Ignacio Méndez Pérez
IIC2283 - 2022 - 2

Aritmética Modular y más

Aritmética Modular

Dados dos números $a, b \in \mathbb{Z}$, si $b > 0$ entonces $\alpha, \beta \in \mathbb{Z}$ tales que $0 \leq \beta \leq b$ y

$$a = \alpha \cdot b + \beta$$

Además, estos números α, β **son únicos**.

β es llamado **el resto de la división entera entre a y b**, y es denotado como

$$a \bmod b := \beta$$

Definición

$$a \equiv_n b \quad \text{si, y solo si, } n \text{ divide a } (b - a)$$

Usamos la notación $n \mid m$ para indicar que n divide a m.

- $a \equiv_n b$ si $n \mid (b - a)$.

Proposición

1. Si $a \equiv_n b$ y $c \equiv_n d$, entonces:

$$(a + c) \equiv_n (b + d)$$
$$(a \cdot c) \equiv_n (b \cdot d)$$

Exponenciación rápida: Calculando $a^b \bmod b$

Utilizamos el siguiente algoritmo:

```
EXP(a, b, n)
    if b = 1 then return a mod n
    else if b es par then
        val := EXP(a, b/2, n)
        return (val · val) mod n
    else
        val := EXP(a, (b−1)/2, n)
        return (val · val · a) mod n
```

Ignacio Méndez Pérez
IIC2283 - 2022 - 2

Máximo común Divisor

Definición

Sea $a, b \in \mathbb{Z} - \{0\}$. Se define el máximo común divisor $\gcd(a, b)$ de a y b como el mayor número d tal que $d \mid a$ y $d \mid b$.

En otras palabras, $\gcd(a, b)$ es el máximo del conjunto

$$D_{a,b} = \{\, c \in \mathbb{Z} \mid c \mid a \wedge c \mid b \,\}$$

Proposición

Para todo $a, b \in \mathbb{Z} - \{0\}$, $\gcd(a, b) = \gcd(b, a \bmod b)$

Demostración

Vamos a demostrar que para $c \in \mathbb{Z} - \{0\}$

$$c \mid a \text{ y } c \mid b \quad \Leftrightarrow \quad c \mid b \text{ y } c \mid (a \bmod b)$$

De esto se concluye que $\gcd(a, b) = \gcd(b, a \bmod b)$.

Sabemos que $a = \alpha \cdot b + (a \bmod b)$

($\Rightarrow$) Suponga que $c \mid a$ y $c \mid b$.
Dado que $(a \bmod b) = a - \alpha \cdot b$, concluimos que $c \mid (a \bmod b)$

($\Leftarrow$) Suponga que $c \mid b$ y $c \mid (a \bmod b)$.
Dado que $a = \alpha \cdot b + (a \bmod b)$, tenemos que $c \mid a$.

De lo anterior, concluimos la siguiente identidad para $a > 0$:

$$\gcd(a, b) = \begin{cases} a & b = 0 \\ \gcd(b, a \bmod b) & b > 0 \end{cases}$$

Usamos esta identidad para generar un algoritmo para calcular el máximo común divisor, el cuales conocido como **Algoritmo de Euclides**.

MCD$(a,\ b)$
  if $a = 0$ **and** $b = 0$ **then return** error
  else if $a = 0$ **then return** b
  else if $b = 0$ **then return** a
  else if $a \geq b$ **then return MCD**$(b,\ a \bmod b)$
  else return MCD$(a,\ b \bmod a)$

Lema

Si $a \geq b$ y $b > 0$, entonces $(a \bmod b) < \frac{a}{2}$

Demostración

Si $b > \frac{a}{2}$:

$$a \bmod b = a - b < a - \frac{a}{2} = \frac{a}{2}$$

Si $b < \frac{a}{2}$:

$$a \bmod b < b < \frac{a}{2}$$

Si $b = \frac{a}{2}$ (a debe ser par):

$$a \bmod b = 0 < b = \frac{a}{2}$$

Definición

b es **inverso de a en módulo n** si

$$a \cdot b \equiv_n 1$$

Identidad de Bézout

Para cada $a, b \in \mathbb{N}$ tales que $a \neq 0$ o $b \neq 0$, existen $s, t \in \mathbb{Z}$ tales que:

$$\gcd(a, b) \;=\; s \cdot a + t \cdot b$$

Demostración:

Sean $a, b \in \mathbb{Z} - \{0\}$. Tomamos:

$$S = \{\, a \cdot x + b \cdot y \mid x, y \in \mathbb{Z} \quad \wedge \quad a \cdot x + b \cdot y > 0 \,\}$$

Sabemos que $S \neq \emptyset$ dado que contiene a a o $-a$ (con $x = \pm 1$ y $y = 0$). Como S es un conjunto no vacío de números positivos, tiene un mínimo elemento $d = a \cdot s + b \cdot t$. Para mostrar que d es el máximo común divisor de a y b, primero debemos mostrar que sí es un común divisor, y que para cualquier otro común divisor c, tenemos que $c \leq d$.

Sabemos que:

$$a = d \cdot q + r \qquad (a \bmod q \;=\; r)$$

Y sabemos que $r \in S \cup \{0\}$ dado que:

$$\begin{aligned} r &= a - q \cdot d \\ &= a - q(a \cdot s + b \cdot t) \\ &= a \cdot (1 - q \cdot s) - b \cdot qt \end{aligned}$$

De este modo, r es de la forma $a \cdot x + b \cdot y$, y por lo tanto $r \in S \cup \{0\}$. Sin embargo, $0 \leq r < d$, y d es el entero positivo más pequeño en S: El resto r puede por lo tanto NO estar en S, haciendo r necesariamente igual a 0. Esto implica que d es divisor de a. Similarmente, d también es divisor de b, y, por lo tanto, $d \mid a$ y $d \mid b$.

Ahora, sea c cualquier divisor común de a y b, esto es, que existen u y v tal que $a = c \cdot u$, y $b = c \cdot v$:

$$\begin{aligned} d &= a \cdot s + b \cdot t \\ &= c \cdot u \cdot s + c \cdot v \cdot t \\ &= c(u \cdot s + v \cdot t) \end{aligned}$$

Luego, $c \mid d$. Y como $d > 0$, esto implica que $c \leq d$.

Teorema

a tiene inverso en módulo n si y sólo si $\gcd(a, n) = 1$.

Demostración

($\Rightarrow$) Suponga que b es inverso de a en módulo n:

$$a \cdot b \equiv_n 1$$

Se deduce que $a \cdot b = \alpha \cdot n + 1$, por lo que $1 = a \cdot b - \alpha \cdot n$.
Concluimos que si $c \mid a$ y $c \mid n$, entonces $c \mid 1$. Por lo tanto c debe ser igual a 1, de lo contrario concluimos que $\gcd(a, n) = 1$.

($\Leftarrow$) Suponga que $\gcd(a, n) = 1$. Por la identidad de Bézout existen $s, t \in \mathbb{Z}$ tales que:

$$1 \quad = \quad s \cdot n + t \cdot a$$

Entonces $a \cdot t \equiv_n 1$. Así a tiene inverso en módulo n.

Sabemos que **MCD** es un algoritmo eficiente para calcular el máximo común divisor entre dos números.

¡Pero este algoritmo puede hacer más! Puede ser extendido para calcular s y t tales que:

$$\gcd(a, b) = s \cdot a + t \cdot b$$

Vamos a usar este algoritmo para calcular inversos modulares.

Ignacio Méndez Pérez
IIC2283 - 2022 - 2

Algoritmo Extendido de Euclides

Suponga que $a \geq b$, y defina la siguiente **sucesión**:

$$\begin{aligned} r_0 &= a \\ r_1 &= b \\ r_i &= r_{i-2} \bmod r_{i-1} \quad (i \geq 2) \end{aligned}$$

y calculamos esta sucesión hasta un número k tal que $r_k = 0$.
Luego, $\boldsymbol{r_{k-1}} = \mathbf{gcd}(\boldsymbol{a}, \boldsymbol{b})$.

Al mismo tiempo, podemos ir calculando dos sucesiones s_i, t_i tales que:

$$r_i = s_i \cdot a + t_i \cdot b$$

Tenemos que:

$$\gcd(a, b) = r_{k-1} = s_{k-1} \cdot a + t_{k-1} \cdot b$$

Sean:

$$\begin{aligned} s_0 &= 1 \qquad & t_0 &= 0 \\ s_1 &= 0 \qquad & t_1 &= 1 \end{aligned}$$

Se tiene que:

$$\begin{aligned} r_0 &= s_0 \cdot a + \cancelto{0}{t_0 \cdot b} = a \\ r_1 &= \cancelto{0}{s_1 \cdot a} + t_1 \cdot b = b \end{aligned}$$

Dado que $r_{i-1} = \overbrace{\left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor}^{\text{División parte entera}} \cdot r_i + \overbrace{r_{i-1} \bmod r_i}^{\text{Resto}}$, tenemos que:

$$r_{i-1} = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot r_i + r_{i+1}$$

Por lo tanto:

$$s_{i-1} \cdot a + t_{i-1} \cdot b = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot (s_i \cdot a + t_i \cdot b) + r_{i+1}$$

Concluimos que:

$$r_{i+1} = \left(s_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot s_i\right) \cdot a + \left(t_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot t_i\right) \cdot b$$

Definimos entonces:

$$\begin{aligned} s_{i+1} &= s_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot s_i \\ t_{i+1} &= t_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot t_i \end{aligned}$$

Ejemplo

Vamos a usar el algoritmo para $a = 60$ y $b = 13$

Inicialmente:

$$
\begin{array}{lll}
r_0 = 60 & s_0 = 1 & t_0 = 0 \\
r_1 = 13 & s_0 = 0 & t_0 = 1
\end{array}
$$

Entonces tenemos que:

$$
\begin{aligned}
r_2 &= r_0 \bmod r_1 \\
s_2 &= s_0 - \left\lfloor \frac{r_0}{r_1} \right\rfloor \cdot s_1 \\
t_2 &= t_0 - \left\lfloor \frac{r_0}{r_1} \right\rfloor \cdot t_1
\end{aligned}
$$

Por lo tanto:

$$r_2 = 8 \qquad s_2 = 1 \qquad t_2 = -4$$

Y el proceso continúa:

$$
\begin{array}{lll}
r_3 = 5 & s_3 = -1 & t_3 = 5 \\
r_4 = 3 & s_4 = 2 & t_4 = -9 \\
r_5 = 2 & s_5 = -3 & t_5 = 14 \\
r_6 = 1 & s_6 = 5 & t_6 = -23 \\
r_7 = 0 & s_7 = -13 & t_7 = 60
\end{array}
$$

Tenemos que: $1 = 5 \cdot 60 + (-23) \cdot 13$

Dados dos números naturales a y n, con $n \geq 2$, si el inverso de a en módulo n existe el siguiente algoritmo lo retorna, y en caso contrario indica que no existe.

Inverso(a, n)
  if MCD(a, n) > 1 **then** *no_existe_inverso*
  else
    $s_0 := 1$
    $t_0 := 0$
    $s_1 := 0$
    $t_1 := 1$
    $r_0 := n$
    $r_1 := a$
    while $r_1 > 1$ **do**
      $aux\_s := s_0 - \left\lfloor \frac{r_0}{r_1} \right\rfloor \cdot s_1$
      $s_0 := s_1$
      $s_1 := aux\_s$
      $aux\_t := t_0 - \left\lfloor \frac{r_0}{r_1} \right\rfloor \cdot t_1$
      $t_0 := t_1$
      $t_1 := aux\_t$
      $r_0 := r_1$
      $r_1 := s_1 \cdot n + t_1 \cdot a$
    return t_1

Ejemplo para mortales:

¿Cuántas veces cabe el 54 en 888?

$$\gcd(888{,}54) = \ 6:$$

$$888 = 54(16) + 24$$

$$54 = 24(2) + \boxed{6}$$

Nuesto último resto $\neq 0$ será el gcd

$$24 = 6(4) + 0$$

Pequeño Teorema de Fermat

Sea p un número primo. Si $a \in \{0, \dots, p-1\}$, entonces:

$$a^p \equiv a \pmod p$$

Corolario

Sea p un número primo. Si $a \in \{1, \dots, p-1\}$, entonces:

$$a^{p-1} \equiv 1 \pmod p$$

Demostración

Por teorema anterior sabemos que

$$a^p \equiv a \pmod p$$

Por lo tanto, existe $\alpha \in \mathbb{Z}$ tal que:

$$a^p - a = \alpha \cdot p$$

Dado que $a \mid (a^p - a)$, se tiene que $a \mid (\alpha \cdot p)$

Por lo tanto, dado que $a \in \{1, \dots, p-1\}$ y p es un número primo, se concluye que $a \mid \alpha$. Entonces $(a^p - 1) = \frac{\alpha}{a} \cdot p$, donde $\frac{\alpha}{a}$ es un número entero.

- Concluimos que $a^{p-1} \equiv 1 \quad mod\ p$.

Test de primalidad: Primera versión

El test de primalidad que vamos a estudiar está basado en estas propiedades ($n \geq 2$):

1. Si n es primo y $a \in \{1, \dots, n-1\}$, entonces:

$$a^{n-1} \equiv 1 \pmod{n}$$

2. Si n es compuesto, entonces existe $a \in \{1, .., n-1\}$ tal que

$$a^{n-1} \not\equiv 1 \pmod{n}$$

Demostración

Suponga que n es compuesto. Sea $a \in \{1, \dots, n-1\}$ tal que $\gcd(a, n) > 1$.
Luego, a no tiene inverso en módulo n.
Concluimos que

$$a^{n-1} \not\equiv 1 \pmod{n}$$

Dado que a^{n-2} no puede ser inverso de a en módulo n.

Para $n \geq 2$, defina el conjunto $\mathbb{Z}_n^*$ como:

$$\mathbb{Z}_n^* = \{a \in \{1, \dots, n-1\} \mid \gcd(a, n) = 1\}$$

Es decir, $\mathbb{Z}_n^*$ es el conjunto de todos los primos relativos de n que son menores a él.

Suponga que n es compuesto. Luego, si $a \in \{1, \dots, n-1\} - \mathbb{Z}_n^*$, entonces $a^{n-1} \not\equiv 1 \pmod{n}$.

Test de primalidad entonces depende de qué tan grande es $\mathbb{Z}_n^*$.
Supongamos que $|\mathbb{Z}_n^*| \leq \left\lfloor \frac{n}{2} \right\rfloor$ para cada número compuesto $n \geq 2$.

```
TestPrimalidad1(n)
    sea a un número elegido de manera uniforme desde {1, ..., n − 1}
    if EXP(a, n − 1, n) ≠ 1
        then return COMPUESTO
    else
        return PRIMO
```

Ignacio Méndez Pérez
IIC2283 - 2022 - 2

Ejercicio:

De un algoritmo que reciba como parámetros a dos números enteros $n \geq 2$ y $k \geq 1$, y determina si n es un número primo con probabilidad de error menor o igual a $\left(\frac{1}{2}\right)^k$.

```
TestPrimalidad2(n, k)
    sea a_1, ..., a_k una secuencia de números elegidos de
                manera uniforme e independiente desde {1, ..., n − 1}
    for i := 1 to k do
        if EXP(a_i, n − 1, n) ≠ 1
            then return COMPUESTO
    return PRIMO
```

¿Pero la probabilidad de error de **TestPrimalidad2** está bien acotada?

Supusimos que $|\mathbb{Z}_n^*| \leq \left\lfloor \frac{n}{2} \right\rfloor$ para cada número compuesto $n \geq 2$.

¿Es correcta esta suposición?

Considere la **función de Euler**: $\phi: \mathbb{N} \to \mathbb{N}$ definida como:

$$\phi(n) \begin{cases} 0 & n = 1 \\ |\mathbb{Z}_n^*| & n > 1 \end{cases}$$

Teorema

$$\phi(n) \in \Omega\left(\frac{n}{\log_2(\log_2 n)}\right)$$

Conclusión

Para cada número n, el conjunto $\mathbb{Z}_n^*$ tiene un número de elementos cercano a n.

- No es cierto que $|\mathbb{Z}_n^*| \leq \left\lfloor \frac{n}{2} \right\rfloor$ para cada número compuesto $n \geq 2$.
- No podemos basar nuestro test en los elementos del conjunto $(\{1, \dots, n-1\} - \mathbb{Z}_n^*)$.

Test de primalidad: Segunda versión

Una observación importante: si n es compuesto, entonces puede existir $a \in \mathbb{Z}_n^*$ tal que $a^{n-1} \not\equiv 1 \pmod{n}$.

- Por ejemplo: $3^{15} \bmod 16 = 11$.df

En lugar de considerar $\mathbb{Z}_n^*$ en el test de primalidad, consideramos:

$$J_n \quad = \quad \{a \in \mathbb{Z}_n^* \mid a^{n-1} \equiv 1 \pmod{n}\}$$

Si demostramos que para cada número compuesto n se tiene que $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$, entonces tenemos un test de primalidad.

- Puesto que para p primo: $|J_p| = |\mathbb{Z}_p^*| = p - 1$.

Recuerde que en nuestros algoritmos consideramos $n \geq 2$.

```
TestPrimalidad3(n, k)
    sea a_1, ..., a_k una secuencia de números elegidos de
                manera uniforme e independiente desde {1, ..., n − 1}
    for i := 1 to k do
        if MCD(a_i, n) > 1 then return COMPUESTO
        else
            if EXP(a_i, n − 1, n) ≠ 1
                then return COMPUESTO
    return PRIMO
```

Teoría de Grupos

Definición

Un conjunto G y una función (total) $\circ : G \times G \to G$ forman un **grupo** si:

1. (**Asociatividad**) Para cada $a, b, c \in G$:

$$(a \circ b) \circ c = a \circ (b \circ c)$$

2. (**Elemento neutro**) Existe $e \in G$ tal que para cada $a \in G$:

$$a \circ e = e \circ a = a$$

3. (**Inverso**) Para cada $a \in G$, existe $b \in G$:

Propiedades básicas

- Neutro es único.
- Inverso de cada elemento a es único.

Recordatorio...

$\subset \rightarrow$ Subconjunto.
$\subseteq \rightarrow$ Subconjunto o igual.
$\subsetneq \rightarrow$ Subconjunto pero no igual.
$\not\subseteq \rightarrow$ Ni subconjunto ni igual.

Definición

$(H,\circ)$ es un subgrupo de un grupo $(G,\circ)$, para cada $\emptyset \subsetneq H \subseteq G$, si $(H,\circ)$ es un grupo.

Propiedades básicas

- Si e_1 es el neutro en $(G,\circ)$ y e_2 es el neutro en $(H,\circ)$, entonces $e_1 = e_2$.
- Para cada $e \in H$, si b es el inverso de a en $(G,\circ)$ y c es el inverso de a en $(H,\circ)$, entonces $c = b$.

Teorema de Lagrange

Si $(G,\circ)$ es un grupo finito y $(H,\circ)$ es un subgrupo de $(G,\circ)$, entonces $|H|$ divide a $|G|$.

Demostración

Suponga que e es el elemento neutro de $(G,\circ)$ y a^{-1} es el inverso de a en $(G,\circ)$.
Sea $\sim$ una relación binaria sobre G definida como:

$$a \sim b \quad \text{si y sólo si} \quad b \circ a^{-1} \in H$$

Lema

$\sim$ es una relación de equivalencia.

Demostración

(**Refleja**) $a \sim a$ ya que $a \circ a^{-1} = e$ y $e \in H$.

(**Simétrica**) Suponga que $a \sim b$. Demostramos que $b \sim a$.

Dado que $a \sim b$: $b \circ a^{-1} \in H$, tenemos que demostrar que $a \circ b^{-1} \in H$.
Tenemos que:

$$\begin{aligned}(b \circ a^{-1}) \circ (a \circ b^{-1}) &= \big(b \circ (a^{-1} \circ a)\big) \circ b^{-1} \\ &= (b \circ e) \circ b^{-1} \\ &= b \circ b^{-1} \\ &= e\end{aligned}$$

De la misma forma que $(a \circ b^{-1}) \circ (b \circ a^{-1}) = e$. Por lo tanto,

$$(b \circ a^{-1})^{-1} = a \circ b^{-1}$$

Concluimos que $a \circ b^{-1}$ está en H, ya que $(H,\circ)$ es un subgrupo de $(G,\circ)$.

(**Transitiva**) Suponga que $a \sim b$ y $b \sim c$. Tenemos que demostrar que $a \sim c$.

Por hipótesis: $b \circ a^{-1} \in H$ y $c \circ b^{-1} \in H$. Tenemos que demostrar que $c \circ a^{-1} \in H$.
Pero $(c \circ b^{-1}) \circ (b \circ a^{-1}) = c \circ a^{-1}$ y $\circ$ es cerrada en H.
Por lo tanto: $c \circ a^{-1} \in H$.

Lema

Sea $[a]_\sim$ la clase de equivalencia de $a \in G$ bajo la relación $\sim$. Luego:

1. $[e]_\sim = H$.
2. Para cada $a, b \in G$: $|[a]_\sim| = |[b]_\sim|$.

Demostración

1. Se tiene que:

$$\begin{aligned} a \in [e]_\sim &\Leftrightarrow e \sim a \\ &\Leftrightarrow a \circ e^{-1} \in H \\ &\Leftrightarrow a \circ e \in H \\ &\Leftrightarrow a \in H \end{aligned}$$

2. Sean $a, b \in G$, y defina la función f de la siguiente forma:

$$f(x) = x \circ (a^{-1} \circ b)$$

Se tiene que:

$$\begin{aligned} x \in [a]_\sim &\Rightarrow a \sim x \\ &\Rightarrow x \circ a^{-1} \in H \\ &\Rightarrow (x \circ a^{-1}) \circ e \in H \\ &\Rightarrow (x \circ a^{-1}) \circ (b \circ b^{-1}) \in H \\ &\Rightarrow \big(x \circ (a^{-1} \circ b)\big) \circ b^{-1} \in H \\ &\Rightarrow f(x) \circ b^{-1} \in H \\ &\Rightarrow b \sim f(x) \\ &\Rightarrow f(x) \in [b]_\sim \end{aligned}$$

Por lo tanto: $f: [a]_\sim \to [b]_\sim$

Vamos a demostrar que f es una **biyección**, de lo cual concluimos que $|[a]_\sim| = |[b]_\sim|$.

f es inyectiva (1-1):

$$
\begin{aligned}
f(x) = f(y) \quad &\Rightarrow \quad x \circ (a^{-1} \circ b) = y \circ (a^{-1} \circ b) \\
&\Rightarrow \quad \big(x \circ (a^{-1} \circ b)\big) \circ (b^{-1} \circ a) = \big(y \circ (a^{-1} \circ b)\big) \circ (b^{-1} \circ a) \\
&\Rightarrow \quad \big(x \circ (a^{-1} \circ (b \circ b^{-1}) \circ a)\big) = \big(y \circ (a^{-1} \circ (b \circ b^{-1}) \circ a)\big) \\
&\Rightarrow \quad \Big(x \circ \big((a^{-1} \circ e) \circ a\big)\Big) = \Big(y \circ \big((a^{-1} \circ e) \circ a\big)\Big) \\
&\Rightarrow \quad \big(x \circ (a^{-1} \circ e)\big) = \big(y \circ (a^{-1} \circ e)\big) \\
&\Rightarrow \quad x \circ e = y \circ e \\
&\Rightarrow \quad x = y
\end{aligned}
$$

f es sobreyectiva:

$$
\begin{aligned}
y \in [b]_{\sim} \quad &\Rightarrow \quad b \sim y \\
&\Rightarrow \quad y \circ b^{-1} \in H \\
&\Rightarrow \quad (y \circ b^{-1}) \circ (a \circ a^{-1}) \in H \\
&\Rightarrow \quad \big((y \circ b^{-1}) \circ a\big) \circ a^{-1} \in H \\
&\Rightarrow \quad a \sim \big((y \circ b^{-1}) \circ a\big) \\
&\Rightarrow \quad \big((y \circ b^{-1}) \circ a\big) \in [a]_{\sim}
\end{aligned}
$$

Sea $x = \big((y \circ b^{-1}) \circ a\big)$. Tenemos que:

$$
\begin{aligned}
f(x) \quad &= \quad x \circ (a^{-1} \circ b) \\
&= \quad \big((y \circ b^{-1}) \circ a\big) \circ (a^{-1} \circ b) \\
&= \quad y \circ (b^{-1} \circ (a \circ a^{-1}) \circ b) \\
&= \quad y \circ \big((b^{-1} \circ e) \circ b\big) \\
&= \quad y \circ (b^{-1} \circ b) \\
&= \quad y \circ e \\
&= \quad y
\end{aligned}
$$

Dejamos pendiente la siguiente pregunta:

¿Qué enfoque podríamos usar para demostrar que $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$?

R: Usamos el Teorema de Lagrange.

Dado que $(J_n, \ \cdot\ )$ es un subgrupo de $(\mathbb{Z}_n^*, \ \cdot\ )$:

Si existe $a \in (\mathbb{Z}_n^* \setminus J_n)$, entonces $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$

Definición

Un **número n es de Carmichael** si $n \geq 2$, n es compuesto y $|J_n| \leq |\mathbb{Z}_n^*|$.

Ejemplo
561, 1105 y 1792 son números de Carmichael.

Teorema
Existe un número infinito de números de Carmichael.

Conclusión: Este test de primalidad NO va a funcionar, pero no todo esta perdido:

En lugar de utilizar J_n, vamos a usar las herramientas que desarrollamos sobre el siguiente conjunto (n impar):

$$S_n \quad = \quad \left\{ a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n} \ \text{ ó } \ a^{\frac{n-1}{2}} \equiv -1 \pmod{n} \right\}$$

Vamos a diseñar un test de primalidad considerando los conjuntos:

$$\begin{aligned} S_n^+ &= \left\{ a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n} \right\} \\ S_n^- &= \left\{ a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv -1 \pmod{n} \right\} \end{aligned}$$

Así, podemos definir S_n a partir de estos conjuntos:

$$S_n = S_n^+ \cup S_n^-$$

Para hacer esto necesitamos estudiar algunas propiedades de los conjuntos S_n^+, S_n^- y S_n.

- Consideramos primero el caso en que n es primo, y luego el caso en que n es compuesto.

Proposición 1

Si $n \geq 3$ es primo, entonces $S_n = \mathbb{Z}_n^*$.

Demostración

Si $a \in \{1, \ldots, n-1\}$, tenemos que $a^{n-1} \equiv 1 \pmod{n}$

Por lo tanto $\left(a^{\frac{n-1}{2}}\right)^2 \equiv 1 \pmod{n}$, de lo cual se deduce que:

$$\left(a^{\frac{n-1}{2}} + 1\right) \cdot \left(a^{\frac{n-1}{2}} - 1\right) \equiv 0 \pmod{n}$$

Así, dado que n es primo se concluye que $a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$ ó $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$.

Proposición 2

Si $n \geq 3$ es primo: $|S_n^+| = |S_n^-| = \frac{n-1}{2}$

Para demostrar la proposición, primero vamos a demostrar un lema.

Sea $p(x)$ un polinomio:

$$p(x) = \sum_{i=0}^{k} a_i x^i,$$

donde $k \geq 1$, cada $a_j \in \{0, \dots, n-1\}$ para $0 \leq j \leq k-1$, y $a_k \neq 0$.

Decimos que a es una **raíz de $p(x)$ en módulo n** si:

$$p(a) \equiv_n 0$$

Lema

$p(x)$ tiene a lo más k raíces en módulo n.

Demostración

Decimos que dos polinomios $p_1(x)$ y $p_2(x)$ son **congruentes en módulo n** si para todo $a \in \{0, \dots, n-1\}$:

$$p_1(a) \equiv_n p_2(a)$$

Sea a una raíz de $p(x)$ en módulo n. Vamos a demostrar que existe un polinomio $q(x)$ de grado $k-1$ tal que:

$$p(x) \equiv_n (x-a) \cdot q(x)$$

Veamos que al demostrar esta propiedad se concluye que el lema es cierto.

Si c es una raíz de $p(x)$ en módulo n, entonces $p(c) \equiv_n 0$.

- Como $p(x) \equiv_n (x-a) \cdot q(x)$, concluimos que $(c-a) \cdot q(c) \equiv_n 0$.

Dado que n es primo, si $d \cdot e \equiv_n 0$, entonces $d \equiv_n 0$ o $e \equiv_n 0$.

- Tenemos entonces que $c \equiv_n a$ o $q(c) \equiv_n 0$.

Así, tenemos que c es la raíz a que ya habíamos identificado o es una raíz de $q(x)$ en módulo n.

Concluimos que el número de raíces de $p(x)$ en módulo n es menor o igual a uno más el número de raíces de $q(x)$ en módulo n.

- Como $q(x)$ tiene grado $k-1$, si continuamos usando este argumento (o usamos inducción) concluimos que el número de raíces de $p(x)$ es menor o igual a k.

Nótese que el argumento anterior no funciona si n es compuesto.

- Dado que podemos tener d y e tales que $d \cdot e \equiv_n 0$, $d \not\equiv_n 0$ y $e \not\equiv_n 0$.

De hecho, si n es compuesto no es necesariamente cierto que el número de raíces de un polinomio está acotado superiormente por su grado.

Ejemplo

Si $n = 35$, tenemos que $5 \cdot 7 \equiv_{35} 0$, pero $5 \not\equiv_{35} 0$ y $7 \not\equiv_{35} 0$.
En este caso tenemos cuatro raíces para el polinomio $p(x) = x^2 - 1$.

- Ya que $1^2 \equiv_{35} 1$, $6^2 \equiv_{35} 1$, $29^2 \equiv_{35} 1$ y $34^2 \equiv_{35} 1$.

Volvemos entonces a la demostración de que existe un polinomio $q(x)$ de grado $k-1$ tal que:

$$p(x) \equiv_n (x-a) \cdot q(x)$$

Definimos $q(x)$ como:

$$q(x) = \sum_{i=0}^{k-1} b_i x^i$$

donde $b_i = a_{i+1} + a_{i+2} \cdot a + \cdots + a_k \cdot a^{k-1-i}$
Se tiene que:

$$\begin{aligned}
(x-a) \cdot q(x) &= \left(\sum_{i=0}^{k-1} b_i x^{i+1}\right) + \left(\sum_{i=0}^{k-1} (-a \cdot b_i) x^{i+1}\right) \\
&= \left(\sum_{i=1}^{k} b_{i-1} x^{i}\right) + \left(\sum_{i=0}^{k-1} (-a \cdot b_i) x^{i+1}\right) \\
&= b_{k-1} \cdot x^k + \left(\sum_{i=1}^{k-1} (b_{i-1} - a \cdot b_i) x^{i+1}\right) - a \cdot b_0
\end{aligned}$$

Así, dado que:

$$b_{k-1} = a_k$$

Y dado que para $i \in \{1, \dots, k-1\}$:

$$\begin{aligned}(b_{i-1} - a \cdot b_i) &= a_i + a_{i+1} \cdot a + \cdots + a_k \cdot a^{k-i} - a \cdot \left(a_{i+1} + \cdots + a_k \cdot a^{k-1-i}\right) \\ &= a_i + a_{i+1} \cdot a + \cdots + a_k \cdot a^{k-i} - a_{i+1} \cdot a - \cdots - a_k \cdot a^{k-1} \\ &= a_i\end{aligned}$$

Concluimos que:

$$(x-a) \cdot q(x) = \left(\sum_{i=1}^{k} a_i \cdot x^i\right) - a \cdot b_0$$

Pero:

$$\begin{aligned}-a \cdot b_0 &= -a \cdot (a_1 + a_2 \cdot a + \cdots + a_k \cdot a^{k-1}) \\ &= -a_1 \cdot a - a_2 \cdot a^2 - \cdots - a_k \cdot a^k\end{aligned}$$

De lo cual deducimos que:

$$a_0 \equiv_n - a \cdot b_0$$

ya que $a_k \cdot a^k + \cdots + a_1 \cdot a + a_0 \equiv_n 0$.

Tenemos entonces que:

$$(x-a) \cdot q(x) \equiv_n p(x)$$

Sea $R = \left\{\{b^2 \mid 1 \leq b \leq \frac{n-1}{2}\right\}$

Por el Teorema de Fermat, tenemos que:

$$R \subseteq \left\{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv_n 1\right\}$$

Además, sabemos que si $1 \leq b < c \leq \frac{n-1}{2}$ y $b^2 \equiv_n c^2$:

$$(c-b) \cdot (c+b) \equiv_n 0$$

Así, dado que $2 \leq b + c \leq n-1$, concluimos que $b \equiv_n c$.

- Dado que n es primo.

Pero $b \equiv_n c$, no puede ser cierto puesto que $1 \leq (c-b) \leq \frac{n-1}{2}$.

- Por lo tanto: $|R| = \frac{n-1}{2}$

Además, sabemos que $p(x) = x^{\frac{n-1}{2}} - 1$ tiene a lo más $\frac{n-1}{2}$ raíces en módulo n.

- Por lo tanto: $\left|\left\{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv_n 1\right\}\right| \leq \frac{n-1}{2}$

Concluimos que:

$$\frac{n-1}{2} = |R| \leq \left|\left\{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv_n 1\right\}\right| \leq \frac{n-1}{2}$$

Por lo tanto:

$$|S_n^+| = \left|\left\{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv_n 1\right\}\right| = \frac{n-1}{2}$$

Así, dado que $|S_n| = |\mathbb{Z}_n^*| = n - 1$ y $|S_n^+| + |S_n^-| = |S_n|$, concluimos que:

$$|S_n^-| \;=\; \frac{n-1}{2}$$

■

Teorema

Sea $n = n_1 \cdot n_2$, donde $n_1, n_2 \geq 3$ y $\gcd(n_1, n_2) = 1$. Si existe $a \in \mathbb{Z}_n^*$ tal que $a^{\frac{n-1}{2}} \not\equiv_n -1$, entonces:

$$|S_n| \;\leq\; \frac{1}{2} \cdot |\mathbb{Z}_n^*|$$

Para demostrar el teorema necesitaremos el **Teorema Chino del resto**.

Teorema Chino del Resto:

Suponga que $\gcd(m, n) = 1$. Para todo a y b, existe c tal que:

$$c \equiv_m a$$
$$c \equiv_n b$$

Demostración:

Dado que $\gcd(m, n) = 1$, existen d y e tales que:

$$n \cdot d \equiv_m 1$$
$$m \cdot e \equiv_n 1$$

Sea $c = a \cdot n \cdot d + b \cdot m \cdot e$. Se tiene que:

$$c \equiv_m a$$
$$c \equiv_n b$$

Ahora, para el demostrar el teorema anterior, suponga que $e \in \mathbb{Z}_n^*$ y $a^{\frac{n-1}{2}} \equiv_n -1$

Por Teorema Chino del Resto, existe b tal que:

$$b \equiv_{n_1} a$$
$$b \equiv_{n_2} 1$$

Entonces: $a = \alpha \cdot n_1 + b$ y $1 = \beta \cdot n_2 + b$

- Por lo tanto $\gcd(b, n) = 1$, ya que $n = n_1 \cdot n_2$ y $a \in \mathbb{Z}_n^*$

Además, tenemos que:

$$\begin{aligned} b^{\frac{n-1}{2}} \quad \equiv_{n_1} \quad a^{\frac{n-1}{2}} \quad &\equiv_{n_1} -1 \\ b^{\frac{n-1}{2}} \quad &\equiv_{n_2} \quad 1 \end{aligned}$$

Dado que $n = n_1 \cdot n_2$ concluimos que:

$$b^{\frac{n-1}{2}} \not\equiv_n \quad 1$$
$$b^{\frac{n-1}{2}} \not\equiv_n -1$$

Sea $c = (b \mod n)$. Concluimos que $c \notin S_n$ y $c \in \mathbb{Z}_n^*$. Es decir:

$$S_n \subsetneq \mathbb{Z}_n^*$$

Pero se tiene que $(S_n, \ \cdot)$ es un subgrupo de $(\mathbb{Z}_n^*, \ \cdot)$.
Entonces por Teorema de Lagrange:

$$|S_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$$

Ya tenemos los ingredientes esenciales para el test de primalidad

- Sólo nos falta implementar algunas funciones auxiliares

Necesitamos desarrollar un algoritmo eficiente para determinar si un número n es la potencia (no trivial) de otro número.

Primero necesitamos una función para calcular n^k

- Usamos el algoritmo de exponenciación rápida sin considerar el módulo

```
EXP(n, k)
    if k = 1 then return n
    else if k es par then
        val := EXP(n, k/2)
        return val · val
    else
        val := EXP(n, (k-1)/2)
        return val · val · n
```

Dado un número natural $n \geq 2$, la siguiente función verifica si existen $m, k \in \mathbb{N}$ tales que $k \geq 2$ y $n = m^k$

```
EsPotencia(n)
    if n ≤ 3 then return no
    else
        for k := 2 to ⌊log₂(n)⌋ do
            if TieneRaízEntera(n, k, 1, n) then return sí
        return no
```

La siguiente función verifica si existe $m \in \{i, \ldots, j\}$ tal que $n = m^k$.

- Vale decir, la llamada **TieneRaízEntera**$(n, k, 1, n)$ verifica si n tiene raíz k-ésima entera.

```
TieneRaízEntera(n, k, i, j)
    if i = j then
        if EXP(i, k) = n then return sí
        else return no
    else if i < j then
        p := ⌊(i+j)/2⌋
        val := EXP(p, k)
        if val = n then return sí
        else if val < n then return TieneRaízEntera(n, k, p + 1, j)
        else return TieneRaízEntera(n, k, i, p − 1)
    else return no
```

Consideramos la multiplicación de números enteros como la operación básica a contar.

Tenemos que:

- En el peor caso **EsPotencia**(n) realiza $(\lfloor \log_2 n \rfloor - 1)$ llamadas a la función **TieneRaízEntera**.

- Existe $c \in \mathbb{N}$ tal que la llamada **TieneRaízEntera**$(n, k, 1, n)$ realiza en el peor caso a lo más $c \cdot \log_2 n$ llamadas a la función **EXP**.

- **EXP**(n, k) en el peor caso es $\mathcal{O}(\log_2 k)$.

Concluimos que **EsPotencia**(n) en el peor caso es $\mathcal{O}([\log_2 n]^3)$

Vale decir, **EsPotencia** en el peor caso es de orden polinomial en el tamaño de la entrada.

- Se puede llegar a la misma conclusión si consideramos todas las operaciones realizadas por **EsPotencia**.

El siguiente algoritmo aleatorizado determina si un número entero $n \geq 2$ es primo.

```
TestPrimalidad(n, k)
    if n = 2 then return PRIMO
    else if n es par then return COMPUESTO
    else if EsPotencia(n) then return COMPUESTO
    else
        sea a_1, ..., a_k una secuencia de números elegidos de
                manera uniforme e independiente desde {1, ..., n − 1}
        for i := 1 to k do
            if MCD(a_i, n) > 1 then return COMPUESTO
            else b_i := EXP(a_i, (n−1)/2, n)
        neg := 0
        for i := 1 to k do
            if b_i ≡ −1 mod n then neg := neg + 1
            else if b_i ≢ 1 mod n then return COMPUESTO
        if neg = 0 then return COMPUESTO
        else return PRIMO
```

El algoritmo recibe como entrada un valor entero $k \geq 1$ que es usado para controlar la probabilidad de error.

TestPrimalidad se puede equivocar de dos formas:

1. Suponga que $n \geq 3$ es primo. En este caso **TestPrimalidad** da una respuesta incorrecta si $b_i \equiv_n 1 \ \forall i \in \{1, \ldots, k\}$.

   Dado que $|S_n^+| = |S_n^-| = \frac{n-1}{2}$

   - La probabilidad de que para un número a elegido con distribución uniforme desde $\{1, \ldots, n-1\}$ se tenga que $a^{\frac{n-1}{2}} \equiv_n 1$ es $\frac{1}{2}$.

   Por lo tanto, la probabilidad de que **TestPrimalidad** diga COMPUESTO para $n \geq 3$ primo es $\left(\frac{1}{2}\right)^k$.

2. Suponga que n es compuesto, n es impar y n NO es de la forma m^l con $l \geq 2$.

   a. Si n es par o m es de la forma m^l con $l \geq 2$, entonces **TestPrimalidad** da la respuesta correcta COMPUESTO.

Tenemos entonces que $n = n_1 \cdot n_2$ con $n_1 \geq 3$, $n_2 \geq 3$ y $\gcd(n_1, n_2) = 1$.

Además, debe existir $a \in \{1, \dots, n-1\}$ tal que $\gcd(a, n) = 1$ y $a^{\frac{n-1}{2}} \equiv_n -1$

   b. Si esto NO es cierto **TestPrimalidad** retorna COMPUESTO, dado que si **TestPrimalidad** logra llegar a la última instrucción **if** entonces neg necesariamente es igual a 0.

Concluimos que $|S_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$.

   c. Por la caracterización que dimos de S_n para n compuesto.

Vamos a utilizar este resultado para acotar la probabilidad de error:

$$\Pr\left(\left(\bigwedge_{i=1}^{k} \gcd(a_i, n) = 1 \wedge (b_i \equiv_n 1 \vee b_i \equiv_n -1)\right) \wedge \left(\bigvee_{j=1}^{k} b_j \equiv_n -1\right)\right)$$

Tenemos que:

$$\Pr\left(\left(\bigwedge_{i=1}^{k} \gcd(a_i, n) = 1 \wedge (b_i \equiv_n 1 \vee b_i \equiv_n -1)\right) \wedge \left(\bigvee_{j=1}^{k} b_j \equiv_n -1\right)\right)$$
$$\leq \Pr\left(\left(\bigwedge_{i=1}^{k} \gcd(a_i, n) = 1 \wedge (b_i \equiv_n 1 \vee b_i \equiv_n -1)\right)\right)$$

Por lo tanto, sólo necesitamos una cota superior para la última expresión.

Tenemos que:

$$\Pr\left(\left(\bigwedge_{i=1}^{k} \gcd(a_i, n) = 1 \wedge (b_i \equiv_n 1 \vee b_i \equiv_n -1)\right)\right)$$
$$= \prod_{i=1}^{k} \Pr\big(\gcd(a_i, n) = 1 \wedge (b_i \equiv_n 1 \vee b_i \equiv_n -1)\big)$$

$$= \prod_{i=1}^{k} \Pr\big((b_i \equiv_n 1 \lor b_i \equiv_n -1) \,\big|\, \gcd(a_i, n) = 1\big) \cdot \Pr(\gcd(a_i, n) = 1)$$
$$\leq \prod_{i=1}^{k} \Pr\big((b_i \equiv_n 1 \lor b_i \equiv_n -1) \,\big|\, \gcd(a_i, n) = 1\big)$$
$$= \prod_{i=1}^{k} \Pr(a_i \in S_n \mid a_i \in \mathbb{Z}_n^*) \leq \prod_{i=1}^{k} \frac{1}{2} = \left(\frac{1}{2}\right)^k$$

Concluimos que la probabilidad de que el test diga PRIMO para el valor compuesto n está acotada por $\left(\frac{1}{2}\right)^k$.

En ambos casos (si n es primo o compuesto) la probabilidad de error del algoritmo está acotada por $\left(\frac{1}{2}\right)^k$.

¡Si $k = 100$, está probabilidad está acotada por $\left(\frac{1}{2}\right)^{100} \approx 7.9 \times 10^{-31}$!

Caso promedio de un algoritmo:

Para definir el caso promedio, para cada $n \in \mathbb{N}$ usamos una **variable aleatoria** X_n tal que para cada $w \in \Sigma^n$ se tiene que:

$$X_n(w) = \text{tiempo}_{\mathcal{A}}(w)$$

Para las entradas de largo n, el número de pasos $\mathcal{A}$ en el caso promedio es el **valor esperado** de la variable X_n:

$$E(X_n) = \sum_{w \in \Sigma^n} X_n(w) \cdot \Pr_n(w)$$

Definición: Complejidad en el caso promedio

Decimos que $\mathcal{A}$ en el caso promedio es $\mathcal{O}\big(f(n)\big)$ si $E(X_n) \in \mathcal{O}\big(f(n)\big)$

Notación

Las definiciones de peor caso y caso promedio pueden ser modificadas para considerar las notaciones Θ y Ω

3. Simplemente reemplazando $\mathcal{O}\big(f(n)\big)$ por $\Theta\big(f(n)\big)$ u $\Omega\big(f(n)\big)$, respectivamente.

Por ejemplo, decimos que $\mathcal{A}$ en el caso promedio es $\Theta\big(f(n)\big)$ si $E(X_n) \in \Theta\big(f(n)\big)$.

Quicksort es un algoritmo de ordenación muy utilizado en la práctica.

La función clave para la definición. de Quicksort $\Longrightarrow$

```
Partición(L, m, n)
    pivote := L[m]
    i := m
    for j := m + 1 to n do
        if L[j] ≤ pivote then
            i := i + 1
            intercambiar L[i] con L[j]
    intercambiar L[m] con L[i]
    return i
```

La definición de Quicksort $\Longrightarrow$

```
Quicksort(L, m, n)
    if m < n then
        ℓ := Partición(L, m, n)
        Quicksort(L, m, ℓ − 1)
        Quicksort(L, ℓ + 1, n)
```

Nótese que en la definición de **Partición** la lista L es pasado por referencia.

Vamos a contar el **número de comparaciones** al medir la complejidad de Quicksort.

Si **Partición** siempre divide a una lista en dos listas del mismo tamaño, entonces la complejidad de **Quicksort** estaría dada por la siguiente ecuación de recurrencia:

$$T(n)\begin{cases} 1 & n = 0 \\ 2 \cdot T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c \cdot n & n \geq 1 \end{cases}$$

Esto está en la clase 24, pero no en la 25.

Dado que $c \cdot n \in \Theta\big(n^{\log_2 2}\big)$, concluimos usando el Teorema Maestro que:

$$T(n) \in \Theta(n \cdot \log_2(n))$$

Vamos a demostrar que **Quicksort** en el caso promedio es $\Theta(n \cdot \log_2 n)$ suponiendo una distribución uniforme en las entradas.

Vamos a considerar listas **sin elementos repetidos**.

- Queda como ejercicio el pensar en cómo obtener la misma complejidad para listas con elementos repetidos.

La complejidad de **Quicksort** en el caso promedio está dad por $E(X_n)$.

¿Por qué nos restringimos a las listas con elementos sacados desde el conjunto $\{1, \dots, n\}$?

- ¿En qué sentido estamos considerando todas las listas posibles con n elementos (sin repeticiones)?

- ¿Cómo refleja esto el hecho de que estamos considerando las entradas de un cierto largo?

Sea $L \in \mathcal{E}_n$.

Para cada $i, j \in \{1, \dots, n\}$ con $i < j$ defina la siguiente **variable aleatoria**:

$\boldsymbol{Y_{i,j}(L)}$: número de veces que i es comparado con j en la llamada **Quicksort**$(L, 1, n)$.

Entonces tenemos lo siguiente:

$$X_n(L) = \sum_{i=1}^{n} \sum_{j=1}^{n} Y_{i,j}(L)$$

Dado que el valor esperado de una variable aleatoria es una función lineal, concluimos que:

$$E(X_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} E\left(Y_{i,j}\right)$$

Para calcular $E(X_n)$ basta entonces calcular $E\left(Y_{i,j}\right)$ para cada $i. j \in \{1, \dots, n\}$ con $i \leq j$.

Por la definición de **Partición** no es posible comparar un elemento consigo mismo, por lo que $Y_{i,i}(L) = 0$ para cada $i \in \{1, \dots, n\}$ y $L \in \mathcal{E}_n$.

- Tenemos entonces que $E\left(Y_{i,i}\right) = 0$

Consideremos ahora el caso $i = 1$ y $j = n$.

Los elementos 1 y n sólo pueden ser comparados en la llamada **Partición**$(L, 1, n)$.

- Estos elementos son comparados si $L[1] = 1$ o $L[1] = n$.
- Se realiza a lo más una comparación entre ellos.

Tenemos entonces que $Y_{1,n}$ es igual a 0 o 1.

Además, $\Pr(Y_{1,n} = 1)$ es igual a la probabilidad de que $L[1] = 1$ o $L[1] = n$ dado que L es escogido al azar y con distribución uniforme desde el conjunto $\mathcal{E}_n$.

Tenemos entonces que:

$$\Pr(Y_{1,n} = 1) = \frac{1 \cdot (n-1)!}{n!} = \frac{2}{n}$$

Nótese que esta probabilidad puede cambiar si consideramos otra distribución de probabilidades sobre las listas en $\mathcal{E}_n$.
Concluimos que:

$$E(Y_{1,n}) = 0 \cdot \Pr(Y_{1,n} = 0) + 1 \cdot \Pr(Y_{1,n} = 1) = \frac{2}{n}$$

Consideramos ahora el caso general $1 \leq i < j \leq n$.

Mientras las llamadas a **Partición** escojan un valor para *pivote* tal que *pivote* $< i$ o *pivote* $> j$, los elementos i, j no son comparados y están en una parte de la lista que va a ser ordenada en **Quicksort**.

- i y j pueden ser comparados en las siguientes llamadas a **Partición**.

Si **Partición** escoge un valor para *pivote* tal que $i <$ *pivote* $< j$, entonces i no es comparado con j en la ejecución completa de **Quicksort**.

La única forma en que **Quicksort** puede comparar i con j es que el primer elemento que escoja **Partición** desde el conjunto $\{i, i+1, \ldots j\}$ sea i o j.

- Se realiza a lo más una comparación entre i y j en la ejecución completa de **Quicksort**.

Tenemos entonces que $Y_{i,j}$ es igual a 0 o 1, y además que:

$$\Pr(Y_{i,j} = 1) = \frac{2}{j - i + 1}$$

Concluimos que:

$$E(Y_{i,j}) = 0 \cdot \Pr(Y_{i,j} = 0) + 1 \cdot \Pr(Y_{i,j} + 1) = \frac{2}{j - i + 1}$$

Concluimos que:

$$\begin{aligned}
E(X_N) &= \sum_{i=1}^{n}\sum_{j=1}^{n} E\left(Y_{i,j}\right) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E\left(Y_{i,j}\right) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1} \frac{2}{k} \\
&= \sum_{k=2}^{n-i+1} (n+1-k)\cdot\frac{2}{k} \\
&= 2\cdot(n+1)\cdot\left(\sum_{k=2}^{n}\frac{1}{k}\right) - 2\cdot(n-1) \\
&= 2\cdot(n+1)\cdot\left(\sum_{k=1}^{n}\frac{1}{k}\right) - 4\cdot n
\end{aligned}$$

Para terminar el cálculo de $E(X_n)$ tenemos que acotar la sumatoria armónica $\sum_{k=1}^{n}\frac{1}{k}$.

Tenemos que:

$$\sum_{k=2}^{n}\frac{1}{k} \leq \int_{1}^{n}\frac{1}{x}\,dx \leq \sum_{k=1}^{n}\frac{1}{k}$$

Dado que $\int_{1}^{n}\frac{1}{x}\,dx = \ln(n) - \ln(1) = \ln(n)$, concluimos que:

$$\ln(x) \leq \sum_{k=1}^{n}\frac{1}{k} \leq \ln(n) = 1$$

Por lo tanto:

$$2\cdot(n+1)\cdot\ln(n) - 4\cdot n \leq E(X_n) \leq 2\cdot(n+1)\cdot(\ln(n)+1) - 4\cdot n$$

De lo cual concluimos que $E(X_n) \in \Theta(n\cdot\log_2 n)$

- Vale decir, **Quicksort** en el caso promedio es $\Theta(n\cdot\log_2 n)$.