Test de primalidad y el pequeño teorema de Fermat

Desde ahora en adelante suponga que n es un número impar

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

El pequeño teorema de Fermat nos dice que si p es primo, entonces $\left|W_{p}\right|=p-1$

Test de primalidad y el pequeño teorema de Fermat

¿Qué podemos decir sobre W_n si n es un número compuesto?

Test de primalidad y el pequeño teorema de Fermat

¿Qué podemos decir sobre W_n si n es un número compuesto?

Demuestre que si n es compuesto, entonces $\left|W_{n}
ight| < n-1$

¿Pero cuán pequeño es W_n ?

Test de primalidad: primer intento

Suponga que si n es compuesto, entonces

$$|W_n| \leq rac{n-1}{2}$$

¿Cómo puede usar este resultado para construir un test de primalidad?

Test de primalidad: primer intento

Dados n y k:

- 1. Elija a_1, \ldots, a_k en $\{1, \ldots, n-1\}$ con distribución uniforme y de manera independiente
- 2. Calcule $b_i = a_i^{n-1} mod n$ para cada $i \in \{1, \dots, k\}$
- 3. Si $b_i \neq 1$ para algún $i \in \{1, ..., k\}$, entonces retorne **false**. En el caso contrario, retorne **true**.

Test de primalidad: primer intento

¿Con cuáles entradas se puede equivocar el algoritmo? ¿Cuál es la probabilidad de error del algoritmo?

¿Utilizaría este algoritmo en la práctica?

Pero ...

 ${\bf No}$ es cierto que si n es compuesto, entonces

$$|W_n| \leq \frac{n-1}{2}$$

Algunos contraejemplos:

$$egin{aligned} |W_{561}| &= 320 \ |W_{1729}| &= 1296 \ |W_{2465}| &= 1792 \end{aligned}$$

Pero ...

Se puede demostrar que existe una cantidad infinita de números compuestos n tales que:

$$|W_n|>\frac{n-1}{2}$$

Pero ...

Se puede demostrar que existe una cantidad infinita de números compuestos n tales que:

$$|W_n|>\frac{n-1}{2}$$

¿Cómo podemos solucionar el problema?

Una segunda noción de testigo

Vamos a considerar testigos de la forma

$$a^{\frac{n-1}{2}} mod n$$

Estos testigos están bien definidos ya que n es impar

¿Qué valores toman los testigos si n es un número primo? ¿Qué valores toman si n es compuesto?

Un ejemplo para el caso primo

Para p = 7:

```
1^3 \mod 7 = 1 \mod 7 = 1
2^3 \mod 7 = 8 \mod 7 = 1
3^3 \mod 7 = 27 \mod 7 = 6
4^3 \mod 7 = 64 \mod 7 = 1
5^3 \mod 7 = 125 \mod 7 = 6
6^3 \mod 7 = 216 \mod 7 = 6
```

Vale decir, $a^3 \equiv 1 \mod 7$ o $a^3 \equiv -1 \mod 7$

Un ejemplo para el caso primo

¿Se puede generalizar el resultado a cualquier número primo impar?

Definiendo una segunda noción de testigo

Si p es un número primo impar y $a \in \{1, \dots, p-1\}$, entonces

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

Veamos la demostración en la pizarra

 Usted tiene las herramientas necesarias para demostrar este teorema

Definiendo una segunda noción de testigo

Sea

$$egin{align} W_n^+ &= \{a \in \{1,\dots,n-1\} \mid a^{rac{n-1}{2}} \equiv 1 \mod n \} \ W_n^- &= \{a \in \{1,\dots,n-1\} \mid a^{rac{n-1}{2}} \equiv -1 \mod n \} \ \end{cases}$$

Definiendo una segunda noción de testigo

Sea

$$egin{aligned} W_n^+ &= \{a \in \{1, \dots, n-1\} \mid a^{rac{n-1}{2}} \equiv 1 \mod n \} \ W_n^- &= \{a \in \{1, \dots, n-1\} \mid a^{rac{n-1}{2}} \equiv -1 \mod n \} \end{aligned}$$

Teorema: si p es un número primo impar, entonces

$$|W_p^+| = |W_p^-| = rac{p-1}{2}$$

Tenemos que ir al mundo de los polinomios en módulo un número dado

Sea
$$r(x) = x^2 - 1$$

¿Cuántas raíces tiene r(x) en módulo 7? 2

Para a=1 y a=6 se tiene que $r(a)\equiv 0 \mod 7$

¿Cuántas raíces tiene r(x) en módulo 77?

¿Cuántas raíces tiene r(x) en módulo 77? 4 1, 34, 43 y 76 son raíces del polinomio

¿Qué pasó? ¿Por qué la diferencia entre 7 y 77?



Los polinomios tienen las propiedades usuales cuando los evaluamos en módulo un primo

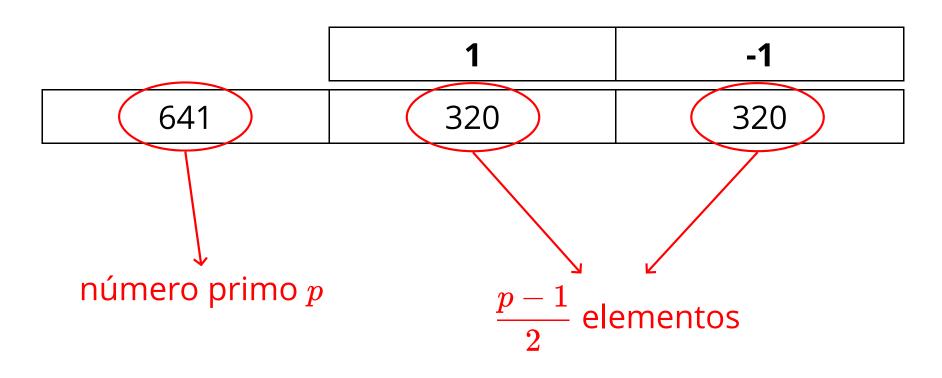
Teorema: si p es un número primo y r(x) es un polinomio de grado k, entonces r(x) tiene a lo más k raíces en módulo p

Con esto podemos demostrar la caracterización de W_p^+ y W_p^- para p primo impar

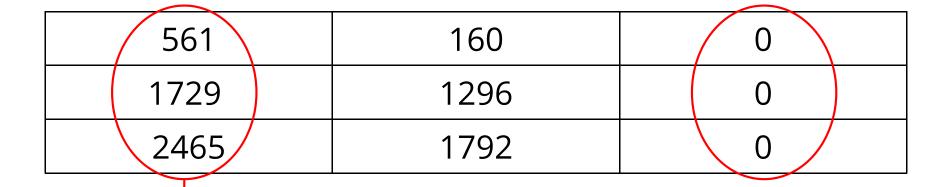
Vamos nuevamente a la pizarra

 Y nuevamente tiene las herramientas necesarias para demostrar el teorema

	1	-1
641	320	320
561	160	0
1729	1296	0
2465	1792	0
1891	225	225
2047	121	121
2701	324	324

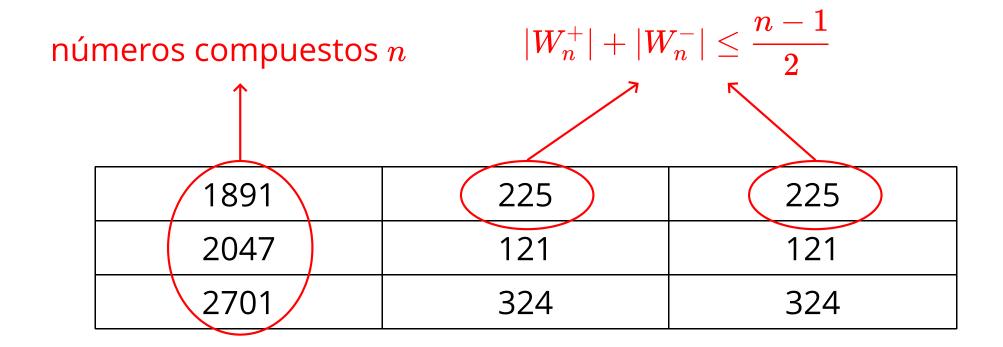


1 -1



números compuestos

1 -1



La formalización de los patrones anteriores

Teorema: Sea $n=n_1\cdot n_2$, donde $n_1,n_2\geq 3$ y $MCD(n_1,n_2)=1.$ Si existe $a\in\{2,\ldots,n-1\}$ tal que $a^{\frac{n-1}{2}}\equiv -1\mod n$, entonces:

$$|W_n^+| + |W_n^-| \leq rac{n-1}{2}$$

Y un último ingrediente

Necesitamos distinguir el caso cuando un número $n \geq 2$ no es primo porque es potencial no trivial de otro número

ullet Vale decir, $n=a^b$ con $b\geq 2$

Por ejemplo, 36 no es primo ya que $36=6^2$

Veamos en la pizarra un algoritmo de tiempo polinomial para resolver este problema

Finalmente el código

```
def test primalidad(n: int, k: int) -> bool:
       Argumentos:
           n: int - n >= 1
           k: int - k >= 1
       Retorna:
           bool - True si n es un numero primo, y False en caso contrario.
           La probabilidad de error del test es menor o igual a 2**(-k),
 9
           y esta basado en el test de primalidad de Solovay-Strassen
       0.00
10
11
       if n == 1:
12
           return False
13
       elif n == 2:
14
           return True
15
       elif n \% 2 == 0:
                                         Verifica si n es potencia no
16
17
       elic es potencia(n):
                                             trivial de un número
18
19
       else:
```

Finalmente el código

```
neg = 0
           for i in range(1, k+1):
                a = random.randint(2, n-1)
                if mcd(a, n) > 1:
                    return False
               else:
                    b = pow(a, (n-1)//2, n)
                    if b == n - 1:
 9
                        neg = neg + 1
10
                    elif b != 1:
11
                        return False
12
           if neg > 0:
13
               return True
14
           else:
15
                return False
```

Y en la práctica ...

```
1 if name == " main ":
   594363236250881445679738443300610044271230329506694061456935493654987499908267
   837823162990672937913416793547138262131162027654525159743671145416885026759510
   967807798396037679273587887606706633886423937222779033920335019140885692470045
   389062224534954730489613866855218857728804741777937870098279279181986655311360
   896681010943076506752842990211660721362674656217273071452543976542283204562818
   9761714003
       0 =
   594853230520675412480703452656865320286661579071458125732625920078118304621513
   327159370949369506632630618864234462347497091715658932838218340863551767851586
   359234950892172187401638485582112316673768053774021786114956665205696743968971
   215530824202884686449465635361231768163225461580241273009020222619625670052199
   141974408714888109491926351891091769195114042400795276168620861606789205367727
   8599271911
       t1 = time.time()
       rP = test primalidad(P, 100)
      t2 = time.time()
       rQ = test primalidad(Q, 100)
       t3 = time.time()
       rPQ = test primalidad(P*Q, 100)
       t4 = time.time()
10
       print("P: ", rP, "- tiempo: ", t2 - t1, " segundos")
11
       print("Q: ", rQ, "- tiempo: ", t3 - t2, " segundos")
12
       print("P*Q: ", rPQ, "- tiempo: ", t4 - t3, " segundos")
13
```

Y en la práctica ...

P: True - tiempo: 1.2840402126312256 segundos

Q: True - tiempo: 1.2941510677337646 segundos

P*Q: False - tiempo: 0.343120813369751 segundos

¿Cuál es la probabilidad de error del algoritmo?

Muestre que la probabilidad de error del algoritmo está acotado superiormente por $\left(\frac{1}{2}\right)^k$

¿Utilizaría este algoritmo en la práctica?

¿Cuál es la probabilidad de error del algoritmo?

```
neq = 0
           for i in range(1, k+1):
               a = random.randint(2, n-1)
               if mcd(a,n) > 1:
                   return False
 6
               else:
                   b = pow(a, (n-1)//2, n)
                   if b == n - 1:
 9
                       neg = neg + 1
10
                   elif b != 1:
11
                       return False
                                             El algoritmo solo puede
12
           if neg > 0:
13
               return True
                                            equivocarse en este paso
14
           else:
15
               return False
```

¿Cuál es la probabilidad de error del algoritmo?

¿Cómo se puede equivocar el algoritmo si n es primo? ¿Cuál es la probabilidad de retornar **false** en este caso?

¿Cómo se puede equivocar el algoritmo si n es compuesto? ¿Cuál es la probabilidad de retornar **true** en este caso?