PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

Criptografía y Seguridad Computacional - IIC3253
Ayudantía 8
Alexander Pinto

Test de primalidad

1. Realice una demostración práctica del test de primalidad visto en clases para el número $n = 137$, con una probabilidad de error menor a 0.02.

   Utilizaremos la noción de testigo para probar la primalidad de n. En clases se vió que el test tenía una probabilidad de error igual a $(\frac{1}{2})^k$, donde k es el número de "iteraciones" de nuestro test. Entonces si tomamos $k = 6$, tenemos un $error < 0.02$.

   Nuestro primer intento se basa en el pequeño teorema de Fermat. Sea:

   $$W_n = \{a \in \{1, ..., n-1\} \mid a^{n-1} \equiv 1 \mod n\}$$

   , si n es primo entonces $|W_n| = n - 1$.

   Luego, elegimos $a_1, ..., a_k$ en $\{1, ..., n-1\}$ de manera aleatoria, Por ejemplo $a = \{3, 5, 7, 11, 13, 17\}$, y probamos $b_i = a_i^{n-1} \mod n$ para cada $i \in a$. Si $b_i \neq 1$, nuestro n será compuesto y detenemos nuestro algoritmo, caso contrario continuamos con el test. Por ejemplo, tomando $a_i = 11$:

   $$b_i = 11^{136} \mod 137 = 1$$

   Nuestra segunda noción de testigo nos indica que si n es un número impar y $a \in \{1, ..., n-1\}$ entonces $a^{\frac{n-1}{2}} \equiv \pm 1 \mod n$. Además de esto, si n es primo se debe cumplir que:

   $$|W_n^+| = |W_n^-| = \frac{n-1}{2} \quad \text{,donde:}$$
   $$W_n^+ = \{a \in \{1, ..., n-1\} \mid a^{\frac{n-1}{2}} \equiv 1 \mod n\}$$
   $$W_n^- = \{a \in \{1, ..., n-1\} \mid a^{\frac{n-1}{2}} \equiv -1 \mod n\}$$

   Por ejemplo, tomando $a_i = 11$:

   $$11^{136/2} \equiv 1 \mod n$$

   Aquí debemos tomar en cuenta que esto no se cumple para todo número primo impar. Por lo tanto, nuestro algoritmo exige que al menos exista un $W_n^- \equiv -1 \mod n$. Por ejemplo, tomando $a_i = 3$, tenemos:

   $$3^{136/2} \equiv -1 \mod n$$

Estas dos nociones deben repetirse para cada uno de nuestros a_i elegidos. La siguiente tabla resume este cálculo.

a_i	3	5	7	11	13	17
$b_i = a_i^{136} \mod 137$	1	1	1	1	1	1
$a^{\frac{136}{2}} \equiv \pm 1 \mod 137$	-1	-1	1	1	-1	1

Nuestra tercera noción nos pide distinguir cuando n no es primo porque es potencia no trivial de otro número, es decir $n = a^b$, con $b \geq 2$. Una forma de realizar esto eficientemente, es por medio de una búsqueda binaria de la base por cada posible exponente de acuerdo al siguiente algoritmo. Realizado este test podemos indicar que $n = 137$ es un número primo con una probabilidad de error menor a 0.02.

Algorithm 1 Comprobar si n es potencia de otro número

Input: n

Output: True, si n es potencia de otro número, c.c. False.

```
power ← 2
avoid ← {}                                   ▷ un conjunto de números compuestos
while 2^power ≤ n do                         ▷ mientras la mínima base elevada a power sea menor igual que n
    if power no está en avoid then           ▷ sólo considerar exponentes primos
        up ← 2                               ▷ límite superior de intervalo
        low ← 1                              ▷ límite inferior de intervalo
        i ← 2
        while up^power < n do                ▷ mientras no encontremos el límite superior de intervalo
            low ← up
            up ← up^2                        ▷ la búsqueda es de forma binaria creciente
            avoid = avoid ∪ {i × power}      ▷ evitamos exponentes no primos
            i ← i + 1
        end while
        while low ≠ up do                    ▷ búsqueda binaria en el intervalo definido
            mid = (up + low)/2
            if mid^power < n then
                low ← mid + 1
            else if mid^power > n then
                up ← mid
            else return True                 ▷ mid^power = n, n es potencia de otro número
            end if
        end while
        if up^power = n then
            return True                      ▷ caso borde, n es potencia de otro número
        end if
    end if
    power ← power + 1                        ▷ continua el bucle con el siguiente exponente
end while
return False                                 ▷ n no es potencia de otro número
```

2. Como parte del enunciado para el desarrollo de la Tarea 2, usted cuenta con el código de algunas funciones auxiliares. Explique en detalle las funciones `_is_probably_prime` y `es_potencia`.

   De la revisión del código podemos notar lo siguiente:

   - La función `es_potencia` es una implementación del Algoritmo 1 mostrado en la pregunta anterior.
   - La función `_is_probably_prime` contiene los pasos descritos en la pregunta anterior.
     - Comprueba el caso borde $n = 2$.
     - Se asegura que n no sea par.
     - Comprueba que n no sea potencia trivial de otro número.
     - Repite las dos primeras nociones en k iteraciones para diferentes a_i tomados al azar.
     - Se asegura que al menos se encuentre un $W_n^- \equiv -1 \mod n$.
     - Cumplidas todas estas condiciones retorna `True` si n es un probable primo, caso contrario retorna `False` n es compuesto.

3. Realice una demostración práctica del test de primalidad para los números:

   $P = 113863597804193638684746233303866275862959799351116319454932839076728438946896147861014045230382038476640030287985809455541219549331656514489547597100776866463781814271673557022224794131215064237956990930271588711457443015898310055178700526009643888726738585998120562566732806649886086397912653034050178893859$

   $Q = 115838909791943748922987753851050334560193343608838943167465762721872484238907283305609260452195433132247935218033024924981579405051273360371753983885753654957234722064446587754445946080621943088160497578956151867063126988652321672161448439688550488904372167035366626051841175064590746743483812434310620662793$

   Reporte el tiempo de ejecución.

   Utilizando la implementación del test de primalidad provisto, obtenemos respuestas positivas para P y Q en un tiempo $t \approx 0.4s$. Es decir, P y Q son probablemente primos.