Teste de Primalidade Miller-Rabin.

Autores

- Thiago Senhorinha Rose (12100774)

O que é?

Miller-Rabin é algoritmo para teste de primalidade, ou seja, dado um número, ao final do algoritmo este será categorizado em primo ou composto. Vale lembrar que se um número não passar no teste, é certo que ele é um número composto porém se o resultado der primitude não é certo que é primo.

Esse algoritmo tem uma grande aplicação no campo da segurança da computação em que grandes números primos são essenciais para segurança dos algoritmos.

Algoritmo

```
Input: n > 3, an odd integer to be tested for primality;
Input: k, a parameter that determines the accuracy of the test
Output: composite if n is composite, otherwise probably prime
write n − 1 as 2s·d with d odd by factoring powers of 2 from n − 1
WitnessLoop: repeat k times:
   pick a random integer a in the range [2, n − 2]
   x ← ad mod n
   if x = 1 or x = n − 1 then do next WitnessLoop
   repeat s − 1 times:
      x ← x2 mod n
      if x = 1 then return composite
      if x = n − 1 then do next WitnessLoop
   return composite
return probably prime
```

Implementação em Java

```
import java.math.BigInteger;
import java.util.Random;
```

```
public class MillerRabin {

    private static final BigInteger ZERO = BigInteger.ZERO;
    private static final BigInteger ONE = BigInteger.ONE;
    private static final BigInteger TWO = BigInteger.valueOf(2);

    public static boolean isProbablyPrime(BigInteger n, int precision) {
        if (n.compareTo(ONE) <= 0 || n.mod(TWO) == ZERO) {
            return false;
        } else if (n.intValue() == 3) {
            return true;
        }
        Random random = new Random();

        BigInteger nLessOne = n.subtract(ONE);
        BigInteger x;
        int s = 0;
        BigInteger d = nLessOne;
        BigInteger randomInteger;
        while (d.mod(TWO).equals(ZERO)) {
            s++;
            d = d.divide(TWO);
        }

        for (int k = 0; k <= precision; k++) {
            randomInteger = BigInteger.valueOf(random.nextInt(n.intValue() - 4) +
 2);
            x = randomInteger.pow(d.intValue()).mod(n);
            if (x.compareTo(ONE) == 0 || x.compareTo(nLessOne) == 0) {
                continue;
            }
            for (int r = 0; r < s - 1; r++) {
                x = x.pow(TWO.intValue()).mod(n);
                if (x.compareTo(ONE) == 0) {
                    return false;
                }
                if (x.compareTo(nLessOne) == 0) {
                    continue;
                }
            }
            return false;
        }
        return true;
    }
}
```

Resolução em Papel para número 11 com precisão 3

Número 11, precisão 3

$2^1 \cdot 5$

1° - n-1 as $2^s \cdot d$
2° - Repeat K Times
3° Pik a random Int
4° $x \leftarrow a^d \bmod n$

$K = 3$ (Precisão)
$a = 3$
$x = 3^5 \bmod 11$
$x = 1$
TRUE

5° Se $x = 1$ ou $x = n-1$

3°
4°
5°

$a = 4$
$x = 4^5 \bmod 11$
$x = 1$
TRUE

3°
4°
5°

$a = 5$
$x = 5^5 \bmod 11$
$x = 1$
TRUE

2°

FIM DO LAÇO

7°

RETORNA TRUE (Primo detectado)