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 são 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 a milhares de dígitos binários. 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 de 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) Entregável

Você deve entregar no Moodle um ÚNICO arquivo PDF contendo contendo as seguintes seções:

A. Relatório dos requisitos (veja itens 2 e 3 abaixo)
B. Os códigos devem estar no PDF (incluído no PDF) e devidamente documentados (**comentados**);
C. Qualquer outro documento que deva ou precise ser entregue, necessários ao entendimento do trabalho. Procurar, no entanto, referenciar o material.

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

Requisitos:

- Deve ser possível 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;
- O código implementado deve estar documentado. Dá-se preferência para documentar no próprio corpo do programa, na forma de comentários;
- Justifique a escolha e compare os métodos;
- Usando os métodos, gere uma tabela de até 10 números pseudo-aleatórios de cada um dos tamanhos acima definidos;
- Faça uma análise de complexidade dos algoritmos;
- Quanto tempo do seu computador é necessário para se gerar um número pseudo-aleatório (considerando os diversos tamanhos)?

2) 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 qualquer

(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?