Trabalho Individual sobre Números Primos

Números inteiros maiores que um são ditos primos se seus únicos divisores são 1 e o próprio número. Em segurança computacional utilizamos números primos em várias algoritmos e protocolos. Para isso, é necessário manter-se uma tabela de números primos (uma lista pré computada) ou, gerar tais números quando tais números necessários.

Não é simples a geração de números primos para uso em sistemas de segurança computacional. Normalmente, estamos interessados em números grandes, da ordem de grandeza de centenas de dígitos. No Brasil, por exemplo, para assinar documentos eletrônicos, você vai precisar ter chaves criptográficas geradas a partir de números primos de 2048 bits.

Uma forma de se gerar números primos é primeiro gerar um número aleatório ímpar (grande) e depois testá-lo para saber se é primo. Caso não seja, gera-se outro número aleatório até que seja primo.

Temos duas tarefas para fazer:

a. gerar números aleatórios;
b. testar se esse número gerado é primo.

Neste trabalho individual, vamos explorar técnicas para se gerar números pseudo-aleatórios e para se verificar a primalidade desses números.

1) Quanto ao Entregável

Você deve entregar no Moodle um ÚNICO arquivo PDF com os seguinte requisitos:

A. Uma seção para Números Aleatórios;
B. Uma seção para Números Primos;
C. Os códigos devem estar no PDF (incluído no PDF) e devidamente documentados (**comentados**);
D. Referências a cada um dos algoritmos.

LEMBRE-SE: Entregue um **único documento PDF**, contendo o relatório desse trabalho

individual (incluindo os códigos dos programas, tabelas, saídas, …)

2) Gerar Números Pseudo-aleatórios

Você deve escolher dois (2) dos seguintes algoritmos geradores de números pseudo-aleatórios e implementá-lo em Java (*Ou qualquer outra linguagem de programação, desde que justifique seu uso no relatório*) .

- Blum Blum Shub
- Complementary-multiply-with-carry
- Inversive congruential generator
- ISAAC (cipher)
- Lagged Fibonacci generator
- Linear congruential generator
- Linear feedback shift register
- Maximal periodic reciprocals
- Mersenne twister
- Multiply-with-carry
- Naor-Reingold Pseudorandom Function
- Park–Miller random number generator
- Well Equidistributed Long-period Linear
- Xorshift

Você deve executar cada um dos algoritmos escolhidos para:

- Gerar números pseudo-aleatórios grandes. Entende-se como grande, números de até 4096 bits. Experimente gerar números da seguintes ordens de grandeza: 40, 56, 80, 128, 168, 224, 256, 512, 1024, 2048, 4096 bits binários;
    - Caso não seja possível gerar números aleatórios de um determinado tamanho usando um dos algoritmos, você deve justificar;
    - Monte uma tabela explicitando o algoritmo, o tamanho do número e o tempo necessário para gerá-lo;
        - Exemplo:
        -

Algoritmo	Tamanho do Número	Tempo para gerar
Blum blum Shib	40	10 micro seg
xxx	56	

- O código implementado deve estar documentado. Dá-se preferência para documentar no próprio corpo do programa, na forma de comentários;
- Compare os dois algoritmos escolhidos;
- Faça uma análise de complexidade dos algoritmos;

3) Verificação de Primalidade

Miller-Rabin é um método clássico usado para se verificar se um número é ou não primo. Neste trabalho individual você deve experimentar com esse método e um outro (da lista abaixo, ou ainda outro qualquer, justificando a sua escolha). Descreva e implemente em Java ou outra linguagem de sua escolha.

Eis uma sugestão de métodos de teste de primalidade em adição ao Miller-Rabin:

- Teste de Primalidade de Fermat
- Solovay-Strassem
- Frobenius

Usando os algoritmos de geração de números pseudo-aleatórios e de teste de primalidade que você implementou,

Requisitos:

- Justifique a escolha do segundo método e compare-os;
- Gere uma tabela com números primos de 40, 56, 80, 128, 168, 224, 256, 512, 1024, 2048 e 4096 bits. Provavelmente, você vai ter dificuldade em gerar números dessas ordens de grandeza. Procure gerar o máximo possível;
- Escreva sobre as dificuldades encontradas e o tempo necessário em seu computador para gerar esses números;
- O código implementado deve estar documentado. Dá-se preferência para documentar no próprio corpo do programa, na forma de comentários;
- Faça uma análise de complexidade dos algoritmos;
- Quanto tempo do seu computador é necessário para se gerar números primos?