Algoritmos MonteCarlo

750098M Simulación computacional

Contenido

Introducción

“Inocente hasta que se demuestre lo contrario”

- Comprobar si una hipótesis es verdadera o falsa
- Se comprueban instancias, retornando verdadero si no existe una contradicción y falso si la hipótesis es falsa
- Análisis basado en probabilidad. Tiene razón con probabilidad p y no la tiene con probabilidad 1 - p
- Se reduce probabilidad de error si se aumenta número de iteraciones

Son dos vectores iguales?

Algoritmo MonteCarlo:

Seleccionar aleatoriamente un componente i

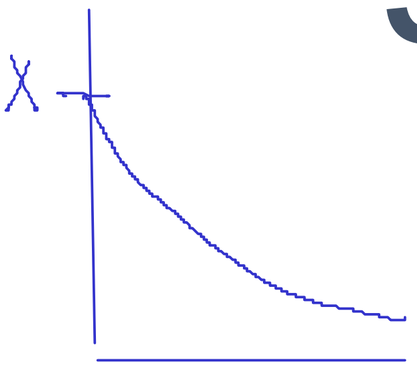
Si vector1(i) = vector2(i) devolver: Son iguales

Sino Devolver: Son diferentes

- Si el algoritmo devuelve SON IGUALES, puede ser que este sea cierto o que coinciden por casualidad en este componente

- Si el algoritmo devuelve SON DIFERENTES, este resultado es acertado: vector1(i) != vector2(i), los vectores son diferentes con certeza

Son dos vectores iguales?



Repetir el algoritmo puede aumentar la confiabilidad en que la respuesta sea correcta

- *p* es igual a la probabilidad de que la respuesta sea correcta

  q = 1 - p es igual a la probabilidad de error en **1** ensayo

- $q^n = (1 - p)^n$ es la probabilidad de equivocación después de n iteraciones (aumento de probabilidad estocástica)

- Aumentamos considerablemente la probabilidad de que la respuesta SON IGUALES sea correcta, pero no llegamos a tener certeza

Definición

- Un algoritmo MonteCarlo es p-correcto, si la respuesta es verdadera con probabilidad p, independiente de la estancia de entrada instancia

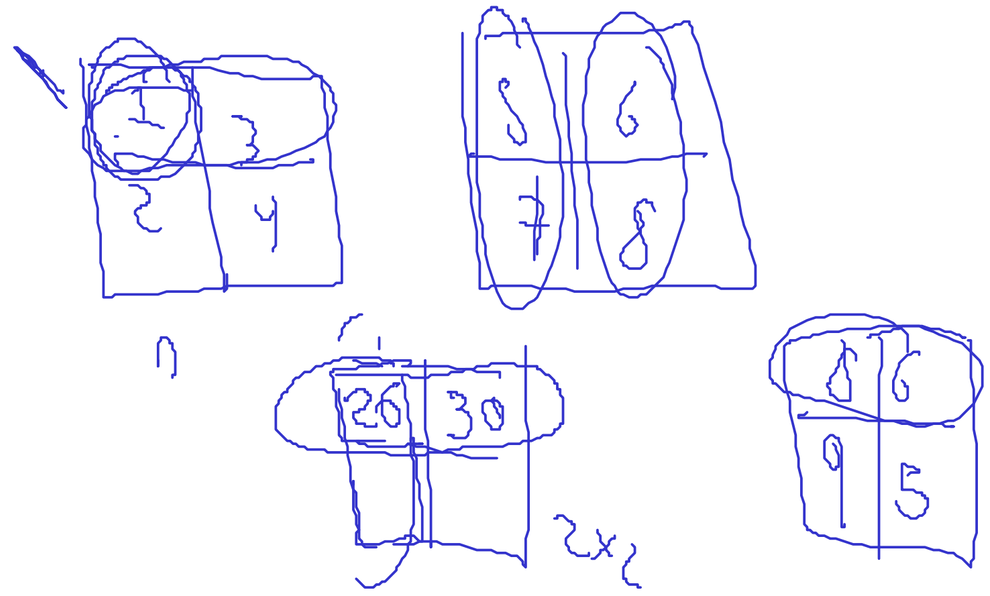
- A veces se permite que p depende del tamaño del input, pero nunca dependerá de sus datos específicos

Verificación de la multiplicación de matrices

Dadas las matrices (nxn) A,B y C, se quiere verificar C = AB

Algoritmos determinísticos	Algoritmo MonteCarlo
La multiplicación matricial tradicional requiere n^3 operaciones, es decir el algoritmo es de $O(n^3)$ Los mejores algoritmos determinísticos de multiplicación de matrices son de $O(n^{2.37})$	La multiplicación puede llegar a ser $O(n^2)$ Implica asumir el riesgo de que no sea verdad Sin embargo, se puede controlar el error

Algoritmo de Freivalds



Principio del algoritmo

Se suman arbitrariamente algunas filas de la matriz AB y las mismas filas de la matriz C y se comparan los resultados.
Si las sumas son iguales, el algoritmo devuelve: SON IGUALES; sino, devuelve NO SON IGUALES

Ejemplo

$$A = \begin{matrix} 5 & 2 & 6 & 1 \\ 0 & 6 & 2 & 0 \\ 3 & 8 & 1 & 4 \\ 1 & 8 & 5 & 6 \end{matrix} \qquad B = \begin{matrix} 7 & 5 & 8 & 0 \\ 1 & 8 & 2 & 6 \\ 9 & 4 & 3 & 8 \\ 5 & 3 & 7 & 9 \end{matrix}$$

$$AB = \begin{matrix} 96 & 68 & 69 & 69 \\ 24 & 56 & 18 & 52 \\ 58 & 95 & 71 & 92 \\ 90 & 107 & 81 & 141 \end{matrix} \qquad C = \begin{matrix} 96 & 68 & 69 & 69 \\ 24 & 56 & 18 & 52 \\ 58 & 95 & 71 & 92 \\ 90 & 107 & 81 & 142 \end{matrix}$$

Ejemplo

$$AB = \begin{matrix} 96 & 68 & 69 & 69 \\ 24 & 56 & 18 & 52 \\ 58 & 95 & 71 & 92 \\ 90 & 107 & 81 & 141 \end{matrix} \qquad C = \begin{matrix} 96 & 68 & 69 & 69 \\ 24 & 56 & 18 & 52 \\ 58 & 95 & 71 & 92 \\ 90 & 107 & 81 & 142 \end{matrix}$$

Suma de las filas 1 y 2:
Para AB: [110, 124, 87, 121];
para C : [110, 124, 87, 121].
Respuesta: SON IGUALES

Suma de las filas 1,3 y 4:
Para AB: [244, 270, 221, 302];
para C : [244, 270, 221, 301].
Respuesta: NO SON IGUALES

Ejemplo

$$AB = \begin{matrix} 96 & 68 & 69 & 69 \\ 24 & 56 & 18 & 52 \\ 58 & 95 & 71 & 92 \\ 90 & 107 & 81 & 141 \end{matrix}$$

Suma de las filas 1 y 2:
X:[1,1,0,0]
X(AB)[110, 124, 87, 121].

Suma de las filas 1,3 y 4:
X:[1,0,1,1]
X(AB): [244, 270, 221, 301].

Algoritmo de Freivalds

La multiplicación requiere menos operaciones si en vez de X(AB) se evalúa (XA)B

Para X(AB)	Para (XA)B
La multiplicación de 2 matrices (n x n): $O(n^3)$ La multiplicación de 1 vector (1 x n) con una matriz (n x n): $O(n^2)$ X(AB) es de $O(n^3)$	2 multiplicaciones de 1 vector (1 x n) con una matriz (n x n): $O(n^2)$ (XA) B es de $O(n^2)$

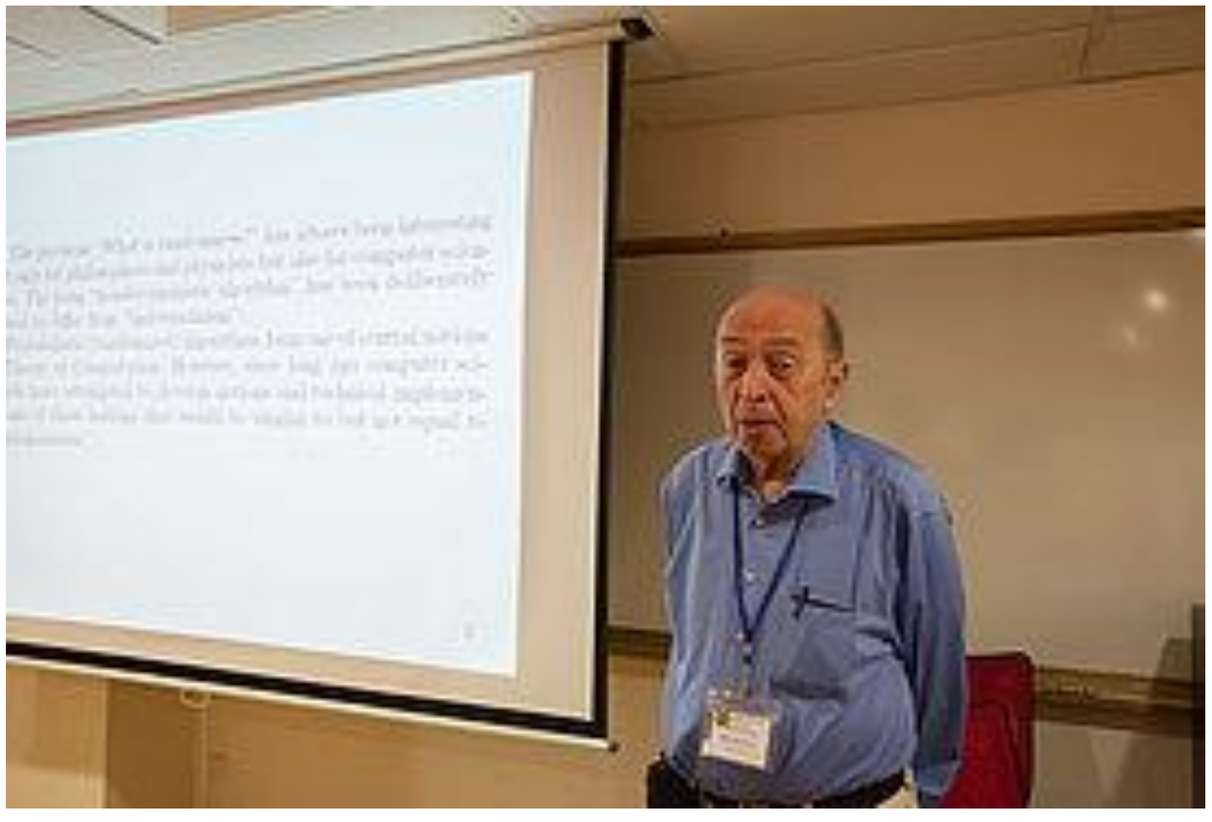
Algoritmo de Freivalds

Algoritmo Freivalds(A,B,C,n):

Generar vector binario X en (1,n)

Si (XA)B = XC devolver VERDADERO

Sino Devolver: Son diferentes



Rūsiņš Mārtiņš Freivalds

(10 November 1942 – 4 January 2016)

Algoritmo de Freivalds

Se ha seleccionado aleatoriamente X binario y se encontró XAB = XC

Analicemos las instancias AB /= C

En este caso existe una fila i tal que AB_i /= C_i

Se define el vector X' como

$$X'_j = \begin{cases} X_j & \text{si } j \neq i \\ 1 - X_j & \text{si } j = i \end{cases}$$

Para X' tenemos

VERDADERO

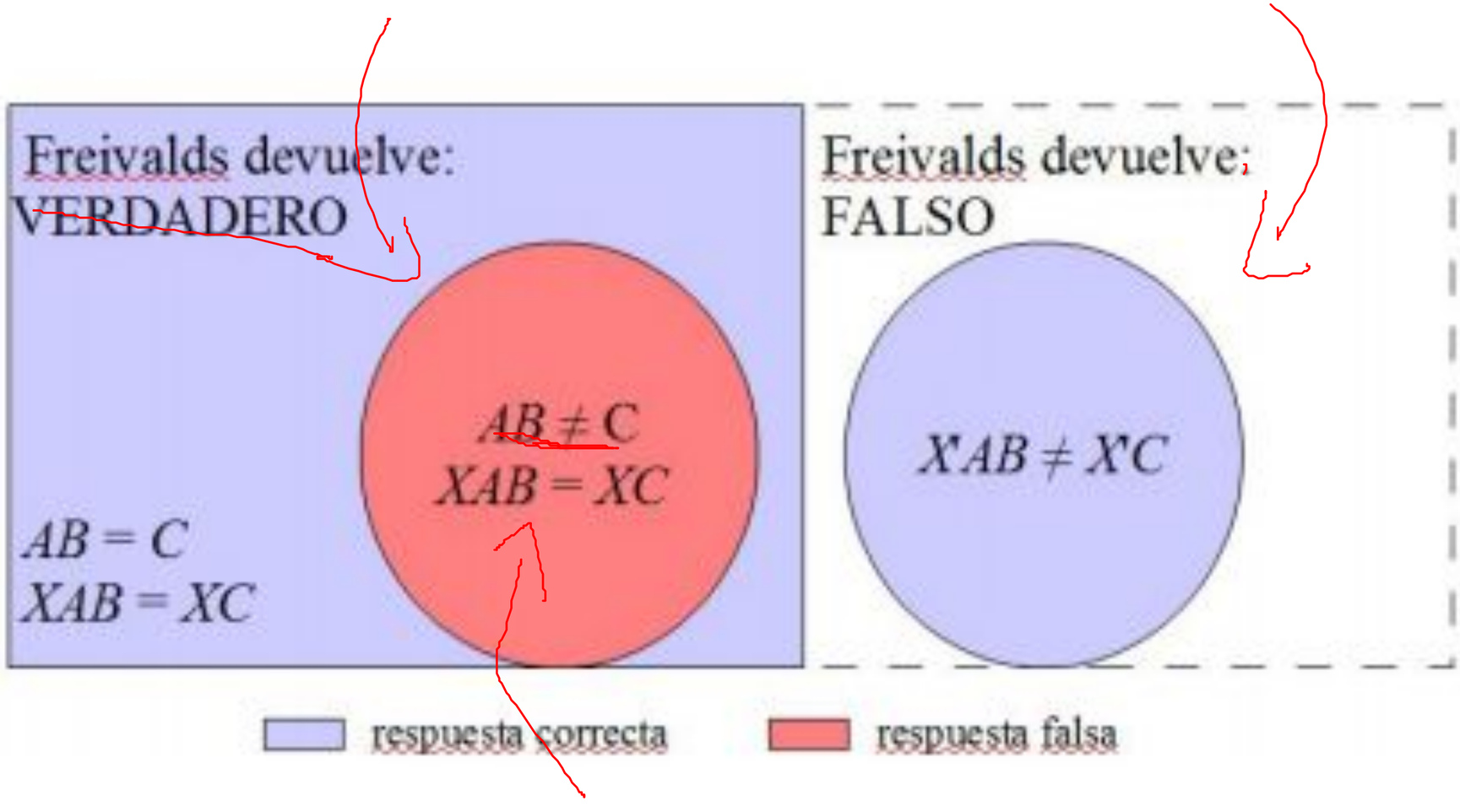
X'AB = XAB +- AB_i /= XAB = XC

Es decir Freivalds daría FALSO

Algoritmo de Freivalds

Es decir, Freivalds devuelve VERDADERO en la mitad de los casos donde AB /= C, y siempre si AB = C.

Entonces Freivalds es ½ - correcto



Iteraciones

```
Algoritmo RepetirFreivalds(A,B,C,n):
Repetir k veces
Si Freivalds(A,B,C,n) = FALSO
    Devolver FALSO
    Salir
Devolver VERDADERO
```

La probabilidad de error en una llamada de Freivalds es ½.
Diferentes llamadas son estadísticamente independientes, por eso las probabilidades de error se multiplican.
Después de k llamadas, la probabilidad de error es por ende $(1/2)^k$.
En consecuencia RepetirFreivalds es $1 - (1/2)^k$ - correcto, un valor que converge a 1.

Tolerancia para el error

Para garantizar que la probabilidad de error quede por debajo de una tolerancia ε dada, se puede determinar el número adecuado de iteraciones k:

El tiempo de ejecución en este caso es de

$$\epsilon \leq (1/2)^k \quad \Longrightarrow \quad k >= \log_2 (1/\epsilon)$$

El tiempo de ejecución es:

$$O\left(n^2 \log_2 (1/\epsilon)\right)$$

Limitaciones

El algoritmo de Freivalds significa un ahorro en el número de multiplicaciones sólo si las matrices son muy grandes:
si la probabilidad p de que el resultado sea correcto debe ser 0.99 o mayor, RepetirFreivalds debe usar por o menos 7 iteraciones.
Es más rápido que la comprobación si:

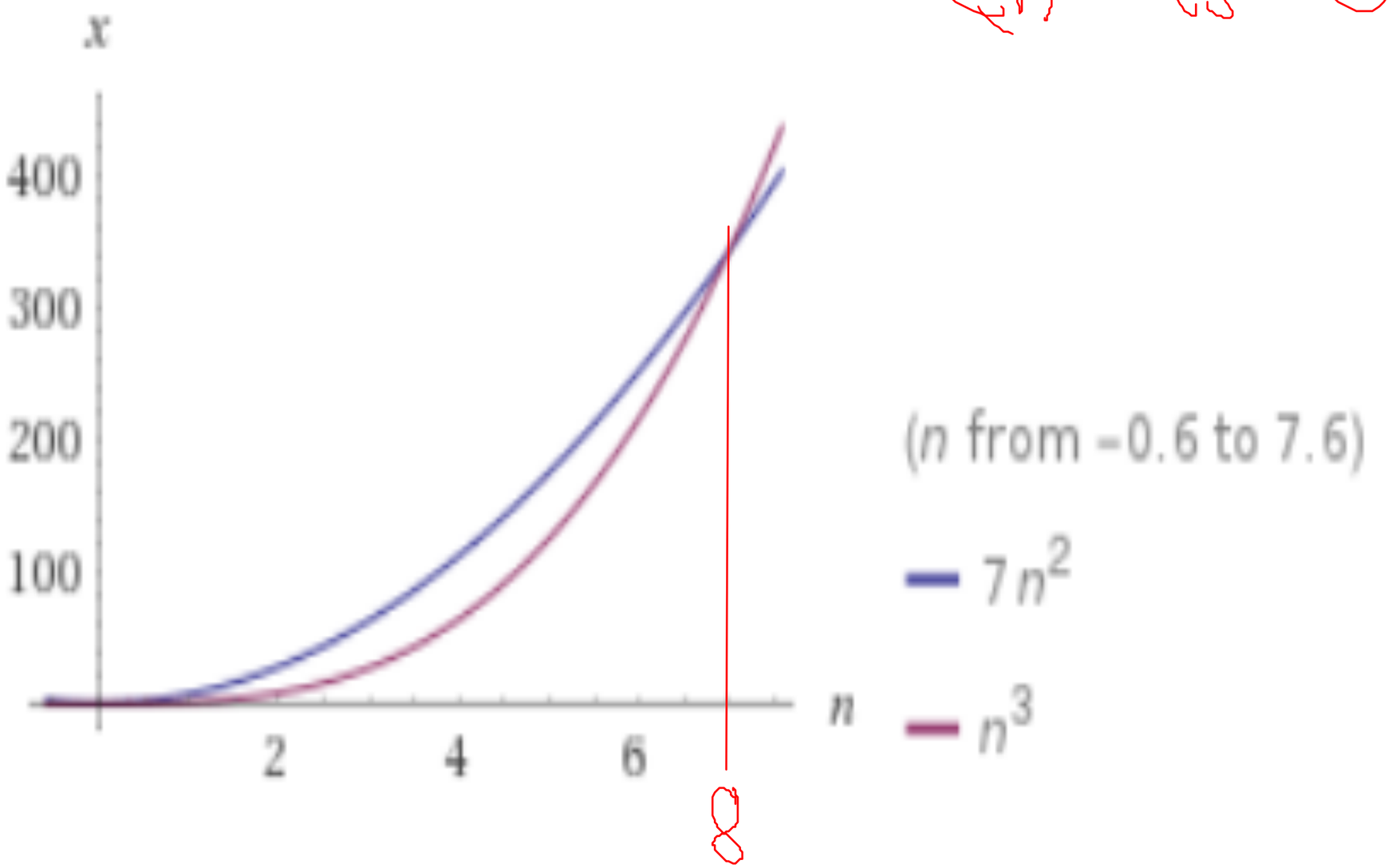
$$7 * 2 * n^2 <= n^3$$

Es decir que

$$n >= 14$$

$7n^2$ vs n^3

$7n^2$ is $O(n^3)$



Primaldad

Hay un algoritmo importante de criptografía, que es muy difícil de decodificar. Este algoritmo tiene la especialidad que:

- El quien quiere recibir el mensaje manda información públicamente que sirve para codificar el mensaje.
- Conserva privado la información para de codificar.
- La persona que manda el mensaje lo codifica y lo manda públicamente.

Primaldad

El algoritmo está basado en el producto de dos números primos grandes.

- La persona que quiere recibir el mensaje necesita encontrar dos primos muy grandes
- Una persona que no tiene la información para decodificar, tendrá que probar con muchos productos de primos hasta poder romper el código.

Comprobación

Ejemplo: Desarrollar un algoritmo MonteCarlo basado en el

Teorema menor de Fermat:

Sea n un número primo. Entonces para cualquier a entero tal que $1 \leq a \leq n - 1$

Se tiene:

$$a^{n-1} \bmod n = 1$$

Ejemplo

$n = 5$	
$a = 1$	$a^{n-1} \bmod n = 1^4 \bmod 5 = 1 \bmod 5 = 1$
$a = 2$	$a^{n-1} \bmod n = 2^4 \bmod 5 = 16 \bmod 5 = 1$
$a = 3$	$a^{n-1} \bmod n = 3^4 \bmod 5 = 81 \bmod 5 = 1$
$a = 4$	$a^{n-1} \bmod n = 4^4 \bmod 5 = 256 \bmod 5 = 1$

¿Cómo se podría diseñar un algoritmo MonteCarlo basado en este Teorema?

Algoritmo MonteCarlo

Se usa la negación.

Teorema:
Sea n un entero. Si n es primo, entonces se tiene para cada a $\in$ [2, n − 1]

$$a^{n-1} \bmod n = 1$$

Negación:
Sea n un entero. Si existe a $\in$ [2, n − 1] tal que

$$a^{n-1} \bmod n \mathrel{/}= 1$$

entonces n no es primo

Algoritmo MonteCarlo

Algoritmo Fermant(n):

Generar aleatorio a $\in$ [2, n − 1]
Evaluar a^{n-1} mod n:
Si a^{n-1} mod n /= 1 devolver “no es primo”
Sino devolver “n es primo”

Sea n el entero a comprobar

(el algoritmo verifica la proposición ”n es primo”)

Ejemplo

$n = 15$

a	a^{n-1}	a^{n-1} mod n
9	22 876 792 454 961	6
8	4 398 046 511 104	4
12	1 283 918 464 548 864	9
6	78 364 164 096	6
12	1 283 918 464 548 864	9
10	100 000 000 000 000	10
2	16 384	4
9	22 876 792 454 961	6
14	11 112 006 825 558 016	1
12	1 283 918 464 548 864	9

$n = 17$

a	a^{n-1}	a^{n-1} mod n
9	1 853 020 188 851 841	1
8	281 474 976 710 656	1
12	184 884 258 895 036 416	1
6	2 821 109 907 456	1
12	184 884 258 895 036 416	1
10	10 000 000 000 000 000	1
2	65 536	1
9	1 853 020 188 851 841	1
14	2 177 953 337 809 371 136	1
12	184 884 258 895 036 416	1

Si el algoritmo devuelve ”no es primo” tiene razón y para.

Si el algoritmo devuelve ”es primo” puede ser equivocado. Se repite.

Cuanto se detiene?

Falsos testigos

Falsos testigos para la primaldad de n son aleatorios a $\in [2, n - 1]$ tal que $a^{n-1} \bmod n = 1$.
La mayoría de los no-primos tiene pocos falsos testigos.
Sin embargo: hay números no-primos con muchos falsos testigos: producen muchas veces el resultado ”1”, aunque no son primos.
Cuantitativamente: para cada p existen infinitos números tal que la probabilidad de detectar que no son primos es menor que p.

Es decir: El algoritmo Fermat no es p-correcto para ningún p. **¡No funciona la idea de MonteCarlo!**

Extensión del teorema de Fermat

Teorema: Sea n entero. Si n es primo representado en la forma $n = 2^s t + 1$, entonces para cada $a \in [2, n-2]$ se tiene:

$$a^t \bmod n = 1 \text{ ó } a^{2^i t} \bmod n = n - 1 \text{ para un } i \text{ tal que } 0 \leq i < s$$

Negación:

Sea n entero impar de la forma a $a = 2^s t + 1$,.
si existe $a \in [2, n-2]$ tal que :

$$a^t \bmod n \neq 1 \quad \text{y} \quad a^{2^i t} \bmod n = n - 1 \text{ para cada } i \text{ tal que } 0 \leq i < s,$$

entonces n no es primo

Ejemplo

Ejemplo: *n* primo

$n=13=2^2\,3+1$,
$s=2$, $t=3$

a	$a^t \bmod n$	$a^{2t} \bmod n$
2	8	12
3	1	1
4	12	1
5	8	12
6	8	12
7	5	12
8	5	12
9	1	1
10	12	1
11	5	12

$9^6 \bmod 13 =$

$3^6 \bmod 13$

$4^6 \bmod 13$

Ejemplo: *n* no primo

$n=21=2^2\,5+1$,
$s=2$, $t=5$

a	$a^t \bmod n$	$a^{2t} \bmod n$
2	11	16
3	12	18
4	16	4
5	17	16
6	6	15
7	7	7
8	8	1
9	18	9
10	19	4
11	2	4
12	3	9
13	13	1
14	14	7
15	15	15
16	4	16
17	5	4
18	9	18
19	10	16

Algoritmo MillerRabin

AlgoritmoMillerRabin(n)

- encontrar s, t tal que $n = 2^s t + 1$.
- generar aleatorio $a \in [2, n-2]$.
- si $a^t \bmod n = 1$ ó $a^{2^i t} \bmod n = n-1$ para un i tal que $0 \leq i < s$: devolver "n es primo "
  sino: devolver "n no es primo "

Si el algoritmo devuelve ”n es primo ”, puede ser equivocado
Si se devuelve ”n no es primo ” es cierto que así es.

Algoritmo MillerRabin

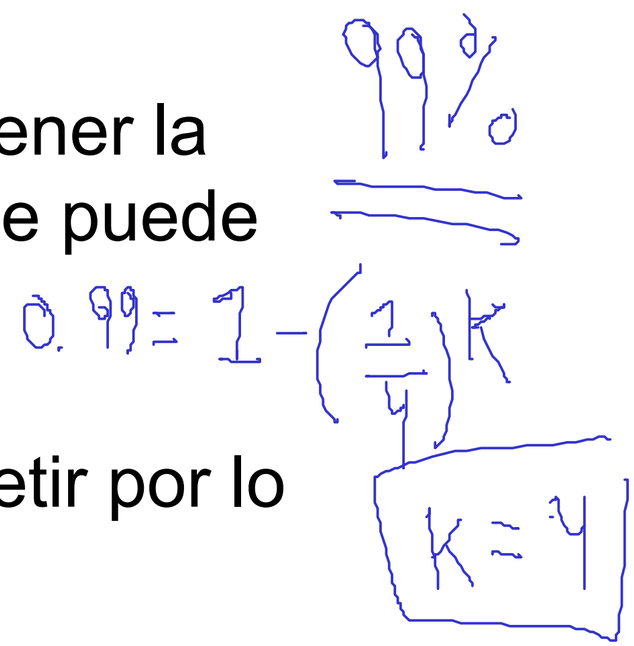
La extensión del teorema de Fermat es más fuerte.

Hay menos falsos testigos.

Consecuencia: Se puede mostrar que el algoritmo Miller-Rabin es ¾ - correcto.

Dado que Miller-Rabin es 3/4- correcto, la aplicación repetida aumenta la probabilidad de obtener la respuesta correcta.

Después de k iteraciones, la probabilidad de obtener la respuesta correcta es $p = 1 - (1/4)^k$; es decir: se puede controlar el error.



Para una tolerancia del error de $\in$, se debe repetir por lo menos $k = 1/2 \log_2 1/\in$ veces.

Conclusiones

Los algoritmos MonteCarlo sirven para validar una proposición o hipótesis

Un algoritmo MonteCarlo está seguro, si la proposición es falsa; en esta caso devuelve la respuesta correcta .

No está seguro, si no ha podido encontrar la contradicción a la proposición; en este caso devuelve verdadero, que puede o no ser correcto.

Para que funcione el algoritmo, se debe asegurar que es p-correcto para una probabilidad p.

Si es p-correcto, se puede ampliar la ventaja estocástica y aplicar el algoritmo tantas veces que el error sea no pequeño como se necesite.