

BCC241 - Projeto e Análise de Algoritmos

# Aula 3:

# Teste de Primalidade - Algoritmos Probabilísticos

BCC241/2012-2

DECOM/UFOP

2012/2 – 5<sup>o</sup>. Período

Anderson Almeida Ferreira

Material desenvolvido por Andréa

Iabrudi Tavares



# Princípios para Análise

- **Pior caso**
  - Genérico
  - Maior facilidade matemática
  - Alternativas: caso médio, *benchmarks*
- **Notação assintótica**
  - Ignorar constantes multiplicativas e termos de baixa ordem
    - Independência da arquitetura
    - Maior facilidade
  - Comportamento para grandes entradas

# Primalidade x Fatoração

- Dado um número  $N$  de  $n = \lceil \log N \rceil$  bits
  - Ele é primo?
  - Quais seus fatores primos?
- Base da segurança digital
- Origem do termo algoritmo

# Teste de primalidade

- $N$  é primo iff divisível apenas por 1 e  $N$ .
  - 2, 3, 5, 7, 11, 13, ...
  - 1299709, 15485863, 22801763489, ...
- Número infinito de número primos...
  - $2^{101}-1=2535301200456458802993406410751?$
- Perguntas que vamos responder hoje:
  - Como testar se um número é primo?

# Solução ingênua: menor divisor

```
function primo(N)
  for i= 2 .. N-1:
    if (N mod i == 0):
      return false;
  return true;
```

$N \bmod i == 0$  -  $\theta(n^2)$

# Existe um jeito mais eficiente de testar se um número é primo?

**Sim!** Se permitirmos uma chance infinitamente pequena de erro.

- Provar que o número é primo sem fatoração...  
Como?
- Teoria e Aleatoriedade!

# O teste de primalidade de Fermat

- Pequeno teorema de Fermat (1640)

*If  $p$  is prime, then for every  $1 \leq a < p$ ,*

$$a^{p-1} \equiv 1 \pmod{p}.$$

- Para melhor, tenho que usar o corolário



# Primeira tentativa

```
function primality( $N$ )
```

```
Input: Positive integer  $N$ 
```

```
Output: yes/no
```

```
Pick a positive integer  $a < N$  at random
```

```
if  $a^{N-1} \equiv 1 \pmod{N}$ :
```

```
    return yes
```

```
else:
```

```
    return no
```

$x^y \bmod N - \theta(n^3)$

Complexidade?

Correção?



# O teste de primalidade de Fermat

- Pequeno teorema de Fermat (1640)

*If  $p$  is prime, then for every  $1 \leq a < p$ ,*

$$a^{p-1} \equiv 1 \pmod{p}.$$



- Lema de Rabin (1977) (sem Carmichael)

**Lemma** *If  $a^{N-1} \not\equiv 1 \pmod{N}$  for some  $a$  relatively prime to  $N$ , then it must hold for at least half the choices of  $a < N$ .*

# Algoritmo

```
function primality2( $N$ )
```

```
Input: Positive integer  $N$ 
```

```
Output: yes/no
```

```
Pick positive integers  $a_1, a_2, \dots, a_k < N$  at random
```

```
if  $a_i^{N-1} \equiv 1 \pmod{N}$  for all  $i = 1, 2, \dots, k$ :
```

```
    return yes
```

```
else:
```

```
    return no
```

Correção:

Complexidade:

# Algoritmo Probabilístico

Algoritmo que utiliza escolhas **aleatórias**, mas cujo resultado é **correto** com **alta probabilidade** para **todas as entradas!**

Aleatoriedade é uma ferramenta/estratégia algorítmica poderosa!

# Correção

- Se falha, então  $N$  certamente não é primo.
- Mas e se não falha? Qual é a **probabilidade** de que um número composto seja identificado como primo?
  - Se Carmichael, sempre.
  - Senão: probabilidade  $2^{-k}$  (Lema)

Um erro de  $2^{-100}$  é aceitável?

# Limite superior, na realidade é melhor...

