Tarea Automatizada #1

Diseño de la Aplicación

Jhon Sebastian Rojas Rodriguez

El diseño de la aplicación se realizó entorno a la implementación del test de Miller-Rabin en el lenguaje de programación Python, siguiendo el pseudocódigo en el Libro *Introduction to Algorithms[1]* y en la escritura de una función para generar números aleatorios de una longitud dada para luego ser probados con el test. El algoritmo para realizar el test de Miller-Rabin a un número dado es*[2]*:

Sea $n>1$ un número impar, y sean k el número natural y m el número impar para los que se cumple que $n-1=2^k m$ y a un entero escogido aleatoriamente entre 2 y n – 2.

Paso 0:

Calcular $b_0 \equiv a^m (mod\, n)$, en caso de que $b_0 \equiv \pm 1 (mod\, n)$ se concluye que n es probable primo.

Paso $1 \leq i < k-1$:

Calcular $b_i = b_{i-1}^2 (mod\, n)$

Si $b_i = -1 (mod\, n)$ se concluye que n es probable primo y se continua.

Si $b_i = 1 (mod\, n)$ se concluyen que n no es primo y se detiene el algoritmo.

Paso k – 1:

Si $b_{k-1} = -1 (mod\, n)$ el test termina y se concluye que n es probable primo, en cualquier otro caso se concluye que n no es primo.

El siguiente Pseudocódigo implementa el algoritmo ():

```
WITNESS(a, n)
let t and u be such that t ≥ 1, u is odd, and n − 1 = 2^t u
x_0 = MODULAR-EXPONENTIATION(a, u, n)
for i = 1 to t
    x_i = x_{i−1}^2 mod n
    if x_i == 1 and x_{i−1} ≠ 1 and x_{i−1} ≠ n − 1
        return TRUE
if x_t ≠ 1
    return TRUE
return FALSE
```

```
MILLER-RABIN(n, s)
for j = 1 to s
    a = RANDOM(1, n − 1)
    if WITNESS(a, n)
        return COMPOSITE          // definitely
return PRIME                      // almost surely
```

A partir de la implementación de lo anterior y una función para evaluar números aleatorios con el test se generó el código fuente de la aplicación.

Bibliografia

[1] Introduction to algorithms / Thomas H. Cormen . . . [et al.].—3rd ed. p 968 – 975.

[2] Rabin, Michael (1980). «Probabilistic algorithm for testing primality». *Journal of Number Theory*: 128-138