

Crittografia e sicurezza delle reti

1. Classi complessità
probabilistiche e OWF
2. Test primalità

Classi di complessità

Una classe di complessità è una classe di problemi di decisione (linguaggi) che possono essere risolti con determinate risorse di calcolo

P classe dei problemi risolubili in tempo polinomiale (algoritmi **deterministici**)

NP classe dei problemi risolubili in tempo polinomiale (algoritmi **NONdeterministici**)

Classi di complessità probabilistiche

Un algoritmo probabilistico è un algoritmo che effettua **scelte casuali**

- sullo stesso input può dare risposte diverse (a seconda delle scelte casuali fatte)
- un algoritmo probabilistico può fare errori

RP classe dei linguaggi L per cui esiste un algoritmo probabilistico A tale che

- $x \in L : \text{Prob}[A(x) \text{ accetta}] > .5$
- $x \text{ non in } L : \text{Prob}[A(x) \text{ rifiuta}] = 1$

Classi di complessità probabilistiche

- Algoritmo Monte Carlo: sbaglia da lato solo
- Algoritmo Las Vegas: senza errori (cambia tempo esecuzione; talvolta risponde 'NON SO') (classe ZPP)
- Caso generale (errori da due lati):
PP classe dei linguaggi L per cui esiste un algoritmo probabilistico A tale che
 - $x \in L : \text{Prob}[A(x) \text{ accetta}] > .5$
 - $x \text{ non in } L : \text{Prob}[A(x) \text{ rifiuta}] > .5$

OWF: definizione

Definizione: $f: D \rightarrow R$ è *one way function* se è

- **facile da calcolare** (tempo polinomiale con algoritmo probabilistico)
- **difficile da invertire:**

Per ogni algoritmo polinomiale probabilistico A esiste una funzione trascurabile $g(n)$ - n dimens. input, $g(n) < 1/\text{polin.}(n)$ - tale che

se si sceglie a caso x in D , con x di dimensione n

$$\text{Prob} [A \text{ trovi } z, \text{ tale che } f(z) = f(x)] < g(n)$$

(è poco probabile riuscire a invertire f)

NB: non si richiede di trovare x

Trap-door OWF

Definizione: $f: D \rightarrow R$ è una *trap-door one way function* se esiste una trap-door s tale che

- Senza conoscenza di s , la funzione è one way
- Dato s , invertire f è facile

Si assume che RSA sia OWF. Si può provare che se RSA è OWF allora è anche trap-door OWF

Test di primalità e fattori primi

Input: Un intero dispari M , $2^{n-1} < M < 2^n$

Problema di Decisione: Composto

$M \in \text{Composto}$ se M non è primo

Il linguaggio Composto è in NP

(prova esaustivamente tutti i possibili fattori & verifica)

Problema di Decisione: Primo

$M \in \text{Primo}$ se M è primo

Primo è in NP

$M \in \text{Primo}$ se e solo se M non appartiene a Composto

Test di primalità e ricerca di fattori primi

Problema di Decisione: M è composto ?

Problema di Ricerca: Trova i fattori primi di M .

Nota: il metodo semplice (imparato a scuola) per risolvere i due problemi è lo stesso:

dato M verifica esaustivamente (per enumerazione) i fattori primi

se non ci sono fattori primi

allora è primo

altrimenti hai trovato i fattori primi

Test di primalità e ricerca di fattori primi

Input: Un intero dispari M , $2^{n-1} < M < 2^n$

Problema di Decisione: Composto

($M \in$ Composto se M non è primo)

Il linguaggio Composto è in RP

Problema di Ricerca: Trova i fattori primi di M .

La fattorizzazione di interi è considerato un problema intrattabile

(si crede che non appartenga a RP o alcune delle classi probabilistiche ad es. PP, BPP..)

Primality Testing

Domanda

Esiste un metodo migliore di decidere se un numero è primo senza risolvere il problema della fattorizzazione (che richiede una ricerca esaustiva)?

[Solovay-Strassen, 1977]:

per mostrare che M sia primo è sufficiente trovare evidenza che M non si comporti come numero primo.

Primality Testing

Piccolo teorema di Fermat: se M è primo allora $a^{M-1} = 1$ (si applica il teorema di Eulero con $\varphi(M) = n-1$).

Quindi un qualunque $1 < a < M$ che soddisfa $a^{M-1} \neq 1$ mostra che M non è primo (senza fattorizzare)

Esempio:

```
> M:=78888880997:
```

```
> 769967665 &^ (M-1) mod M;  
10621956220
```

M è composto

Il "Test di Fermat" funziona sempre?

Primality Testing

In alcuni casi fallisce !

Esempio:

```
> M:= 225593397919:  
> 769967665 &^ (M-1) mod M;  
1  
> 3222223664 &^ (M-1) mod M;  
1
```

Infatti M non è primo

```
> gcd(6619,M) ;  
6619
```

Numeri di Carmichael

Numeri composti per cui il test di Fermat fallisce ($a^{M-1} = 1$) per la maggior parte degli interi a , $1 < a < M-1$.

$M = p_1 p_2 p_3 \dots p_k$ ($k > 2$), dove p_i sono primi distinti, e ogni p_i soddisfa $p_i - 1$ divide $M - 1$.

Esempio

```
> M:= 225593397919: ifactor(M);  
                (15443) (6619) (2207)  
> (M-1) mod 15442 ; (M-1) mod 6618; (M-1) mod 2206;  
                0  
                0  
                0
```

Numeri di Carmichael: Rari, ma infiniti

Smart witness che M sia composto

Sia $M-1=2^k r$ con r dispari.

Scegli $1 < b < M$.

Calcola mod M

$$a_0 = b^r, a_1 = (a_0)^2, a_2 = (a_1)^2, \dots, a_k = (a_{k-1})^2.$$

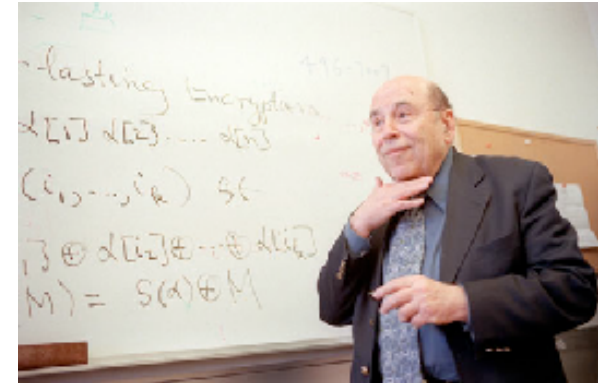
1. Se $a_k \neq 1$ allora M è composto

Sia j il più piccolo indice con $a_j = 1 \pmod{M}$.

2. Se $0 < j$ e $a_{j-1} \neq M-1$ allora M è composto

b che soddisfa (1) o (2) è detto smart witness.

Rabin Theorem (1980)



Sia $M=2^k r+1$ con r dispari
Se M è composto allora almeno
 $3M/4$ di tutti i b nell'intervallo
 $1 < b < M$ sono **smart witnesses**.

Nessuna assunzione, la prova usa metodi
elementari.

Miller-Rabin Primality Testing

Input: intero dispari M ($2^{n-1} < M < 2^n$).

Ripeti 100 volte:

Scegli b a caso ($1 < b < M$).

Verifica se b è smart witness
(tempo $\text{poly}(n)$).

Se uno o più b sono smart witness, allora

" M è composto".

Altrimenti " M è primo".

Miller-Rabin Primality Testing

Proprietà dell'algoritmo:

- **Probabilistico** (usata nella scelta di b).
- Tempo esecuzione polin. in $n = \log M$.
- Se M è **primo** l'algoritmo fornisce **sempre** una risposta corretta
- Se M è composto l'algoritmo **può fornire una risposta errata**. Questo succede se tutte le scelte di b non sono witness; con 100 iterazioni
- **Probabilità di errore** $< (0.25)^{100} \lll 1$.

Primality Testing

In termini di classi di complessità l'algoritmo
implica **Composto** \square **RP**

RP=Random Poly Time, one sided error

Esercizi (facili)

1. Teorema: RP è contenuta in NP.
2. Algoritmo di Miller Rabin prova che

Composto \square **RP** prova anche che

Primo \square **RP?**

Fondamentale risultato Primality Testing è in P

Manindra Agrawal, Neeraj Kayal, Nitin
Saxena , India Inst. of Tech., Kanpur:

