Computações

Paula Cristina Valença

March 8, 2007

1 Algoritmo de Potenciação

Algoritmo 1 Algoritmo de Potenciação

Input: um inteiro n e $g \in G$.
Output: $g^n \in G$

```
y ← 1.
if n < 0 then [Inicializa]
  N ← −n, z ← g^{−1}.
else
  N ← n, z ← g
end if
while N ≠ 0 do [Terminou?]
  if N is odd then
    y ← z.y
  end if
  N ← ⌊N/2⌋, z ← z.z
end while
return y
```

2 Teste de Rabin-Miller (não primalidade)

Definition 2.1. *Seja N um inteiro positivo impar e a um inteiro. Escreva $N-1 = 2^t q$, com q impar. Diz-se que N é um* ***pseudo-primo forte com base*** **a** *se $a^q \equiv 1 \pmod{N}$ ou existe e tal que $a^{2^e q} \equiv -1 \pmod{N}$ e $0 \leq e < t$.*

No que se segue *Rand(N)* denota um gerador de números inteiros aleatórios no intervalo $\{1..N\}$.

3 Geração de um número primo aleatório

Suponha que tem um teste *ePrimo(N)* que identifica se N é ou não primo. Na prática podemos usar um teste *eProvavelmentePrimo(N)* do tipo de Miller-

Algoritmo 2 Teste de Rabin-Miller

Input: um inteiro ímpar $N \geq 3$

Output: Determina se N é um número não primo ou provavelmente primo (não prova primalidade)

$q \leftarrow N-1, t \leftarrow 0, c \leftarrow 20$
while q é par **do** [$N-1 = 2^t q$, q impar]
$q \leftarrow q/2, t \leftarrow t+1$
end while
while $c > 0$ **do** [escolhe um novo a]
$a \leftarrow$*Rand(N)* $(1 < a < N)$
$e \leftarrow 0, b \leftarrow a^q \mod N$
if $b \neq 1$ **then**
while $b \not\equiv \pm 1 \mod N$ and $e \leq t-2$ **do** $[0 < e < t]$
$b \leftarrow b^2 \mod N, e \leftarrow e+1$
end while
if $b \neq N-1$ **then**
return N não é primo
end if
end if
$c \leftarrow c-1$
end while
return N é provavelmente primo

Rabin durante o ciclo e reservar a prova para o fim, quando temos já bastante certeza de que N é de facto primo.

Algoritmo 3 Geração de um número primo aleatório

Input: o número de bits n.

Output: um número primo N com n bits

$a \leftarrow$*Rand(*2^n*)*
while !*ePrimo(a)* **do** [a é primo?]
$a \leftarrow$*Rand(*2^n*)*
end while
return a