L' INFORMATICA: DUE CONCETTI DI BASE

Fabrizio Luccio, Pisa 2019

La decidibilità

l'algoritmo

non tutti i problemi sono risolubili

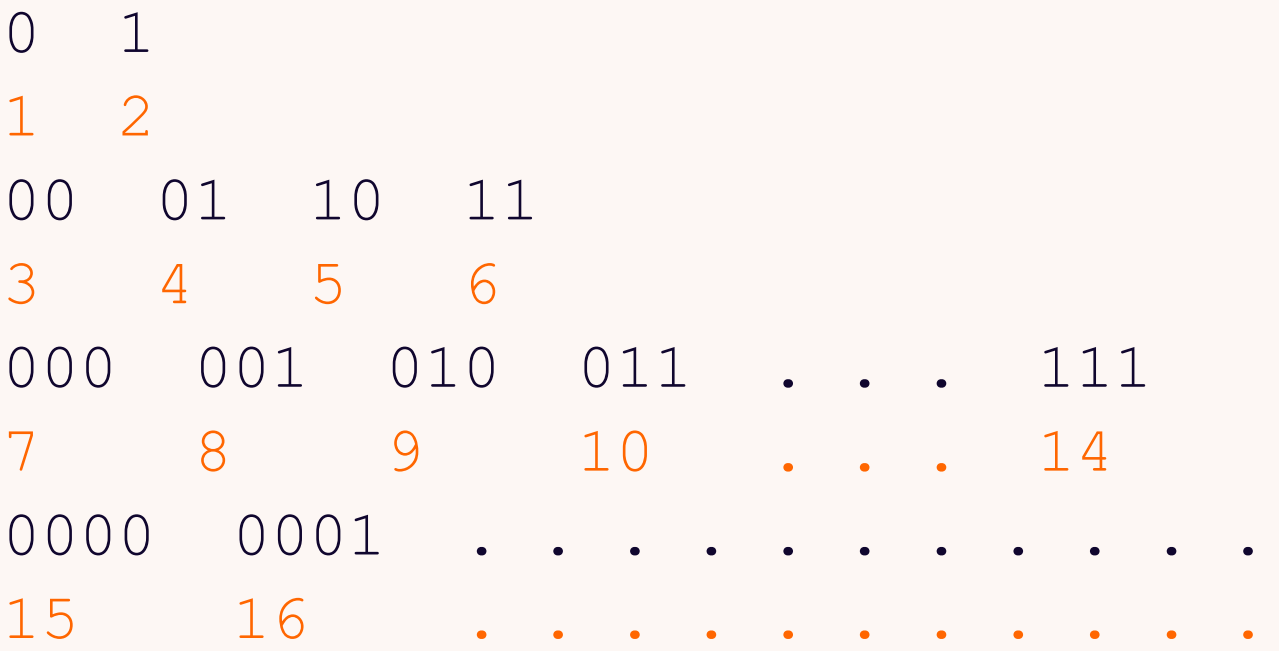
La complessità

la rappresentazione

problemi trattabili e intrattabili

La decidibilità

nasce alla fine del 1800 dalla teoria degli insiemi infiniti che comporta il concetto di numerabilità



Le sequenze sono numerabili

	0	1	2	3	4	5	6	.	.
F0	1	1	0	1	0	1	1	.	.
F1	0	1	0	0	0	1	0	.	.
F2	0	0	0	1	1	0	1	.	.
F3	1	0	1	1	1	0	1	.	.
..	.	.	.	.	.	.	.	.	.

Tabella delle funzioni Fi: N ➔ {0,1}

	0	1	2	3	4	5	6	.	.
F0	1	1	0	1	0	1	1	.	.
F1	0	1	0	0	0	1	0	.	.
F2	0	0	0	1	1	0	1	.	.
F3	1	0	1	1	1	0	1	.	.
..	.	.	.	.	.	.	.	.	.

F3(4) = 1

	0	1	2	3	4	5	6	.	.
F0	1	1	0	1	0	1	1	.	.
F1	0	1	0	0	0	1	0	.	.
F2	0	0	0	1	1	0	1	.	.
F3	1	0	1	1	1	0	1	.	.
..	.	.	.	.	.	.	.	.	.

La funzione Fj: Fj(i)=0 se e solo se Fi(i)=1

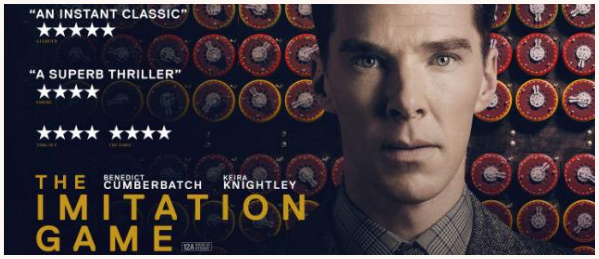
Fj non appartiene all'elenco

Le funzioni non sono numerabili

La numerabilità degli algoritmi (sequenze finite) e la non numerabilità delle funzioni (sequenze infinite) implica l'esistenza di funzioni non calcolabili

ma richiede di porre una definizione formale al concetto di algoritmo

Nel 1936 Alan Turing definisce l'algoritmo attraverso una famosa macchina



È questa la "Macchina di Turing" ?

La Macchina di Turing: MT = {S, A, B, T}

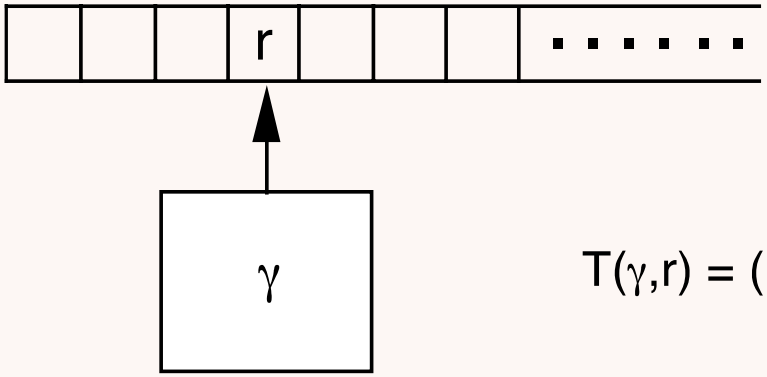
S = {α, β, γ, . . . , ω} α stato iniziale
ω stato finale

A = {a, b, c, ... , z} z = blank

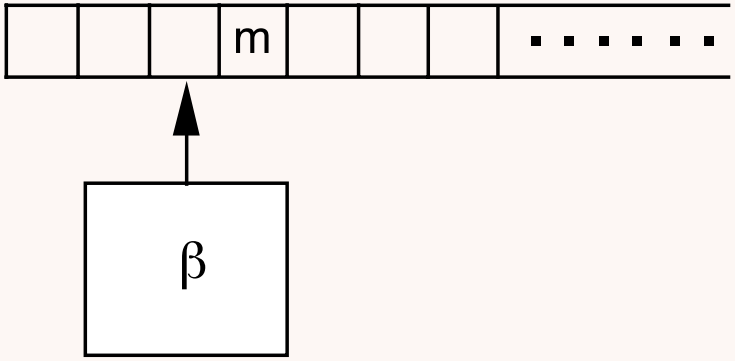
B = {<, -, >} < sinistra, - fermo, > destra

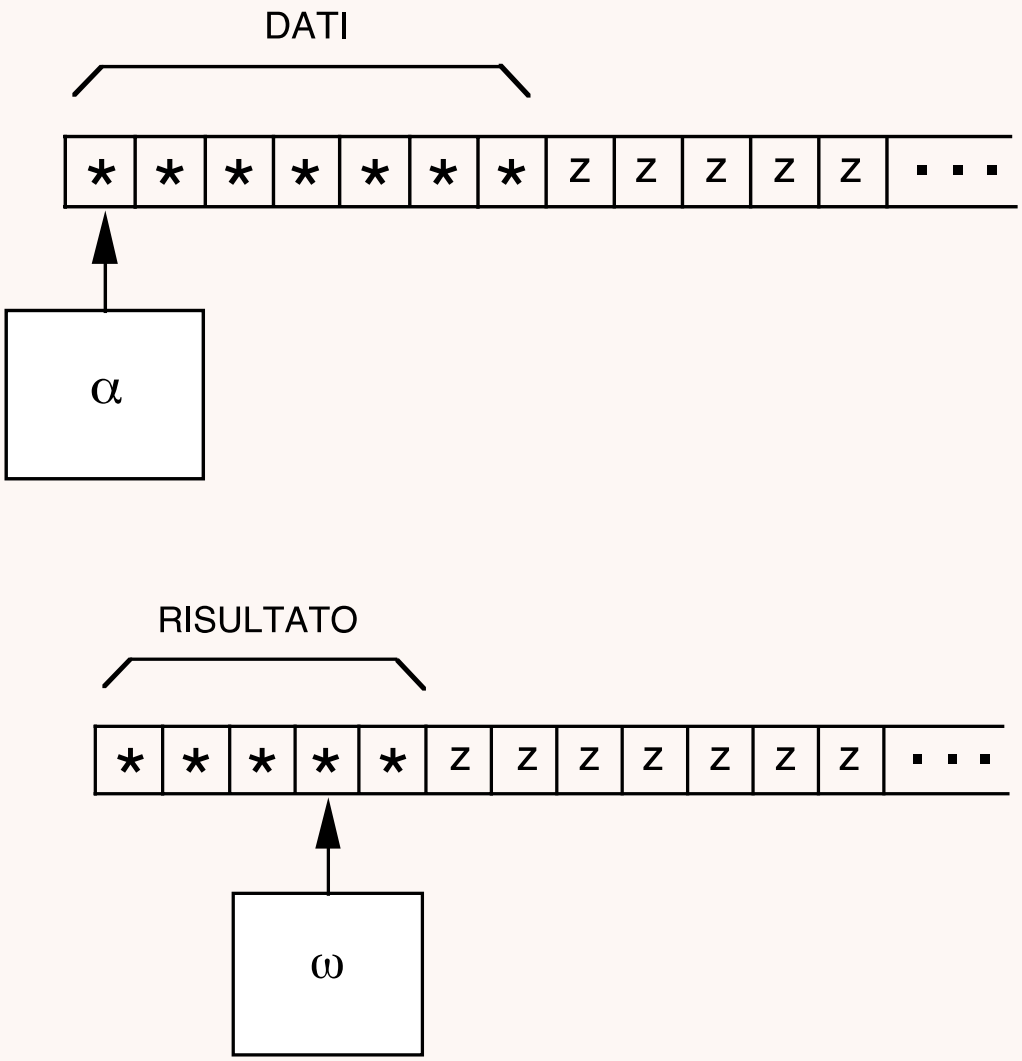
T: S × A ➔ S × A × M

per esempio: T(γ,r) ➔ (β,m,<)



$T(\gamma,r) = (\beta,m,<)$

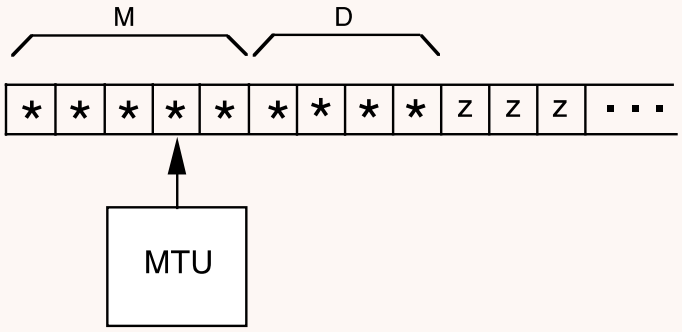



DATI
* * * * * * * Z Z Z Z Z ...
α
RISULTATO
* * * * * Z Z Z Z Z Z Z ...
ω

Una Macchina di Turing MT che termina il suo calcolo in tempo finito è un algoritmo. Algoritmi diversi corrispondono a MT diverse.

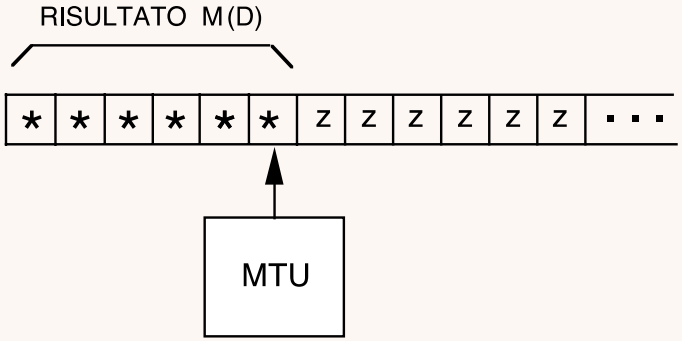
Poiché una MTsi descrive con una sequenza finita di simboli, essa ha la stessa natura della sequenza D dei suoi dati. Dunque si può fornire la M come sequenza di dati per a un'altra MT.

Tutto questo ha permesso a Turing di definire la Macchina Universale MTU che accetta come dati M e D e simula il calcolo di M su D



MTU simula il

calcolo di M su D



Dunque la Macchina Universale

è un calcolatore

Stabilita una definizione di algoritmo, cerchiamo una funzione non calcolabile

Problema della terminazione. Per una MT arbitraria M e i suoi dati arbitrari D decidere (in tempo finito) se il calcolo di M su D termina in tempo finito.

Non esiste un algoritmo HALT che decide (in tempo finito) se un altro algoritmo arbitrario A, operando su dati arbitrari D, termina o no:

```
HALT(A,D)=true,  se A(D) termina

HALT(A,D)=false, se A(D) non termina
```

La dimostrazione è basata sul paradosso del mentitore, combinazione di autoriferimento e negazione

Consideriamo un nuovo algoritmo P che lavora su una sequenza A che rappresenta un algoritmo:

```
P(A)
   while (HALT(A,A)==true} A = A;
   print "A(A) non termina";
```

P(A) termina se e solo se A(A) non termina

P(P) termina se e solo se P(P) non termina !!!

È un'antinomia: il punto debole
è l'ammissione che HALT esista

Ma vediamo una conseguenza forse inaspettata del problema della terminazione

La congettura di Goldbach afferma che ogni numero pari n si può esprimere come somma di due numeri primi

per esempio: 8 = 4+4 (non primi), ma anche 8 = 3+5 (primi)

La congettura non è stata mai dimostrata, ma è stata verificata fino a valori altissimi di n.

Consideriamo allora l'algoritmo:

```
GOLD
    per ogni n pari a partire da 2
      costruisci tutte le coppie a,b con a+b=n;
      if esiste una coppia a e b di primi
         procedi con il prossimo valore di n
      else stampa n e arresta il calcolo.
```

Se la congettura fosse falsa, GOLD si arresterebbe sul primo valore di *n* che non la verifica, altrimenti continuerebbe a girare all'infinito. Quindi GOLD termina se e solo se la congettura è falsa per un certo valore di n.

Se esistesse l'algoritmo HALT:

HALT(GOLD)= true (cioè GOLD si arresta)

$\Rightarrow$ la congettura è falsa

HALT(Gold)= false (cioè GOLD non si arresta)

$\Rightarrow$ la congettura è vera

Se esistesse (e io conoscessi) l'algoritmo HALT,
avrei un modo banale di dimostrare quasi gratis
moltissimi teoremi e congetture sui numeri interi !

Successivamente è stato stabilito che molti problemi sono indecidibili per trasformazione con il problema della terminazione.

Il teorema di Rice dimostra che qualunque questione "non banale" nella teoria dei linguaggi è indecidibile.

La rappresentazione

è basata sulla crescita esponenziale, una legge che permette di comunicare se si usa un "alfabeto" di almeno due simboli

Impiegando k simboli il numero p di "parole" di lunghezza n cresce esponenzialmente con n:

$$p = k^n$$

Alfabeto inglese: $k = 26,\ n = 3,\ p = 26^3 = 17.576$

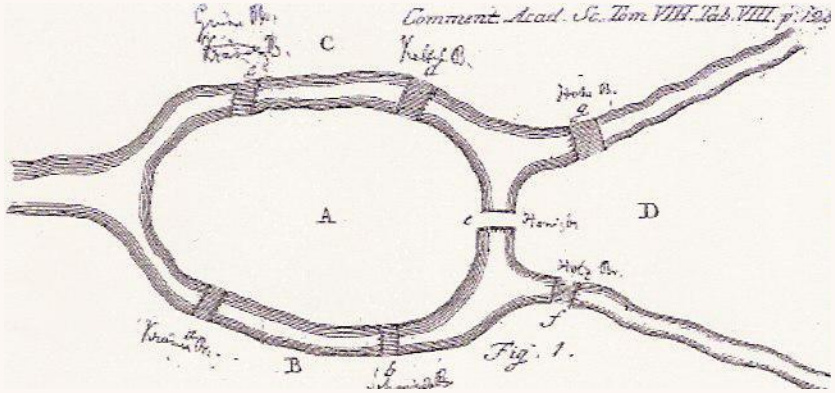
Parole: aaa aab aac . . . . zzz

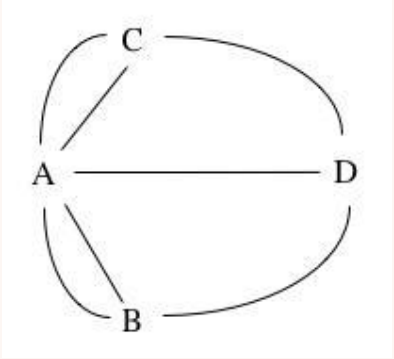
Cifre decimali: $k = 10,\ n = 3,\ p = 10^3 = 1000$

Numeri: 000 001 002 999

La complessità di calcolo

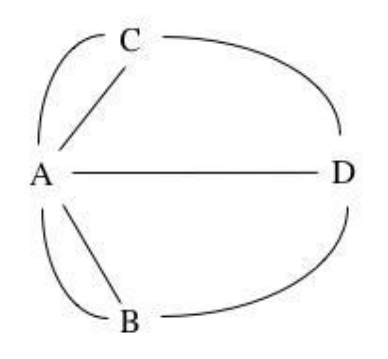
Königsberg, 1735

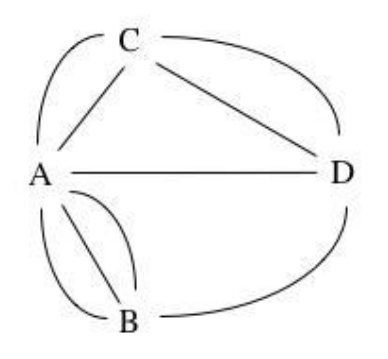




La condizione di Eulero:

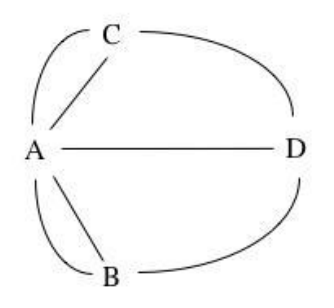
Esiste un ciclo che traversa tutti gli archi (ponti) esattamente una volta se e solo se tutti i nodi (zone della città) hanno grado pari.



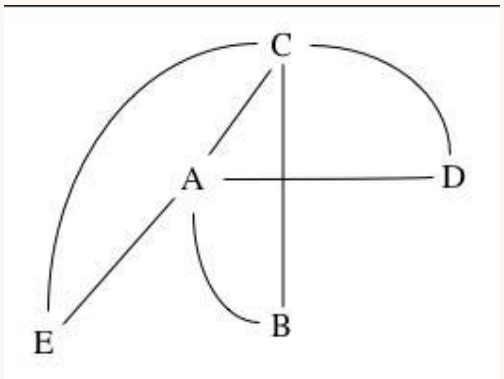


A C D B A C D A B A

Il ciclo Hamiltoniano traversa tutti i nodi esattamente una volta



A B D C A



A B C D E ?
A B C E D ?
. . . . . .

In linea di principio si devono fare n! = 120 prove

n =	2	3	4	5	6	7	8	9	10
n! =	2	6	24	120	720	5040	40320	362880	> 3.5M

n! cresce come $(n/e)^n$

La complessità (di calcolo)

è il "tempo" (cioè il numero di operazioni elementari) necessario per risolvere un problema.

Sono intrattabili i problemi che richiedono tempo esponenziale nella dimensione dell'input

Complessità polinomiale e esponenziale

Un algoritmo di complessità polinomiale risolve un certo problema su n dati in tempo $t = cn^s$ su un computer A.

Lo stesso algoritmo, su un computer k volte "più veloce" di A, risolve in tempo t lo stesso problema su m dati:

$$cn^s = t \qquad cm^s = kt \qquad \text{dunque} \qquad m = k^{1/s} n$$

Ripetiamo il ragionamento con un algoritmo di complessità $c2^n$:

$$c2^n = t \qquad c2^m = kt \qquad \text{dunque} \qquad m = n + \log_2 k$$

Le due principali classi di complessità

P è la classe dei problemi che si risolvono in tempo polinomiale.

NP è la classe dei problemi che si verificano in tempo polinomiale (ma si sanno risolvere solo in tempo esponenziale).

P = NP ?

Per la risoluzione di questo problema è in palio un premio di 1M$

Tre problemi tra i "più difficili" in NP

Biologia molecolare: dato un insieme arbitrario di frammenti di sequenze di DNA stabilire se esiste una sequenza incognita ma di lunghezza assegnata che contenga al suo interno ognuno dei frammenti dati.

Trasporti: stabilire se un insieme di casse di pesi diversi possono essere caricate su un assegnato numero di camion senza superare il peso massimo sopportabile da ciascun camion.

Algebra: stabilire se un'equazione algebrica di secondo grado in due variabili x, y, a coefficienti interi, ammette una soluzione in cui x e y hanno valori interi (se l'equazione è di primo grado il problema è in P).

Un campo di applicazione ove la distinzione tra problemi polinomiali e esponenziali è alla base:

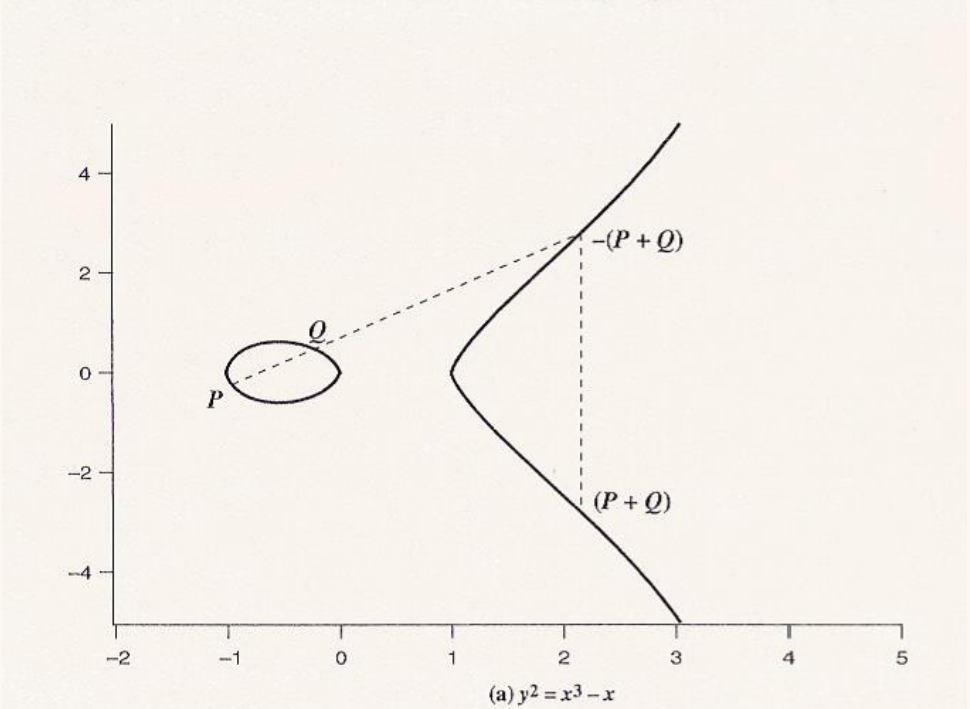
la crittografia, che utilizza funzioni "one way" cioè "facili da calcolare" e "difficili" da invertire

Vediamo come due partner possano costruire una loro chiave segreta con un metodo noto a tutti, anche se tutti ne intercettano la comunicazione

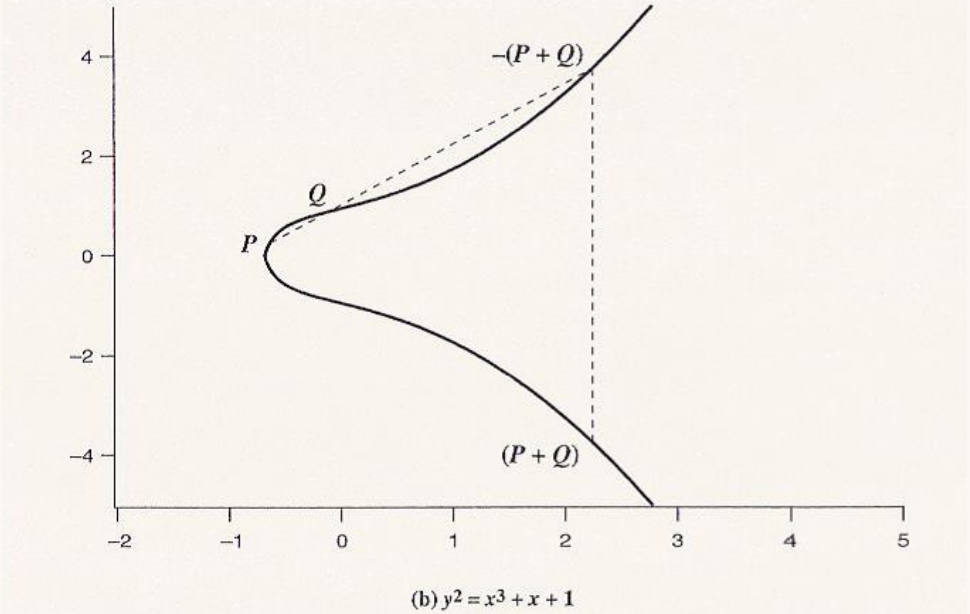
Le curve ellittiche

$$y^2 = x^3 + ax + b$$

Si considerano solo i punti a coordinate intere (e si opera "in modulo")



(a) $y^2 = x^3 - x$



(b) $y^2 = x^3 + x + 1$

Un punto P può essere sommato a se stesso mediante la tangente in P alla curva

Moltiplicazione di P per una un numero intero k:

$$P + P + \ldots + P = k\,P$$

La moltiplicazione è eseguita con raddoppi e addizioni. Per esempio se $k = 13$:

$$13\,P = P + 4P + 8P = P + (2(2P)) + (2(4P))$$

Per $R = k P$

dati k e P, si calcola R in tempo "breve"

dati R e P, si sa calcolare k solo in tempo esponenziale

Costruzione di una chiave tra Alice e Bob

- I due concordano su una curva da usare e su un suo punto P (che possono essere noti a tutti)
- Alice sceglie a caso un intero segreto α, calcola αP e lo invia a Bob
- Bob sceglie a caso un intero segreto β, calcola βP e lo invia ad Alice
- Alice calcola la chiave comune $k = \alpha\beta P$
  Bob calcola la chiave comune $k = \beta\alpha P$