método propuesto por Garner en 1958.

Calculamos $\binom{k}{2}$ constantes c_{ij} con $1 \leq i < j \leq k$ tales que

$$c_{ij} m_i \equiv 1 \mod m_j$$

Conversión base $\beta \leftrightsquigarrow$ modular (2)

Asignamos

$$\begin{aligned}
y_1 &\leftarrow x_1 \mod m_1 \\
y_2 &\leftarrow (x_2 - y_1)c_{12} \mod m_2 \\
y_3 &\leftarrow ((x_3 - y_1)c_{13} - y_2)\, c_{23} \mod m_3 \\
&\vdots \\
y_k &\leftarrow (\ldots((x_k - y_1)c_{1k} - y_2)\, c_{2k} - \cdots - y_{k-1})\, c_{(k-1)k} \mod m_k
\end{aligned}$$

El valor buscado es

$$x = y_k m_{k-1} \ldots m_2 m_1 + \cdots + y_3 m_2 m_1 + y_2 m_1 + y_1$$

Conversión base $\beta \leftrightsquigarrow$ modular (y 3)

Cálculo rápido de los c_{ij}'s: si $e \equiv d \mod f$ y $ce \equiv 1 \mod f$ entonces

$$(1 + 2^d + \cdots + 2^{(c-1)d}) \cdot (2^e - 1) \equiv 1 \mod 2^f - 1$$

Por tanto los elementos c_{ij} son rápidos de calcular cuando $m_i = 2^{e_i} - 1$

2.4 El cálculo de potencias.

Definición

Si queremos calcular una potencia a^b, en donde a y b son números enteros, procedemos como sigue:

$$a^b = a \cdot a^{b-1}.$$

Por consiguiente tenemos que calcular previamente a^{b-1}, a^{b-2}, etc. Se puede comprobar que este proceso es lento si, como es el caso, pretendemos que b sea un número grande.

Idea principal

Existe otro método que es descomponer a^b en productos más pequeños. Si b es par, por ejemplo $b = 2c$, podemos hacer

$$a^b = a^{2c} = (a^c)^2;$$

de esta forma necesitamos hacer sólo algo más de la mitad de las operaciones.

¿Pero qué pasa si b es impar? Esto es, si $b = 2c + 1$.

En este caso basta con hacer la siguiente cuenta:

$$a^b = a^{2c+1} = (a^c)^2 a,$$

y observamos que realmente no hacemos muchas más operaciones, esto es, el número de operaciones que tenemos que hacer es aproximadamente la mitad que al principio, y no importa si b es par o impar.

El proceso sigue calculando ahora a^c.

Proceso de cálculo

Observad que realmente en este paso sólo hemos hecho $(a^c)a^\varepsilon$, en donde ε es 0 o 1. Si nos fijamos ésta es la cifra de las unidades cuando escribimos b en base 2.

$$b = b_t 2^t + b_{t-1}2^{t-1} + \cdots + b_1 2 + b_0.$$

Esto suponer que podemos escribir

$$b_{(2} = b_t b_{t-1} \ldots b_1 b_0 = b_t b_{t-1} \ldots b_1 0 + b_0.$$

Y en consecuencia se obtiene:

$$a^b = a^{b_{(2}} = a^{b_t b_{t-1} \ldots b_1 b_0} = (a^{b_t b_{t-1} \ldots b_1})^2 a^{b_0}.$$

Se trata ahora de hacer un algoritmo que permite el cálculo de a^b de forma rápida. Para ello basta observar que se tiene:

$$a^b = (a^{b_t b_{t-1} \ldots b_1})^2 a^{b_0} = \left((a^{b_t b_{t-1} \ldots b_2})^2 a^{b_1}\right)^2 a^{b_0} =$$

Siguiendo con este proceso se tiene la expresión:

$$a^b = ((\cdots((a^{b_t})^2 a^{b_{t-1}})^2 a^{b_{t-2}} \cdots)^2 a^{b_1})^2 a^{b_0}.$$

Recordemos que los b_i valen 0 o 1.

Podemos en este punto plantearnos la pregunta de cuantas operaciones es necesario realizar para calcular a^b.

En el primer caso ($a^b = a a^{b-1}$), tenemos que hacer $b-1$ multiplicaciones.

En el segundo tenemos que hacer como máximo dos por cada dígito de $b_{(2}$, y por tanto haremos como máximo $2(t+1)-1 = 2t+1$ multiplicaciones. El valor de t es conocido, ya que es el logaritmo de b en base 2, por tanto el número de operaciones es: $2\log_2(b)+1$.

Tenemos las expresiones:

$$b = 2^{\log_2(b)} = (10^{\log_{10}(2)})^{\log_2(b)} = 10^{\log_{10}(2)\log_2(b)}$$
$$b = 10^{\log_{10}(b)}$$
$$\log_{10}(b) = \log_{10}(2)\log_2(b)$$
$$\log_2(b) = \frac{\log_{10}(b)}{\log_{10}(2)} \approx 3\log_{10}(b).$$

Algunos ejemplos

.

Si $b = 1{,}000$, entonces $\log_2(b) = 3\log_{10}(1{,}000) = 9$. En el primer caso haríamos 999 multiplicaciones y en el segundo solamente $2 \times 9 + 1 = 19$.

Si $b = 1{,}000{,}000$, entonces $\log_2(b) = 3\log_{10}(1{,}000{,}000) = 18$. En el primer caso haríamos 999.999 multiplicaciones y en el segundo solamente $2 \times 18 + 1 = 37$.

TEMA 3

Polinomios

3.1 Formas normales y canónicas

Definición

Definición 7. Sea R un anillo conmutativo. El anillo de polinomios $R[x]$ sobre R se define como el conjunto de las expresiones formales

$$\sum_{i=0}^{n} r_i x^i = r_0 + r_1 x + \cdots + r_n x^n$$

junto con la aritmética

- $\sum_{i=0}^{n} r_i x^i + \sum_{i=0}^{m} s_i x^i = \sum_{i=0}^{\max(n,m)} (r_i + s_i) x^i$,
- $(\sum_{i=0}^{n} r_i x^i)(\sum_{i=0}^{m} s_i x^i) = \sum_{k=0}^{n+m} t_k x^k$ donde $t_k = \sum_{i+j=k} r_i s_j$

Una variable

Ejemplos: $\mathbb{Z}[x], \mathbb{Q}[x], \mathbb{R}[x], \mathbb{C}[x]$

- Forma normal:
$$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \quad \text{con } e_i \neq e_j \text{ cuando } i \neq j$$
- Forma canónica:
$$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \quad \text{con } e_i < e_j \text{ cuando } i > j$$

Varias variables

Tratamos con $R[x_1][x_2][x_3]\ldots[x_\nu] = R[x_1, x_2, \ldots, x_\nu]$.

Recursiva $\leftrightsquigarrow$ Distributiva

Recursiva:

$$a(x) = \sum_{i=0}^{\deg_\nu(a(x))} a_i(x_1, \ldots, x_{\nu-1}) x_\nu^i$$

Distributiva:

$$a(x) = \sum_{\alpha \in \mathbb{N}^\nu} a_i x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_\nu^{\alpha_\nu}$$

Dispersa $\leftrightsquigarrow$ Densa

- Dispersa: Sólo se almacenan los términos no nulos
- Densa: Se almacenan todos los términos

Formas asociadas a factorización

Si R es un dominio, otra forma normal asociada a $R[x_1, \ldots, x_\nu]$ es representar cada polinomio como

$$\prod_{i=1}^{k} p_i$$

- Si los elementos p_i no están en forma factorizada la forma obtenida es una forma normal
- Si R es un DFU y los p_i son irreducibles y normalizados la forma obtenida es canónica

Estructuras de Datos: Listas enlazadas

Adecuadas a la representación recursiva:

$$a(x) = \sum_{i=0}^{\deg_\nu(a(x))} a_i(x_1, \ldots, x_{\nu-1}) x_\nu^i$$

- Nodo inicial:

VAR_LINK	NEXT_LINK

- Nodo normal:

COEF_LINK	EXPONENT	NEXT_LINK

Estructuras de Datos: Vectores dinámicos

Para representación distributiva:

$$a(x) = \sum_{\alpha \in \mathbb{N}^\nu} a_i x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_\nu^{\alpha_\nu}$$

- Estructura SUMA:

TYPE/LENGTH	COEF	TERM	$\cdots$	COEF	TERM

- Estructura PRODUCTO:

TYPE/LENGTH	EXP	FACT	$\cdots$	EXP	FACT

Estructuras de Datos: Ejemplo

$$\begin{aligned} a(x, y, z) &= 3y^2z^2 - 2x^3yz^2 + 5x^2z^2 + 4z - x^4 + 1 \\ &= (3y^2 + (-2x^3)y + 5x^2)z^2 + 4z + (-x^4 + 1) \end{aligned}$$

Estructuras de Datos: Factorización

Es posible utilizar listas enlazadas y vectores dinámicos.

3.2 Aritmética clásica

Suma: Algoritmo de la escuela (1)

Suma de polinomios de una variable

procedure SumPoly(a, b)
Require: $a = a_0 + a_1x + \cdots + a_{m-1}x^{m-1}$
Require: $b = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$
  $l \leftarrow \max(m, n)$
  for i **from** 0 **to** $l - 1$ **do**
    $c_i \leftarrow a_i + b_i$
  end for
  return $c_0 + c_1x + \cdots + c_{l-1}x^{l-1}$
end procedure

Suma: Algoritmo de la escuela (y 2)

Para cada $a = \sum_{\alpha\in\mathbb{N}^v} a_\alpha x_1^{\alpha_1}\cdots x_v^{\alpha_v} \in R[x_1, \ldots, x_v]$, llamamos $\operatorname{sop}(a) = \{\alpha \in \mathbb{N}^v \mid a_\alpha \neq 0\}$

Suma de polinomios de varias variables

procedure SumPolyVars(a, b)
Require: $a = \sum_{\alpha\in\mathbb{N}^v} a_\alpha x_1^{\alpha_1}\cdots x_v^{\alpha_v}$
Require: $b = \sum_{\alpha\in\mathbb{N}^v} b_\alpha x_1^{\alpha_1}\cdots x_v^{\alpha_v}$
  $\mathcal{B} \leftarrow \operatorname{sop}(a) \cup \operatorname{sop}(b)$
  for each $\alpha \in \mathcal{B}$ **do**
    $c_\alpha \leftarrow a_\alpha + b_\alpha$
  end for
  return $\sum_{\alpha\in\mathcal{B}} c_\alpha x_1^{\alpha_1}\cdots x_v^{\alpha_v}$
end procedure

Multiplicación: Algoritmo de la escuela (1)

Multiplicación de dos polinomios en una variable

procedure ProdPoly(a, b, β)
Require: $a = a_0 + a_1x + \cdots + a_{m-1}x^{m-1}$
Require: $b = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$
  for i **from** 0 **to** $m - 1$ **do** $c_i \leftarrow 0$
  end for
  for k **from** 0 **to** $n - 1$ **do**
    for i **from** 0 **to** $m - 1$ **do**
      $c_{i+k} \leftarrow a_ib_k + c_{i+k}$
    end for
  end for
  return $c = c_0 + c_1x + \cdots + c_{m+n-1}x^{m+n-1}$
end procedure

Multiplicación: Algoritmo de la escuela (y 2)

Multiplicación de dos polinomios en varias variables

procedure ProdPoly(a, b, β)
Require: $a = \sum_{\alpha\in\mathbb{N}^v} a_\alpha x_1^{\alpha_1}\cdots x_v^{\alpha_v}$
Require: $b = \sum_{\alpha\in\mathbb{N}^v} b_\alpha x_1^{\alpha_1}\cdots x_v^{\alpha_v}$

```
    for each α ∈ sop(a) do c_α ← 0
    end for
    for each β ∈ sop(b) do
        for each α ∈ sop(a) do
            c_{α+β} ← a_α b_β + c_{α+β}
        end for
    end for
    return c = Σ_{α∈ℕ^ν} c_α x_1^{α_1} ... x_ν^{α_ν}
end procedure
```

3.3 Transformada rápida de Fourier

Representación modular de un polinomio

A lo largo de esta parte $\mathbb{F}$ representa un cuerpo, es decir, $\mathbb{F}$ puede ser $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ y $\mathbb{Z}_p$ con p un número primo. Muchas de las afirmaciones son válidas para dominios como $\mathbb{Z}$.

Sea

$$a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \in \mathbb{F}[x]$$

Dividimos $a(x)$ entre $x - u$

$$a(x) = q(x)(x-u) + a(u)$$

donde

$$a(u) = a_0 + a_1 u + \cdots + a_{n-1}u^{n-1} \in \mathbb{F}$$

¡El resto de la división entre $x - u$ coincide con la evaluación en u!

Evaluación de polinomios

Teorema 8 (Método de Horner). ▪ *Sea* $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \in \mathbb{F}[x]$.

- *Sea* $b_n = 0$.
- *Sea* $b_k = a_k + b_{k+1}u$ *para* $k = n-1, n-2, \ldots, 1, 0$.

Entonces

- $a(u) = b_0$,
- *si* $b(x) = b_1 + b_2 x + \cdots + b_{n-1}x^{n-2}$, $a(x) = b(x)(x-u) + a(u)$

Para calcular b_0 necesitamos $2n$ operaciones en $\mathbb{F}$, luego

Complejidad $\mathcal{O}(n)$

Teorema chino del resto

Teorema 9. *Si $u_0, \dots, u_{n-1} \in \mathbb{F}$ son distintos, entonces el sistema de ecuaciones*

$$\begin{cases} a(x) \equiv b_0 \pmod{(x-u_0)} \\ \qquad \vdots \\ a(x) \equiv b_{n-1} \pmod{(x-u_{n-1})} \end{cases}$$

tiene solución única módulo $M(x) = (x-u_0)(x-u_1)\dots(x-u_{n-1})$.

Existe un único polinomio $a(x)$ de grado menor que n tal que

$$a(u_0) = b_0, \dots, a(u_{n-1}) = b_{n-1}$$

Transformaciones genéricas

Fijamos $u_0, \dots, u_{n-1} \in \mathbb{F}$ distintos.

Directa

Dado el polinomio $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, calcular $b_0 = a(u_0), \dots, b_{n-1} = a(u_{n-1})$, es decir,

$$\begin{aligned} T_{(u_0,\dots,u_{n-1})} : \mathbb{F}^n &\longrightarrow \mathbb{F}^n \\ (a_0, \dots, a_{n-1}) &\longmapsto (b_0, \dots, b_{n-1}) \end{aligned}$$

Inversa

Dados $b_0, \dots, b_{n-1} \in \mathbb{F}$ encontrar el único polinomio $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$ tal que $a(u_0) = b_0, \dots, a(u_{n-1}) = b_{n-1}$, es decir,

$$\begin{aligned} T^{-1}_{(u_0,\dots,u_{n-1})} : \mathbb{F}^n &\longrightarrow \mathbb{F}^n \\ (b_0, \dots, b_{n-1}) &\longmapsto (a_0, \dots, a_{n-1}) \end{aligned}$$

Simetría/Evaluación

Sea $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$ con n par. Podemos representar $a(x)$ como

$$a(x) = b(x^2) + x c(x^2)$$

donde

$$b(y) = a_0 + a_2 y + \cdots + a_{n-2} y^{\frac{n}{2}-1} \text{ y } c(y) = a_1 + a_3 y + \cdots + a_{n-1} y^{\frac{n}{2}-1}$$

Lema 10. *Sea $\{u_0, \dots, u_{n-1}\}$ un conjunto de n elementos distintos de un cuerpo $\mathbb{F}$ que satisfacen la condición de simetría*

$$u_{\frac{n}{2}+i} = -u_i \quad \textit{para todo } i \in \{0, 1, \dots, \frac{n}{2} - 1\} \tag{sym}$$

Si $T(n)$ es el coste de evaluar un polinomio de grado $n-1$ en los puntos anteriores, entonces

$$T(1) = 0, \text{ y } T(n) = 2T\left(\frac{n}{2}\right) + c\frac{n}{2}$$

Raíces de la unidad

Estrategia:
Elegir $\{u_0, \ldots, u_{n-1}\}$ de modo que podamos aplicar el lema de forma recursiva.

Definición 11. Un elemento ω de un cuerpo $\mathbb{F}$ es una raíz n-ésima primitiva de la unidad si

$$\omega^n = 1, \text{ pero } \omega^k \neq 1 \text{ para cada } 0 < k < n$$

Cuando ω es una raíz n-ésima primitiva de la unidad, el conjunto $\{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$ recibe el nombre de *puntos de Fourier*.

Transformada Discreta de Fourier

Recordemos la aplicación

$$\begin{aligned} T_{(1,\omega,\ldots,\omega^{n-1})} : \mathbb{F}^n &\longrightarrow \mathbb{F}^n \\ (a_0, \ldots, a_{n-1}) &\longmapsto (b_0, \ldots, b_{n-1}) \end{aligned}$$

donde

$$b_i = a_0 + a_1\omega^i + a_2(\omega^i)^2 + \cdots + a_{n-1}(\omega^i)^{n-1},$$

es decir, b_i es la evaluación del polinomio $a(x) = a_0 + a_1 x + \ldots, a_{n-1}x^{n-1}$ en ω^i.

La aplicación $T_{(1,\omega,\ldots,\omega^{n-1})}$ se llama *Transformada Discreta de Fourier (DFT).*

Resultados técnicos

Lema 12. *Sea ω una raíz n-ésima primitiva de la unidad donde n es par. Entonces*

1. *El conjunto de puntos de Fourier $\{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$ satisface la condición de simetría* (sym)*,*
2. *ω^2 es una raíz $\frac{n}{2}$-primitiva de la unidad,*
3. *Si $\frac{n}{2}$ es par entonces el conjunto de puntos $\{1, \omega^2, \omega^4, \ldots, \omega^{n-2}\}$ satisface la condición de simetría* (sym)*.*

Transformada Rápida de Fourier

FFT
procedure FFT($2^m, \omega, a(x)$)
Require: ω es una raíz 2^m-ésima primitiva de la unidad
Require: $a(x)$ es un polinomio de grado menor que 2^m
  if $2^m = 1$ **then**
    $A_0 \leftarrow a_0$
  else
    $b(x) \leftarrow \sum_{i=0}^{2^{m-1}-1} a_{2i}x^i$ y $c(x) \leftarrow \sum_{i=0}^{2^{m-1}-1} a_{2i+1}x^i$
    $B \leftarrow \text{FFT}(2^{m-1}, \omega^2, b(x))$ y $C \leftarrow \text{FFT}(2^{m-1}, \omega^2, c(x))$
    for i **from** 0 **to** $2^{m-1} - 1$ **do**
      $A_i \leftarrow B_i + \omega^i C_i$ y $A_{2^{m-1}+i} \leftarrow B_i - \omega^i C_i$
    end for
  end if
  return $(A_0, A_1, \ldots, A_{2^m-1})$
end procedure

Transformada Discreta de Fourier Inversa

Definición 13. La *Transformada Discreta de Fourier Inversa (IDFT)* para un conjunto de puntos de Fourier es

$$S_{(1,\omega,\ldots,\omega^{n-1})}(q_0,\ldots,q_{n-1}) = (r_0,\ldots,r_{n-1})$$

donde

$$r_j = n^{-1}\sum_{k=0}^{n-1} q_k(\omega^{-j})^k$$

Teorema 14. *DFT y IDFT son inversas una de la otra.*

Complejidad

$$S_{(1,\omega,\ldots,\omega^{n-1})} = n^{-1}T_{(1,\omega^{-1},\ldots,\omega^{-n+1})},$$

es decir, ambas tienen la misma naturaleza.

La complejidad es $\mathcal{O}(n\log n)$

Multiplicación rápida de polinomios

Multiplicación rápida mediante FFT

procedure MULTIPLICACIÓNFFT($a(x), b(x), m, n$)
Require: $a(x)$ es un polinomio de grado m
Require: $b(x)$ es un polinomio de grado n
$N \leftarrow$ primer entero tal que $m+n < 2^N$
$\omega \leftarrow$ raíz 2^N-ésima primitiva de la unidad
$A \leftarrow \mathrm{FFT}(2^N, \omega, a(x))$
$B \leftarrow \mathrm{FFT}(2^N, \omega, b(x))$
for i **from** 0 **to** $2^N - 1$ **do** $C_i = A_iB_i$
end for
$c \leftarrow (2^N)^{-1}\mathrm{FFT}(2^N, \omega^{-1}, C(x))$
$c(x) \leftarrow \sum_{i=0}^{2^N-1} c_ix^i$
return $c(x)$
end procedure

Raíces primitivas de la unidad: Caso complejo

Una raíz n-ésima primitiva de la unidad es

$$\omega = e^{\frac{2\pi i}{n}}$$

o en representación cartesiana

$$\omega = e^{\frac{2\pi i}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$$

Raíces primitivas de la unidad: Caso finito (1)

Teorema 15. *El cuerpo $\mathbb{Z}_p$ tiene una raíz n-ésima primitiva de la unidad si y sólo si n divide a $p-1$*

Nos interesan entonces *Primos de Fourier*, que son de la forma

$$p = 2^r k + 1$$

Para un r fijo existen aproximadamente

$$\frac{x}{\log x \cdot 2^{r-1}}$$

primos de Fourier menores que x

Raíces primitivas de la unidad: Caso finito (y 2)

Proposición 16. *Si α es un generador del grupo multiplicativo $\mathbb{Z}_p^\times = \mathbb{Z}_p \setminus \{0\}$ entonces $\omega = \alpha^{\frac{p-1}{n}}$ es una raíz n-ésima primitiva de la unidad.*

Teorema 17. *Un elemento a es un generador de $\mathbb{Z}_p^\times$ si y sólo si*

$$a^{\frac{p-1}{q}} \not\equiv 1 \mod p$$

para cada factor primo q de $p-1$

Los generadores de $\mathbb{Z}_p^\times$ se pueden obtener por fuerza bruta ya que existen

$$\frac{\phi(p-1)}{p-1} \approx \frac{3}{\pi^2} \approx 0{,}3$$

Reinterpretación

Sea $\{1, \omega, \ldots, \omega^{2^N-1}\}$ un conjunto de puntos de Fourier. Como

$$(x-1)(x-\omega)\ldots(x-\omega^{2^N-1}) = x^{2^N} - 1$$

la multiplicación usando FFT es módulo $x^{2^N} - 1$. Por tanto, dados dos polinomios a, b de grado menor que 2^N, tenemos

$$a(x)b(x) \equiv \mathrm{IDFT}(2^N, \omega, \mathrm{DFT}(2^N, \omega, a(x))\mathrm{DFT}(2^N, \omega, b(x))) \pmod{x^{2^N} - 1}$$

3.4 Schönhage–Strassen

Schönhage-Strassen: Idea

Este algoritmo utiliza la FFT para multiplicar enteros. La idea consiste en hacer una serie de reducciones en el producto:

$$\mathbb{Z} \Rightarrow \mathbb{Z}_{2^N+1} \Rightarrow \mathbb{Z}[x]_{x^K+1} \Rightarrow \mathbb{Z}_n[x]_{x^K+1} \Rightarrow \mathbb{Z}_n$$

El primer paso es sencillo. Basta con seleccionar N suficientemente grande como para que el producto de los dos enteros sea menor que $2^N + 1$. Explicamos las demás reducciones a realizar.

$\mathbb{Z}_{2^N+1} \Rightarrow \mathbb{Z}[x]_{x^K+1}$

Escribimos $N = 2^k M$ y llamamos $K = 2^k$. Cada entero $u \in [0, 2^N]$ se puede expresar de manera única como

$$a = \sum_{i=0}^{K-1} a_i 2^{iM} \quad \text{donde } 0 \leq a_i < 2^M \text{ para } i < K-1 \text{ y } 0 \leq a_{K-1} \leq 2^M$$

Tenemos que $a = a(2^M)$ para el polinomio $a(x) = \sum_{i=0}^{K-1} a_i x^i$

$\mathbb{Z}[x]_{x^K+1} \Rightarrow \mathbb{Z}_n[x]_{x^K+1}$

Para multiplicar $a, b \in \mathbb{Z}_{2^N+1}$, multiplico $a(2^M)b(2^M) \pmod{2^N+1}$, lo que puedo hacer evaluando el polinomio

$$c(x) = a(x)b(x) \pmod{x^K+1}$$

Cada coeficiente de $c(x)$ se escribe como

$$c_i = \sum_{j=0}^{i} a_j b_{i-j} - \sum_{j=0}^{i+k} a_j b_{i+k-j},$$

por lo que los coeficientes están en un intervalo de rango $K2^{2M}$. Tomamos $n > 2M + k$ para que los coeficientes no estén fuera de rango.

$\mathbb{Z}[x]_{x^K+1} \Rightarrow \mathbb{Z}_n[x]_{x^K+1}$

Tomamos como n un múltiplo de K. De esta forma $\omega = 2^{2n/K}$ es una raíz K-ésima primitiva de la unidad. Llamamos $\theta = 2^{n/K}$, es decir, $\omega = \theta^2$. Llamamos $a'(x) = \sum_{i=0}^{K-1} \theta^i a_i x^i$, y de idéntica manera realizamos la operación para $b(x)$.

Lema 18. *Si* $c'(x) \equiv a'(x)b'(x) \pmod{2^n+1, x^{2K}-1}$ *entonces* $c'_i = K\theta^i c_i$.

De esta forma obtenemos los datos necesarios para reconstruir $c(2^M)$.

Multiplicación de enteros mediante FFT y tres primos

Una dificultad no utilizada es el problema del cálculo de raíces primitivas de la unidad.

FFT y Tres primos

procedure MULTIPLICACIÓNFFT(a, b)
Require: $a, b < 2^{64 \cdot 2^s}$ donde $s \leq 62$
Require: se han precalculado tres parejas $\{(p_i, \omega_i\}_{1 \leq i \leq 3}$ tales que $2^{63} \leq p_i < 2^{64}$ y $0 < \omega_i < p_i$ es una raíz 2^s-ésima primitiva en $\mathbb{Z}_{p_i}$
sean $A, B \in \mathbb{Z}[x]$ tales que sus coeficientes están entre 0 y 2^{64},
$a = A(2^{64})$, $b = B(2^{64})$ y $t = \lceil \log_2(1 + \deg(AB)) \rceil$
for i **from** 1 **to** 3 **do**$C_i = AB \pmod{p_i}$ donde $\omega = \omega_i^{2^{s-t}}$
end for
Encontramos mediante el TCR $C \in \mathbb{Z}[x]$ tal que $C \equiv C_i \pmod{p_i}$ para cada $1 \leq i \leq 3$
return $C(2^{64})$
end procedure

TEMA 4

Factorización de polinomios

Generalidades

El problema de la factorización de polinomios está ligado a la naturaleza del anillo sobre el que trabajamos. Así, en polinomios sobre $\mathbb{R}$ o sobre $\mathbb{C}$, los métodos que funcionan son eminentemente numéricos, ya que la propia naturaleza de los reales y los complejos hace que un ordenador sólo pueda manipularlos por aproximación.

El caso de $\mathbb{Z}$, $\mathbb{Q}$ o $\mathbb{F}_q$ es distinto ya que cualquiera de sus elementos puede representarse.

Vamos a centrarnos en dos casos, $\mathbb{Z}[x]$ y $\mathbb{F}[x, y]$ que se estudian de forma similar. El caso racional puede ser reducido a los enteros mediante el Lema de Gauss.

4.1 Factorización libre de cuadrados

Conceptos

Recordemos que un polinomio $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} \in \mathbb{D}[x]$ se dice primtivo si $\mathrm{mcd}(a_0, a_1, \ldots, a_{n-1}) = 1$

Siempre podemos suponer que nuestros polinomios son primitivos

Definición 19. Un polinomio primitivo $a(x) \in \mathbb{D}[x]$ se dice libre de cuadrados si no existe $b(x)$ con $\deg b(x) \geq 1$ tal que $b(x)^2 \mid a(x)$

Definición 20. La descomposición libre de cuadrados de $a(x)$ es $a(x) = \prod_{i=1}^{k} a_i(x)^i$ donde cada $a_i(x)$ es libre de cuadrados y para cada $i \neq j$ $\mathrm{mcd}(a_i(x), a_j(x)) = 1$

Derivación

El concepto de derivada es un concepto simbólico. La derivada de $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ se define como

$$a'(x) = a_1 + 2a_2x + \cdots + (n-1)a_{n-1}x^{n-2}$$

Satisface

- linealidad: $(a(x) + b(x))' = a'(x) + b'(x)$, $(\lambda a(x))' = \lambda a'(x)$,
- regla del producto: $(a(x)b(x))' = a'(x)b(x) + a(x)b'(x)$,
- regla de la potencia: $(a(x)^n)' = na(x)^{n-1}a'(x)$

Derivación y factores múltiples: característica 0 (1)

Teorema 21. *Sea $\mathbb{D}$ un DFU de característica 0. Un polinomio $a(x) \in \mathbb{D}[x]$ tiene factores múltiples si y sólo si* $\mathrm{mcd}(a(x), a'(x)) \neq 1$.

Sea $a(x) = \prod_{i=1}^{k} a_i(x)^i$. Entonces

$$a'(x) = \sum_{i=1}^{k} a_1(x) \ldots i a_i(x)^{i-1} a_i'(x) \ldots a_k(x)^k$$

Por tanto

$$c(x) = \mathrm{mcd}(a(x), a'(x)) = \prod_{i=2}^{k} a_i(x)^{i-1}$$

Derivación y factores múltiples: característica 0 (2)

Si

$$w(x) = a(x)/c(x), \qquad y(x) = \mathrm{mcd}(c(x), w(x))$$

entonces $a_1(x) = w(x)/y(x)$. El segundo factor libre de cuadrados de $a(x)$ es el primero libre de cuadrados de $c(x)$. Ahora

$$\mathrm{mcd}(c(x), c'(x)) = \prod_{i=3}^{k} a_i(x)^{i-1} = c(x)/y(x)$$

y el producto de los demás factores libres de cuadrados es $y(x)$.

Derivación y factores múltiples: característica 0 (y 3)

Factorización libre de cuadrados

procedure LibreDeCuadrados($a(x)$)
  $i \leftarrow 1$, $Output \leftarrow 1$, $b(x) \leftarrow a'(x)$
  $c(x) \leftarrow \mathrm{mcd}(a(x), b(x))$, $w(x) \leftarrow a(x)/c(x)$
  while $c(x) \neq 1$ **do**
    $y(x) \leftarrow \mathrm{mcd}(w(x), c(x))$, $z(x) = w(x)/y(x)$
    $Output \leftarrow Output \cdot z(x)^i$, $i \leftarrow i + 1$
    $w(x) \leftarrow y(x)$, $c(x) \leftarrow c(x)/y(x)$
  end while
  $Output \leftarrow Output \cdot w(x)^i$
  return $Output$
end procedure

Derivación y factores múltiples: característica p (1)

En este caso puede ocurrir lo siguiente:

Teorema 22. *Sea $a(x) \in \mathbb{F}_{p^t}[x]$. Entonces*

$$a'(x) = 0 \iff a(x) = b(x)^p$$

para algún $b(x) \in \mathbb{F}_{p^t}[x]$.

En este caso, $a(x) = a_0 + a_p x^p + a_{2p} x^{2p} + \cdots + a_{kp} x^{kp}$. Necesariamente

$$\sqrt[p]{a(x)} = b(x) = a_0 + a_p x + a_{2p} x^2 + \cdots + a_{kp} x^k$$

El algoritmo se puede escribir de la siguiente forma

Derivación y factores múltiples: característica p (y 2)

Factorización libre de cuadrados en Cuerpos Finitos

procedure LibreDeCuadradosCF($a(x)$)
  $i \leftarrow 1$; $Output \leftarrow 1$; $b(x) \leftarrow a'(x)$
  if $b(x) \neq 0$ **then**
    $c(x) \leftarrow \mathrm{mcd}(a(x), b(x))$; $w(x) \leftarrow a(x)/c(x)$
    while $w(x) \neq 1$ **do**

```
        y(x) ← mcd(w(x), c(x)); z(x) ← w(x)/y(x)
        Output ← Output · z(x)^i; i ← i + 1
        w(x) ← y(x); c(x) ← c(x)/y(x)
      end while
      if c(x) ≠ 1 then
        c(x) ← ∛c(x); Output ← Output · (LibreDeCuadradosCF(c(x)))^p
      end if
    else
      a(x) ← ∛a(x); Output ← Output · (LibreDeCuadradosCF(a(x)))^p
    end if
    return Output
  end procedure
```

4.2 Algoritmo de Berlekamp

Número de factores

En esta sección $a(x) = a_0 + a_1 x + \cdots + x^n \in \mathbb{F}_q[x]$ es un polinomio libre de cuadrados. No hemos perdido generalidad al suponer que es mónico, es decir, tiene coeficiente líder 1.

Vamos a trabajar en el cociente $V = (\mathbb{F}_q[x])_{a(x)}$. Llamamos

$$W = \{v(x) \in \mathbb{F}_q[x] \mid v(x)^q \equiv v(x) \mod a(x)\}$$

Teorema 23. *W es un $\mathbb{F}_q$–subespacio vectorial de V. Además $\dim_{\mathbb{F}_q} W$ coincide con el número de factores irreducibles de $a(x)$*

W y factores irreducibles

Teorema 24. *Sea $a(x)$ un polinomio libre de cuadrados en $\mathbb{F}_q[x]$, y sea $v(x)$ un polinomio no constante en W. Entonces*

$$a(x) = \prod_{s \in \mathbb{F}_q} \mathrm{mcd}(v(x) - s, a(x))$$

Para encontrar los factores irreducibles de $a(x)$ hay que determinar los elementos no constantes de W y calcular ciertos mcd. El número de elementos no constantes en W y el propio número de elementos en $\mathbb{F}_q$ puede hacer este cálculo prohibitivo.

Simplificación en W

La primera idea para simplificar consiste en reducir la selección de los elementos no constantes de W a una base del mismo. El cálculo de la base lo basamos en el conocimiento de ecuaciones cartesianas (o implícitas) de W. Observemos que

$$W = \{v(x) \in \mathbb{F}_q[x] \mid v(x^q) \equiv v(x) \mod a(x)\}$$

ya que $v(x)^q = v(x^q)$ (¿por qué?)

Teorema 25. *Sea $Q = (q_{i,j})$ con $0 \le i, j \le n-1$ la matrix con coeficientes en $\mathbb{F}_q$ definida por*

$$x^{q \cdot j} \equiv q_{j,0} + q_{j,1} x + \cdots + q_{j,n-1} x^{n-1} \mod a(x)$$

Entonces, identificando un polinomio con la lista de sus coeficientes

$$W = \{v = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_q^n \mid v(Q - I) = 0\}$$

donde I es la matriz identidad.

Cálculo de Q

Podemos utilizar un método iterativo. Observemos que

$$x^n \equiv -a_0 - a_1 x - \cdots - a_{n-1}x^{n-1} \mod a(x)$$

Supongamos que

$$x^m \equiv r_{m,0} + r_{m,1}x + \cdots + r_{m,n-1}x^{n-1} \mod a(x)$$

Entonces

$$\begin{aligned} x^{m+1} &\equiv r_{m,0}x + r_{m,1}x^2 + \cdots + r_{m,n-1}x^n \mod a(x) \\ &\equiv r_{m,0}x + r_{m,1}x^2 + \cdots + r_{m,n-1}(-a_0 - a_1 x - \cdots - a_{n-1}x^{n-1}) \mod a(x) \\ &\equiv -r_{m,n-1}a_0 + (r_{m,0} - r_{m,n-1}a_1)x + \\ &\quad \cdots + (r_{m,n-2} - r_{m,n-1}a_{n-1})x^{n-1} \mod a(x) \end{aligned}$$

Esto nos da un procedimiento iterativo. Si llamamos

$$r_m = (r_{m,0}, r_{m,1}, \ldots, r_{m,n-1}),$$

entonces

$$\begin{aligned} r_0 &= (1, 0, \ldots, 0) \\ &\vdots \\ r_{m+1} &= (-r_{m,n-1}a_0, r_{m,0} - r_{m,n-1}a_1, \ldots, r_{m,n-2} - r_{m,n-1}a_{n-1}), \end{aligned}$$

lo que nos permite calcular las filas de Q.

Otra aproximación es utilizar cuadrados iterados (de forma similar a la potenciación rápida de enteros). Esta aproximación supone una mejora siempre que p sea grande.

Cálculo de Q, algoritmo

Construcción de Q

procedure MatrizQ($a(x), q$)
Require: polinomio $a(x)$ sobre $\mathbb{F}_q$ de grado n
Require: La matriz Q es la requerida en el algoritmo de Berlekamp
  $n \leftarrow \deg(a(x))$; $\mathbf{r} \leftarrow (1, 0, \ldots, 0)$; Fila$(0, Q) \leftarrow \mathbf{r}$
  for m **from** 1 **to** $(n-1)q$ **do**
    $\mathbf{r} \leftarrow (-r_{n-1}a_0, r_0 - r_{n-1}a_1, \ldots, r_{n-2} - r_{n-1}a_{n-1})$
    if $q \mid m$ **then**
      Fila$(\frac{m}{q}, Q) \leftarrow \mathbf{r}$
    end if
  end for
  return Q
end procedure

Cálculo de W

El cálculo de W puede hacerse con técnicas de álgebra lineal. Necesitamos calcular una base del subespacio nulo asociado a una matriz. Una de las formas es hacer operaciones sobre las columnas de $Q - I$ hasta transformarla en una matriz triangular inferior e idempotente L. Una base de W es entonces el conjunto de las filas no nulas de $L - I$.

Algoritmo de Berlekamp

procedure Berlekamp($a(x), p$)

Require: polinomio $a(x)$ sobre $\mathbb{F}_q$ de grado n libre de cuadrados
  $Q \leftarrow$ MatrizQ($a(x), q$); $\mathbf{v}^{[1]}, \mathbf{v}^{[2]}, \ldots, \mathbf{v}^{[k]} \leftarrow$ CalculaBase(Q-I)
Require: Podemos suponer $\mathbf{v}^{[1]} = (1, 0, \ldots, 0)$
  $factores \leftarrow \{a(x)\}$; $r \leftarrow 2$
  while Size($factores$) $< k$ **do**
    for each $u(x) \in factores$ **do**
      for each $s \in \mathbb{F}_q$ **do**
        $g(x) \leftarrow \gcd(\mathbf{v}^{[r]}(x) - s, u(x))$
        if $g(x) \neq 1$ **or** $g(x) \neq u(x)$ **then**
          Quitar($u(x), factores$); Poner($\{\frac{u(x)}{g(x)}, g(x)\}, factores$)
        end if
        if Size($factores$) $= k$ **then**
          return $factores$
        end if
      end for
      $r \leftarrow r + 1$
    end for
  end while
end procedure

Simplificación en elementos de $\mathbb{F}_q$

De nuevo, si q es un entero elevado el cálculo de la descomposición

$$a(x) = \prod_{s \in \mathbb{F}_q} \operatorname{mcd}(v(x) - s, a(x))$$

obliga a realizar demasiados mcd. Para simplificar el cálculo debemos buscar los elementos no triviales. Llamemos

$$S = \{s \in \mathbb{F}_q \mid \operatorname{mcd}(a(x), v(x) - s) \neq 1\}, \qquad m_v(x) = \prod_{s \in S} (x - s)$$

El polinomio $m_v(x)$ es el polinomio mínimo para $v(x)$, es decir, el polinomio de menor grado tal que

$$m_v(v(x)) \equiv 0 \mod a(x)$$

y puede ser calculado de varias formas. Por ejemplo, calculamos para cada $r \geq 1$ $v(x)^r \mod a(x)$. La primera solución no trivial de la ecuación

$$m_0 + m_1 v(x) + \cdots + m_r v(x)^r \equiv 0 \mod a(x)$$

proporciona el polinomio mínimo.

El conocimiento de $m_v(x)$ reduce el número de mcd necesarios, pero el cálculo de $m_v(x)$ también tiene un coste. En general es conveniente calcular el polinomio mínimo si q es grande.

4.3 Factorización en distinto grado

Resultado principal

La factorización en distinto grado de un polinomio mónico libre de cuadrados $f \in \mathbb{F}_q[x]$ es $f = g_1 \ldots g_s$ donde para cada $1 \leq i \leq s$ g_i es el producto de todos los polinomios mónicos irreducibles de grado i que dividen a f. Entendemos que $g_s \neq 1$ aunque para otros valores de $i < s$ puede ocurrir $g_i = 1$.

Teorema 26. *Para cualquier $d \geq 1$, $x^{q^d} - x \in \mathbb{F}_q[x]$ es el producto de todos los polinomios mónicos irreducibles en $\mathbb{F}_q[x]$ cuyo grado divide a d.*

Este resultado generaliza al teorema pequeño de Fermat, $a^{p-1} \equiv 1 \pmod p$ para cualquier $0 < a < p$.

Factorización en distinto grado

procedure DistintoGrado($f(x)$)
Require: $f(x) \in \mathbb{F}_q[x]$ libre de cuadrados.
  $h_0(x) \leftarrow x$; $f_0(x) \leftarrow f(x)$; $i \leftarrow 0$
  repeat
    $i \leftarrow i + 1$
    $h_i(x) \leftarrow h_{i-1}(x)^q \pmod{f(x)}$
    $g_i(x) \leftarrow \text{mcd}(h_i(x) - x, f_{i-1})$; $f_i(x) \leftarrow f_{i-1}(x)/g_i(x)$
  until $f_i(x) = 1$
  $s \leftarrow i$
  return $g_1, (x) \ldots, g_s(x)$
end procedure

4.4 Algoritmo de Cantor-–Zassenhaus

Input y Output

Input: Un polinomio $f(x) \in \mathbb{F}_q[x]$ libre de cuadrados y cuyos factores irreducibles tienen el mismo grado.

Output: Un polinomio $g(x) \in \mathbb{F}_q[x]$ tal que $g(x) \mid f(x)$

Llamamos $R = \mathbb{F}_q[x]_{f(x)}$ y supongamos que $f(x) = p_1(x) \ldots p_s(x)$ donde $p_i(x)$ es irreducible y $\deg(p_i) = d$ para todo $1 \leq i \leq s$.

Caso q impar: resultado base.

Lema 27. *Sea $a(x) \in \mathbb{F}_q[x]$ tal que $\deg(a) < \deg(f)$. Entonces*

$$f(x) = \text{mcd}(f(x), a(x)) \, \text{mcd}(f(x), a(x)^{\frac{q^d-1}{2}} - 1) \, \text{mcd}(f(x), a(x)^{\frac{q^d-1}{2}} + 1)$$

La probabilidad de que la descomposición anterior no produzca factores no triviales es menor que $\frac{1}{2}$.

Caso q impar: cálculo de un factor

Cantor-Zassenhaus Splitting
procedure CZSplitting($f(x)$)
Require: q impar
Require: $f(x) \in \mathbb{F}_q[x]$ libre de cuadrados cuyos factores irreducibles tienen todos grado d.
  $a(x) \in \mathbb{F}_q[x]$, $0 < \deg(a) < \deg(f)$ aleatorio
  $g(x) \leftarrow \text{mcd}(f(x), a(x))$
  if $g(x) \neq 1$ **then**
    return $g(x)$
  end if
  $b(x) \leftarrow a(x)^{\frac{q^d-1}{2}} \pmod{f(x)}$
  $g(x) \leftarrow \text{mcd}(f(x), b(x) - 1)$
  if $g(x) \neq 1$ y $g(x) \neq f(x)$ **then**
    return $g(x)$
  else
    return `fallo`
  end if
end procedure

Caso $q = 2^n$: resultado base.

Lema 28. *Sea $a(x) \in \mathbb{F}_{2^n}[x]$ tal que $\deg(a) < \deg(f)$. Entonces*

$$f(x) = \operatorname{mcd}(f(x), a(x) + a(x)^2 + \cdots + a(x)^{2^{nd-1}}) \operatorname{mcd}(f(x), 1 + a(x) + a(x)^2 + \cdots + a(x)^{2^{nd-1}})$$

Se debe a que como $q = 2^n$,

$$(a(x) + a(x)^2 + \cdots + a(x)^{2^{nd-1}})^2 = a(x)^2 + a(x)^{2^2} + \cdots + a(x)^{2^{nd}},$$

lo que implica que

$$(a(x) + a(x)^2 + \cdots + a(x)^{2^{nd-1}})(1 + a(x) + a(x)^2 + \cdots + a(x)^{2^{nd-1}}) = a(x)^{2^{nd}} - a(x).$$

La probabilidad de que la descomposición no produzca factores no triviales es menor que $\frac{1}{2}$.

Caso $q = 2^f$: cálculo de un factor

Cantor-Zassenhaus Splitting

procedure CZSplitting($f(x)$)
Require: $q = 2^n$
Require: $f(x) \in \mathbb{F}_{2^n}[x]$ libre de cuadrados cuyos factores irreducibles tienen todos grado d.
  $a(x) \in \mathbb{F}_{2^n}[x]$, $0 < \deg(a) < \deg(f)$ aleatorio
  $g(x) \leftarrow \operatorname{mcd}(f(x), a(x))$
  if $g(x) \neq 1$ **then**
    return $g(x)$
  end if
  $b(x) \leftarrow a(x) + a(x)^2 + \cdots + a(x)^{2^{nd-1}} \pmod{f(x)}$
  $g(x) \leftarrow \operatorname{mcd}(f(x), b(x) + 1)$
  if $g(x) \neq 1$ y $g(x) \neq f(x)$ **then**
    return $g(x)$
  else
    return `fallo`
  end if
end procedure

4.5 Levantamiento de Hensel

Factorización en $\mathbb{Q}[x]$ y $\mathbb{Z}[x]$

Una vez que sabemos factorizar en $\mathbb{Z}_p[x]$ nos centramos en el caso racional. Multiplicando por la fracción adecuada podemos transformar cualquier polinomio con coeficientes racionales en otro con coeficientes enteros tal que el máximo común divisor de los coeficientes sea 1. Esto es lo que se llama polinomio primitivo. Dado que las fracciones son unidades en $\mathbb{Q}[x]$ podemos limitarnos a estudiar la factorización de polinomios primitivos en $\mathbb{Z}[x]$. Sea entonces $a(x) = a_0 + a_1 x + \cdots + a_n x^n \in \mathbb{Z}[x]$ primitivo.

Método

La estrategia será:

1. Factorizar el polinomio en $\mathbb{Z}_p[x]$ para cierto primo p,
2. levantar la factorización anterior hasta $\mathbb{Z}_{p^k}$ para un p^k suficientemente grande,
3. combinar los factores anteriores hasta dar con la factorización requerida en $\mathbb{Z}[x]$

Para el primer paso la única condición requerida es que $p \nmid a_n$, aunque simplifica el procedimiento el que $a_n \equiv 1 \pmod{p}$

Lema de Hensel

El segundo paso se basa en el

Teorema 29 (Lema de Hensel). *Sea $a(x) \in \mathbb{Z}[x]$ primitivo y $p \in \mathbb{Z}$ primo. Sean $u^{(1)}(x), w^{(1)}(x) \in \mathbb{Z}_p[x]$ relativamente primos tales que*

$$a(x) \equiv u^{(1)}(x)w^{(1)}(x) \mod p.$$

Entonces para cada $k \geq 1$ podemos construir polinomios $u^{(k)}(x), w^{(k)}(x) \in \mathbb{Z}_{p^k}[x]$ tales que

$$a(x) \equiv u^{(k)}(x)w^{(k)}(x) \mod p^k$$

y

$$u^{(k)}(x) \equiv u^{(1)}(x) \mod p, \quad w^{(k)}(x) \equiv w^{(1)}(x) \mod p$$

Lema de Hensel: Construcción de las soluciones

Recurrencia en k. La pareja de soluciones para $k = 1$ viene dada de principio. Supongamos que tenemos una pareja de soluciones $u^{(k)}(x), w^{(k)}(x) \in \mathbb{Z}_{p^k}[x]$ y vamos a ver cómo podemos construir la pareja correspondiente al entero $k+1$. Como

$$a(x) \equiv u^{(k)}(x)w^{(k)}(x) \mod p^k,$$

vamos a llamar $c^{(k)}(x)$ al polinomio cuyos coeficientes son los restos que se obtienen al dividir entre p los coeficientes del polinomio $\frac{a(x)-u^{(k)}(x)w^{(k)}(x)}{p^k}$. Calculamos polinomios $\sigma^{(k)}(x), \tau^{(k)}(x) \in \mathbb{Z}_p[x]$ tales que $\deg \sigma^{(k)} < \deg w^{(1)}$ y

$$\sigma^{(k)}(x)u^{(1)}(x) + \tau^{(k)}(x)w^{(1)}(x) \equiv c^{(k)}(x) \mod p$$

Los polinomios buscados son

$$u^{(k+1)}(x) = u^{(k)}(x) + \tau^{(k)}(x)p^k, \quad w^{(k+1)}(x) = w^{(k)}(x) + \sigma^{(k)}(x)p^k$$

módulo p^{k+1}.

Cota de Landau–Mignotte

Es una cota superior para el valor absoluto de los coeficientes de cualquier factor no trivial de un polinomio $a(x) \in \mathbb{Z}[x]$. Sea $a(x) = a_0 + a_1x + \cdots + a_nx^x$, y sean

$$d = \left\lfloor \frac{n}{2} \right\rfloor, \quad |\, a \,|_2 = \sqrt{a_0^2 + a_1^2 + \cdots + a_n^2}.$$

La cota es

$$B = \binom{d-1}{\lfloor \frac{d}{2} \rfloor - 1} + \binom{d-1}{\lfloor \frac{d}{2} \rfloor} |\, a \,|_2$$

Debemos aplicar el levantamiento de Hensel hasta que $p^k > 2B$

Combinación de las soluciones

El levantamiento presentado para dos factores podemos extenderlo a r si es necesario. Los factores irreducibles de $a(x)$ módulo $p^k > 2B$ no tienen por qué corresponder con factores en $\mathbb{Z}[x]$. En cualquier caso no puede haber más factores de los calculados.

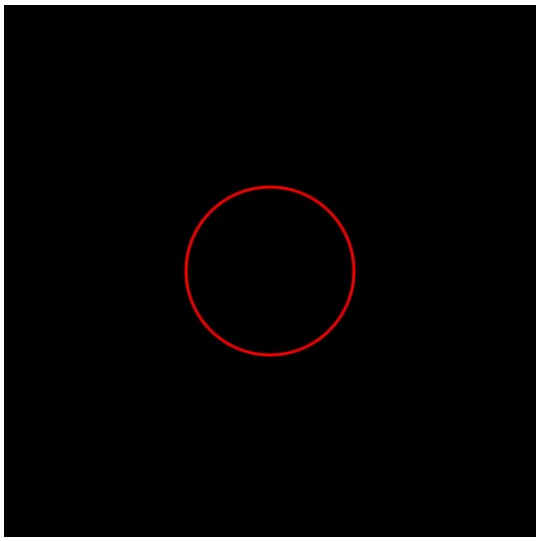
El paso final consiste en combinar los factores obtenidos para calcular los factores verdaderos de $a(x)$ en $\mathbb{Z}[x]$. En esencia tenemos que multiplicar agrupaciones de los factores obtenidos y dividir $a(x)$ entre las agrupaciones anteriores para encontrar los factores verdaderos.

TEMA 5

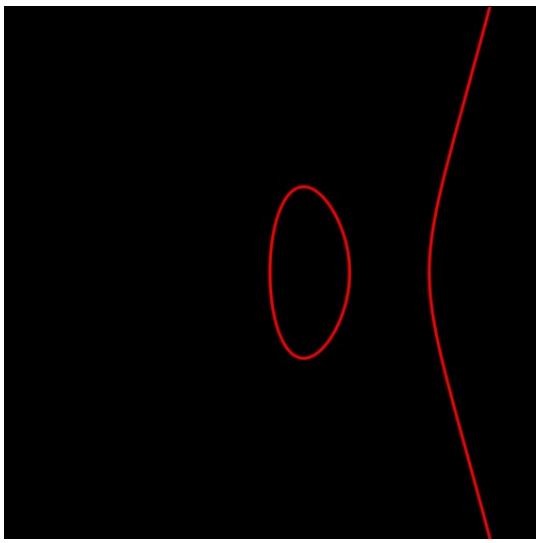
Ecuaciones y variedades algebraicas

5.1 Variedades afines

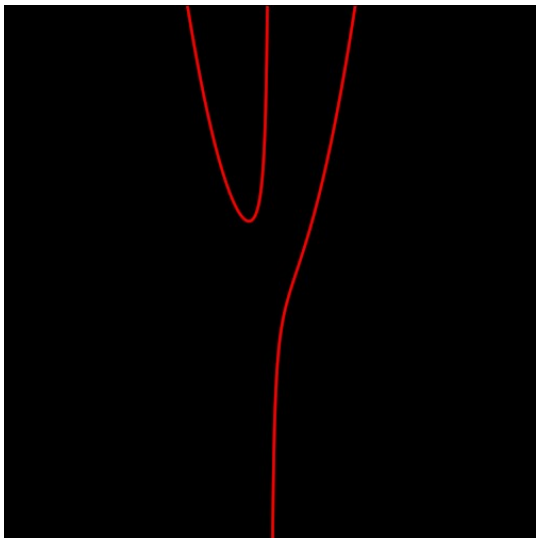
$x^2 + y^2 - 1$



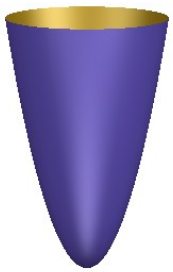
$y^2 - x(x-1)(x-2)$



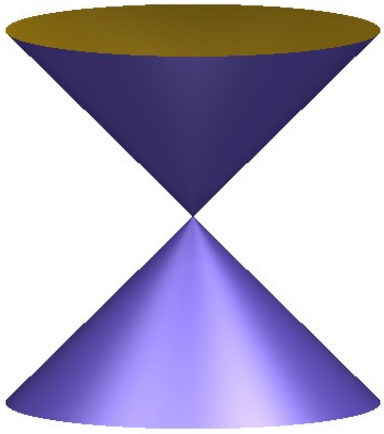
$xy - x^3 + 1$



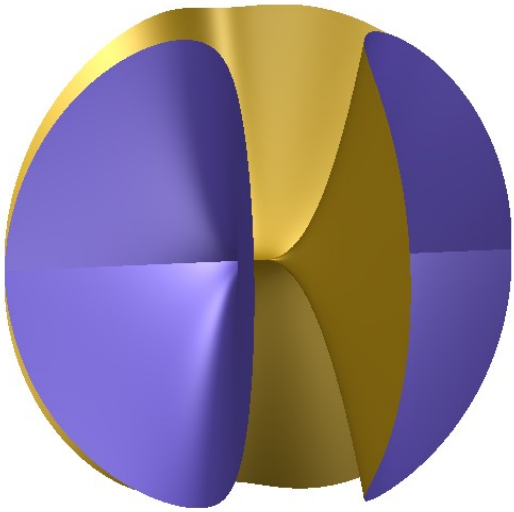
$z - x^2 - y^2$



$z^2 - x^2 - y^2$



$x^2 - y^2z^2 + z^3$



Conceptos

1. $\Bbbk$ es un cuerpo, es decir, $\mathbb{C}$, $\mathbb{R}$, $\mathbb{Q}$ o $\mathbb{F}_q$.
2. $\Bbbk^n$ el $\Bbbk$-espacio vectorial de dimensión n.
3. $\Bbbk[x_1, \ldots, x_n] = \Bbbk[X]$ el anillo de polinomios en n variables.
4. $X^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ los monomios estándar ($\alpha \in \mathbb{N}^n$)

Definición 30. Sean $f_1, \ldots, f_s \in \Bbbk[X]$. El conjunto

$$V(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in \Bbbk^n \mid f_i(a_1, \ldots, a_n) = 0, 1 \leq i \leq s\}$$

se llama *variedad afín*.

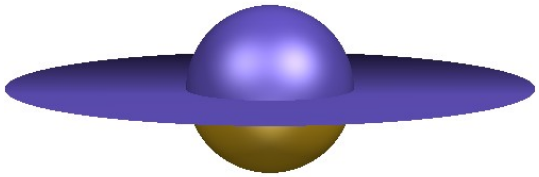
Operaciones con variedades

Proposición 31. *Si $V, W \subseteq \mathbb{k}^n$ son variedades afines entonces $V \cup W$ y $V \cap W$ también lo son.*

Supongamos que $V = V(f_1, \ldots, f_s)$ y $W = V(g_1, \ldots, g_t)$

$$V \cap W = V(f_1, \ldots, f_s, g_1, \ldots, g_t)$$
$$V \cup W = V(f_i g_j \mid 1 \leq i \leq s, 1 \leq j \leq t)$$

$z(x^2 + y^2 + z^2 - 1)$



5.2 Ideales

Definición 32. Un subconjunto $I \subseteq \mathbb{k}[X]$ es un *ideal* si satisface

1. $0 \in I$,
2. Si $f, g \in I$ entonces $f + g \in I$,
3. Si $f \in I$ y $h \in \mathbb{k}[X]$ entonces $hf \in I$.

Definición 33. Sean $f_1, \ldots, f_s \in \mathbb{k}[X]$. El conjunto

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i \mid h_1, \ldots, h_s \in \mathbb{k}[X] \right\}$$

es un ideal, el llamado *ideal generado por* $f_1, \ldots, f_s$.

Si $I = \langle f_1, \ldots, f_s \rangle$ decimos que $\{f_1, \ldots, f_s\}$ es una *base* para I.

Ideales y variedades

Proposición 34. *Si $\langle f_1, \ldots, f_s \rangle = \langle g_1, \ldots, g_t \rangle$ entonces $V(f_1, \ldots, f_s) = V(g_1, \ldots, g_t)$.*

Proposición 35. *$V(f_1, \ldots, f_s) = \varnothing$ si y sólo si $\langle f_1, \ldots, f_s \rangle = \langle 1 \rangle = \mathbb{k}[X]$*

Ideal asociado a una variedad

Definición 36. Sea V una variedad afín. El conjunto

$$I(V) = \{f \in \Bbbk[X] \mid f(a_1, \ldots, a_n) = 0 \forall (a_1, \ldots, a_n) \in V\}$$

es un ideal, llamado el *ideal de* V.

Es claro que

$$\langle f_1, \ldots, f_s \rangle \subseteq I(V(f_1, \ldots, f_s))$$

La inclusión puede ser estricta

5.3 Órdenes admisibles

Monomios y exponentes

Hemos visto que un monomio de $\Bbbk[X]$ puede escribirse como

$$X^\alpha = x_1^{\alpha_1} \ldots x_n^{\alpha_n}$$

Podemos identificar cada monomio con su exponente. La suma de exponentes se corresponde con el producto de monomios y viceversa.

Definición 37. Una relación de orden total $\preceq$ en $\mathbb{N}^n$ se dice que es un *orden admisible* si satisface:

1. $0 \preceq \alpha$ para cualquier $\alpha \in \mathbb{N}^n$.
2. Si $\alpha \preceq \beta$ y $\gamma \in \mathbb{N}^n$ entonces $\alpha + \gamma \preceq \beta + \gamma$.

Lema 38. *Un orden admisible es un buen orden, es decir, todo subconjunto no vacío tiene un primer elemento.*

Ejemplos

Definición 39 (Lexicográfico)**.** Decimos que $\alpha \preceq_{\text{lex}} \beta$ si y sólo si la primera posición no nula de $\beta - \alpha$ es positiva.

Definición 40 (Lexicográfico graduado)**.** Llamamos para cada $\alpha \in \mathbb{N}^n$ $|\alpha| = \sum_{i=1}^n \alpha_i$. Decimos que $\alpha \preceq_{\text{grlex}} \beta$ si

$$|\alpha| < |\beta|, \text{ o } |\alpha| = |\beta| \text{ y } \alpha \preceq_{\text{lex}} \beta$$

Definición 41 (Lexicográfico graduado inverso)**.** Decimos que $\alpha \prec_{\text{grevlex}} \beta$ si $|\alpha| < |\beta|$,o $|\alpha| = |\beta|$ y la última posición no nula de $\beta - \alpha$ es negativa.

Exponentes

Fijamos un orden admisible $\preceq$ en $\mathbb{N}^n$. Sea $0 \neq f = \sum_{\alpha \in \mathbb{N}^n} a_\alpha X^\alpha$

1. El *diagrama de Newton* de f es $\mathcal{N}(f) = \{\alpha \in \mathbb{N}^n \mid a_\alpha \neq 0\}$.
2. El *exponente* de f es $\exp(f) = \max_{\preceq} \mathcal{N}(f)$
3. El *coeficiente líder* es $\text{lc}(f) = a_{\exp(f)} \in \Bbbk$
4. El *monomio líder* es $\text{lm}(f) = X^{\exp(f)}$
5. El *término líder* es $\text{lt}(f) = \text{lc}(f)\,\text{lm}(f)$

Lema 42. *Sean* $f, g \in \Bbbk[X]$ *polinomios no nulos. Entonces*

1. $\exp(fg) = \exp(f) + \exp(g)$.

2. Si $f + g \neq 0$ *entonces* $\exp(f + g) \leq \max\{\exp(f), \exp(g)\}$.

Sea $F \subseteq \Bbbk[X]$. Llamamos $\exp(F) = \{\exp(f) \mid f \in F \setminus \{0\}\} \subseteq \mathbb{N}^n$. Si I es un ideal $\exp(I)$ es lo que se llama un monoideal, es decir,

$$\exp(I) + \mathbb{N}^n = \exp(I)$$

5.4 División en polinomios

Partición asociada a una lista

Sea $F = \{f_1, \dots, f_m\}$ un conjunto finito de polinomios no nulos. Definimos la partición $\mathbf{\Delta}^{[F]}$ en $\mathbb{N}^n$:

$$\begin{aligned}
&\Delta_1 = \exp(f_1) + \mathbb{N}^n \\
&\vdots \\
&\Delta_i = \big(\exp(f_i) + \mathbb{N}^n\big) \setminus \big(\Delta_1 \cup \cdots \cup \Delta_{i-1}\big) \\
&\vdots \\
&\overline{\Delta} = \mathbb{N}^n \setminus \big(\Delta_1 \cup \cdots \cup \Delta_m\big).
\end{aligned} \tag{1}$$

Observemos que $\Delta_1 \cup \cdots \cup \Delta_m = \exp(F) + \mathbb{N}^n$, el monoideal generado por $\{\exp(f_1), \dots, \exp(f_m)\}$.

Concepto de división

Sea $\preceq$ un orden admisible sobre $\mathbb{N}^n$ y $F = \{f_1, \dots, f_m\}$ un conjunto finito de polinomios no nulos. Para todo $f \in \Bbbk[X]$, existen $q_1, \dots, q_m, r \in \Bbbk[X]$ tales que

1. $f = \sum_{i=1}^m q_i f_i + r$.
2. O bien $r = 0$, o bien $\mathcal{N}(r) \subseteq \overline{\Delta}$ y $\exp(r) \preceq \exp(f)$.
3. O bien $q_i = 0$, o bien $\exp(f_i) + \mathcal{N}(q_i) \subseteq \Delta_i$ y $\exp(q_i f_i) \preceq \exp(f)$.

Estas condiciones nos dicen, en particular, que

$$\exp(f) = \max\{\exp(r), \exp(q_1 f_1), \dots, \exp(q_m f_m)\}.$$

Tenemos que $\exp(f) = \exp(r)$ o existe un único i_0 tal que $\exp(f) = \exp(q_{i_0} f_{i_0})$. El primer caso es equivalente a que $\exp(f) \in \overline{\Delta}$, y el segundo a que $\exp(f) \in \Delta_{i_0}$, es decir, $\exp(f) = \exp(r)$ si y sólo si $\exp(f) \in \overline{\Delta}$ y $\exp(q_i f_i) = \exp(f)$ si y sólo si $\exp(f) \in \Delta_i$.

Unicidad y existencia de la división

Los cocientes $q_1, \dots, q_m$ y el resto r son únicos.

Esta unicidad es consecuencia de el orden establecido en los polinomios.

La existencia es consecuencia del algoritmo de cálculo. Primeramente podemos suponer que todos los coeficientes líderes de los f_i son 1. Primeramente $q_1 = \cdots = q_m = r = 0$. Dado que los conjuntos $\Delta_1, \dots, \Delta_m, \overline{\Delta}$ constituyen una partición de $\mathbb{N}^n$, tenemos que $\exp(f) \in \Delta_{i_0}$ para un único i_0 o $\exp(f) \in \overline{\Delta}$.

- Si $\exp(f) \in \overline{\Delta}$, llamamos $r \leftarrow r + \operatorname{lm}(f)$ y $f \leftarrow f - \operatorname{lm}(f)$.
- Si $\exp(f) \in \Delta_{i_0}$, tenemos que $\exp(f) = \exp(f_{i_0}) + \beta$. Las asignaciones son entonces $q_{i_0} \leftarrow q_{i_0} + \operatorname{lc}(f)X^\beta$ y $f \leftarrow f - \operatorname{lc}(f)X^\beta f_{i_0}$

En ambos casos el nuevo valor de f tiene un exponente menor, por lo que este procedimiento recursivo termina.

Denotamos el resto: $r = \operatorname{rem}(f, F)$

5.5 Bases de Groebner

Concepto de base de Groebner

Definición 43. Un subconjunto G de un ideal I es una base de Groebner si $\exp(I) = \exp(G) + \mathbb{N}^n$, es decir, si $\exp(G)$ es un conjunto de generadores de $\exp(I)$.

Proposición 44. *Si $G = \{g_1, \ldots, g_t\}$ es una base de Groebner de I entonces $I = \langle g_1, \ldots, g_t \rangle$, es decir, G es una base para I.*

Podemos comprobarlo mediante la división definida anteriormente.

Problema de la pertenencia

Proposición 45 (Pertenencia). *Sea $G = \{g_1, \ldots, g_t\}$ una base de Groebner de I. Para cualquier polinomio $f \in \Bbbk[X]$, $f \in I$ si y sólo si el resto obtenido al dividir f entre G es cero.*

S-polinomios y caracterización

Sean $\alpha, \beta \in \mathbb{N}^n$. Llamamos $\operatorname{lcm}(\alpha, \beta) = \gamma$ donde $\gamma_i = \operatorname{máx}(\alpha_i, \beta_i)$ para cualquier $i = 1, \ldots, n$.

Definición 46. Sean $f, g \in \Bbbk[x_1, \ldots, x_n]$, $\gamma = \operatorname{lcm}(\exp(f), \exp(g))$. El S-polinomio de f y g es

$$S(f, g) = \frac{X^\gamma}{\operatorname{lt}(f)} f - \frac{X^\gamma}{\operatorname{lt}(g)} g$$

Teorema 47. *G es una base de Groebner (para el ideal que genera) si y sólo si para cualesquiera $i \neq j$, el resto obtenido al dividir $S(g_i, g_j)$ entre G es cero.*

Cálculo de Bases de Groebner

procedure BUCHBERGER(F)
Require: Entrada: $F = \{f_1, \ldots, f_s\}$ un conjunto de generadores de un ideal I
Require: Salida: Una base de Groebner para el ideal $\langle F \rangle$.
  $G \leftarrow F$
  repeat
    $G' \leftarrow G$
    for each $p, q \in G'$ **do**
      $r \leftarrow \operatorname{rem}(S(p, q), G)$
      if $r \neq 0$ **then**
        $G \leftarrow G \cup \{r\}$
      end if
    end for
  until $G = G'$
  return G
end procedure

Unicidad en Bases de Groebner

Definición 48. Una base de Groebner G se dice minimal si para cualquier $g \in G$, $\exp(g) \notin \exp(G \setminus \{g\}) + \mathbb{N}^n$

El procedimiento para encontrar una base de Groebner minimal es inmediato, quitar aquellos elementos cuyo exponente pertenece al monoideal generado por los exponentes de los demás.

Definición 49. Una base de Groebner minimal se dice reducida si para cualquier $g \in G$, $g = \operatorname{rem}(g, G \setminus \{g\})$

Una vez que tenemos una base de Goebner minimal reemplazamos cada elemento por el resto obtenido al dividirlo entre los demás.

5.6 Aplicación: Coloreo de Grafos

Qué es un grafo y una coloración

Un *grafo (no dirigido)* G está definido por dos conjuntos $V(G)$ y $E(G)$ junto con una aplicación $\gamma_G : E(G) \to \{\{u, v\} \mid u, v \in V(G)\}$. $V(G)$ son los *vértices* del grafo, y $E(G)$ son los *lados* o aristas del grafo.

Dos vértices $u \neq v \in V(G)$ se dicen adyacentes si existe $e \in E(G)$ tal que $\gamma_G(e) = \{u, v\}$

Una *coloración* en un grafo G es una aplicación $c : V(G) \to C$ tal que para cada $e \in E(G)$, si $\gamma_G(e) = \{u, v\}$ con $u \neq v$ entonces $c(u) \neq c(v)$. Es decir, una coloración es una asignación de valores a los vértices de G de tal manera vértices adyacentes tienen distinto color.

Planteamiento del problema

Dado un grafo G, un conjunto de colores C y una asignación de colores a algunos vértices, ¿existe una coloración que complete la asignación anterior?

En términos matemáticos, dados $v_1, \ldots, v_t \in V(G)$ distintos, y $c_1, \ldots, c_t \in C$, ¿existe una coloración $c : V(G) \to C$ tal que $c(v_i) = c_i$ para todo $1 \leq i \leq t$?

Traducción a polinomios

- Sea $\mathbb{K}$ un cuerpo numérico con al menos tantos elementos como colores hay en C. Podemos asociar a cada color un número, por lo que podemos suponer que $C = \{0, 1, \ldots, n-1\}$.
- Asociamos a cada vértice una variable x_i, así si $V = \{v_1, \ldots, v_p\}$ vamos a trabajar con el anillo de polinomios $\mathbb{K}[x_1, \ldots, x_p]$.
- Para cada variable x_i consideramos los polinomios $F_i = x_i(x_i - 1) \cdots (x_i - (n-1))$.
- Para cada pareja de variables x_i, x_j se puede demostrar que $F_i - F_j = (x_i - x_j)G_{ij}$ donde G_{ij} es un polinomio tal que $x_i - x_j \nmid G_{ij}$.

Colores y adyacencias.

- Indicamos que los posibles colores del vértice v_i son los presentes en C mediante la ecuación $F_i = 0$.
- Supongamos que v_i y v_j son adyacentes. Dado que $F_i = F_j = 0$, necesariamente $(x_i - x_j)G_{ij} = F_i - F_j = 0$, y como queremos que $x_i \neq x_j$ debemos imponer la ecuación $G_{ij} = 0$.
- Finalmente, si queremos que el vértice v_i tenga el color etiquetado como k debemos añadir la ecuación $x_i - k = 0$.

Ecuaciones

Tenemos por tanto las ecuaciones

$$\begin{cases} F_i = 0 & \text{para cada } 1 \leq i \leq p \\ G_{ij} = 0 & \text{para cada } i < j \text{ tales que } v_i \text{ y } v_j \text{ son vértices adyacentes} \\ x_i - k = 0 & \text{para cada asignación inicial del color } k \text{ al vértice } v_i. \end{cases}$$

Es decir, la variedad que queremos resolver viene definida por el ideal

$$\langle F_i (1 \leq i \leq p), G_{ij} (i < j, v_i \frown v_j), x_i - k (v_i \to k) \rangle$$

Sudokus

Un puzzle con bastante aficionados en los últimos años.

	2			3		9		7
	1							
4		7				2		8
		5	2				9	
			1	8		7		
	4				3			
				6			7	1
	7							
9		3		2		6		5

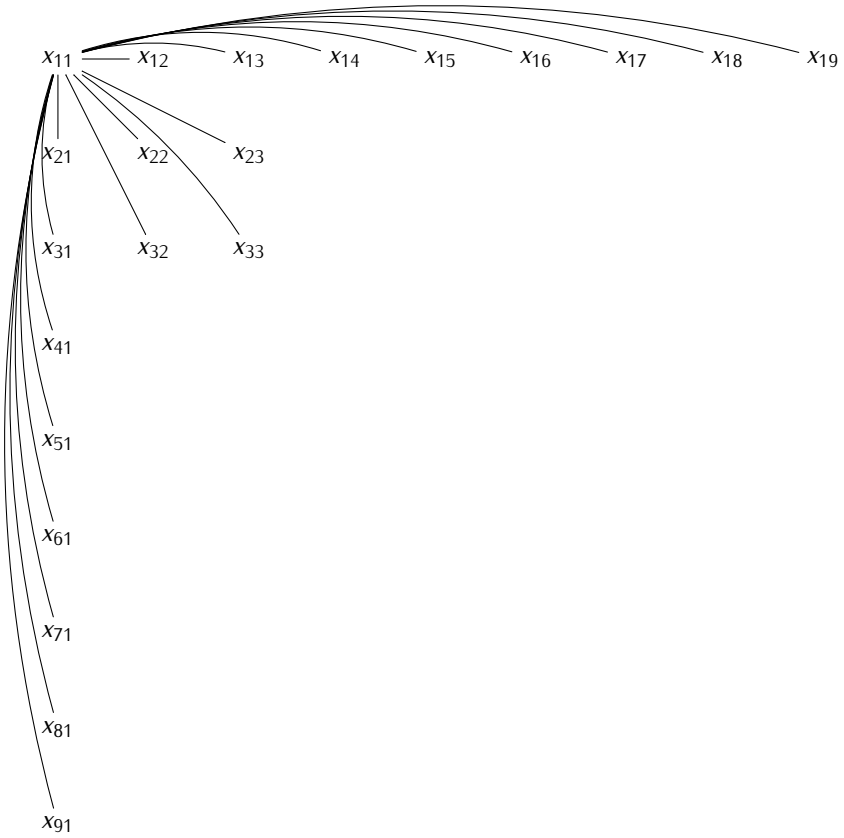
Sudokus: Vértices

Podemos representarlo con un grafo con 81 vértices

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}	x_{39}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}	x_{49}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	x_{57}	x_{58}	x_{59}
x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	x_{67}	x_{68}	x_{69}
x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	x_{77}	x_{78}	x_{79}
x_{81}	x_{82}	x_{83}	x_{84}	x_{85}	x_{86}	x_{87}	x_{88}	x_{89}
x_{91}	x_{92}	x_{93}	x_{94}	x_{95}	x_{96}	x_{97}	x_{98}	x_{99}

Sudoku: Adyacencias

Las adyacencias de x_{11} son



Más juegos

- Killer Sudoku.
- Sudoku par/impar.
- Sudoku mayor que.
- Sudoku geométrico.
- Buscaminas.
- Cuadrados mágicos.
- Cuadrados latinos.

5.7 Aplicación: GPS

Sistema de Posicionamiento Global (GPS)

- 24 satélites (y 6 de refuerzo) en órbita estable, conocida y transmitida.
- Todos *perfectamente* sincronizados con relojes atómicos.
- Cada uno de ellos emite una señal pseudoaleatoria única junto con otra información referente a la posición.
- El receptor GPS mide el momento en que recibe las señales de los satélites.
- Como el receptor sabe en qué instante se ha enviado la señal, el retardo en el tiempo que tarda la señal en llegar multiplicado por la velocidad de la luz nos da la distancia *exacta* a la que están los satélites.

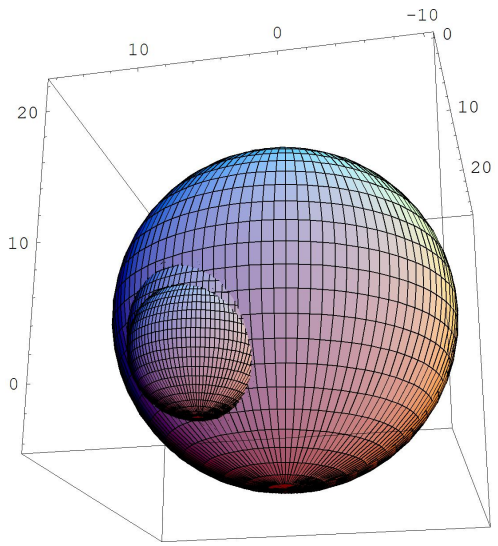
Ecuaciones para un satélite con receptor sincronizado

Si (x_i, y_i, z_i) es la posición del satélite, t_i el tiempo que tarda en llegar la señal y (x, y, z) la posición del receptor, entonces

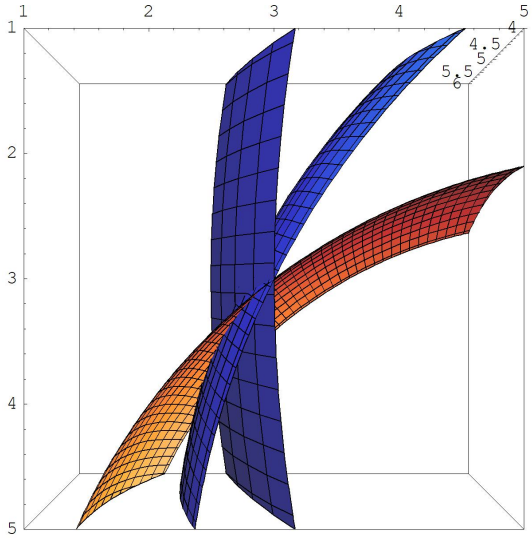
$$(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = (ct_i)^2$$

donde c es la velocidad de la luz. Si tomamos (x, y, z) como variable tenemos la ecuación de una esfera centrada en (x_i, y_i, z_i) y de radio ct_i.

Tres esferas



Tres esferas: Intersección

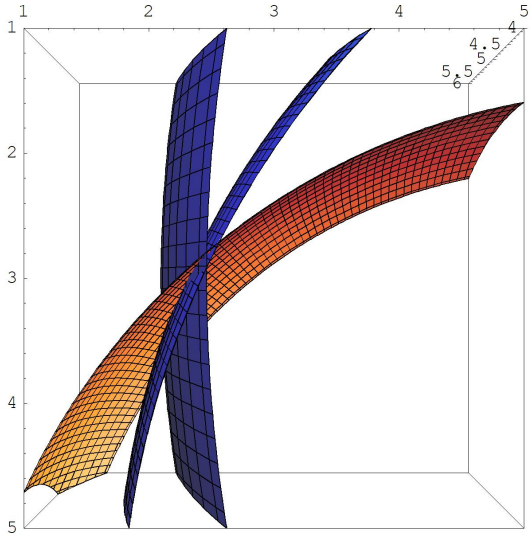


Demora del receptor

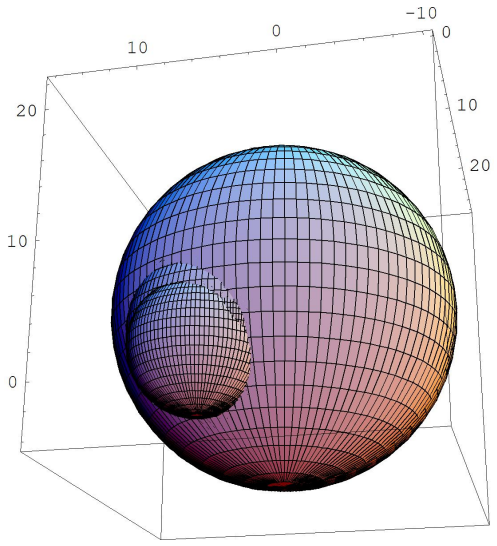
La sincronización de los relojes (atómicos) de los satélites es extrema, pero el receptor no dispone de un reloj atómico y por tanto su sincronización no es perfecta. Eso quiere decir que hay un retraso o adelanto d del reloj del receptor con respecto de los relojes de los satélites. Si t_i es el tiempo medido por el receptor, la ecuación se convierte en

$$(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = (c(t_i + d))^2$$

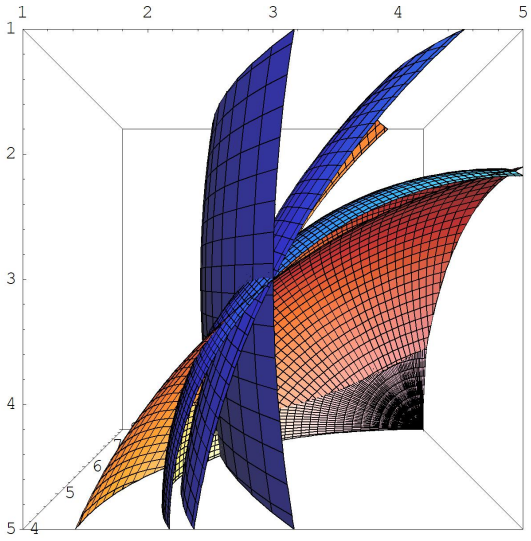
Tres esferas: Desplazamiento



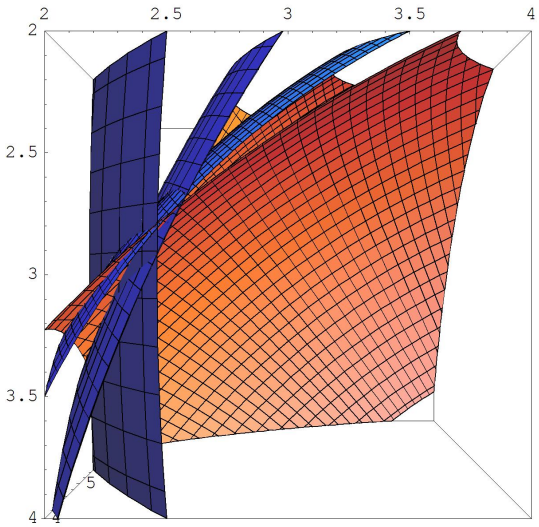
Cuatro esferas



Cuatro esferas: Intersección



Cuatro esferas: Desplazamiento



Resincronización

Con al menos cuatro satélites podemos considerar d como variable, por tanto tenemos averiguar la variedad asociada al ideal

$$\langle (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 - (c(t_i + d))^2 \mid 1 \leq i \leq 4 \rangle$$

En una primera resolución calculamos el valor de d y sincronizamos el reloj del receptor con los satélites. A partir de ese momento podemos trabajar con ideales de la forma

$$\langle (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 - (ct_i)^2 \mid 1 \leq i \leq 3 \rangle$$