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.

Neste trabalho individual, vamos explorar técnicas para se gerar números primos.

1) Quanto ao Entregável

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

A. Uma seção para apresentação dos algoritmos de geração de Números Primos;
B. Os códigos devem estar no PDF (incluído no PDF) e devidamente documentados (**comentados**);
C. 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) 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 dois outros (da lista abaixo), ou ainda outros quaisquer, justificando a sua escolha. Descreva e implemente em Pyhton 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
- Lucas Primality Test
- Pocklington–Lehmer primality test
- Proth's theorem
- Pépin's test
- Quadratic Frobenius Test (QFT)
- AKS primality test
- Adleman–Pomerance–Rumely primality test
- Baillie–PSW primality test

Use os algoritmos de geração de números pseudo-aleatórios quando necessário para gerar números aleatórios.

Requisitos:

- Justifique a escolha dos métodos 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;
    - Monte uma tabela para mostrar quanto tempo levou para gerar cada número de cada tamanho por cada um dos algoritmos;
- 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?