Pontificia Universidad Católica de Chile
Facultad de Ingenieria
Departamento de Ciencias de la Computación
IIC2283 - Diseño y Análisis de Algoritmos

Ayudantía 10

5 de Noviembre, 2018

Dados números naturales a, b, n con $n > 0$, decimos que b es una raíz cuadrada de a en modulo n si $b^2 \equiv a \mod n$. Por ejemplo, 2 y 9 son raíces cuadradas de 4 en módulo 11 puesto que $2^2 \equiv 4 \mod 11$ y $9^2 \equiv 4 \mod 11$. Dado un número primo impar p y un número $a \in \{1, ..., p-1\}$, en clases se demostró que a tiene raíz cuadrada en módulo p si y sólo si

$$a^{\frac{p-1}{2}} \equiv 1 \mod p$$

Por ejemplo, tenemos que $4^5 = 1024$ y $1024 \equiv 1 \mod 11$, por lo que 4 tiene raíz cuadrada módulo 11. Por el contrario, $2^5 = 32$ y $32 \equiv -1 \mod 11$, por lo que 2 no tiene raíz cuadrada en módulo 11.

En este ejercicio se va a desarrollar un algoritmo para calcular raíces cuadradas en módulo un número primo impar p. La idea del algoritmo está basada en los siguientes pasos, los cuales usted tiene que generalizar y demostrar que son correctos.

Sea $a \in \{1, ..., p-1\}$ tal que $a^{\frac{p-1}{2}} \equiv 1 \mod p$, lo cual nos asegura que a tiene raíz cuadrada en módulo p. Dado que p es impar, tenemos que $p - 1 = 2^t \cdot s$, donde $t \geq 1$ y s es un número impar. Además, dado que a tiene raíz cuadrada en módulo p, concluimos que

$$a^{2^{t-1} \cdot s} \equiv 1 \mod p$$

Si $t = 1$, entonces tenemos que $a^s \equiv 1 \mod p$ En este caso tenemos los ingredientes necesarios para calcular una raíz cuadrada de a:

$$\begin{aligned} a^s \equiv 1 \mod p &\implies a^{s+1} \equiv a \mod p \\ &\implies \left(a^{\frac{s+1}{2}}\right)^2 \equiv a \mod p \end{aligned}$$

Tenemos que $a^{\frac{s+1}{2}}$ es una raíz cuadrada de a en módulo p, la cual está bien definida puesto que $\frac{s+1}{2}$ es un número natural ya que s es impar.

Nos queda considerar el caso en que $t > 1$. Suponemos en este caso que tenemos un número natural γ tal que

$$\gamma^{\frac{p-1}{2}} \equiv -1 \mod p$$

Vale decir, γ no tiene raíz cuadrada en módulo p. En este caso es posible demostrar que existe $k_1 \in \{0, 1\}$ tal que

$$(a^s)^{2^{t-2}} \cdot (\gamma^s)^{2^{t-1} \cdot k_1} \equiv 1 \mod p$$

Si $t = 2$, nuevamente tenemos una forma de calcular una raíz cuadrada de a:

$$\begin{aligned} (a^s)^{2^{t-2}} \cdot (\gamma^s)^{2^{t-1} \cdot k_1} \equiv 1 \mod p &\implies (a^s) \cdot (\gamma^s)^{2 \cdot k_1} \equiv 1 \mod p \\ &\implies (a^{s+1}) \cdot (\gamma^s)^{2 \cdot k_1} \equiv a \mod p \\ &\implies (a^{\frac{s+1}{2}})^2 \cdot (\gamma^s)^{2 \cdot k_1} \equiv a \mod p \\ &\implies (a^{\frac{s+1}{2}})^2 \cdot (\gamma^{s \cdot k_1})^2 \equiv a \mod p \\ &\implies (a^{\frac{s+1}{2}} \cdot \gamma^{s \cdot k_1})^2 \equiv a \mod p \end{aligned}$$

Así, $a^{\frac{s+1}{2}} \cdot \gamma^{s \cdot k_1}$ es una raíz cuadrada de a en módulo p que está bien definida porque s es un número impar.

Pero nuevamente nos puede pasar que no se cumpla la condición y que $t > 2$. En este caso es posible demostrar que existe $k_2 \in \{0, 1\}$ tal que

$$(a^s)^{2^{t-3}} \cdot (\gamma^s)^{2^{t-2} \cdot k_1} \cdot (\gamma^s)^{2^{t-1} \cdot k_2} \equiv 1 \mod p$$

Notar que aquí estamos considerando el mismo valor γ utilizado en la iteración anterior. Si $t = 3$, nuevamente tenemos una forma de calcular una raíz cuadrada de a:

$$\begin{aligned}
(a^s)^{2^{t-3}} \cdot (\gamma^s)^{2^{t-2} \cdot k_1} \cdot (\gamma^s)^{2^{t-1} \cdot k_2} \equiv 1 \mod p &\implies (a^s) \cdot (\gamma^s)^{2 \cdot k_1} \cdot (\gamma^s)^{4 \cdot k_2} \equiv 1 \mod p \\
&\implies (a^{s+1}) \cdot (\gamma^s)^{2 \cdot k_1} \cdot (\gamma^s)^{4 \cdot k_2} \equiv a \mod p \\
&\implies (a^{\frac{s+1}{2}})^2 \cdot (\gamma^s)^{2 \cdot k_1} \cdot (\gamma^s)^{4 \cdot k_2} \equiv a \mod p \\
&\implies (a^{\frac{s+1}{2}})^2 \cdot (\gamma^{s \cdot k_1})^2 \cdot (\gamma^{2 \cdot s \cdot k_2})^2 \equiv a \mod p \\
&\implies (a^{\frac{s+1}{2}} \cdot \gamma^{s \cdot k_1} \cdot \gamma^{2 \cdot s \cdot k_2})^2 \equiv a \mod p
\end{aligned}$$

Nuevamente, $a^{\frac{s+1}{2}} \cdot \gamma^{s \cdot k_1} \cdot \gamma^{2 \cdot s \cdot k_2}$ es una raíz cuadrada de a en módulo p que está bien definida porque s es un número *impar*. Finalmente, si la condición no se cumple y tenemos que $t > 3$, el proceso puede continuar de la misma forma hasta que el primer término de la multiplicación sea a^s.

1. Formalice la idea mostrada anteriormente como un algoritmo aleatorizado de Las Vegas. En particular, utilizado la notación desarrollada en clases de un pseudocódigo para una función `RaízCuadrada`(p, a, k), la cual recibe cómo parámetros un número primo impar p, un número natural $a \in \{1, ..., p-1\}$ y un número natural $k \geq 1$, y retorna uno de los siguientes valores: si a no tiene raíz cuadrada en módulo p entrega `sin_raíz_cuadrada`, y si a tiene raíæ cuadrada en módulo p entonces con probabilidad $(1 - \frac{1}{2^k})$ entrega un número $b \in \{1, ..., p-1\}$ tal que $b^2 \equiv a \mod p$, y con probabilidad $\frac{1}{2^k}$ entrega `sin_resultado`.
2. Formalice y demuestre las propiedades necesarias para establecer que el algoritmo dado en (1) es correcto, vale decir, las propiedades que muestran que si el algoritmo entrega un valor distinto de `sin_resultado`, entonces ese valor es correcto.
3. Analice la complejidad del algoritmo considerando como operaciones básicas a contar la suma, resta, multiplicación, división y cálculo del resto para números enteros. Además, demuestre que la probabilidad de que el algoritmo entregue `sin_resultado` es $\frac{1}{2^k}$.

Respuestas

Pregunta 1

Algunos comentarios:

- La función **EXP**(a, b, n) calcula a^b mod n utilizando exponenciación modular rápida, como se vio en clases. Toma $O(log(b))$ multiplicaciones.
- Los índices parten en 1 (i.e. $L[1]$ es el primer elemento de L) para ser consistente con la notación de clases
- **Append** es una función que recibe dos listas y concatena la segunda a la primera (modificando la primera lista)
- $[i]$ es una lista con un único elemento i. Similarmente, [ ] es una lista vacía.
- Los **for** son inclusivos en el límite inferior y superior (**for** $i := j$ **to** k **do** itera desde j hasta k, con j y k incluidos)

Con esto, el procedimiento **RaizCuadrada** es como sigue:

RaizCuadrada(p, a, k)
1: $test\_exp := (p-1)/2$
2: **if EXP**$(a, text\_exp, p) \neq 1$ **then return** $sin\_raiz\_cuadrada$
3: $s := p - 1$
4: $t := 0$
5: **while** $s \bmod 2 = 0$ **do**
6: $s := s/2$
7: $t := t + 1$
8: **if** $t = 1$ **then return EXP**$(a, (s+1)/2, p)$
9: $found := 0$
10: $counter := 0$
11: **while** $found = 0$ **and** $counter < k$ **do**
12: $\gamma :=$ número aleatorio elegido de manera uniforme desde $\{1, ..., p-1\}$
13: **if EXP**$(\gamma, test\_exp, p) \neq 1$ **then** $found := 1$
14: $counter := counter + 1$
15: **if** $found = 0$ **then return** $sin\_resultado$
16: $a_s = [\textbf{EXP}(a, s, p)]$
17: $\gamma_s = [\textbf{EXP}(\gamma, s, p)]$
18: **for** $i := 1$ **to** t **do**
19: **Append**$(a_s, [\textbf{EXP}(a_s[i], 2, p)])$
20: **Append**$(\gamma_s, [\textbf{EXP}(\gamma_s[i], 2, p)])$
21: $k := [\,]$
22: **for** $i := 1$ **to** $t - 1$ **do**
23: $r := a_s[t - i]$
24: **for** $j := 2$ **to** i **do**
25: **if** $k[i + 1 - j] = 1$ **then**
26: $r := (r \cdot \gamma_s[t - j + 1]) \bmod p$
27: **if** $r = 1$ **then**
28: **Append**$(k, [0])$
29: **else**
30: **Append**$(k, [1])$
31: $value :=$ **EXP**$(a, (s+1)/2, p)$
32: **for** $i := 1$ to $t - 1$ **do**
33: **if** $k[i] = 1$ **then**
34: $value := (value \cdot \gamma_s[i]) \bmod p$
35: **return** $value$

Pregunta 2

El algoritmo retorna algo distinto de *sin_resultado* en 3 partes:

1. Se retorna *sin_raiz_cuadrada* en la línea 2: Si retorna *sin_raiz_cuadrada*, esto es porque se cumplió la condición $a^{\frac{p-1}{2}} \not\equiv 1 \mod p$. Como se vio en clases, a tiene raíz cuadrada modulo p si y solo si $a^{\frac{p-1}{2}} \equiv 1 \mod p$. Por lo tanto, si se cumple que $a^{\frac{p-1}{2}} \not\equiv 1 \mod p$ efectivamente a no tiene raíz cuadrada en módulo p, por lo que el valor retornado es correcto.

2. Se retorna en la línea 8: Si $t = 1$ entonces se retorna $\mathbf{EXP}(a, (s+1)/2, p) = a^{\frac{s+1}{2}} \mod p$. En este caso se tiene que $p - 1 = 2 \cdot s$, y por lo tanto

$$\begin{aligned}\left(a^{\frac{s+1}{2}} \mod p\right)^2 &\equiv \left(a^{\frac{s+1}{2}}\right)^2 \mod \text{p} \\ &\equiv \left(a^{s+1}\right) \mod \text{p} \\ &\equiv \left(a^{\frac{2\cdot s}{2}+1}\right) \mod \text{p} \\ &\equiv \left(a^{\frac{p-1}{2}+1}\right) \mod \text{p} \\ &\equiv \left(a^{\frac{p-1}{2}}\right) \cdot a \mod \text{p} \\ &\equiv 1 \cdot a \mod \text{p} \\ &\equiv a \mod \text{p}\end{aligned}$$

   Donde $a^{\frac{p-1}{2}} \equiv 1 \mod p$ pues de lo contrario se habría retornado *sin_raiz_cuadrada* al comienzo del algoritmo.

   Así, se tiene que $a^{\frac{s+1}{2}} \mod p$ es efectivamente una raíz cuadrada de a en módulo p, por lo que el valor retornado es correcto.

3. Se retorna en la línea 35: Dado que no se retornó *sin_resultado* en la linea 15, γ es tal que $\gamma^{\frac{p-1}{2}} = \gamma^{s \cdot 2^{t-1}} \equiv -1 \mod p$. Además, $t >= 2$.

   La clave del algoritmo son los k_i mencionados en el enunciado: ¿Cómo sabemos que efectivamente existen $k_i \in \{0, 1\}$ tales que cumplan con las congruencias planteadas, y cómo los calculamos? Una vez obtenidos los k_i, obtener la raiz cuadrada es directo. Primero se demostrará la existencia de $k_1, ..., k_{t-1}$, luego se demostrará como obtener la raíz modular utilizando estos k_i y por último se demostrará que el algoritmo llega a los valores correctos de $k_1, ..., k_{t-1}$ y calcula la raíz modular.

Proposición: Para todo $i \in \{1, ..., t-1\}$ existen $k_1, ..., k_i$ tales que

$$(a^s)^{2^{t-i-1}} \cdot (\gamma^s)^{2^{t-i} \cdot k_1} \cdot (\gamma^s)^{2^{t-(i-1)} \cdot k_2} \cdot ... \cdot (\gamma^s)^{2^{t-1} \cdot k_i} \equiv 1 \bmod p$$

Demostración: Se hará inducción sobre i.

- Caso Base: $i = 1$

  Sabemos que $a^{s \cdot 2^{t-1}} = a^{\frac{p-1}{2}} \equiv 1 \bmod p$. Así,

  $$\begin{gathered} a^{s \cdot 2^{t-1}} \equiv 1 \bmod p \\ a^{s \cdot 2^{t-1}} - 1 \equiv 0 \bmod p \\ \left(a^{s \cdot 2^{t-2}} - 1\right) \cdot \left(a^{s \cdot 2^{t-2}} + 1\right) \equiv 0 \bmod p \end{gathered}$$

  Como p es un número primo, se concluye que $a^{s \cdot 2^{t-2}} - 1 \equiv 0 \bmod p$ o $a^{s \cdot 2^{t-2}} + 1 \equiv 0 \bmod p$. Esto implica que $a^{s \cdot 2^{t-2}} \equiv 1 \bmod p$ o $a^{s \cdot 2^{t-2}} \equiv -1 \bmod p$.

  Si $a^{s \cdot 2^{t-2}} \equiv 1 \bmod p$ estamos listos: basta elegir $k_1 = 0$, con lo que

  $$(a^s)^{2^{t-2}} \cdot (\gamma^s)^{2^{t-1} \cdot k_1} = (a^s)^{2^{t-2}} \cdot (\gamma^s)^{2^{t-1} \cdot 0} = (a^s)^{2^{t-2}} \equiv 1 \bmod p$$

  Por el otro lado, si $a^{s \cdot 2^{t-2}} \equiv -1 \bmod p$ basta elegir $k_1 = 1$, con lo que

  $$(a^s)^{2^{t-2}} \cdot (\gamma^s)^{2^{t-1} \cdot k_1} = (a^s)^{2^{t-2}} \cdot (\gamma^s)^{2^{t-1}} \equiv -1 \cdot -1 = 1 \bmod p$$

- Hipótesis de Inducción: Se cumple para $1 \leq i < t - 1$
- Tesis de Inducción: Se debe demostrar la proposición para $i + 1$

  Por la hipótesis de inducción tenemos que

  $$\begin{gathered} (a^s)^{2^{t-i-1}} \cdot (\gamma^s)^{2^{t-i} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-1} \cdot k_i} \equiv 1 \bmod p \\ (a^s)^{2^{t-i-1}} \cdot (\gamma^s)^{2^{t-i} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-1} \cdot k_i} - 1 \equiv 0 \bmod p \\ \left((a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} - 1\right) \cdot \\ \left((a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} + 1\right) \equiv 0 \bmod p \end{gathered}$$

  Al igual que antes, como p es un número primo tenemos dos casos:

  (a) $(a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} \equiv 1 \bmod p$

  (b) $(a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} \equiv -1 \bmod p$

  En el caso a) basta tomar $k_{i+1} = 0$:

  $$\begin{gathered} (a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} \cdot (\gamma^s)^{2^{t-1} \cdot k_{i+1}} \\ = (a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} \cdot 1 \equiv 1 \bmod p \end{gathered}$$

  En el caso b) basta tomar $k_{i+1} = 1$:

  $$\begin{gathered} (a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} \cdot (\gamma^s)^{2^{t-1} \cdot k_{i+1}} \\ = (a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} \cdot (\gamma^s)^{2^{t-1}} \equiv -1 \cdot -1 = 1 \bmod p \end{gathered}$$

  Esto concluye la demostración de la proposición. ■

Tomando $i = t-1$, la proposición anterior nos dice que existen $k_1, ..., k_{t-1}$ tales que

$$a^s \cdot (\gamma^s)^{2^1 \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-1} \cdot k_{t-1}} \equiv 1 \bmod p$$

De esto se concluye que $a^{\frac{s+1}{2}} \cdot \gamma^{s \cdot 2^0 \cdot k_1} \cdot ... \cdot \gamma^{s \cdot 2^{t-2} \cdot k_{t-1}}$ es una raíz de a en modulo p. En efecto,

$$\begin{aligned}
\left(a^{\frac{s+1}{2}} \cdot \gamma^{s \cdot 2^0 \cdot k_1} \cdot ... \cdot \gamma^{s \cdot 2^{t-2} \cdot k_{t-1}}\right)^2 &= a^{s+1} \cdot \gamma^{s \cdot 2^1 \cdot k_1} \cdot ... \cdot \gamma^{s \cdot 2^{t-1} \cdot k_{t-1}} \\
&= a \cdot a^s \cdot \gamma^{s \cdot 2^1 \cdot k_1} \cdot ... \cdot \gamma^{s \cdot 2^{t-1} \cdot k_{t-1}} \\
&= a \cdot a^s \cdot (\gamma^s)^{2^1 \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-1} \cdot k_{t-1}} \\
&\equiv a \cdot 1 \bmod p \\
&\equiv a \bmod p
\end{aligned}$$

Esto demuestra que existen los $k_1, ..., k_{t-1}$ buscados y muestra como obtener una raíz de a en modulo p utilizando $a, \gamma, k_1, ..., k_{t-1}$. Falta demostrar que el algoritmo calcula efectivamente los valores correctos.

En las líneas 16 a 20 del algoritmo, lo que se hace es precomputar los valores de $a^{s \cdot 2^i} \bmod p$ y $\gamma^{s \cdot 2^i} \bmod p$ para todo i desde 0 hasta t. En $a_s[i]$ queda guardado el valor $a^{s \cdot 2^{i-1}} \bmod p$, y es análogo para γ. Además en la línea 21 se inicializa la lista donde se guardarán los k_i (en $k[i]$ se encuentra k_i). Es en el **for** de la linea 22 donde se calcula el valor de $k_1, ..., k_{t-1}$. El procedimiento es muy similar al paso inductivo realizado en la demostración de la página anterior. Así, utilizando nuevamente inducción se puede demostrar que el el cálculo de los k_i es correcto:

En la primera iteracion del for ($i = 1$), $r = a^{s \cdot 2^{t-2}} \bmod p$ (no se entra al **for** de la linea 24). Luego, si $r = 1$, entonces $k_1 = 0$. Por el otro lado, si $r = -1$ entonces $k_1 = 1$. Este valor para k_1 es correcto por el argumento entregado en la página anterior en la demostración del caso base.

Luego, si asumimos que los valores $k_1, ..., k_i$ fueron calculados correctamente, en la iteración $i+1$ del **for** de la linea 21 (cuando se calcula k_{i+1}) se hace lo siguiente: Se calcula (haciendo un juego con los índices)

$$r = (a^s)^{2^{t-(i+1)-1}} \cdot (\gamma^s)^{2^{t-(i+1)} \cdot k_1} \cdot ... \cdot (\gamma^s)^{2^{t-2} \cdot k_i} \bmod p$$

Luego, si $r = 1$, entonces $k_{i+1} = 0$. Por el otro lado, si $r = -1$ entonces $k_{i+1} = 1$. Este valor para k_{i+1} es correcto por el argumento entregado en la página anterior en la demostración del paso inductivo (tesis de inducción).

Así, los valores $k_1, ..., k_{t-1}$ calculados por el algoritmo son correctos.

Ahora que tenemos $k_1, ..., k_{t-1}$, en las líneas 31 a 34 el algoritmo calcula el valor final de retorno de manera directa: inicializa *value* en $a^{\frac{s+1}{2}} \bmod p$, y luego para cada $k_i = 1$ multiplica *value* por $\gamma^{s \cdot 2^{i-1}}$ y saca módulo p. Así, al llegar a la línea 35 se retorna *value* con un valor igual a

$$a^{\frac{s+1}{2}} \cdot \gamma^{s \cdot 2^0 \cdot k_1} \cdot ... \cdot \gamma^{s \cdot 2^{t-2} \cdot k_{t-1}} \bmod p$$

que como ya vimos es efectivamente una raiz de a en módulo p. Por lo tanto, en este caso el valor retornado también es correcto.

Así, cada vez que el algoritmo retorna un valor distinto de *sin_resultado* este valor es correcto. ■

Pregunta 3

Calculo de Complejidad del Algoritmo

El mejor caso del algoritmo es que el número no tenga raíz. En ese caso se hace 1 resta, 1 división y se ejecuta 1 vez **EXP**, lo que tiene costo $O(log(p))$ multiplicaciones. Finalmente el costo total en el mejor caso es $O(log(p))$ operaciones.

El peor caso es más interesante. Este se da cuando el último intento de calcular γ tiene éxito, y luego se tienen que calcular todos los k_i para poder llegar a la raíz modular. Los distintos pasos del algoritmo tienen el siguiente costo:

- En la línea 2 se ejecuta una vez **EXP**, lo que tiene costo $O(log(p))$
- En las líneas 3 a 7 se calculan s y t, lo que nuevamente tiene costo $O(log(p))$
- No se entra al **if** de la linea 8, pues estamos en el peor caso
- En las lineas 9 a 14 se encuentra el número γ. Asumiendo que tomar un número aleatorio uniforme en un rango no toma más que tiempo logarítmico en el rango, tomar 1 valor de γ y comprobar que $\gamma^{\frac{p-1}{2}} \bmod p \neq 1$ toma $O(log(p))$. En el peor caso hay que realizar las k iteraciones para encontrar γ, por lo que esta parte toma tiempo $O(k \cdot log(p))$.
- En las líneas 16 a 20 se precomputan los valores de $a^{s \cdot 2^i} \bmod p$ y $\gamma^{s \cdot 2^i} \bmod p$. El proceso completo de calcular $a^s \bmod p$ y luego elevar al cuadrado t veces (guardando los resultados intermedios en a_s) toma en total $O(log(p))$ pasos. Para γ se sigue el mismo proceso, que también toma $O(log(p))$ pasos.
- En las lineas 22 a 30 se calculan los k_i. El peor caso se da cuando todos los k_i son iguales a 1, pues en ese caso hay que realizar el máximo número de multiplicaciones y calculo de restos. Obviando las operaciones sobre los índices i y j (no cambian el orden final), para calcular r en la iteración i en el peor caso se deben hacer $i-1$ multiplicaciones e $i-1$ calculo de restos para un total de $2 \cdot i - 2$ operaciones. Se hacen $t-1$ iteraciones, donde t está acotado superiormente por $log(p)$.

  Así, en total en el **for** de la línea 22 se hacen $0 + 2 + 4 + 6 + ... + 2 \cdot t - 4 = \frac{(t-1)(2 \cdot t-4)}{2}$ operaciones en el peor caso, lo que está acotado por $O(log(p)^2)$.
- En las líneas 31 a 34 se calcula el valor de retorno final del algoritmo. La exponenciación rápida de la linea 31 toma $O(log(p))$, y el **for** de la línea 32 hace como máximo $2 \cdot t - 2$ operaciones, lo que también está acotado por $O(log(p))$.

Así, si sumamos los costos del algoritmo completo se obtiene que el algoritmo toma $O(k \cdot log(p) + log(p)^2)$ pasos en el peor caso.

Cálculo de la probabilidad de entregar *sin_resultado*

Como se vio en clases, dado p primo se tiene que $|S_p^+| = |S_p^-| = \frac{p-1}{2} = \frac{|Z_n^*|}{2}$. Así, dado un número γ aleatorio e uniforme en el rango $1, ..., p-1$, la probabilidad que $\gamma^{\frac{p-1}{2}} = 1$ (o sea, $\gamma \in S_p^+$) es igual a $\frac{1}{2}$. Para que el algoritmo retorne *sin_resultado* tendría que ocurrir que $\gamma^{\frac{p-1}{2}} = 1$ para cada uno de los k intentos de sacar γ aleatoriamente de forma independiente. La probabilidad de que esto ocurra es igual a la probabilidad que $\gamma^{\frac{p-1}{2}} = 1$ en el primero intento, multiplicado por la probabilidad que $\gamma^{\frac{p-1}{2}} = 1$ en el segundo intento, etc. debido a que cada intento es independiente. Así, se tiene que

$$Pr[\text{entregar } sin\_resultado] = \prod_{i=1}^{k} \frac{1}{2} = \frac{1}{2^k}$$