**Tema 1: Algoritmos**

**Definición**

* Un algoritmo es un proceso diseñado para llevar a cabo una tarea o resolver un problema.
* Por ejemplo: encontrar el máximo de una sucesión de números, ordenar nombres en orden alfabético o encriptar un mensaje.
* Es una sucesión finita de instrucciones precisas y organizadas.

**Propiedades** (de un algoritmo útil)

* Existe una entrada y salida de datos. (input/output)

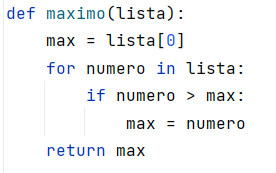
Para cada conjunto de valores de entrada, el algoritmo produce valores de salida que son la solución del problema.

* Debe estar definido con precisión.
* Debe ser correcto.
* Debe ser de duración finita para cualquier entrada.
* Debe ser efectivo: realizar cada paso del algoritmo con exactitud en un intervalo finito de tiempo.
* Debe poseer generalidad: ser aplicable a todos los problemas y no sólo a un conjunto particular de entrada
* Debe ser eficiente: ejecutarse en tiempo polinómico, es decir, el tiempo de duración es menor o igual que un valor calculado a partir de ciertas variables de entrada.

**Ejemplos**

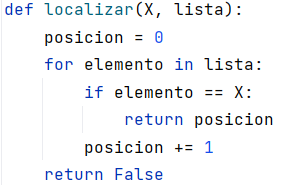
* **Encontrar máximo:** Permite encontrar el mayor número de una lista.

1. Tomamos el primer número de la lista y lo fijamos como máximo
2. Iteramos por cada número comprobando si es mayor que el máximo. De serlo, se establece como el máximo.
3. Finaliza cuando se han comprobado todos los números introducidos.

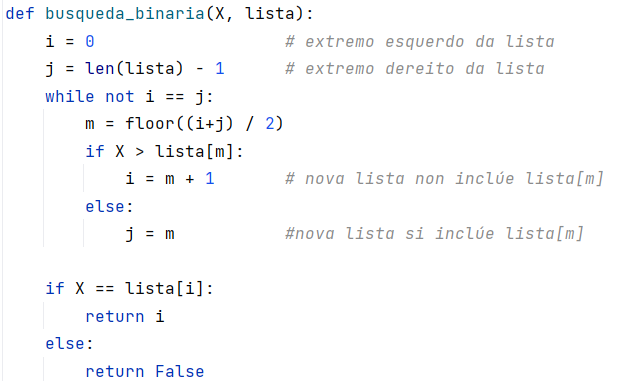
****

* **Algoritmos de búsqueda:** Intentan localizar un elemento X en una lista de elementos distintos, o determinar que no está en la lista. Devuelve True o False.

1. Compara cada elemento de la lista con el dado
2. Si son iguales, devuelve la localización del elemento. Si no, continúa.
3. Si se llega al fin sin encontrar coincidencias, devuelve False

.

* **Búsqueda binaria:** Permite ordenar una lista (palabras o números). Se procede comparando el elemento que se quiere localizar con el central en la lista, para dividirla en dos sublistas.
  + Para buscar x en una lista [a0, a1, a2… an], comparamos x con am, donde m = ⌊(n/2)⌋
    - **NOTA:** ⌊x⌋ es una operación ‘floor’ (truncamiento), ⌈x⌉ es ‘ceiling’ (menor entero tal que z ≥ x)
  + Si x > am, la búsqueda se restringe a la 2ª mitad. Si x≤ am, se restringe a la 1ª mitad, donde se incluye am.



**Algoritmos de ordenación**

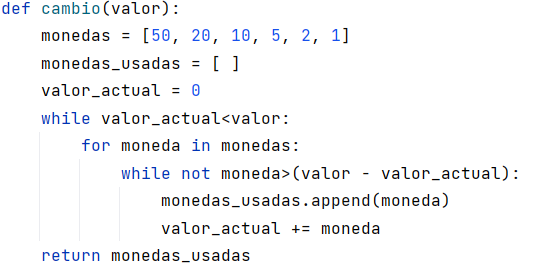
* Bubble sort: Complejidad cuadrática
* Selection sort: C. cuadrática
* Insertion sort: C. cuadrática
* Merge sort: C. cuasi-lineal
* Quick sort: C. cuadrática

**Tipos de algoritmos por método**

* De fuerza bruta (técnicas de enumeración, búsqueda exhaustiva)
* ‘Divide y vencerás’ (búsqueda binaria)
* ‘Transforma y vencerás’ (reformulación del problema)
* Algoritmos voraces: Consisten en encontrar la solución local que parece óptima en paso para crear una solución global, que luego comprueban si es adecuada.
  + Se suelen utilizar en problemas de optimización.

**Ejemplos de algoritmos voraces**

**Ejemplo 1:** Algoritmo que da cambio con monedas de 1,2,5,10,20,50 céntimos, proporcionando el menor número de monedas posible.



**Ejemplo 2:** Algoritmo que programa el numero máximo de conferencias que es posible incluír en una franja horaria sin que se solapen.

* La solución óptima es tomar las conferencias que terminen antes.

**Problemas indecidibles**

* Aquellos que no pueden ser resueltos por algoritmos.
* **Ejemplo**: el problema de la parada
  + No es posible diseñar un programa que analiza si otro programa se detiene o cuando lo hará, ya que al analizar este programa consigo mismo se produciría una paradoja.

**Notación Big-O**

* Sean f y g funciones, se dice que **f(x) es O(g(x))** si existen dos constantes (‘testigos) c y k tales que:

|f(x)| ≤ c\*|g(x)| en x > k

* Es decir, f(x) no crece más rápido que g(x). Se lee ‘f es Big-O de g’.
* **Ejemplos**:
  + f(x) es O(xn)
  + n! es O(nn)
  + loga(n) es O(logb(n)), siendo ‘a’ y ‘b’ cualquier base.
  + log2(n) es O(n)
  + log(n!) es O(n \* log(n))

**Teoremas de Big-O**

* Si f1(x) es O(g(x)) y f2(x) es O(g2(x)), f1 y f2 son Big-O de lamáxima entre g1 y g2.
  + **Corolario:** Si f1 y f2 son O(g(x)), f1 + f2 (x) también.
* Si f1(x) es O(g(x)) y f2(x) es O(g2(x)), f1\*f2(x) es O(g1(x)\*g2(x))
* Si f(x) es O(g(x) y g(x) es O(h(x)), f(x) es O(h(x))
* Si f(x) es O(g(x)), a\*f(x) es O(g(x))

**Big-Ω**

* Sean f y g funciones, se dice que **f(x) es Ω(g(x))** si existen dos constantes (testigos) c y k tales que:

|f(x)| ≥ c\*|g(x)| en x > k

**Big-Ө**

* Sean f y g funciones, se dice que **f(x) es Ө(g(x))** si f(x) es **O(g(x))** **y** **Ω(g(x))**

**Complejidad computacional en tiempo**

* Un análisis del tiempo requerido para resolver un problema está relacionado con la complejidad en tiempo del algoritmo.
* Se expresa en términos de operaciones usadas por el algoritmo, analizando el peor caso.
  + También existe la complejidad en el caso promedio o en el mejor caso.
* **Ejemplo:** Busqueda lineal es de c. lineal, busqueda binaria es de c. logarítmica.
* **Problema resoluble:** Resoluble por un algoritmo
* **Problema tratable:** Resoluble por un algoritmo con complejidad polinómica en el peor caso
* **Problema intratable:** Resoluble por un algoritmo pero no tratable

**Clases de algoritmos**

* **Clase NP:** Algoritmo para el cual es posible comprobar una solución en tiempo polinómico. Puede ser, además, clase P o no.
  + **Ejemplos:** Si un grafo tiene un ciclo hamiltoniano[[1]](#footnote-0), Torres de Hanoi, factorización en enteros.
* **Clase P:** Algoritmo que se ejecuta en tiempo polinómico.
* **Clase NP completa:** Algoritmo NP para el cual no se conoce la forma de hallar una solución en tiempo polinómico (Pertenece a NP, pero no a P).
  + De poder resolver uno, sería posible resolverlos todos.
  + **Ejemplos:** ciclo hamiltoniano, problema del mercader viajante, candy crush

**Tema 2: Aritmética entera**

**División de enteros**

* Dados dos **enteros** a,b , se dice que a divide a b (a|b) si existe un entero c tal que c\*a=b.
* En este caso, a es factor de b y b es múltiplo de a.
* **Teoremas:**
  + Si a|b y a|c => a|(b+c)
  + Si a|b => a|(b\*c) para todo entero c.
  + Si a|b y b|c => a|c
    - Corolario: si a|b y a|c, a|(n\*b + m\*c) para todo m,n enteros.

**Algoritmo de la división**

* Sea a entero, d>0. Existen enteros q,r únicos con 0≤r<d tal que:

a = q\*d + r

* **Nota:** De ser el dividendo o divisor negativos, |q\*d| tendrá que ser mayor que |a|, debido a que el resto siempre es positivo.

**Números primos**

* Sea p>1 entero, p es un número primo si sus únicos divisores son p y 1.
  + 1 no es primo por ser la única unidad de los enteros positivos.
  + **Unidad:** Número que puede ser multipliicado por su ‘inverso’ para dar como resultado 1.
* Un número entero positivo no primo es compuesto.
* **Teorema de Euclídes:** Existen infinitos números primos
* **Primos de Mersenne:** aquellos de la forma 2^p.1 con p primo (no todos son primos)
  + El mayor número primo conocido es el M51, 2^825899333 - 1

**Teorema fundamental de la aritmética** (Euclídes)

* Todo entero positivo mayor que 1 se puede factorizar de manera única como un primo o producto de primos, donde los factores primos están escritos en orden creciente.
* **Teorema:** Si n es un número compuesto, los divisores primos de n son menores o iguales que su raíz.

**Criba de Eratóstenes**

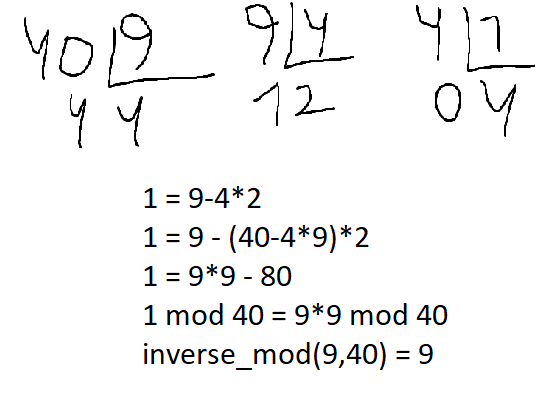
* Consiste en escribir todos los números de un conjunto, marcar los primos y tachar sus múltiplos para encontrar todos los primos.

**MCM y MCD**

* Sean a y b enteros, al menos uno de ellos distinto de 0, el mayor entero D tal que d|a y d|b, es el máximo común divisor. gcd(a,b)
  + Se encuentra mediante **algoritmo de Euclídes:**
    - Lema: Sean a,b, a>b y r=a%b
    - gcd(a,b) = gcd(b, r).
    - Demostración: Sea d un divisor de a y de b, también lo es de r, ya que a-qb = r.
* Sean a y b enteros, al menos uno de ellos distinto de 0, el menor entero m tal que a|m y b|m es el mínimo común múltiplo lcm(a,b)
  + Se encuentra multiplicando a\*b y dividiéndolo entre su gdc..
* gcd(a,b) \* lcm(a,b) = producto de las descomposiciones en primos de a y b

**Algoritmo de Euclides para calcular inverso modular**

* Nota: existe inverso modular <-> gcd(a,m) = 1
* Ejemplo: inverso modular de 9 en 40



**Teorema de Bézout:**

* Sean a y b enteros, existen enteros s,t tales que gcd(a,b)=s\*a+t\*b
* Para encontrarlos, se realiza la división de 252 y 198 mediante el algoritmo de Euclídes
  + Ejemplo: gcd(252,198) = 18
  + 18 = 54 - 36 = 54 - (198 - 3\*54) = 4\*54 - 198 = 4 \* (252 - 198) - 198 =

4\*252 - 5 \* 198 - > s=4, t=-5

**Primos entre sí:**

* Sean a y b enteros, se dice que a,b son primos entre sí si gcd(a,b) = 1
* Sean a1,a2,...an son primos entre sí si gcd(ai, aj) = 0 para todo i y j
* **Lema:** Si a,b,c son enteros tal que gcd(a,b) = 1 y a|bc -> a|c
* **Lema:** Si p es primo y p|(a1\*…\*an), p|ai para algún i

**Tema 3: Aritmética modular**

**División de enteros**

* m entero y positivo, sean a,b enteros, de dice que a es congruente con b módulo m **a ≡ b (mód m)** si a-b es múltiplo de m
  + Ejemplo: 36 ≡ 12 (mód 12)
* En un cuerpo de módulo 2:
  + 2k+1 ≡ 1 (mód 2), 2k ≡ 0 (mód 2)
* **Teorema:** Sean a,b enteros, m entero positivo a≡b mód m <-> a mód m = b mód m
  + a mód m = resto de a/m
  + Demostración:
    - a mód m = b mód m
    - a - qm = b - q’m
    - a - b = (q-q’) m -> a-b es múltiplo de m.
  + Otra demostración:
    - a - b = k\*m, k entero
    - a - b = (qm+r) - (q’m+r’) = (q-q’)m `(r-r’)
* **Teorema:** Sean a,b,c,d enteros, m entero positivo a≡b mód m y c≡d mód m;
  + a+c≡b+d mód m
  + a\*c≡b\*d mód m
  + Ejemplo: a=13, b=25, c=5, d=17, m=12
    - a mod m = b mod m = 1
    - c mod m = d mod m = 5
    - a+c mod m = b+d mod m = 6 (= [a mod m + c mod m] mod m)
    - a\*c mod m = b\*d mod m = 5yc (= [a mod m \* c mod m] mod m)

**Aritmética modular**

* a +m b = (a + b) mod m
* a \*m b = (a \* b) mod m
* Z /mZ : Conjunto de enteros con módulo entero
  + En un conjunto de enteros con un módulo determinado, los primos relativos de m son las unidades.
    - Si m es primo, todos los números del conjunto son unidades
  + En el caso de mod 12, 1,5,7,11 son unidades y al multiplicarse por si mismas dan **1** mod 12 (tienen inverso multiplicativo, 1/a = a)
  + Los que **no** son primos relativos con m son “malos”
* Nº de unidades para un módulo m según Euler:
  + ф (p^k) = p^(k-1) (p-1) para un primo con exponente
  + Si no es primo, m se descompone en primos p1, p2 …ps (con el correspondiente exponente en cada primo)
    - ф(m) = ф(p1) \* ф(p2)...
    - Ejemplo: m = 360
      * 360 = 2^3 \* 3^2 \* 5^1
      * ф(360) = ф(2^3) \* ф(3^2) \* ф(5^1)
      * 𝝫(360) = (4\*1) \* (3\*2) \* (1\*4) = 96

**Congruencias lineales**

* Considerando la ecuación ax≡b (mód m).
  + Siendo d = gcd(a,m)
  + Si d|b, la ecuación tiene **d** soluciones en [0, m). Si no, no tiene solución.
  + Para resolverla, se dividen ambos lados por d.
  + Luego, se puede realizar un cambio de variable para convertir b en 1.
  + Finalmente, se encuentra el inverso modular de a en m. Ejemplo:
    - 45x = 15 mod 200, gcd(45,200) = 5 soluciones
    - 9x = 3 mod 40
    - x = 3y, 9y = 1 mod 40
      * y = inverso modular de 9 en 40, se encuentra mediante Euclídes. Ver anterior ejemplo, demuestra que y=9
    - x = 3\*y = 27.
    - Hay 5 soluciones: 27 + k, k € [0,m)
    - non sei se hai algun motivo polo que non o fixen asi que non o borro, pero 90% seguro que se poderia facer:
    - 3x = 1 mod 40 e resolver esa ecuacion por euclides. nn sei por que fixen o cambio de variable

**Pequeño teorema de Fermat**

* Si P es un número primo y P no divide a A, **A(P-1) ≡ 1 mód P**
* Entonces, a^p ≡ a (mód p).
  + Ambas se cumplen también si P si divide a A (en este caso, a^p-a sería también múltiplo de p).

**Teorema de Euler:**

* Si gcd(a,m)=1, entonces aф(m)  ≡ 1 mód m, siendo ф la función de Euler
  + **Función de Euler**: Cantidad de enteros positivos menores a n y primos entre si con n. Si m primo, ф(m) = m-1

**Teorema chino de los restos:**

* Sean m1, m2, … mk primos relativos entre ellos.
* Sean a1, a2, … ak enteros cualesquiera. Entonces, el sistema:
  + x ≡ a1(mód m1)
  + x ≡ a2(mód m2)
  + …
  + x ≡ ak(mód mk)
* Tiene una **única solución** tal que x pertenece a [0, m), donde m = m1 \* m2 \* … \* mk. La solución es:
  + x =
  + Subíndice m1 => módulo m1
* Ejemplo:
  + x ≡ 2 (mód 3)
  + x ≡ 3(mód 5)
  + x ≡ 2(mód 7)
  + 3,5,7 son primos entre sí. m = 3\*5\*7 = 105
  + x =
    - [35 mod 3]-1 = [2 mod 3]-1 = 2
    - [21 mod 5]-1 = [1 mod 5]-1 = 1
    - [15 mod 7]-1 = [1 mod 7]-1 = 1
  + x = 140 + 63 + 30 = 233
  + x debe pertenecer a [0, 105). Entonces, x = 233 mód 105 = **23**

**Aplicaciones**

* **NIF:** La letra del final se decide así:
  + Se retiran la I, O, U y Ñ por confusión, quedando 23 letras (primo).
  + La letra depende del **resto** de dividir el NIF por 23, las letras están desordenadas.
* Letras/números de la parte de atrás:
  + Empezóa por IDESP, luego el número de soporte, NIF, incluye sexo, fechha de caducidad…
  + Se multiplica cada bloque de grupos de números/letras por (7,3,1) en grupos.
    - A=0, B=1…
  + Se suman los resultados y se hace mód 10. El resultado es el número de control que aparece tras los 3 grupos.
  + El último dígito de control se obtiene de la suma de todo el número
* **Billetes de euro:**
  + La primera letra es el país que emite el billete.
  + El último dígito es el dígito de control.
    - Se calcula el módulo 9 de la suma de dígitos y letras (en ascii) y se hace su complemento a 0.
    - No existen billetes que acaban en 0, los que sumen 9 mod 9 ponen 9.
  + No puedeh haber billetes de un país con números consecutivos.
* **IBAN tarjeta de crédito:**
  + Los dígitos en posición impar se multiplican por dos y se reducen a único dígito, sumando sus cifras. Luego se suman todos
  + Los dígitos en posición par se multiplican por 1, luego se suman todos y se hace módulo 10.
  + Se suma todo y es el dígito de control.
* **Ean-13 (código de barras)**
  + Los dígitos se multiplican por pesos, (1,3,1,3,1…) y se realiza módulo 10 y su complemento a 0 para hallar el dígito de control.

**Criterios de divisibilidad**

* **N=2**: Divisible si la última cifra es par.
  + ak10k + ak-110k-1 +…+ a3103 + a2102 + a110 + a0 mód 2 ≡ a0 mód 2
* **N=3**: Divisible si la suma de las cifras es divisible entre 3. 10k mod 3 = 1
  + ak10k + … + a2102 + a110 + a0 mód 3 ≡ ak + … + a2 + a1 + a0 mód 3
* **N=4**: Divisible si las dos últimas cifras son divisibles entre 4.
  + ak10k + ak-110k-1 +…+ a3103 + a2102 + a110 + a0 mód 4 ≡ a110 + a0 mód 4
* **N=5**: Divisible si la última cifra es divisible entre 5.
  + ak10k + ak-110k-1 +…+ a3103 + a2102 + a110 + a0 mód 5 ≡ a0 mód 5
* **N=6**: Divisible si lo es entre 2 y 3.
* **N=7**: Se elimina el último dígito y este se multiplica por 2 y resta al número. Luego, se comprueba si el resultado es divisible entre 7. Para calcular el resto:
  + … + a6106 + a5105 + a4104+ a3103 + a2102 + a110 + a0 mód 7 ≡

a6 + 5a5 + 4a4 + 6a3 + 2a2 + 3a1 + a0 mód 7

* + Explicación primer método:
    - n\*10 + a0 = 0 mód 7
    - n\*3 + a0 = 0 mod 7
    - n + 5a0 = 0 mod 7
    - n - 2a0 = 0 mod 7
* **N=8**: Divisible si las tres últimas cifras son divisibles entre 8.
  + ak10k + ak-110k-1 +…+ a3103 + a2102 + a110 + a0 mód 8≡ 100a2+10a1+a0 mód 8
* **N=9**: Divisible si la suma de las cifras es divisible entre 9. 10k mod 9 = 1
  + ak10k + … + a2102 + a110 + a0 mód 9 ≡ ak + … + a2 + a1 + a0 mód 9
* **N=11**: Divisible si la suma de las cifras pares menos las impares es 0 mod 11.
  + … + a3103 + a2102 + a110 + a0 mód 11 ≡ … - a3 + a2 - a1 + a0 mód 11

**Tema 4: Criptografía**

**Criptografía**

* Disciplina del estudio de los códigos cifrados.
* Usada para proteger tarjetas de crédito, documentos electrónicos, copyright etc.
* Consiste en la creación de dos operaciones, mutuamente inversas.

E: P —> C, D: P←–C. D\*E = idp

**Tipos de sistemas criptográficos**

* **Clave simétrica** (o clave privada): La llave de cifrado es idéntica a la de descifrado.
  + Por sustitución: consiste en sustituir cada letra del texto plano por una letra, cifra, número o signo convencional
    - El alfabeto plano se representa en minúsculas, el cifrado en mayusculas.
    - Por traslación: Desplazar cada letra un número de posiciones en el alfabeto.
      * Es un caso particular de cifrado afín, que cifra un mensaje como C = a\*P + b. En este caso a=1.
      * E: A → A, E: Z/26Z → Z/26Z, x → x+k
      * D: A → A, E: Z/26Z → Z/26Z, x → x-k
      * Cifrado afín: Se multiplica además por un entero **a** la posición de cada número, donde mcd(a,mod) = 1. De lo contrario, dos números cifrados se correspondrían con la misma solución.
  + Fácil de romper, en desuso
* **Clave asimétrica** (o clave pública): La llave de descifrado es diferente.
  + Algoritmos basados en funciones matemátricas, funciones unidireccionales: fáciles de calcular en un sentido pero dificil en sentido contrario.
    - Ejemplo: multiplicar varios primos | factorizar el resultado. (Esquema de cifrado **RSA**)

**RSA**

* Sea m el mensaje original, y sean p y q dos números primos.
* Sea n del receptor el producto de ambos números: n = p\*q
* Dado que el receptor conoce p y q, puede calcular fácilmente dos números e y d tal que **(me)d☰m mod n**
  + e tiene inverso multiplicativo módulo Φ(n), donde Φ(n) será (p-1)\*(q-1)
  + d = e-1 mod [Φ(n)]. (**inverso modular**, (d\*e-1 -1) es múltiplo de Φ(n)). Calculado mediante algoritmo de Euclides.
* La clave pública será e,n. La clave privada será d
* Para encriptar el mensaje, se realiza E(m) ≡ me mod n
* Para descifrarlo, el receptor calcula E(m)d ≡ m mod n
  + El emisor utiliza la clave pública del receptor, el receptor descifra el mensaje con su clave privada.

1. **Ciclo hamiltoniano:** Cualidad de un grafo en el que se pueden pasar por todos los puntos una sóla vez y volver al origen. [↑](#footnote-ref-0)