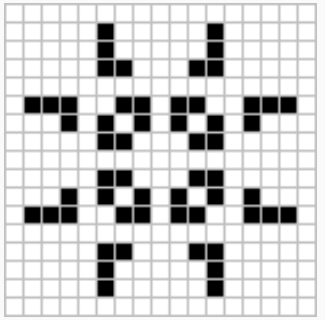


Algebra Computazionale

Minicorso Matematica - 2022

Luca Amata

30 maggio 2022

Dipartimento di Scienze Matematiche e Informatiche,
Scienze Fisiche e Scienze della Terra,
Università degli Studi di Messina

Sistemi di computer algebra

Introduzione

- I termini *algebra* e *algoritmo* vengono fatti risalire al matematico persiano **al-Khwarizmi** [2], il quale con *al-gabr* e *al-muqabalah* descriveva, rispettivamente, le trasformazioni simboliche e le riduzioni a termini comuni utilizzate per risolvere equazioni algebriche.

- Fino alla fine del XIX secolo, la manipolazione simbolica delle equazioni ha mantenuto una fondamentale importanza negli studi algebrici.

- Con l'avvento dell'algebra astratta, il cambio di paradigma ha dirottato l'interesse comune verso i sistemi formali assiomatici, oscurando il calcolo simbolico.

- Negli ultimi decenni, gli aspetti algoritmici dell'algebra sono nuovamente tornati in auge grazie, soprattutto, al rapidissimo sviluppo dei sistemi informatici e delle attività ad essi collegati.

Introduzione

- Per *sistema di computer algebra* (CAS) si intende un software che permette di manipolare automaticamente espressioni matematiche rappresentate in forma *simbolica*.

- Nasce così una nuova disciplina: la *computer algebra* (o *calcolo simbolico*), dedita allo studio di metodi per la risoluzione automatica di problemi espressi da una formulazione "matematica" tramite l'implementazione di opportuni algoritmi.

- Le applicazioni dell'algebra computazionale sono rivolte prevalentemente allo studio di proprietà di specifiche strutture e spesso anche alla produzione di esempi notevoli, esotici o controesempi.

- Si può affermare che in questi anni si stia vivendo una nuova età aurea dei metodi algoritmici nell'algebra, non soltanto come strumento accessorio, ma soprattutto come metodologia applicabile all'analisi di strutture matematiche.

Rappresentazione dei dati

- L'utilizzo di CAS è reso possibile dall'esistenza di *rappresentazioni finite esatte* di oggetti e strutture matematiche, e dalla possibilità di manipolarle in tempi finiti.

- Gli oggetti matematici più "semplici" sono i numeri ed è quindi basilare partire dalla loro rappresentazione. In generale si parla di *dati*.

- I dati sono informazioni memorizzate in frammenti chiamati **parole** (words). Attualmente la maggior parte degli elaboratori utilizzano parole a 32 o 64 bit.

- Consideriamo di lavorare su un processore (con parole) a 64 bit. Questo significa, in pratica, che ogni parola contiene un intero a **singola precisione** compreso fra 0 e $2^{64} - 1$.

- Come è possibile rappresentare numeri interi al di fuori del range della singola parola di 64 bit?

Rappresentazione dei dati

- Gli interi che superano la dimensione di una parola sono detti interi a **precisione multipla** e possono essere rappresentati tramite *vettori* di interi a 64 bit, cioè come sequenze finite di parole a precisione singola.
- Consideriamo la seguente decomposizione dell'intero n:

$$a = (-1)^s \sum_{i=0}^{k} a_i \cdot 2^{64i},$$

  dove $s \in \{0, 1\}$ (il segno), $0 \leq k + 1 \leq 2^{63}$ (la molteplicità di precisione) e $0 \leq a_i \leq 2^{64} - 1$ (le cifre in base 2^{64} di a).
- La **rappresentazione standard** di n è data dal seguente vettore:

$$[s \cdot 2^{63} + k + 1, a_0, a_1, \ldots, a_k],$$

  tale rappresentazione è unica imponendo che $a_k \neq 0$ (quando $a \neq 0$).
- Con tale notazione è possibile rappresentare interi compresi tra

$$-2^{64 \cdot 2^{63}} + 1 \quad \text{e} \quad 2^{64 \cdot 2^{63}} - 1.$$

Rappresentazione dei dati

- Si definisce **lunghezza** di un intero a precisione multipla n:

$$\lambda(a) = \lfloor \log_{2^{64}} |a| \rfloor + 1 = \left\lfloor \frac{\log_2 |a|}{64} \right\rfloor + 1,$$

  quindi $k + 1 = \lambda(a)$ (cardinalità del vettore).

Esempi

1. La rappresentazione standard di $10^{30} + 1$ è:

$$[2, 5076944270305263617, 54210108624],$$

   con $k = 1$ e $\lambda(10^{30} + 1) = 2$;
2. La rappresentazione di -1 è $[2^{63} + 1, 1]$, con $k = 0$ e $\lambda(-1) = 1$.

Operazioni a precisione multipla

Siano dati due interi a precisione multipla in notazione standard:

$$a = [s_a \cdot 2^{63} + k_a + 1, a_0, a_1, \ldots, a_{k_a}] \quad \text{e} \quad b = [s_b \cdot 2^{63} + k_b + 1, b_0, b_1, \ldots, b_{k_b}].$$

Se $s_a = s_b = s$, la **somma** $c = a + b$ è un intero a precisione multipla con $k = \max\{k_a, k_b\} + 1$ dato da:

$$c = [s \cdot 2^{63} + k + 1, a_0 + b_0 - r_0 \cdot 2^{64}, a_1 + b_1 + r_0 - r_1 \cdot 2^{64}, \ldots, a_k + b_k + r_k],$$

dove r_i è il *carry flag* aggiornato all'indice i (il CF è un registro della CPU che assume valore 1 quando la somma fra due parole singole ha un riporto, cioè supera la parola). Similmente si procede per la **differenza** (considerando il concetto di complemento).

Algoritmo

E il **prodotto** a precisione multipla? Osserviamo che il prodotto a singola precisione dà un risultato in precisione doppia: $a \cdot b = c \cdot 2^{64} + d$.
È possibile procedere similmente al prodotto fra polinomi.

Gestione dei polinomi

- Altri oggetti alla base della computer algebra sono i polinomi, usualmente definiti su anelli commutativi. Ricordiamo che un polinomio $p \in R[x]$ è una sequenza finita, $[a_0, a_1, \ldots, a_n]$ con $a_n \neq 0$, di elementi di R. In tal caso diciamo che $n = \deg(p)$ è il grado del polinomio e $a_n = LC(p)$ è il suo coefficiente direttore. Scriveremo:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i.$$

- L'analogia tra la rappresentazione dei polinomi e quella degli interi a precisione multipla è riscontrabile anche negli algoritmi per effettuare le operazioni basilari ed anche alcuni più avanzati.
- Per casi particolari, riguardanti soprattutto la scelta dell'insieme numerico di supporto, esistono algoritmi ottimizzati per eseguire le operazioni fra interi multiprecisione o polinomi.

Divisione fra polinomi

- In un dominio euclideo R, dati a e b, $b \neq 0$, è possibile trovare la scrittura $a = q \cdot b + r$ con $q, r \in R$ e $r = 0$ o $v(r) < v(b)$.
- Sappiamo che se F è un campo allora l'anello dei polinomi $F[x]$ è un dominio euclideo, quindi vale la precedente scrittura (la valutazione associa ad ogni polinomio il proprio grado).

Osservazione

$\mathbb{Z}[x]$ non è un dominio euclideo (non essendo a ideali principali). Ad esempio non è possibile dividere x^2 per $2x + 1$. Infatti in $\mathbb{Z}[x]$ si può procedere soltanto se $LC(b) = \pm 1$ o con quella che viene chiamata **pseudodivisione**.

Algoritmo di divisione con resto

$$
\begin{array}{llll|ll}
x^3 & +2x^2 & -9x & -4 & x^2 & -1 \\
\cline{5-6}
-x^3 & & +x & & x & +2 \\
\cline{1-4}
 & 2x^2 & -8x & -4 & & \\
 & -2x^2 & & +2 & & \\
\cline{2-4}
 & & -8x & -2 & & \\
\end{array}
$$

Algoritmo

Algoritmi classici

Algoritmo di Euclide

- Abbiamo già osservato che in un dominio euclideo R è sempre possibile effettuare la divisione con resto. Inoltre esistono sempre massimo comune divisore e minimo comune multiplo.
- Dati $a, b, d \in R$, si dice che d è il **massimo comune divisore** (a meno di associati) di a e b, $d = \mathrm{MCD}(a, b)$, se valgono le seguenti:

$$(i)\ d|a \text{ e } d|b; \quad (ii)\ \forall d' \in R \text{ tale che } d'|a \text{ e } d'|b \text{ allora } d'|d.$$

- Il **minimo comune multiplo** di a e b può essere definito come:

$$\mathrm{mcm}(a, b) = \frac{a \cdot b}{\mathrm{MCD}(a, b)}.$$

Calcolo del massimo comune divisore

$$\begin{aligned} \mathbf{126} &= 3 \cdot \mathbf{35} + 21, \\ 35 &= 1 \cdot 21 + 14, \\ 21 &= 1 \cdot 14 + \mathbf{7}, \\ 14 &= 2 \cdot 7 + 0. \end{aligned}$$

Algoritmo

Diagramma

Algoritmo di Euclide esteso

- L'algoritmo di Euclide può essere "potenziato" in modo da ottenere ulteriori informazioni. Tale variante permette di calcolare algoritmicamente i coefficienti dell'**identità di Bézout**.
- Siano $a, b \in R$ due qualsiasi elementi di un dominio euclideo e sia $d = \text{MCD}(a, b)$. Allora esistono $x, y \in R$ tali che d si scrive come

$$d = x \cdot a + y \cdot b.$$

Identità di Bézout di $7 = \text{MCD}(126, 35)$

$$\begin{aligned} \mathbf{7} &= 21 - 1 \cdot 14 \\ &= 21 - 1 \cdot (35 - 1 \cdot 21) \\ &= 2 \cdot 21 - 35 \\ &= 2 \cdot (126 - 3 \cdot 35) - 35 \\ &= 2 \cdot \mathbf{126} - 7 \cdot \mathbf{35}. \end{aligned}$$

Algoritmo

Algoritmo di Euclide esteso

- Analizzando passo dopo passo l'algoritmo, sempre guardando all'esempio $7 = \text{MCD}(126, 35)$, otteniamo i seguenti dati:

i:	0,	1,	2,	3,	4,	5
q:	{ 3,	1,	1,	2		}
r:	{$\underline{126}$,	$\underline{35}$,	21,	14,	**7**,	0}
s:	{ 1,	0,	1,	-1,	**2**,	-5}
t:	{ 0,	1,	-3,	4,	-**7**,	18}

- L'algoritmo termina non appena si ottiene resto nullo. Nell'esempio, tale condizione si verifica quando $i = 5$, quindi i valori restituiti dall'algoritmo in questo caso sono $r_{i-1} = r_4 = 7$, $s_4 = 2$ e $t_4 = -7$ che rappresentano rispettivamente d, x e y nella scrittura $d = x \cdot a + y \cdot b$.
- Si ottiene quindi l'identità: $\mathbf{7} = \mathbf{2} \cdot \underline{126} + (-\mathbf{7}) \cdot \underline{35}$.

Inverso "modulare"

- Siano $a, b \in R$ elementi di un dominio euclideo e sia $S = R/aR$. Allora $\overline{b} \in S$ è invertibile se e solo se $\mathrm{MCD}(a, b) = 1$, inoltre il suo inverso in S è $\overline{y}$ (calcolato tramite l'algoritmo esteso: $\overline{1} = \overline{y} \cdot \overline{b}$). Analogamente, se $T = R/bR$ allora $\overline{a} \in T$ è invertibile se e solo se $\mathrm{MCD}(a, b) = 1$ e il suo inverso in T è $\overline{x}$ (infatti $\overline{1} = \overline{x} \cdot \overline{a}$).
- Vi sono importanti applicazioni in campo crittografico, soprattutto nell'algoritmo di cifratura **RSA**. Dati $n = pq$ e la chiave pubblica (e, n), con $\mathrm{MCD}(e, \varphi(n)) = 1$, la chiave privata è (d, n) con $de \equiv 1$ modulo $\varphi(n)$. Il messaggio $c = m^e \bmod n$ si decifra: $c^d = m \bmod n$.

Esempio

```
i1 :  a=2^13-1
o1 =  8191
i2 :  b=2^8+1
o2 =  257
i3 :  ib=esteso(a,b)
o3 =  {1, 109, -3474}
o3 :  List
```

```
i4 :  S=ZZ/a;
i5 :  (b*ib_2)_S
o5 =  1
o5 :  S
i6 :  T=ZZ/b;
i7 :  (a*ib_1)_T
o7 =  1
```

Frazioni continue

In termini formali una **espansione in frazione continua** di un numero reale r è una rappresentazione che utilizza una sequenza di *quozienti successivi*. Spesso è usata una rappresentazione compatta: $r = [a_0; a_1, a_2, a_3, \ldots]$.

$$r = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}$$

$\varphi = [1; 1, 1, 1, 1, 1, 1, 1, 1, \ldots]$; $e = [2; 1, 2, 1, 1, 4, 1, 1, 6, \ldots]$; $\pi = [3; 7, 15, 1, 292, 1, 1, 1, 2, \ldots]$

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}}};$$

$$e = 2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{2}{3 + \cfrac{3}{\cdots}}}}$$

$$\pi = \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{\cdots}}}}$$

Esempio

```
i1 :  a=126;
i2 :  b=35;
i3 :  q=reverse esteso(a,b,Frac=>true)
o3 =  {2, 1, 1, 3}
o3 :  List
i4 :  r=q_0;
```

```
i5 :  qq=take(q,1,#q-1)
o5 =  {1, 1, 3}
o5 :  List
i6 :  for i in qq do r=i+1/r
i7 :  r==a/b
o7 =  true
```

... e i polinomi?

Per i polinomi di $F[x]$, F campo, sappiamo che gli algoritmi di Euclide sono applicabili. Inoltre, disponendo di operazioni opportune (quoziente e resto) tali algoritmi rimangono inalterati rispetto a quelli utilizzati in $\mathbb{Z}$.

Osservazione

È interessante osservare che in $\mathbb{Z}$ le unità sono ± 1, quindi esiste un unico massimo comune divisore positivo (l'unico associato è negativo).
In F^* ogni elemento è un'unità, quindi un polinomio ha infiniti associati. Per ricondurci all'unicità del massimo comune divisore si può considerare il polinomio monico associato al polinomio restituito dall'algoritmo euclideo.

Esempio

```
i1 :  S=QQ[x];
i2 :  a=4*(x^2+1)*(x^2-2*x+4)
o2 =  4x^4-8x^3+20x^2-8x+16
o2 :  S
i3 :  b=(x^2+1)*(x^3+x^2-3)
o3 =  x^5+x^4+x^3-2x^2-3
o3 :  S
i4 :  ee=esteso(a,b)
o4 =  {181x^2+181, 1/2x^2+17/4x+37/4, -2x-11}
o4 :  List
i5 :  ee=ee/(i->i/leadCoefficient ee_0)
o5 =  {x^2+1, 1/362x^2+17/724x+37/724,
       -2/181x-11/181}
o5 :  List
i6 :  ee_0==ee_1*a+ee_2*b
o6 =  true
```

Teorema cinese del resto

- La formulazione originale del problema (Sun-Tsu, 1247) afferma che dati k elementi $m_1, \ldots, m_k$ di un dominio euclideo R a due a due coprimi ($\mathrm{MCD}(m_i, m_j) = 1$, $i \neq j$), il seguente sistema di congruenze

$$x \equiv a_i \bmod m_i, \text{ per } i = 1, \ldots, k$$

  ammette soluzione comunque si scelgano $a_i \in R$. Inoltre tutte le soluzioni del sistema sono congruenti modulo $m = m_1 \cdots m_k$.

- Una formulazione moderna afferma che, sotto le ipotesi precedenti, valgono i seguenti isomorfismi:
  - di anelli $R/(m) \cong R/(m_1) \times \cdots \times R/(m_k)$;
  - di gruppi $(R/(m))^* \cong (R/(m_1))^* \times \cdots \times (R/(m_k))^*$.

Esempio

$$\begin{cases} x \equiv 2 & \bmod 5 \\ x \equiv 3 & \bmod 7 \\ x \equiv 10 & \bmod 11 \end{cases}$$

$$\text{Anelli} \quad \mathbb{Z}_{385} \cong \mathbb{Z}_5 \times \mathbb{Z}_7 \times \mathbb{Z}_{11}$$

$$\text{Gruppi} \quad G_{240} \cong C_4 \times C_6 \times C_{10}$$

$$\varphi(385) = 240$$

Teorema cinese del resto

- Sia $m = 5 \cdot 7 \cdot 11 = 385$ e ad ogni m_i associamo un $\overline{m_i}$ come segue:

$$\overline{m_1} = \frac{m}{m_1} = \frac{385}{5} = 77, \quad \overline{m_2} = \frac{385}{7} = 55, \quad \overline{m_3} = \frac{385}{11} = 35.$$

- Cerchiamo ora l'inverso moltiplicativo s_i per ogni $\overline{m_i}$ modulo m_i:
  - $\overline{m_1} = 77 \equiv 2 \bmod 5$, da cui $s_1 = 3$;
  - $\overline{m_2} = 55 \equiv 6 \bmod 7$, da cui $s_2 = 6$;
  - $\overline{m_3} = 35 \equiv 2 \bmod 11$, da cui $s_3 = 6$;
- Si considerano i prodotti $b_i = s_i \cdot a_i \bmod m_i$:

$$b_1 = s_1 \cdot a_1 \bmod m_1 = 3 \cdot 2 \bmod 5 = 1, \quad b_2 = 4, \quad b_3 = 5.$$

- La soluzione x del sistema è data dalla somma:

$$x = \sum_{i=1}^{k} b_i \cdot \overline{m_i} = 1 \cdot 77 + 4 \cdot 55 + 5 \cdot 35 = 472 \equiv 87 \bmod 385.$$

Algoritmo

Polinomi su campi finiti

Struttura dei campi finiti

- Sia F campo e P il suo **sottocampo fondamentale**
  - se $char(F) = p$, primo, allora $P \cong \mathbb{Z}_p$;
  - se $char(F) = 0$ allora $P \cong \mathbb{Q}$.

- Se F campo finito allora
  - $char(F) = p$, p primo;
  - $|F| = p^n = q$, $n \in \mathbb{N}^+$.

- Dati comunque un primo p e un intero positivo n
  - esiste un campo con $q = p^n$ elementi;
  - è unico a meno di isomorfismi.

  Tale campo viene identificato dal simbolo $\mathbb{F}_{\mathbf{q}}$.

Costruzione di $\mathbb{F}_9$

- Come **campo di spezzamento** del polinomio $x^q - x$:
  - Sia $x^9 - x \in \mathbb{Z}_3[x]$, scomponendolo in fattori irriducibili si ha:

$$x^9 - x = x(x-1)(x+1)(x^2+1)(x^2+x-1)(x^2-x-1).$$

  - Si considerano le nove radici dei fattori nel campo di spezzamento

$$\mathbb{F}_9 = \{0, 1, 2, \alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2\}.$$

- Come **quoziente** dell'anello dei polinomi $\mathbb{Z}_3[x]$:
  - Si quozienta l'anello con un polinomio irriducibile di grado 2

$$\mathbb{Z}_3[x]/(x^2+x-1) = \{a + bx \ : \ a, b \in \mathbb{Z}_3, \ x^2 = -x + 1\}.$$

  - Gli elementi hanno la seguente rappresentazione:

$$\{0, 1, 2, x, 1+x, 2+x, 2x, 1+2x, 2+2x\}.$$

  - Considerati β, α tali che $\beta^2 + \beta - 1 = 0$, $\alpha^2 + 1 = 0$, vale che

$$\mathbb{Z}_3(\beta) = \mathbb{Z}_3[x]/(x^2+x-1) \cong \mathbb{Z}_3[x]/(x^2+1) = \mathbb{Z}_3(\alpha)$$

    con $\varphi : \ \mathbb{Z}_3(\beta) \to \mathbb{Z}_3(\alpha)$ tale che $\beta \mapsto \alpha + 1$.

Costruzione di $\mathbb{F}_9$

Con $g_1 = 0, g_2 = x, g_3 = 2x, g_4 = 1, g_5 = 1 + x, g_6 = 1 + 2x, g_7 = 2, g_8 = 2 + x, g_9 = 2 + 2x$.

(GF[9], +) x + y

x \ y	g1	g2	g3	g4	g5	g6	g7	g8	g9
g1	g1	g2	g3	g4	g5	g6	g7	g8	g9
g2	g2	g3	g1	g5	g6	g4	g8	g9	g7
g3	g3	g1	g2	g6	g4	g5	g9	g7	g8
g4	g4	g5	g6	g7	g8	g9	g1	g2	g3
g5	g5	g6	g4	g8	g9	g7	g2	g3	g1
g6	g6	g4	g5	g9	g7	g8	g3	g1	g2
g7	g7	g8	g9	g1	g2	g3	g4	g5	g6
g8	g8	g9	g7	g2	g3	g1	g5	g6	g4
g9	g9	g7	g8	g3	g1	g2	g6	g4	g5

(a) Somma

(GF[9], x) x * y

y \ x	g1	g2	g3	g4	g5	g6	g7	g8	g9
g1	g1	g1	g1	g1	g1	g1	g1	g1	g1
g2	g1	g6	g8	g2	g4	g9	g3	g5	g7
g3	g1	g8	g6	g3	g7	g5	g2	g9	g4
g4	g1	g2	g3	g4	g5	g6	g7	g8	g9
g5	g1	g4	g7	g5	g8	g2	g9	g3	g6
g6	g1	g9	g5	g6	g2	g7	g8	g4	g3
g7	g1	g3	g2	g7	g9	g8	g4	g6	g5
g8	g1	g5	g9	g8	g3	g4	g6	g7	g2
g9	g1	g7	g4	g9	g6	g3	g5	g2	g8

(b) Prodotto

Figura 1: Tavole delle operazioni di $\mathbb{F}_9$.

Automorfismi e gruppo moltiplicativo

- Sia F un campo di caratteristica p. La mappa $\Phi : F \to F$ definita da $a \mapsto a^p$ è detta **omomorfismo di Frobenius**.
  - Φ è sempre iniettivo;
  - Se F è finito allora Φ è un automorfismo, $F = F^p$;
  - Se $\mathbb{F}_q$, $q = p^n$, si ha $\Phi^r : a \mapsto a^{p^r}$, $r \geq 1$.

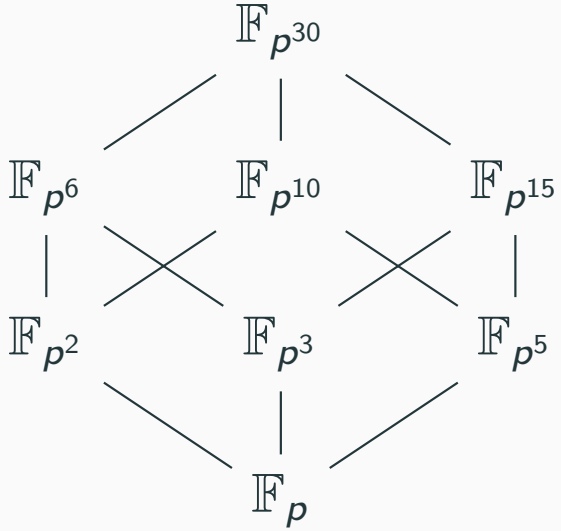
- Il **gruppo moltiplicativo** di un campo finito $\mathbb{F}$ è ciclico.
  - Un elemento u che lo genera è detto **elemento primitivo**;
  - Se $char(F) = p$ allora $F = \mathbb{Z}_p(u)$;

Esempi

- In $\mathbb{F}_9 = \mathbb{Z}_3[x]/(x^2 + 1)$ la classe $1 + x$ è un elemento primitivo.
- In generale, il gruppo moltiplicativo di un campo non è ciclico. In $(\mathbb{Q}^*, \cdot)$ si ha $o(-1) = 2$, ma $\mathbb{Z}$ non possiede tale elemento.

Sottocampi

- Per classificare i **sottocampi** di un campo finito si trovano i divisori del polinomio che lo identifica.
- Se $m \mid n$ allora $x^{p^m} - x \mid x^{p^n} - x$.
  - Ad esempio $x(x+1)(x-1) = x^3 - x \mid x^9 - x$.
- K è sottocampo di Fq, $q = p^n$, se e solo se $|K| = p^m$ con $m \mid n$.
  - Il campo $\mathbb{F}_{16}$ non ha sottocampi di cardinalità 8.
- I sottocampi di $\mathbb{F}_{p^{30}}$, p primo, rispettano la seguente struttura:



Polinomi irriducibili

- È possibile individuare alcuni **polinomi irriducibili** su un campo finito tramite il polinomio che lo determina. Un polinomio si dice irriducibile se è non nullo, non invertibile e non può essere scritto come prodotto di elementi non invertibili.
- In $\mathbb{F}_p[x]$ si ha che $x^{p^n} - x = \prod p(x)$ al variare di tutti i polinomi monici $p(x)$ irriducibili su F_p di grado m tale che $m \mid n$.
  - $x^9 - x \in \mathbb{Z}_3[x]$ si decompone in $\mathbb{Z}_3[x]$ come:

$$\begin{aligned} x^9 - x =& x(x^8 - 1) = x(x^4 - 1)(x^4 + 1) \\ =& x(x^2 - 1)(x^2 + 1)(x^2 + x - 1)(x^2 - x - 1) \\ =& x(x - 1)(x + 1)(x^2 + 1)(x^2 + x - 1)(x^2 - x - 1). \end{aligned}$$

Fattorizzazione di polinomi su campi finiti

- Fattorizzare un polinomio significa scriverlo come prodotto di polinomi irriducibili. Tale fattorizzazione esiste ed è unica (a meno di unità e dell'ordine) sotto opportune condizioni.
- Se F è un campo allora $F[x]$ è un dominio euclideo e quindi anche un **dominio a fattorizzazione unica** (UFD). Più in generale, se A è un UFD allora lo è anche $A[x]$. Quindi $F[x_1, \ldots, x_n]$ è un UFD.
- Un polinomio $f \in F[x]$ di grado n nel suo campo di spezzamento:

$$f(x) = \alpha(x-a_1)(x-a_2)\cdots(x-a_n) = \alpha(x-a_{i_1})^{t_1}(x-a_{i_2})^{t_2}\ldots(x-a_{i_r})^{t_r}.$$

- Una radice a_j di f si dice **semplice** se $t_j = 1$, **multipla** se $t_j > 1$. Un polinomio privo di radici multiple è detto **squarefree** o **ridotto**.
- La **derivata formale** di $f(x) = a_n x^n + \cdots + a_1 x + a_0 \in F[x]$ è

$$f'(x) = n a_n x^{n-1} + \cdots + 2a_2 x + a_1.$$

- Un polinomio $f \in F[x]$ ha una radice multipla nel suo campo di spezzamento se e solo se $\mathrm{MCD}(f, f') \neq 1$.

```
i1 :  S=QQ[x]
o1 =  S
o1 :  PolynomialRing
i2 :  a=4*(x^2+1)*(x^2-2*x+4)
o2 =  4x^4-8x^3+20x^2-8x+16
o2 :  S
i3 :  ap=diff(x,a)
o3 =  16x^3-24x^2+40x-8
o3 :  S
i4 :  mcd=euclideo(a,ap)
       24843
o4 =   -----
       16
o4 :  S
i5 :  mcd=mcd/leadCoefficient mcd
o5 =  1
o5 :  S
i6 :  roots a
o6 =  {-ii, ii, 1+1.73205*ii, 1-1.73205*ii}
o6 :  List
```

```
i7 :  b=(x^2+1)^2
o7 =  x^4+2x^2+1
o7 :  S
i8 :  bp=diff(x,b)
o8 =  4x^3+4x
o8 :  S
i9 :  mcd=euclideo(b,bp)
o9 =  x^2+1
o9 :  S
i10 :  roots b
o10 =  {-ii, ii, ii, -ii}
o10 :  List
i11 :  b//mcd
o11 =  x^2+1
o11 :  S
```

Fattorizzazione di polinomi su campi finiti

- Un polinomio monico $f \in \mathbb{F}_q[x]$, $q = p^n$, di grado positivo può essere espresso in maniera unica come prodotto dei suoi fattori irriducibili:

$$f = f_1^{e_i} \cdots f_k^{e_k}$$

- Sia $d = \mathrm{MCD}(f, f') \in \mathbb{F}_q[x]$. Esaminiamo le seguenti possibilità:
  - Se $d = 1$ allora f è squarefree.
  - Se $d = f$ allora $f' = 0$, quindi $f = g^p$ con g opportuno polinomio in $\mathbb{F}_q[x]$. La fattorizzazione di f si riduce a quella di g.
  - Se $1 \neq d \neq f$ allora d è un fattore non banale di f, "coprimo" con f/d. Quindi la fattorizzazione di f si riduce a quelle di d e f/d.
- Quindi per fattorizzare un polinomio monico a coefficienti su un campo finito, procedendo ricorsivamente con le osservazioni precedenti, ci si può sempre ricondure, senza perdita di generalità, a considerare le fattorizzazioni di opportuni polinomi squarefree.

Un approccio algebrico

- Sia $f \in \mathbb{F}_q[x]$, $q = p^n$, un polinomio monico squarefree di grado positivo d. f può essere espresso in fattori irriducibili come:

$$f = f_1 \cdots f_k, \quad \deg(f_i) = d_i \text{ per } i = 1, \ldots, k.$$

- Per il teorema cinese del resto, abbiamo la seguente decomposizione:

$$R = \mathbb{F}_q[x]/(f) \cong \mathbb{F}_q[x]/(f_1) \times \cdots \times \mathbb{F}_q[x]/(f_k),$$

  dove $\mathbb{F}_q[x]/(f_i)$ sono dei campi finiti con q^{d_i} elementi.

- L'omomorfismo di Frobenius $\Phi : \mathbb{F}_q \to \mathbb{F}_q$ può essere esteso a $\mathbb{F}_q[x]$ e quindi ristretto a $R = \mathbb{F}_q[x]/(f)$. In particolare, R è uno spazio vettoriale di dimensione d su $\mathbb{F}_q$ e la mappa β è lineare su $\mathbb{F}_q$:

$$\beta = \Phi - \mathrm{id}_R : R \to R \quad \text{definita da} \quad \beta(h) = h^q - h$$

- Il nucleo $\mathcal{B} = \ker \beta$ è chiamato **sottoalgebra di Berlekamp** di R e corrisponde al sottospazio $\mathbb{F}_q \times \cdots \times \mathbb{F}_q = \mathbb{F}_q^k$ di R. Infatti

$$h \bmod f \in \ker \beta \iff h^q \equiv h \bmod f \iff h^q \equiv h \bmod f_i, \ i = 1, \ldots, k$$
$$\iff h \bmod f_i \in \mathbb{F}_q, \ i = 1, \ldots, k$$

Algoritmo di Berlekamp

- Dato $f \in \mathbb{F}_q[x]$ squarefree e preso un elemento della sottoalgebra di Berlekamp $h \in \mathcal{B}$ si ottiene la seguente relazione:

$$f(x) = \prod_{c \in \mathbb{F}_q} \mathrm{MCD}(f(x), h(x) - c)$$

- Diciamo che h mod f è una f-**riduzione**. Osserviamo che se $h(x) \equiv c \bmod f(x)$, $c \in \mathbb{F}_q$, allora h è un fattore banale di f. Altrimenti ogni fattore irriducibile f_i di f divide $h(x) - c_i$ per un opportuno $c_i \in \mathbb{F}_q$.

- Le f-riduzioni h_i non banali, utili per la fattorizzazione di f, sono date dai generatori della sottoalgebra $\mathcal{B}$. Utilizzando quindi *tutti* gli elementi $c \in \mathbb{F}_q$ (per campi piccoli), e calcolando i massimi comuni divisori fra $h_i - c$ e f, otteniamo una scomposizione di f. Procedendo ricorsivamente otteniamo tutti i fattori irriducibili di f.

Algoritmo di Berlekamp

Calcolo della base di $\mathcal{B}$

- Sia $d = \deg f$ allora gli elementi $x^i \bmod f(x)$, per $i = 0, \ldots, d-1$, formano una base di R. Quindi gli elementi $x^{iq} \bmod f(x)$, per $i = 0, \ldots, d-1$, forniscono una rappresentazione per Φ.
- Sia quindi $Q = (q_{ij}) \in \mathcal{M}_{d\times d}(\mathbb{F}_q)$ la matrice che rappresenta Φ:

$$x^{iq} \equiv \sum_{j=0}^{d-1} q_{ij} x^j \bmod f(x), \text{ per } \quad i = 0, \ldots, d-1.$$

- La matrice $B = Q - I_d$, con I_d matrice identica, rappresenta perciò la mappa β, il cui nucleo è $\mathcal{B}$. Calcolando quindi il nucleo della matrice B si ottiene una base per $\mathcal{B}$, $\{h_1, \ldots, h_k\}$, con $k = d - \text{rank}B$.
- Una prima importante osservazione è la seguente:

$$f \text{ irriducibile } \iff k = 1 \iff \text{rank}(Q - I_d) = d - 1$$

Algoritmo di Berlekamp

Calcolo dei fattori

- Osserviamo che il polinomio 1 appartiene sempre a $\mathcal{B}$ ($1^q - 1 = 0$). Escluso 1, si dovranno applicare i metodi già analizzati alle rimanenti f-riduzioni per ottenere la fattorizzazione di f.
- Sappiamo che i fattori irriducibili sono in numero di k, ma non esiste biunivocità fra essi e le f-riduzioni. Quindi è possibile trovare i k fattori irriducibili di f prima di aver esaminato tutte le f-riduzioni.
- Se $k \geq 2$ allora, per ogni f-riduzione $h_i \in \mathcal{B}$, è necessario calcolare:

$$\text{MCD}(f(x), h_i(x) - c), \text{ per ogni } c \in \mathbb{F}_q.$$

- È interessante osservare che, essendo f squarefree, ogni fattore irriducibile sarà anch'esso squarefree nella fattorizzazione. Quindi dopo aver calcolato $\text{MCD}(f(x), h_2(x) - c)$, per la prima f-riduzione non banale h_2 per ogni $c \in \mathbb{F}_q$, gli eventuali fattori trovati $g = f_1 \cdots f_t$ possono essere eliminati da f e passare al calcolo successivo:

$$\text{MCD}(f(x)/g(x), h_3(x) - c), \text{ per ogni } c \in \mathbb{F}_q.$$

Algoritmo di Berlekamp

Algorithm 6: Fattorizzazione di polinomi

Input: Polinomio $f \in \mathbb{F}_q[x]$
Output: Lista di una fattorizzazione di f (da iterare)
begin
  $d \leftarrow \deg f$;
  for $j \leftarrow 0$ **to** $d-1$ **do**
    $q_j \leftarrow x^{j*q} \% f$;
    for $i \leftarrow 0$ **to** $d-1$ **do**
      $Q \leftarrow (q_{ij})$;
    end
  end
  $B \leftarrow Q - I_d$;
  $H \leftarrow$ generatori $\ker B$;
  $k \leftarrow \dim \ker B$;
  if $k = 1$ **then**
    return f;
  end
  foreach $h \in H$ **do** // $|fatt| \neq k$
    foreach $c \in \mathbb{F}_q$ **do**
      $fatt \leftarrow fatt \cup \{\mathrm{MCD}(f(x), h(x) - c)\}$ // $f \leftarrow f / \prod fatt$
    end
  end
  return $fatt$;
end

Esempio

- Sia $f(x) = x^8 + x^6 + x^4 + x^3 + 1 \in \mathbb{F}_2[x]$. Notiamo che f è squarefree, infatti $\text{MCD}(f(x), f'(x)) = 1$.
- Calcoliamo $x^{iq} \bmod f(x)$ con $q = 2$ e $0 \leq i \leq 7$ e la matrice B:

$$
\begin{aligned}
x^0 &\equiv 1 \\
x^2 &\equiv x^2 \\
x^4 &\equiv x^4 \\
x^6 &\equiv x^6 \\
x^8 &\equiv 1 + x^3 + x^4 + x^6 \\
x^{10} &\equiv 1 + x^2 + x^3 + x^4 + x^5 \\
x^{12} &\equiv x^2 + x^4 + x^5 + x^6 + x^7 \\
x^{14} &\equiv 1 + x + x^3 + x^4 + x^5
\end{aligned}
\qquad
\begin{pmatrix}
\mathbf{0} & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & \mathbf{1} & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & \mathbf{1} & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & \mathbf{0} & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & \mathbf{0} & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & \mathbf{0} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{1}
\end{pmatrix}
$$

- Il rango di B su $\mathbb{F}_2$ è 6, quindi il numero di fattori irriducibili di f è $k = d - \text{rank}B = 8 - 6 = 2$.

Esempio

- I generatori di $\mathcal{B} = \ker B$ sono $(1,0,0,0,0,0,0,0)$ e $(0,1,1,0,0,1,1,1)$ e le f-riduzioni corrispondenti sono:

$$h_1(x) = 1, \quad h_2(x) = x + x^2 + x^5 + x^6 + x^7.$$

- Calcoliamo il $\mathrm{MCD}(f(x), h_2(x) - c)$ per ogni $c \in \mathbb{F}_2$:

$$\mathrm{MCD}(f(x), x^7 + x^6 + x^5 + x^2 + x) = x^6 + x^5 + x^4 + x + 1,$$
$$\mathrm{MCD}(f(x), x^7 + x^6 + x^5 + x^2 + x + 1) = x^2 + x + 1.$$

- In questo caso la sola f-riduzione non banale h_2 ha fornito entrambi i fattori irriducibili di f. La fattorizzazione di f in $\mathbb{F}_2[x]$ è la seguente:

$$f(x) = (x^6 + x^5 + x^4 + x + 1)(x^2 + x + 1)$$

Codice

Un'ottimizzazione per grandi campi

- Se la cardinalità di $\mathbb{F}_q$ è molto elevata, l'applicazione dell'algoritmo già descritto diventa computazionalmente onerosa poiché richiede di calcolare $k \cdot q$ volte l'algoritmo di Euclide.
- Tramite il calcolo del **risultante** è possibile caratterizzare a priori gli elementi $c \in \mathbb{F}_q$ per cui si ha che $\text{MCD}(f(x), h(x) - c) \neq 1$. Infatti, sia $\mathcal{R}(f(x), h(x) - c)$ il risultante di $f(x)$ e $h(x) - c$, allora vale che:

$$\mathcal{R}(f(x), h(x) - c) = 0 \quad \Longleftrightarrow \quad \text{MCD}(f(x), h(x) - c) \neq 1.$$

- Considerando quindi $p(y) = \mathcal{R}(f(x), h(x) - y)$ si ottiene un polinomio nell'indeterminata y di grado al più d a valori in $\mathbb{F}_q[x]$. Quindi $\text{MCD}(f(x), h(x) - c) \neq 1$ soltanto se $c \in \mathbb{F}_q$ è una radice di $p(y)$.
- Nella *teoria dell'eliminazione*, la moderna alternativa al risultante è il calcolo della **base di Gröbner** ridotta G, usando un ordinamento monomiale opportuno, dell'ideale $I = (f(x), h(x) - y) \in \mathbb{F}[x, y]$. Un polinomio di G sarà proprio il risultante $p(y)$.
- Esistono diverse strategie per il calcolo del risultante e per la ricerca delle radici distinte di $p(y)$ in $\mathbb{F}_q$. Tali operazioni permettono di calcolare al più $k \cdot d$ volte l'algoritmo di Euclide.

Esempio

- Sia $f(x) = x^6 - 3x^5 + 5x^4 - 9x^3 - 5x^2 + 6x + 7 \in \mathbb{F}_{23}[x]$. Anche in questo caso f è squarefree, infatti $\mathrm{MCD}(f(x), f'(x)) = 1$.
- Calcoliamo $x^{iq} \bmod f(x)$ con $q = 23$ e $0 \leq i \leq 5$ e la matrice B:

$$
\begin{aligned}
x^0 &\equiv 1 \\
x^{23} &\equiv 5 - 1x^2 + 8x^3 - 3x^4 - 10x^5 \\
x^{46} &\equiv -10 + 10x + 10x^2 + x^4 - 9x^5 \\
x^{69} &\equiv 7x + 9x^2 - 8x^3 + 10x^4 - 11x^5 \\
x^{92} &\equiv 11 - 4x^2 + 7x^3 + 7x^4 + 2x^5 \\
x^{115} &\equiv -3 - 10x^2 + 9x^3 + 2x^4 - 9x^5
\end{aligned}
\qquad
\begin{pmatrix}
\mathbf{0} & 5 & -10 & 0 & 11 & -3 \\
0 & \mathbf{-1} & 10 & 7 & 0 & 0 \\
0 & -1 & \mathbf{9} & 9 & -4 & -10 \\
0 & 8 & 0 & \mathbf{-9} & 7 & 9 \\
0 & -3 & 1 & 10 & \mathbf{6} & 2 \\
0 & 10 & -9 & -11 & 2 & \mathbf{-10}
\end{pmatrix}
$$

- Il rango di B su $\mathbb{F}_{23}$ è 3, quindi il numero di fattori irriducibili di f è $k = d - \operatorname{rank} \ker B = 6 - 3 = 3$.
- I tre generatori di $\mathcal{B} = \ker B$ sono $(1, 0, 0, 0, 0, 0)$, $(0, 4, 2, 1, 0, 0)$ e $(0, -2, 9, 0, 1, 1)$. Le f-riduzioni corrispondenti sono:

$$h_1(x) = 1, \quad h_2(x) = 4x + 2x^2 + x^3, \quad h_3(x) = -2x + 9x^2 + x^4 + x^5.$$

Esempio

- Per ottenere il risultante, calcoliamo la base di Gröbner dell'ideale:

$$I = (f(x), h_2(x) - y) \in \mathbb{F}_{23}[x, y].$$

- Il primo polinomio di G sarà proprio $p(y) = y^3 - 5y^2 + 11y - 10$, le cui radici in $\mathbb{F}_{23}$ sono $\{-3, 2, 6\}$.

- Calcoliamo il MCD$(f(x), h_2(x) - c)$ per ogni $c \in \{-3, 2, 6\}$:

$$\begin{aligned}
&\text{MCD}(f(x), h_2(x) + 3) = x - 4,\\
&\text{MCD}(f(x), h_2(x) - 2) = x^2 - x + 7,\\
&\text{MCD}(f(x), h_2(x) - 6) = x^3 + 2x^2 + 4x - 6.
\end{aligned}$$

- Poiché sono stati ottenuti tre polinomi distinti, non bisogna procedere per h_3. La fattorizzazione di f in $\mathbb{F}_{23}[x]$ è la seguente:

$$f(x) = (x - 4)(x^2 - x + 7)(x^3 + 2x^2 + 4x - 6).$$

Codice

Bibliografia

Bibliografia i

[1] J. H. Davenport, Y. Siret, and E. Tournier.
Computer Algebra: Systems and Algorithms for Algebraic Computation (2th ed.).
Academic Press Ltd., GBR, 1993.

[2] J. Grabmeier, E. Kaltofen, and V. Weispfenning, editors.
Computer Algebra Handbook.
Springer-Verlag, Berlin, Heidelberg, 2003.

[3] D. R. Grayson and M. E. Stillman.
Macaulay2, a software system for research in algebraic geometry.
Available at `http://www.math.uiuc.edu/Macaulay2/`, 2020.

[4] D. E. Knuth.
The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms.
Addison-Wesley Longman Publishing Co., Inc., USA, 1997.

Bibliografia ii

[5] R. Lidl and H. Niederreiter.
Introduction to Finite Fields and their Applications.
Cambridge University Press, 2 edition, 1994.

[6] J. von zur Gathen and J. Gerhard.
Modern Computer Algebra.
Cambridge University Press, 3 edition, 2013.

Algoritmo della Somma a precisione multipla

Algorithm 1: Somma a precisione multipla (valori assoluti)

Input: Interi a precisione multipla a, b
Output: Intero a precisione multipla c
begin
  $n \leftarrow 64$;
  $k \leftarrow \max\{a_0 - s_a * 2^{n-1}, b_0 - s_b * 2^{n-1}\}$;
  $r \leftarrow \{0\}$;
  $c \leftarrow \{0 * 2^{n-1} + k\}$;
  for $i \leftarrow 1$ **to** k **do**
    $tmp \leftarrow a_i + b_i + r_{i-1}$;
    if $tmp \geq 2^n$ **then**
      $r \leftarrow r \cup \{1\}$;
    else
      $r \leftarrow r \cup \{0\}$;
    end
    $c \leftarrow c \cup \{tmp - r_i * 2^n\}$;
  end
  $c \leftarrow c \cup \{r_{k+1}\}$;
  return c;
end

Indietro

Algoritmo di divisione fra polinomi in un dominio euclideo $F[x]$

Algorithm 2: Divisione fra polinomi

Input: Polinomi a, b
Output: Polinomi q, r
begin
    $q \leftarrow 0$;
    $r \leftarrow a$;
    while $r \neq 0$ *and* $\deg(\mathrm{LT}(b)) \leq \deg(\mathrm{LT}(r))$ **do**
        $q \leftarrow q + \mathrm{LT}(r)/\mathrm{LT}(b)$;
        $r \leftarrow r - \mathrm{LT}(r)/\mathrm{LT}(b) * b$;
    end
    return $\{q, r\}$;
end

Indietro

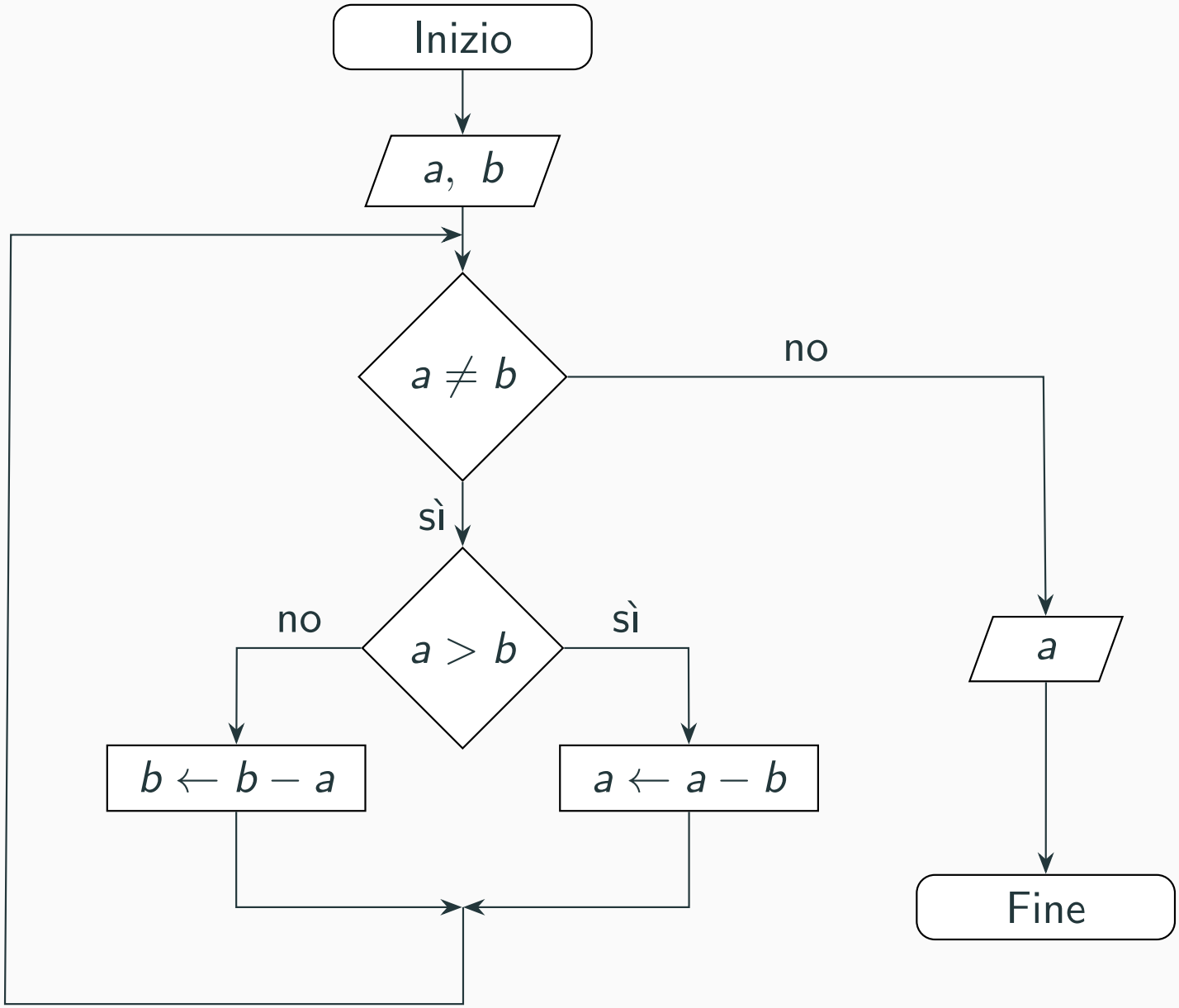
Algoritmo euclideo per il massimo comune divisore

Algorithm 3: Massimo comune divisore (MCD)

Input: Elementi a, b
Output: Massimo comune divisore
begin
    $r \leftarrow \{a, b\}$;
    $i \leftarrow 1$;
    while $r_i \neq 0$ **do**
        $r \leftarrow r \cup \{r_{i-1} \% r_i\}$;
        $i \leftarrow i + 1$;
    end
    return r_{i-1};
end

Indietro

Un diagramma di flusso



Algoritmo euclideo esteso

Algorithm 4: Algoritmo euclideo esteso

Input: Elementi a, b

Output: Massimo comune divisore e coefficienti

begin

$r \leftarrow \{a, b\}$;

$s \leftarrow \{1, 0\}$;

$t \leftarrow \{0, 1\}$;

$i \leftarrow 1$;

while $r_i \neq 0$ **do**

$q \leftarrow q \cup \{r_{i-1} // r_i\}$;

$r \leftarrow r \cup \{r_{i-1} - q_{i-1} r_i\}$;

$s \leftarrow s \cup \{s_{i-1} - q_{i-1} s_i\}$;

$t \leftarrow t \cup \{t_{i-1} - q_{i-1} t_i\}$;

$i \leftarrow i + 1$;

end

return $\{r_{i-1}, s_{i-1}, t_{i-1}\}$;

end

Indietro

Algoritmo del teorema cinese del resto in un DE

Algorithm 5: Teorema cinese del resto (CRT)

Input: Elementi $\{a_i\}_{i\in[k]}$, $\{m_i\}_{i\in[k]}$ (coprimi a coppie)
Output: Soluzione del sistema di congruenze $x \equiv a_i \bmod m_i$
begin
    $m \leftarrow m_1 \cdots m_k$;
    $x \leftarrow 0$;
    for $i \leftarrow 1$ **to** k **do**
        $s \leftarrow \texttt{esteso}(m/m_i, m_i)_1$;
        $c \leftarrow (s \cdot a_i)\%m_i$;
        $x \leftarrow x + c * m/m_i$;
    end
    return $x\%m$;
end

Indietro

```
i1 :  q=2
i2 :  F=ZZ/q
i3 :  S=F[x]
i4 :  f=x^8+x^6+x^4+x^3+1
i5 :  d=(degree f)_0
i6 :  frob=for i to d-1 list x^(i*q)%f
i7 :  Q=transpose matrix for i in frob list
          for j to d-1 list (
              c=flatten entries (coefficients i)_1;
              e=flatten exponents i;
              ind=position(e,u->u==j);
              if ind===null then 0 else c_ind
          )
i8 :  B=Q-id_(F^d)
i9 :  K=mingens kernel B
i10 :  k=numgens source K
i11 :  p=matrix for i to d-1 list x^i
i12 :  H=flatten entries(p*K)
i13 :  H=select(H,i->degree i!=0)
i14 :  HC=unique flatten for h in H list
                   for i to q-1 list h-i
i15 :  fatt=unique for hc in HC list gcd(f,hc)
i16 :  fatt=select(fatt,i->i!=1)
i17 :  fatt==toList factor f
```

Indietro

```
i1 :  q=23; F=ZZ/q; S=F[x..y,MonomialOrder=>Lex]
i2 :  f=x^6-3*x^5+5*x^4-9*x^3-5*x^2+6*x+7; d=(degree f)_0
i3 :  frob=for i to d-1 list x^(i*q)%f
i4 :  Q=transpose matrix for i in frob list for j to d-1 list (
             c=flatten entries (coefficients i)_1;
             e=flatten exponents i; e=for i to #e//2-1 list e_(i*2);
             ind=position(e,u->u==j); if ind===null then 0 else c_ind
          )
i5 :  B=Q-id_(F^d); K=mingens kernel B; k=numgens source K
i6 :  p=matrix for i to d-1 list x^i
i7 :  H=flatten entries(p*K); H=select(H,i->degree i!=0)
i8 :  fatt={}
i9 :  for h in H do (
          I=ideal(f,h-substitute(y,S));
          ris=(flatten entries gens gb I)_0;
          risY=substitute(ris,F[y]);
          sol=flatten rationalPoints ideal risY;
          HC=unique flatten for s in sol list h-promote(s,S);
          fatt=fatt | for hc in HC list gcd(f,hc);
          fatt=select(unique fatt,i->i!=1);
          if #fatt==k then break;
      )
i10 :  fatt==toList factor f
```

Indietro