Primalidad en Tiempo Polinomial

Francisco Gallego Salido

Universidad de Granada

fgallego@correo.ugr.es

24 de noviembre de 2021

Contenidos

Tests de Primalidad

Introducción

¿Qué son los números primos?

Un número primo es aquel que solo es divisible por 1 o por sí mismo.

Example

El número 5 es primo porque solo es divisible por 1 y 5.

Example

El número 12 no es primo porque es divisible por 2, 3, 4 y 6, que son distintos de 1 y 12.

¿Qué es un test de primalidad?

Un test de primalidad es un algoritmo que nos permite determinar si un número es primo o no.

El test más básico es el que se deriva de la definición de primalidad, cuya complejidad es $O(\sqrt{(n)})$:

¿Qué es un test de primalidad?

Un test de primalidad es un algoritmo que nos permite determinar si un número es primo o no.

El test más básico es el que se deriva de la definición de primalidad, cuya complejidad es $O(\sqrt{(n)})$:

- Comprobar todos los números menores que $\lfloor \sqrt{n} \rfloor$ y ver si alguno divide a n.

¿Qué es un test de primalidad?

Un test de primalidad es un algoritmo que nos permite determinar si un número es primo o no.

El test más básico es el que se deriva de la definición de primalidad, cuya complejidad es $O(\sqrt{(n)})$:

- Comprobar todos los números menores que $\lfloor \sqrt{n} \rfloor$ y ver si alguno divide a n.
- Si ninguno lo divide, n es primo.

Un test de primalidad es un algoritmo que nos permite determinar si un número es primo o no.

El test más básico es el que se deriva de la definición de primalidad, cuya complejidad es $O(\sqrt{(n)})$:

- Comprobar todos los números menores que $\lfloor \sqrt{n} \rfloor$ y ver si alguno divide a n.
- Si ninguno lo divide, n es primo.
- Si alguno lo divide, n es compuesto.

Pequeño Teorema de Fermat

Pequeño Teorema de Fermat

Un test mucho más eficiente es el que se deriva del *Pequeño Teorema de Fermat*.

Un test mucho más eficiente es el que se deriva del *Pequeño Teorema de Fermat*.

Theorem (Pequeño Teorema de Fermat)

Si n es primo, entonces $a^n \equiv a \mod (n)$ para todo $a \in \mathbb{Z}$.

Pequeño Teorema de Fermat

Un test mucho más eficiente es el que se deriva del *Pequeño Teorema de Fermat*.

Theorem (Pequeño Teorema de Fermat)

Si n *es primo, entonces* $a^n \equiv a \mod (n)$ *para todo* $a \in \mathbb{Z}$.

El test consiste en comprobar varios valores de a y vemos si se cumple la congruencia.

Pequeño Teorema de Fermat

Un test mucho más eficiente es el que se deriva del *Pequeño Teorema de Fermat*.

Theorem (Pequeño Teorema de Fermat)

Si n es primo, entonces $a^n \equiv a \mod (n)$ para todo $a \in \mathbb{Z}$.

El test consiste en comprobar varios valores de a y vemos si se cumple la congruencia.

Si falla para algún a, entonces n es compuesto. En caso contrario, n probablemente sea primo.

Problema

No es cierto en general que si $a^n \equiv a \mod (n)$ para todo $a \in \mathbb{Z}$, entonces n sea primo.

Problema

No es cierto en general que si $a^n \equiv a \mod (n)$ para todo $a \in \mathbb{Z}$, entonces n sea primo.

De hecho, existe un conjunto de números que no son primos para los que el *Pequeño Teorema de Fermat* siempre se cumple.

Problema

No es cierto en general que si $a^n \equiv a \mod (n)$ para todo $a \in \mathbb{Z}$, entonces n sea primo.

De hecho, existe un conjunto de números que no son primos para los que el *Pequeño Teorema de Fermat* siempre se cumple.

A dicho conjunto se le conoce como *Números de Charmichael*, donde $561 = 3 \cdot 11 \cdot 17$ es el primer elemento de dicho conjunto.

Algoritmo AKS

Historia

El algoritmo **AKS** debe su nombre a los tres matemáticos que lo descubrieron: Manindra Agrawal, Neeraj Kayal y Nitin Saxena.

Historia

El algoritmo **AKS** debe su nombre a los tres matemáticos que lo descubrieron: Manindra Agrawal, Neeraj Kayal y Nitin Saxena.

Se trata del primer test de primalidad que cumple todas las propiedades deseadas:

Historia

El algoritmo **AKS** debe su nombre a los tres matemáticos que lo descubrieron: Manindra Agrawal, Neeraj Kayal y Nitin Saxena.

Se trata del primer test de primalidad que cumple todas las propiedades deseadas:

- **General**. Es válido para cualquier entrada.

Historia

El algoritmo **AKS** debe su nombre a los tres matemáticos que lo descubrieron: Manindra Agrawal, Neeraj Kayal y Nitin Saxena.

Se trata del primer test de primalidad que cumple todas las propiedades deseadas:

- **General**. Es válido para cualquier entrada.
- **Determinista**. Determina con una probabilidad del 100 % la primalidad de un número.

Historia

El algoritmo **AKS** debe su nombre a los tres matemáticos que lo descubrieron: Manindra Agrawal, Neeraj Kayal y Nitin Saxena.

Se trata del primer test de primalidad que cumple todas las propiedades deseadas:

- **General**. Es válido para cualquier entrada.
- **Determinista**. Determina con una probabilidad del 100 % la primalidad de un número.
- **Polinómico**. La complejidad asintótica del test es polinómica en el número de cifras.

Historia

El algoritmo **AKS** debe su nombre a los tres matemáticos que lo descubrieron: Manindra Agrawal, Neeraj Kayal y Nitin Saxena.

Se trata del primer test de primalidad que cumple todas las propiedades deseadas:

- **General**. Es válido para cualquier entrada.
- **Determinista**. Determina con una probabilidad del 100 % la primalidad de un número.
- **Polinómico**. La complejidad asintótica del test es polinómica en el número de cifras.
- **Incondicional**. La validez del test no depende de resultados no probados.

Introducción

Idea Principal

El *Pequeño Teorema de Fermat* no proporciona un test válido, pero una versión general suya sí. Sea el siguiente teorema:

Idea Principal

El *Pequeño Teorema de Fermat* no proporciona un test válido, pero una versión general suya sí. Sea el siguiente teorema:

Theorem

Sea $n > 1$ *y* $a \in \mathbb{Z}$*. Entonces* n *es primo si, y solo si, se cumple*

$$(X + a)^n \equiv X^n + a \mod (n)$$

Idea Principal

El *Pequeño Teorema de Fermat* no proporciona un test válido, pero una versión general suya sí. Sea el siguiente teorema:

Theorem

Sea $n > 1$ *y* $a \in \mathbb{Z}$. *Entonces* n *es primo si, y solo si, se cumple*

$$(X + a)^n \equiv X^n + a \mod (n)$$

Un test que se deriva de esta propiedad es simplemente comprobar si la congruencia se cumple para algún a.

Idea Principal

El *Pequeño Teorema de Fermat* no proporciona un test válido, pero una versión general suya sí. Sea el siguiente teorema:

Theorem

Sea $n > 1$ *y* $a \in \mathbb{Z}$. *Entonces* n *es primo si, y solo si, se cumple*

$$(X + a)^n \equiv X^n + a \mod (n)$$

Un test que se deriva de esta propiedad es simplemente comprobar si la congruencia se cumple para algún a.

Si se cumple, entonces n es primo. En caso contrario, n es compuesto.

¿Es este test polinómico?

Un problema que tiene este test es que tiene complejidad $\Omega(n)$, pues hay que evaluar n coeficientes de los polinomios resultantes.

Este test es muy ineficiente y lejos de ser polinómico en la cantidad de cifras.

¿Es este test polinómico?

Un problema que tiene este test es que tiene complejidad $\Omega(n)$, pues hay que evaluar n coeficientes de los polinomios resultantes.

Este test es muy ineficiente y lejos de ser polinómico en la cantidad de cifras.

¿Podemos reducir el número de coeficientes a evaluar?

Solución

Si la congruencia anterior la evaluamos módulo $(X^r - 1, n)$ para algún r escogido apropiadamente, podremos reducir el número de coeficientes. Sea pues la siguiente congruencia:

Solución

Si la congruencia anterior la evaluamos módulo $(X^r - 1, n)$ para algún r escogido apropiadamente, podremos reducir el número de coeficientes. Sea pues la siguiente congruencia:

$$(X + a)^n \equiv X^n + a \mod (X^r - 1, n)$$

Solución

Si la congruencia anterior la evaluamos módulo $(X^r - 1, n)$ para algún r escogido apropiadamente, podremos reducir el número de coeficientes. Sea pues la siguiente congruencia:

$$(X + a)^n \equiv X^n + a \mod (X^r - 1, n)$$

El problema de esta congruencia es que, aunque se sigue cumpliendo cuando n es primo, algunos compuestos la cumplen también para algunos valores de a y r.

Solución

Si la congruencia anterior la evaluamos módulo $(X^r - 1, n)$ para algún r escogido apropiadamente, podremos reducir el número de coeficientes. Sea pues la siguiente congruencia:

$$(X + a)^n \equiv X^n + a \mod (X^r - 1, n)$$

El problema de esta congruencia es que, aunque se sigue cumpliendo cuando n es primo, algunos compuestos la cumplen también para algunos valores de a y r.

Escogiendo apropiadamente r y probando para ciertos valores de a, si se cumple entonces la congruencia anterior, podemos asegurar que n es una potencia de un primo.

Solución

Si la congruencia anterior la evaluamos módulo $(X^r - 1, n)$ para algún r escogido apropiadamente, podremos reducir el número de coeficientes. Sea pues la siguiente congruencia:

$$(X + a)^n \equiv X^n + a \mod (X^r - 1, n)$$

El problema de esta congruencia es que, aunque se sigue cumpliendo cuando n es primo, algunos compuestos la cumplen también para algunos valores de a y r.

Escogiendo apropiadamente r y probando para ciertos valores de a, si se cumple entonces la congruencia anterior, podemos asegurar que n es una potencia de un primo.

Dichos r y a se pueden elegir de forma que la complejidad del test sea polinómica.

El Algoritmo

Pasos del Algoritmo AKS

Algorithm 1 AKS

procedure $\textsc{IsPrime}(n)$ ▷ Comprobar si $n > 1$ es un número primo

  if $n = a^b$ con $a, b > 1$ **return** COMPUESTO ▷ Paso 1

  Encontrar el menor r tal que $ord_r(n) > \log^2(n)$. ▷ Paso 2

  if $1 < (a, n) < n$ para algún $a \leq r$ **return** COMPUESTO ▷ Paso 3

  if $n \leq r$ **return** PRIMO ▷ Paso 4

  for $a = 1$ hasta $\lfloor \sqrt{\phi(r)} \log(n) \rfloor$ **do** ▷ Paso 5

    if $(X + a)^n \not\equiv X^n + a \mod (n, X^r - 1)$ **return** COMPUESTO

  end for

  return PRIMO ▷ Paso 6

end procedure

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.
4. $31 \not\leq 29$. Pasamos al siguiente paso.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.
4. $31 \not\leq 29$. Pasamos al siguiente paso.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.
4. $31 \not\leq 29$. Pasamos al siguiente paso.
5. - $\lfloor \sqrt{\phi(r)} \log(n) \rfloor = 26$.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.
4. $31 \not\leq 29$. Pasamos al siguiente paso.
5.
   - $\lfloor \sqrt{\phi(r)} \log(n) \rfloor = 26$.
   - $(X + a)^{31} = x^2 + a^{31}$ módulo $(X^{29} - 1, 31)$.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.
4. $31 \not\leq 29$. Pasamos al siguiente paso.
5.
   - $\lfloor \sqrt{\phi(r)} \log(n) \rfloor = 26$.
   - $(X + a)^{31} = x^2 + a^{31}$ módulo $(X^{29} - 1, 31)$.
   - $X^{31} + a = x^2 + a$ módulo $(X^{29} - 1, 31)$.

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.
4. $31 \not\leq 29$. Pasamos al siguiente paso.
5.
   - $\lfloor \sqrt{\phi(r)} \log(n) \rfloor = 26$.
   - $(X + a)^{31} = x^2 + a^{31}$ módulo $(X^{29} - 1, 31)$.
   - $X^{31} + a = x^2 + a$ módulo $(X^{29} - 1, 31)$.
   - $a^{31} \equiv a \mod (X^{29} - 1, 31)$ para todo $1 \leq a \leq 26$. Pasamos al siguiente paso

Ejemplo

Sea el número 31, el cual ya sabemos que es primo.

1. 31 no es una potencia perfecta. Pasamos al siguiente paso.
2. El menor r es 29, pues $ord_{29}(31) = 28 > \log^2(31) \simeq 24{,}54$.
3. $(a, 31) = 1$ para todo $a \leq 29$. Pasamos al siguiente paso.
4. $31 \not\leq 29$. Pasamos al siguiente paso.
5.
   - $\lfloor \sqrt{\phi(r)} \log(n) \rfloor = 26$.
   - $(X + a)^{31} = x^2 + a^{31}$ módulo $(X^{29} - 1, 31)$.
   - $X^{31} + a = x^2 + a$ módulo $(X^{29} - 1, 31)$.
   - $a^{31} \equiv a \mod (X^{29} - 1, 31)$ para todo $1 \leq a \leq 26$. Pasamos al siguiente paso
6. Hemos llegado al último paso, por lo que 31 es primo.

Complejidad

Complejidad Paso a Paso

Cada paso tiene un complejidad determinada:

Cada paso tiene un complejidad determinada:

- $O^{\sim}(\log^3(n))$.

Complejidad Paso a Paso

Cada paso tiene un complejidad determinada:

- $O^{\sim}(\log^3(n))$.
- $O^{\sim}(\log^7(n))$.

Complejidad Paso a Paso

Cada paso tiene un complejidad determinada:

- $O^{\sim}(\log^3(n))$.
- $O^{\sim}(\log^7(n))$.
- $O^{\sim}(\log^6(n))$.

Complejidad Paso a Paso

Cada paso tiene un complejidad determinada:

- $O^{\sim}(\log^3(n))$.
- $O^{\sim}(\log^7(n))$.
- $O^{\sim}(\log^6(n))$.
- $O(\log(n))$.

Complejidad Paso a Paso

Cada paso tiene un complejidad determinada:

- $O^{\sim}(\log^3(n))$.
- $O^{\sim}(\log^7(n))$.
- $O^{\sim}(\log^6(n))$.
- $O(\log(n))$.
- $O^{\sim}(\log^{21/2}(n))$.

Complejidad Paso a Paso

Cada paso tiene un complejidad determinada:

- $O^{\sim}(\log^3(n))$.
- $O^{\sim}(\log^7(n))$.
- $O^{\sim}(\log^6(n))$.
- $O(\log(n))$.
- $O^{\sim}(\log^{21/2}(n))$.
- $O(1)$.

Complejidad Total

La complejidad del quinto paso es la más alta, luego el algoritmo **AKS** tiene una complejidad algorítmica de $O^{\sim}(\log^{21/2}(n))$.

Complejidad Total

La complejidad del quinto paso es la más alta, luego el algoritmo **AKS** tiene una complejidad algorítmica de $O^{\sim}(\log^{21/2}(n))$.

Usando *Teoría de Cribas* se puede probar que $r = O(\log^3(n))$, reduciendo la complejidad hasta $O^{\sim}(\log^{15/2}(n))$.

Complejidad Total

La complejidad del quinto paso es la más alta, luego el algoritmo **AKS** tiene una complejidad algorítmica de $O^{\sim}(\log^{21/2}(n))$.

Usando *Teoría de Cribas* se puede probar que $r = O(\log^3(n))$, reduciendo la complejidad hasta $O^{\sim}(\log^{15/2}(n))$.

Bajo ciertas hipótesis no probadas (como puede ser la *Hipótesis Generalizada de Riemann*), se puede probar que $r = O(\log^2(n))$, reduciendo una vez más la complejidad hasta $O^{\sim}(\log^6(n))$.

Comparaciones

Candidatos

Vamos a comparar el algoritmo AKS con dos tests probabilísticos:

Candidatos

Vamos a comparar el algoritmo AKS con dos tests probabilísticos:

- Test de *Miller-Rabin* con 40 rondas.

Candidatos

Vamos a comparar el algoritmo AKS con dos tests probabilísticos:

- Test de *Miller-Rabin* con 40 rondas.
- Test de *Solovay-Strassen* con 80 rondas.

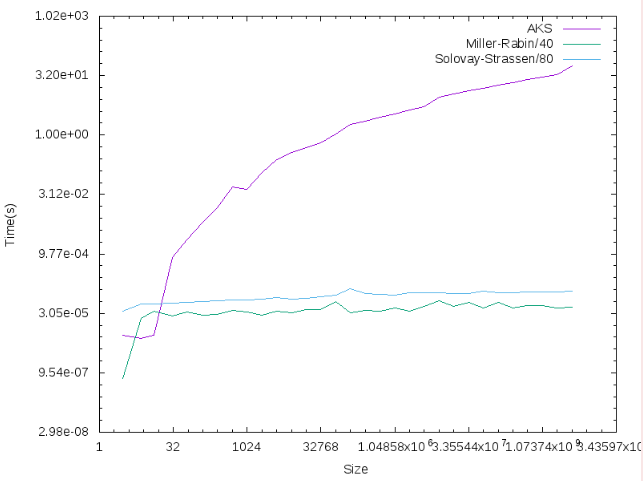
Primos

Conjunto de Prueba

Los números primos con los que se hacen las pruebas tienen una cantidad incremental de bits.

Por ejemplo, el 7 es el mayor primo con 3 bits, 31 el mayor primo con 5 bits, 2147483647 el mayor primo con 32 bits, etc.

Comparación Números Primos



Potencias Perfectas

Conjunto de Prueba

Las potencias perfectas que usaremos serán con los primos presentados anteriorimente, y consisten de dos conjuntos:

Conjunto de Prueba

Las potencias perfectas que usaremos serán con los primos presentados anteriorimente, y consisten de dos conjuntos:

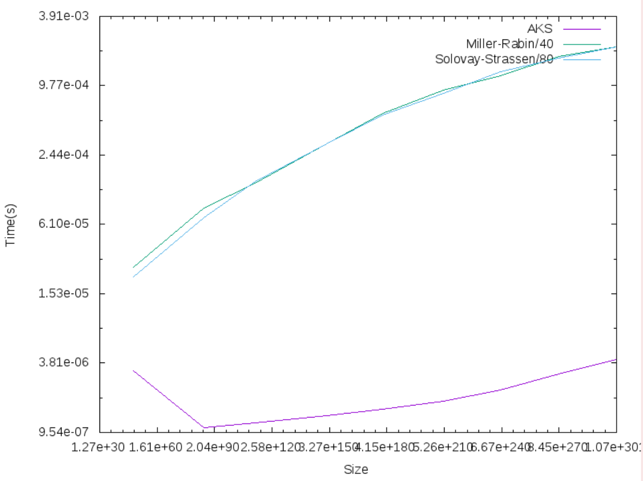
- Primos de hasta 16 bits elevados a 100.

Conjunto de Prueba

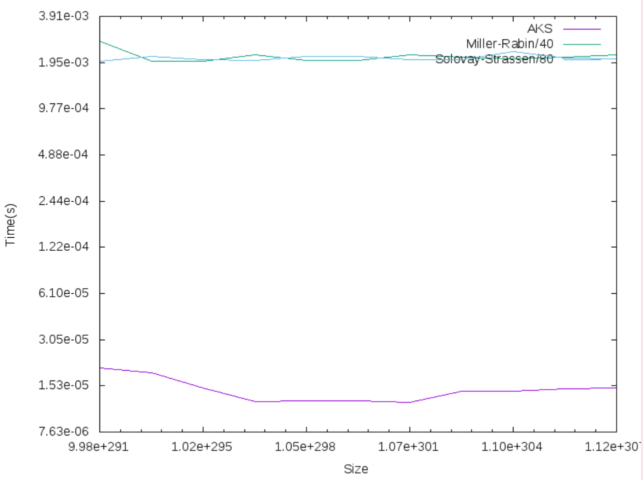
Las potencias perfectas que usaremos serán con los primos presentados anteriorimente, y consisten de dos conjuntos:

- Primos de hasta 16 bits elevados a 100.
- Primos de entre 192 y 256 bits elevados a 5.

Comparación Potencias Perfectas 1



Comparación Potencias Perfectas 2



Compuestos No Potencias Perfectas

Conjunto de Prueba

Los números compuestos que usaremos en estas comparaciones serán producto de los primos mencionados anteriormente. Hay dos conjuntos:

Conjunto de Prueba

Los números compuestos que usaremos en estas comparaciones serán producto de los primos mencionados anteriormente. Hay dos conjuntos:

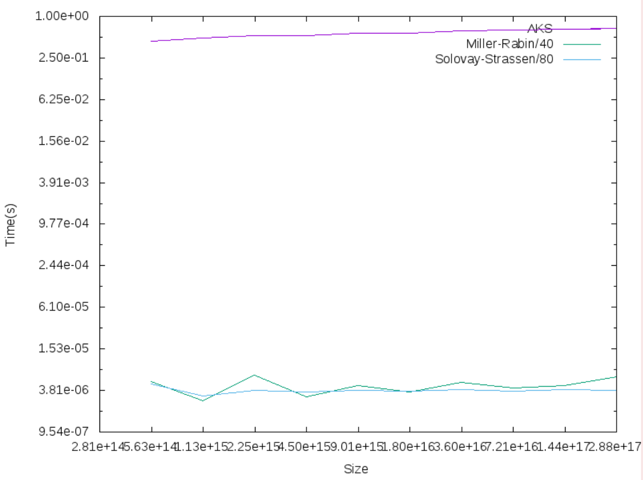
- Primos de entre 32 y 42 bits multiplicados por un primo de 16 bits.

Conjunto de Prueba

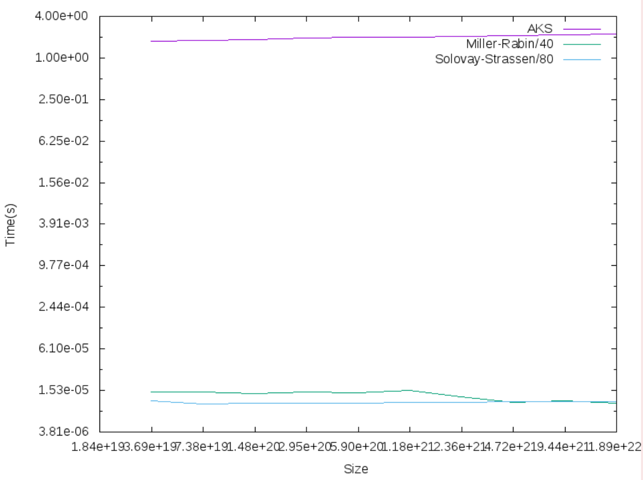
Los números compuestos que usaremos en estas comparaciones serán producto de los primos mencionados anteriormente. Hay dos conjuntos:

- Primos de entre 32 y 42 bits multiplicados por un primo de 16 bits.
- Primos de entre 32 y 42 bits multiplicados por un primo de 32 bits.

Comparación Compuestos No Potencias Perfectas 1



Comparación Compuestos No Potencias Perfectas 1



Conclusiones

Conclusiones

El test es brillante desde el punto de vista matemático, ya que la prueba de la validez del test usa herramientas elementales.

Conclusiones

El test es brillante desde el punto de vista matemático, ya que la prueba de la validez del test usa herramientas elementales.

Sin embargo, a pesar de ser un test polinómico, en la práctica queda muy por detrás de otros test usados en la actualidad.

Referencias

Manindra Agrawal, Neeraj Kayal, and Nitin Saxena.
PRIMES is in P.
Ann. of Math. (2), 160(2):781–793, 2004.

Manindra Agrawal, Neeraj Kayal, and Nitin Saxena.
Errata: PRIMES is in P.
Ann. of Math. (2), 189(1):317–318, 2019.

Francisco Gallego Salido.
Tfg.
https://github.com/fgallegosalido/TFG.

Fin