**1. ¿Qué enteros positivos menores que 30 son primos relativos con 30?**

Los factores primos de 30 son 2, 3 y 5.

Los primos relativos con 30 serán: 1,7,11,13,17,19,23,29

**2. Evaluar las siguientes expresiones:**

(a) −17 mod 2 = 1

(b) 144 mod 7 = 4

(c) −101 mod 13 = 3

(d) 199 mod 19 = 9

**3. Probar que si a, b, c, y m son enteros tales que m ≥ 2, c > 0, y a ≡ b (mod m), entonces ac ≡ bc (mod mc).**

a-b es múltiplo de m (m|a-b)

ac-bc es múltiplo de mc (mc|ac-bc) -> (mc | ac-bc) -> (m|a-b)

**4. Descifrar los siguientes mensajes que fueron encriptados usando la cifra de César:**

(a) HFKH R TXH KDL = ECHE O QUE HAI

(b) RQGH HVWD D WHFOD DQB = ONDE ESTA A TECLA ANY

(c) HVWDU QD KRUWD H QRQ YHU DV EHUCDV = ESTAR NA HORTA E NON VER AS BERZAS

Alfabeto inglés, -3 mod 23

**5. Convertir los siguientes enteros escritos en expansión decimal a expansión binaria:**

(a) 321 = 101000001

(b) 1023 = 11111111111

(c) 100632 = 11000100100011000

**6. Convertir los siguientes enteros escritos en expansión binaria a expansión decimal:**

(a) (1 1111)2 = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 = 31

(b) (10 0000 0001)2 = 2^0 + 2^9 = 513

(c) (1 0101 0101)2 = 2^0 + 2^2 + 2^4 + 2^6 + 2^8 = 3413

(d) (110 1001 0001 0000)2 = 2^4 + 2^8 + 2^11 + 2^13 + 2^14 = 26896

**7. Convertir los siguientes enteros escritos en expansión hexadecimal a expansión binaria:**

(a) (80E)16 = 1000 0000 1110

(b) (135AB)16 = 0001 0011 0101 1010 1011

(c) (ABBA)16 = 1010 1011 1011 1010

(d) (DEF ACED)16 1101 1110 1111 1010 1100 1110 1101

8.

(a) Convertir (807)9 escrito en expansión en base 9 en expansión en base 7.

8079 = 7\*9^0 + 0 + 8\*9^2 = 65510

655 en base 7 = 1624

(b) Convertir (12034)5 escrito en expansión en base 5 en expansión en base 4.

120345 = 4\*5^0 + 3\*5^1 + 0 + 2\*5^3 + 1\*5^4 = 89210

892 en base 4 = 31330

(c) Convertir (5607A)11 escrito en expansión en base 11 en expansión en base 6.

5607A11 = 10\*11^0 + 7\*11^1 + 6\*11^3 + 5\*11^4 = 8137810

81278 en base 6 = 14241426

(d) Convertir (4B07D)14 escrito en expansión en base 14 en expansión en base 8.

4B07D14 = 13\*14^0 + 7\*14^1 + 11\*14^3 + 4\*14^4 = 18395910

183959 en base 8 = 5472278

**9. Probar que 937 es un inverso de 13 módulo 2436.**

Se dice que a es inverso de b (mód m) si a \* b ≡ 1 mód m, es decir, m|(a\*b - 1)

937 \* 13 = 12181

12181 - 1 = 12810

2436 | 12810.

Entonces, 937 es inverso de 13 módulo 2346

**10. Encontrar un inverso de a módulo m para cada uno de los siguientes pares de enteros primos relativos:**

(a) a = 2, m = 17.

2 mód 17 es inverso de **9**.

Mediante algoritmo de Euclides:

Se hace división,

1 = 17 -8\*2

1 = 17 -8\*2 mód 17

1 -8\*2 mód 17

Entonces, inverso multiplicativo es -8 = 9.

9 \* 2 mód 17 = 18 mód 17 = 1 mód 17

(b) a = 34, m = 89.

34 mód 89 es inverso de **55**.

55 \* 34 mód 89 = 1 mód 21

(c) a = 144, m = 233.

144 mód 233 es inverso de **89**.

(d) a = 200, m = 1001.

**11. Resolver la congruencia 2x ≡ 7 (mod 17) usando el inverso de 2 módulo 17 encontrado en la parte (a) del ejercicio anterior.**

Si 2\*9 es 1 mód 17, entonces 7 mód 17 será 2\*9+6 = 2\*12

x=12

**12. Usar el teorema chino de los restos para encontrar todas las soluciones del sistema de congruencias x ≡ 2 (mod 3) x ≡ 1 (mod 4) x ≡ 3 (mod 5)**

x ≡ 2 (mod 3)

x ≡ 1 (mod 4)

x ≡ 3 (mod 5)

3,4,5 son primos entre sí. 3\*4\*5 = 60

x =

x = 2 \* 20 \* 2 + 1 \* 15 \* 3 + 3 \* 12 \* 3 = 233 mód 60 = **53 mód 60**

**13. Probar que si n es un número entero entonces n^2 ≡ 0 o 1 (mod 4).**

Si el número n es un número par, n/2 será entero. Entonces:

n = (n/2) \* 2, donde n/2 es entero

Si (n/2) es entero, (n/2)^2 también lo será.

n^2 = (n^2/4) \* 4, donde n/4 es entero

Entonces, n^2 es múltiplo de 4, y n^2 mód 4 ≡ 0

Si el número n es impar, sabemos que n^2 será un número también impar. Entonces, n^2 + 1 será un número par.

Debido a lo demostrado previamente, (n^2+1) mód 4≡ 0 mód 4.

Entonces, (n^2) mód 4 ≡ 1 mód 4.

Si n es entero, **n^2 ≡ 0 o 1 (mod 4),** sea n par o impar.

**14. Probar que si n es un número entero positivo impar, entonces n^2 ≡ 1 (mod 8).**

Un número entero positivo impar es de la forma n=(2k+1), donde k es cualquier número entero.

Entonces, n^2 = (2k+1)^2 = 4k^2 + 4k +1 = 4(k^2 + k) +1

Tanto para un k par como un k impar, se cumple que k^2 + k es par.

Si k^2+k es par, 4(k^2+k) es múltiplo de 8, y por lo tanto, 4(k^2+k)≡ 0 mód 8

Entonces, n^2 = 4(k^2+k) + 1≡ 1 mód 8

**15. Encontrar contraejemplos a cada una de estos enunciados sobre congruencias.**

(a) Si ac ≡ bc (mod m), donde a, b, c y m son números enteros, con m ≥ 2, entonces a ≡ b (mod m).

Para que se cumpla este enunciado, (ac-bc) | (a-b)

Esto no se cumple necesariamente si **c=0**.

(b) a ≡ b (mod m) y c ≡ d (mod m), donde a, b, c, d y m son números enteros con c y d positivos y m ≥ 2, entonces a^c ≡ b^d (mod m).

No se cumple si uno de los exponentes, c o d, es 0.

1. Describir un algoritmo que tenga como entrada una lista de n números enteros distintos y por

salida la posición del último número entero par en la lista o devuelva NO si no hay números

enteros pares en la lista.

L = [1,2,3,4,5]

L2 = [1,3,5,7]

def ultimo\_par(lista):

for numero in lista[::-1]:

if is\_even(numero):

return numero

return 'NO'

ultimo\_par(L)

ultimo\_par(L2)

2. Describir un algoritmo que tenga como entrada una lista de n enteros y por salida la cantidad

de enteros negativos de la lista.

L = [-2, -1, -0.5, 0, 0.5, 1, 2]

def enteros\_negativos(lista):

total = 0

for numero in lista:

if numero.is\_integer() and numero<0:

total += 1

return total

enteros\_negativos(L)

3. Describir un algoritmo que cuente el número de bits 1 (uno) en una cadena de bits examinando

cada bit de la cadena determinando si es o no un bit 1 (uno).

C = 11010101101

def contar\_bits(cadena):

bits = 0

cadena = str(cadena)

for numero in cadena:

if numero == '1':

bits += 1

return bits

contar\_bits(C)

︡a055ef30-0f13-4c24-9efc-75170610151d︡{"stdout":"7\n"}︡{"done":true}

︠8d4cd4c4-005b-4673-b7ac-434470de7576︠

"""

4. Dar una estimación de big-O del número de comparaciones realizadas por el algoritmo que

determina el número de unos de una cadena de bits examinando cada bit de la cadena si es un

uno o no (véase ejercicio anterior).

El número de operaciones realizadas es idéntico al número de dígitos en la cadena. Será Big(O) de f(x), o O(f(x)).

5. Encontrar el menor entero n tal que f(x) sea O(x^n) para cada una de las siguientes funciones:

a) f(x) = 2x^2 + x^3 \* log(x)

2x^2 es O(x^3 \* log(x))

(x^3 \* log(x)) es O(x^4)

Entonces, n=4

b) f(x) = 3x^5 + (log x)^4

(log x^4) es O(3x^5)

(3x^5)es O(x^5)

Entonces, n=5

c)f(x) = (x^4 + x^2 + 1) / (x^4 + 1)

(x^4 + x^2 + 1) es O(x^4)

(x^4 + 1) es O(x^4)

(x^4 / x^4) es O(x^1)

n=1

d) f(x) = (x^3+ 5logx)/(x^4 + 1)

(x^3 + 5logx) es O(x^3)

(x^4 + 1) es O(x^4)

n=0 (?)

6.

a) f(x) es O(x^2)

b) f(x) es θ(x^2)

c) f(x) es O(x^2)

d) f(x) es Ω(x^2)

e) f(x) es Ω(x^2)

f) f(x) es θ(x^2)

7. Sea k un entero positivo. Probar que 1^k + 2^k + · · · + n^k es O(n^k+1)

1^k + 2^k + · · · + n^k es O(n^k), que es el máximo de todos los elementos del polinomio.

Para (k>0), n^k siempre es O(n^(k+1)), pues es un polinomio de menor grado (k < k+1).

8.

Sería más rápida la búsqeuda lineal, que hace 4 operaciones, mientras que la

binaria debe hacer log2(32) = 5

9. Dado un número real x y un entero positivo k, determinar el número de multiplicaciones usadas para encontrar x^2^k

comenzando con x y elevando al cuadrado sucesivamente (encontrando x2, x4, y así progresivamente). ¿Es este método más eficiente para calcular x^2^k que

multiplicar x por sí mismo el número apropiado de veces?

El número de operaciones usadas será 'k'. Sí, es más eficiente que multiplicar x por si mismo 2^k-1 veces, ya que el segundo ejemplo posee complejidad

exponencial en lugar de polnomica.

10. ¿Cuánto tiempo tarda un algoritmo en resolver un problema de tamaño n si el algoritmo usa2 n^2 + 2^n operaciones con bits y cada una requiere 10^−9segundos, con los siguientes valores de n?

a) 1224 operaciones, 1,224\*10^-6 segundos

b) 1049376 operaciones, 1,05\*10^-3 segundos

c) 1.126\*10^15 operaciones, 1.126\*10^6 segundos

d) 1.268\*10^30 operaciones, 1.268\*10^21 segundos

11. (a) Describir un algoritmo para localizar la última vez que aparece el número más grande en

una lista de enteros.

(b) Estimar el número de comparaciones realizadas.

"""

L = [2,3,20,5,6,20,8]

def localizar\_mayor\_entero(lista):

mayor\_numero = lista[0]

for i in range(len(lista)):

if lista[i] >= mayor\_numero:

mayor\_numero = lista[i]

posicion = i

return(posicion)

localizar\_mayor\_entero(L)

El numero de comparaciones realizadas es igual al número de elementos en la lista (en este ejemplo, 7)

12. (a) Describir un algoritmo para encontrar el mayor elemento y el segundo mayor elemento

en una lista de enteros.

(b) Estimar el número de comparaciones realizadas

L = [9,4,2,6,9,0,-5,1]

def dos\_mayores\_enteros(lista):

mayor\_numero = lista[0]

segundo\_mayor = -infinity

for numero in lista:

if numero > mayor\_numero:

segundo\_mayor = mayor\_numero

mayor\_numero = numero

elif numero > segundo\_mayor:

segundo\_mayor = numero

return mayor\_numero,segundo\_mayor

dos\_mayores\_enteros(L)

El número de comparaciones es aproximadamente igual al número de elementos de la lista.

"""

13. Describir la complejidad en tiempo del algoritmo para encontrar el máximo de una sucesión

finita de enteros.

Complejidad lineal.

14. Describir la complejidad en tiempo del algoritmo de búsqueda lineal.

Complejidad lineal.

15. Describir la complejidad en tiempo del algoritmo de búsqueda binaria.

Complejidad logarítmica.