**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)

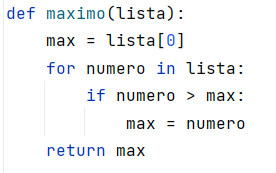
Para cada conjunto de valores de entrada, el algoritmo produce valores de salida que son la solución del problema.

* 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**

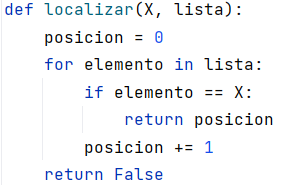
* **Encontrar máximo:** Permite encontrar el mayor número de una lista.

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.

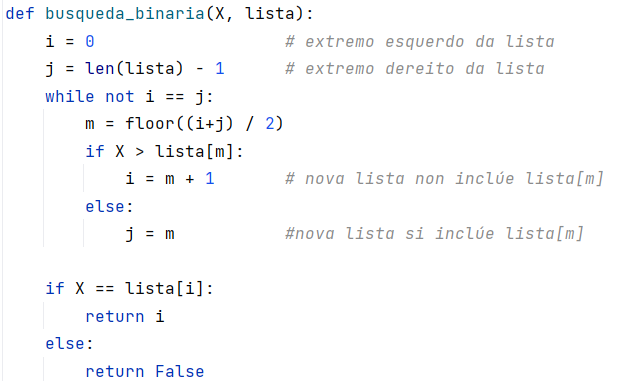
****

* **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**

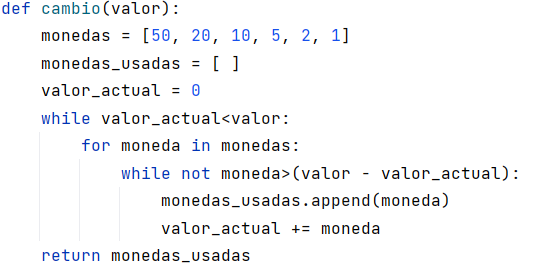
* Bubble sort: Complejidad cuadrática
* Selection sort: C. cuadrática
* Insertion sort: C. cuadrática
* Merge sort: C. cuasi-lineal
* Quick sort: C. cuadrática

**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**

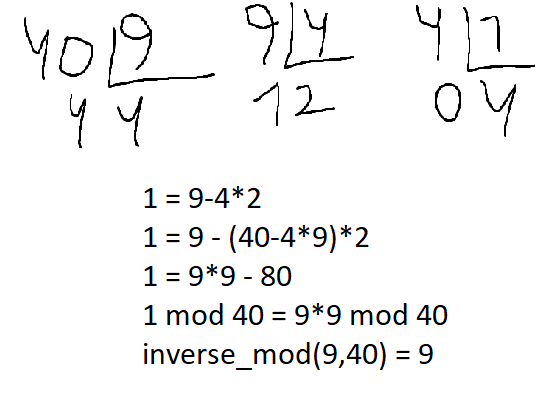
* Consiste en escribir todos los números de un conjunto, marcar los primos y tachar sus múltiplos para encontrar todos los primos.

**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.

**Tema 3: Recursividad**

**Definición**

* Una **definición recursiva** de una sucesión especifica lo siguiente:
  + Uno o más términos iniciales
  + Una regla para determinar los términos siguientes en función de los términos anteriores
    - Esta regla se llama **relación de recurrencia**
* La relación de recurrencia para la sucesión {an} es una ecuación que expresa el término an en función de uno o más anteriores a él.
* Una **recursión** es definir un ‘objeto’ en términos de si mismo.

**Ejemplos**

| **Sucesión** | **Relación de recurrencia** |
| --- | --- |
| 7, 17, 27, 37… | an = 10+an-1 |
| 1, 10, 100, 1000… | an = 10\*an-1 |
| 1, 3, 6, 10, 15 | an = an-1 + n+1 |
| 1, 2, 6, 24, 120, 720 | an = an-1 \* (n+1) |
| 1, 1, 2, 3, 5, 8, 13 | an = an-1 + an-2 |
| 1, 1, 4, 10, 28, 76 | an = 2(an-1+an-2)[[2]](#footnote-1) |
| 1, 1, 1, 3, 5, 9, 17, 31 | an = an-1 + an-2 + an-3 |

* Un **coeficiente binomial** (n|k) es una relación de recurrencia. Cada (n|k) se puede expresar en función de su previo: (n|k) = (n-1|k-1) + (n|k-1)
* Otro ejemplo es el **factorial:** n! = n\*(n-1)!

**Ecuación general**

* Siendo una relación de recurrencia an, la ecuación con an en función de n será la **ecuación general**.
* Por ejemplo, en el primer caso de la tabla, an = 10\*n + 7
* Una sucesión se llama **solución** de una relación de recurrencia si sus términos satisfacen la relación de recurrencia.
  + an = 10\*n+7 es solución de an = 10+an-1. Demostración:
  + 10n + 7 == 10 + [ 10(n-1) + 7 ]
  + 10n+7 == 10 + 10n -10 +7
  + 10n+7 = 10n+7

**Definición**

* Se llama **relación de recurrencia lineal** **homogénea con coeficientes constantes**, de orden k, a una expresión de la forma **an = α1an-1 + α2an-2 + … + αk\*an-k,** donde α1, …, αk son números reales y αk=/=0.
  + Una sucesión como la previa no sería homogénea por incluir un 10.
  + Una sucesión como an=n\*an-1 es de lineal de orden 1, pero no tiene coeficientes constantes.
  + an=2\*an-12 es una r.r. no lineal, por tener un término al cuadrado.
* Ejemplo: la secuencia de Fibonacci, que es de orden 2.

**Solución** de RRLHCC con raíces características distintas (únicas)

* Sea an = α1\*an-1 + … + αk \* an-k, con k=/=0.
* Es una RRLHCC de orden k, tal que las **raíces características** r1, …, rk son todas reales y distintas.
  + Buscamos soluciones {an} de la forma **an=rn**, con r=/=0 (el subíndice pasa a ser el exponente). Sustituyendo:
* rn = α1rn-1 + α2rn-2 + … + αkrn-k.
* Dividimos por rn-k para obtener rk = α1rk-1 + α2rk-2 + … + αk.
* **Ecuación característica** de grado k:
  + .
* **Resolvemos** la ecuación característica y obtenemos uno o más valores de r.
* Estos valores de r se toman como paámetros para la ecuación del enunciado.
  + Por ejemplo, en vez de an = an-1+an-2 se escribe an = α1\*() n + α2\*()n, siendo α1 escalares.
* Se reemplaza an y n por los valores iniciales de la recursión. Se obtiene un sistema que nos permite calcular los valores de las constantes α1 y α2

.

**Ejemplos** de soluciones de RRLHCC con raíces distintas (únicas)

* En **Fibonacci**, an=an-1+an-2. Condiciones iniciales: a0=0, a1=1
  + an = rn
  + rn = rn-1 + rn-2. Dividimos todo por rn-2
  + r2 = r + 1
  + r2 - r - 1 = 0
    - r2 - r - 1 = 0. Calculamos los valores de r que cumplen esta ec.
    - r1 = , r2 =
    - an = α1\*() n + α2\*()n, siendo α escalares.
    - a0 = α1\*() 0 + α2\*()0 para n=0
      * 0=α1+α2
    - a1 = α1\*() 1 + α2\*()1 para n=1
      * 1=α1\*()+α2\*()
    - Resolvemos el sistema de dos ecuaciones. Obtenemos α1=, α2=
  + an = ()n - ()n
* **2º ejemplo**: an = 10an-1
  + an = rn
  + rn = 10 \* rn-1. Dividimos todo por rn-1
  + r = 10
  + **Solución general:** an = α1\*10n
    - Conocemos que a0 = 1.
    - a0 = 1 = α1\*10 -> α1 = 1.
    - **an = 1\*10n**

**Solución** de RRLHCC con raíces características repetidas

* Sea an = c1\*an-1 + c2an-2 + … + ckan-k una relación de recurrencia lineal, homogénea y con coeficientes constantes.
* Sean sus raíces r1, …, rs reales y con multiplicidades respectivas m1, …, ms. Las soluciones serán de la forma:
* an = (α10 + α11\*n + … α1m1-1nm1-1)\*r1n + (α20 + α21\*n + … α2m2-1nm2-1)\*r2n + … + (αs0 + αs1\*n + … αsms-1ns1-1)\*r1s, para cualesquiera números reales αij
  + Cada uno de los términos [ (α10 + α11\*n + … α1m1-1nm1-1)\*r1n ] se corresponde con una de las diferentes raíces de la ecuación general. En la ecuación habrá uno de estos términos por cada raíz distinta.
  + El número de sumandos que habrá dentro es igual a la multiplicidad de esta raíz. El último sumando estará multiplicado por nm-1.

Ejemplo 1

* Sea an = 4an-1 - 4an-2
* Orden 2, r2 = 4r - 4
  + r2-4r+4=0 → (r-2)2 = 0
  + La solución es r=2 (doble).
* Solución general: an = (α + βn)2n
  + El único término es el (α + βn)2n, pues 2 es la única raíz. Dentro de este término hay dos sumandos, pues su multiplicidad es 2.
* Parámetros iniciales: a0=2, a1=8
  + a0 = (α + β\*0)2n = 2 => α = 2
  + a1 = (α + β2n = 8 => (2 + β)\*2 = 8 => β = 4
* an = (2+4n)2n

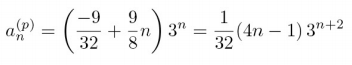
**Solución** de RRLNHCC **no homogénea**

* Sea an = c1\*an-1 + c2an-2 + … + ckan-k + L(n)
* Para solucionar relaciones de este tipo, debemos hallar dos soluciones distintas:
  + **an(p)** es una solución particular de la RRLNHCC.
  + **an(h)** es la fórmula de cualquier solución de la RRLHCC asociada, sin incluír el término L(n).
* Cualquier solución con término general **an = an(h) + an(p)** es solución de la RRLNHCC, y todas las soluciones son de esta forma.

**Ejemplo 2:** Torres de Hanoi

* an = 2an-1 + 1. Debido al + 1, no es homogéneo.
* Para 1 disco debo hacer 1 movimiento. Entonces, a1=1. Al ser an-1, es de orden 1.
* La parte homogénea será an = 2an-1
  + Ec. característica: r = 2.
  + Solución general de la parte homogénea: an(h) = α\*2n
* Buscamos una solución particular a la ec. completa.. Por ejemplo, an(p) = -1.
  + Declaramos que, para cualquier n, an=-1. Entonces, an = -1 y an-1 = -1. Entonces, se cumple la ecuación: an = 2an-1+1, pues -1 = 2\*(-1)+1
  + Esencialmente, reemplazamos tanto an como an-1 por x, y resolvemos la ec.
* Al tener an(p) y an(h), las sumamos:
  + an = an(p) + an(h) = α\*2n -1
  + Parámetro inicial: a1 = 1
  + α\*21 -1 = 1 → α=1.
* Solución: an =α\*2n -1 = **2n-1**

**Ejemplo 3:**

* an = an-2 + n\*3n, y el enunciado dice que la relación tiene una soluciuón particular de la forma an(p) = (β0+β1\*n)3n para algunos valores de β0 y β1.
* Ec. característica de la parte homogénea: r2 = 1. Las raíces son 1,-1. Solución general: an(h) = 1\*α - 1\*ß = α - ß
* La relación tiene una soluciuón particular de la forma an(p) = (β0+β1\*n)3n para algunos valores de β0 y β1. Igualamos las ecuaciones:
  + an = (β0+β1\*n)3n  = an-2 + n\*3n  = (β0+β1\*(n-2))2n-2 + n\*3n.
  + an-2 = (β0+β1\*(n-2))3n-2
* Dividiendo por 3n-2, nos queda (β0+β1\*n)\*9 = β0+β1(n-2) + 9. Creamos un sistema de ecuaciones con los valores n=0 y n=1.
* 8β0+2β1=0, 8β0+10β1=9. La solución es β0=-9/32, β1 = 9/8
* Entonces, una solución particular es: 

**Teorema de soluciones particulares de RRLNHCC**

* En una RRLNHCC an = c1\*an-1 + c2an-2 + … + ckan-k + L(n), donde L(n) = (p0+p1n+..ptnt)sn :
* Si s no es una de las raíces características de la relación homogénea asociada, la relación admite una solución particular de la forma **an(p) = (β0 + β1n +...+βtnt)sn**
* Si s sí es una de las raíces características de la relación homogénea asociada, y tiene multiplicidad m, la relación admite una solución particular de la forma **an(p) = (β0 + β1n +...+βtnt)nmsn**
  + Es decir, aumentamos el grado del polinomio tantas unidades como multiplicidad tenga la base del factor exponencial, s

**Ejemplo: Fórmula de los números triangulares**

* Sea an = 1+2+...+n. Encuentra una RRLCC de la que esta an sea solución, y resuélvela para dar una fórmula de la suma de los n primeros enteros positivos.
* Es de orden 1. Una posible relación sería an = an-1+n, pues en cada iteración sumamos n al número previo. Dividimos esta relación en sus dos partes, con an(h)= an-1 y an(p) una solución particular de an = an-1+q1.
* Empezando por an(h), la ecuación característica sería r-1=0, entonces r=1 y anh = α1n= α.
* Para encontrar la solución particular, utilizamos el teorema previo, viendo L(n) como L(n)=n\*1n, pues r=1 es una raíz característica y su multiplicidad es 1. Tendremos una solución de la forma:
* an(p) = (β0+β1 n)n1n = β0n + β1n2. Sustituímos en la relación de recurrencia previa:
  + (β0n + β1n2) = (β0(n-1)) + β1n2(n-1)2) + n
  + Para n=1 y n=2, β0+β1=1, β0+3β1=2. Resolvemos que β0=β1=½. La solución general de la relación de recurrencia será: an = an(h) + an(p)
* an = α + (n2+n)/2
* Como a1=1 es la condición inicial, concoemos que 1 = α + (1+1)/2, por lo que α=0. Entonces: an = 1+2+...+n = (n+n2)/2. Es la fórmula de los números triangulares.

**Grafos**

**Introducción**

* Estudio de la relación entre objetos (denominados **nodos** o **vértices**) y los puentes (denominados **ejes** o **aristas**) que los unen.
* **Aplicaciones:** Sistemas de aerolíneas, problemas situados en mapas (camino más corto entre 2 puntos), sudoku

**Definición**

* Un grafo es un par **G** **= (V, E)** donde V es el conjunto de vértices (nodos) y E es el conjunto de ejes (aristas).
  + **V** es un conjunto finito no vacío cuyos elementos se llaman vértices.
    - Los vértices se denotan con números: V={1,2,3…}
  + **E** es un conjunto cuyos elementos se llaman ejes.
    - Los ejes se denotan con letras minúsculas: E={a,b,c…}
    - Cada eje se puede denotar como los puntos que une: si b une los puntos 1 y 2, **b = {1,2}**.
  + Se tratará con grafos no dirigidos, donde los ejes no tienen dirección. Aquellos que sí tienen dirección son digrafos, y sus ejes se representan con flechas como vectores.
    - En grafos no dirigidos, los pares no son ordenados.

**Tipos de ejes/vértices**

* Un par de vértices u,v se dicen **adyacentes** si existe un eje e que los conecte.
  + Se dice entonces que el eje e **incide** en los vértices u,v. Entonces, u,v son los **extremos** del eje.
* Un vértice que no está conectado a ningún otro mediante un eje se denomina **aislado**.
* Si dos o más ejes conectan los mismos dos vértices, los ejes se denominan **paralelos** o **múltiples**.
* Un eje que conecta un vértice consigo mismo es un **lazo**.
* Un grafo que no tiene lazos ni ejes múltiples se llama **grafo simple.**

**Grado**

* El **grado** **S(v)** de un vértice v es el número de ejes que inciden en él.
* Un lazo cuenta como dos ejes al contar el grado.
* Los vértices de grado 1 se denominan **hojas**.

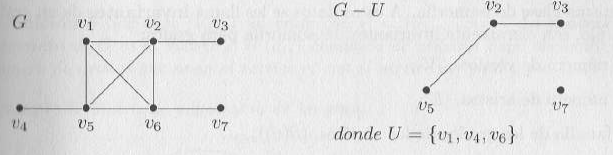
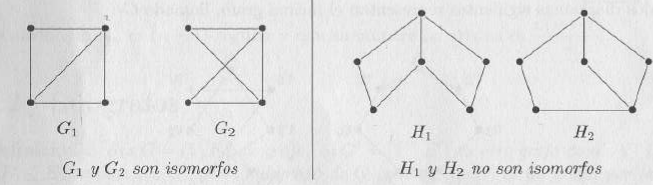
**Sucesión de grados**

* Se llama **sucesión de grados** a la sucesión ( S(v) )v€V, es decir, la lista completa de los grados de cada vértice en el grafo.
* Se pueden ordenar de menor a mayor grado o, más comúnmente, en orden de los vértices.
* **Lema del apretón de manos [handshaking lemma]:** la suma de los grados de todos los vértices de un grafo es igual al **doble** del número de ejes.
  + Esto es debido a que un eje siempre conecta dos puntos.
  + **Corolario:** En cualquier grafo, el número de vértices de grado impar es par.

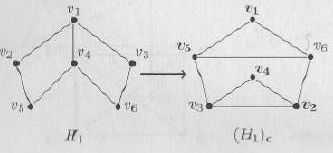
**Propiedades de un grafo**

* |V|: **Orden** del grafo. |E|: **Tamaño** del grafo
* Si en un grafo se cumple que todos los vértices tienen el mismo grado r, el grafo es **r-regular**.
  + Ejemplo: un grafo donde los vértices son de grado 2 es 2-regular.
* Un grafo donde todos los vértices están conectados con el resto de vértices es **completo**. Se denomina con KE.
  + Por ejemplo, el K1 es un sólo punto, el K2 es una línea entre 2 puntos, el K3 es un triángulo, el K4 es un cuadrado con las diagonales marcadas.
  + Un grafo completo tendrá (n|2) ejes.

**Subgrafo de un grafo**

* Sea G(V,E) un grafo. Si tengo otro grafo G’ = (V’,E’) se dice que G’ es un **subgrafo** de G si V’ ⊆ V y E’ ⊆ E.
  + Si V’=V, se denomina **subgrafo recubridor** de G.
* Sea G un grafo, y U C V. Se denota por **G-U** el subbgrafo que se obtiene de G eliminando los puntos que pertenecen a U y por lo tanto también los ejes que conectan con los vértices de V.
  + Es decir, se eliminan los vértices de U y también todos los ejes.que quedan desconectados tras hacerlo.
* Sean G=(V,E) y G’=(V’,E’). dos grafos. Se dice que G y G’ son **isomorfos** si existen dos aplicaciones biyectivas:
  + Φ: V → V’, Ψ: E → E’
  + De tal forma que e={v1, v2} ∈ E ⇔ Ψ(Φ(v1), Φ(v2)) ∈ E’
  + Es decir, la imagen del eje que conecta dos puntos en G conecta la imagen de esos puntos en G’, y viceversa.
  + Esto significa que los dos grafos son el mismo, aunque el dibujo sea distinto.
* Para comprobar si dos grafos son isomorfos, se consideran sus **invariantes:** los datos que son comunes a todos los grafos isomorfos. Los invariantes de un grado son **|V|, |E|** y  **( S(v) )v€V [[3]](#footnote-2)**
  + Es decir, si dos grafos tienen el mismo número de vértices, el mismo número de ejes y la misma lista de grados de vértices, serán isomorfos.

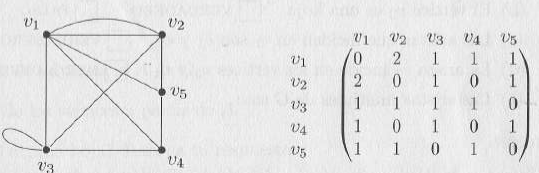
**Grafo complementario**

* Sea G(V,E) un grafo simple. El **grafo complementario de G**, **Gc(V,EC)** será el grafo con los mismos vértices pero sólo los ejes que no están en G.

(nótese los subíndices de los vértices)

* Dos grafos son isomorfos si y solo si sus complementarios lo son.

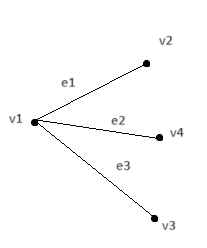
**Lista/matriz de adyacencia de un grafo**

* La **lista de adyacencia** de un grafo es una tabla que asocia cada vértice con los vértices que le son adyacentes (están conectados por un eje).
* La **matriz de adyacencia** de un grafo G = (V,E) tal que |V|=N es una matriz cuadrada de orden N donde la entrada aij nos da el número de ejes que conectan el vértice vi con el vértice vj.
* ****
* Es siempre una matriz simétrica. Los lazos se cuentan como un único eje.

**Matriz de incidencia**

* Sea un grafo G=(V,E) donde |V|=n y |E|=m. Su **matriz de incidencia** es una matriz de orden **n**x**m** donde el elemento bij es:
  + 1 si el eje j incide sobre el vértice i
  + 0 si no.
* Cada fila se corresponde con un vértice y cada columna se corresponde con un eje. En la fila del vértice v1, habrá 1s en los puntos correspontentes a ejes que inciden en v1.

|  | **e1** | **e2** | **e2** |
| --- | --- | --- | --- |
| **v1** | **1** | **1** | **1** |
| **v2** | **1** | **0** | **0** |
| **v3** | **0** | **0** | **1** |
| **v4** | **0** | **1** | **0** |

* **Ejemplo:**

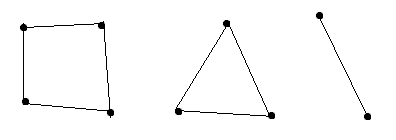
**Caminos**

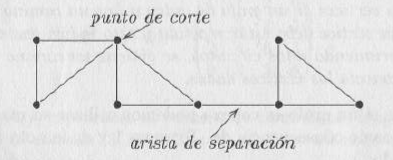
* Sea un grafo G = (V,E), un **camino** o trayectoria de un vértice v0 a otro vértice vn es una secuencia de ejes no necesariamente distintos (se pueden repetir) de G de la forma:
  + e1 = {v0, v1}
  + e2 = {v1, v2}...
  + en = {vn-1, vn}
* El nº de ejes se llama **longitud** del camino**, n.**
* Un camino de longitud mayor o igual que 1 en el que el vértice inicial coincide con el vértice final (v0=vn) se llama **circuito** o **ciclo**.
* Un camino donde todos los ejes son diferentes se llama **camino simple**.

**Caminos y ciclos**

* Un **camino** es una secuencia de aristas que conectan vértices adyacentes. Por ejemplo: [ e1={v0,v1}, e2={v1,v2}, …, en={vn-1, vn} ]
* El **número de caminos** de determinada longitud que existen entre dos vértices de un grafo se puede encontrar con su matriz de adyacencia, A.
  + La entrada aij será el número de caminos de longitud 1 que unen vi y vj.
  + Si calculamos **An**, la entrada aij de esta matriz será el número de caminos de longitud n que unen vi y vj.
* Un camino de longitud mayor o igual a 1 en el que **vn=v0** se denomina **ciclo**, circuito o camino cerrado.
* Un camino o ciclo en el que todas las artistas son distintas se denomina **simple**.

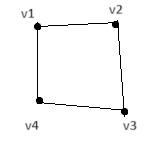
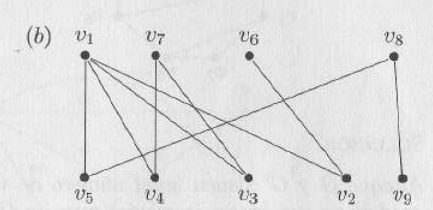
**Grafo conexo**

* Se dice que dos vértices u y v de un grafo G están **conectados** si existe un camino posible entre ellos o si son iguales.
* Un grafo se dice **conexo** si todos los vértices están conectados. En el caso contrario se dice **disconexo**.
*  (siendo todo un mismo grafo)
* En este caso, el grafo no es conexo porque no existe camino entre los vértices de figuras distintas.
* Sin embargo, podemos hablar de **componentes conexas**: un subgrafo conexo maximal (no está contenido en otra)
* Un grafo es conexo si y solo si tiene 1 única componente conexa.
* Sea G un grafo de orden n, y A su matriz de adyacencia. G es conexo si y solo si todas las entradas de la matriz (A+A2+...+An-1), son distintas de 0.
  + Con la excepción de la diagonal (lazos), que pueden ser 0 y el grafo sigue siendo conexo.

**Arista puente**

* Una arista e={vi , vj} de un grafo se dice **arista puente** o **arista de separación** si al suprimir la arista se rompe la conexión entre vi y vj.
* Un vértice v de un grafo G es un **punto de corte** si al suprimirlo aumenta el número de componentes conexas.

**Grafo bipartito**

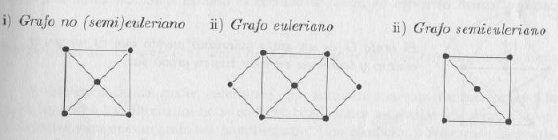
* Una **partición** de un conjunto V se trata de dividir un conjunto V en dos subconjuntos V1, V2, que son suplementarios.
* Sea un grafo G=(V,E). Si se puede realizar una partición de los vértices tal que no existen ejes entre los vértices de V1 ni ejes entre los vértices de V2, el grafo es **bipartito.**
* Ejemplo:
  + Dividimos el grafo entre V1={v1, v3} y V2={v2, v4}. No existen ejes entre los vértices de V1 ni entre los vértices de V2. El grafo es bipartito.
* Un grafo bipartito se dice **completo** si todo vértice de V1 es adyacente con todos los vértices de V2, y recíprocamente.
  + El ejemplo es bipartito completo,
  + Siendo |V1|=n y |V2|=m, el grafo bipartito completo se denomina **Kn,m**. O seu número de aristas é n\*m.
  + Para dibujar cualquier grafo Kn,m, se pueden dibujar n vértices arriba y m vértices abajo y luego conectar todos los de arriba con todos los de abajo. Ejemplo: K4,5:
  + 

**Teorema/algoritmo para grafos bipartitos**

* Un grafo es bipartito si y sioloo si todos sus ciclos son de longitud par.
* Algoritmo para dar una partición de un grafo bipartito:
  + Sea v€V un vertice cualquiera
  + V1={v’€V | d(v,v’) es impar}
  + V2={v’’€V|d(v,v’’) es par}
  + Es decir, tomamos un vértice del grafo y dividimos al grafo en dos partes: los vértices que tienen una distancia par con él y los que tienen distancia impar.

**Grafos Eulerianos**

* Sea G un grafo. Se llama un **camino Euleriano** a un camino simple que recorre todos los ejes del grafo, sin repetir.
* Un **circuito Euleriano** es un camino Euleriano cerrado (empieza y acaba en el mismo vértice).
* Un grafo es un **grafo Euleriano** si tiene un circuito Euleriano.
  + Un grafo no Euleriano que tiene un camino Euleriano se dice **semieuleriano**.
* **Teorema de Euler:** Sea G un grafo conexo. G es Euleriano si y solo si todo vértice tiene **grado par**.
  + **Corolario:** G es Semieuleriano si y solo si todos los vértices tienen grado par excepto 2. Estos dos serán los vértices de origen y final del camino euleriano.



**Algoritmo de Fleury** para construir un circuito o camino Euleriano

* Elegimos un vértice cualquiera
* Se recorren todos los ejes que forman un camino con las siguientes condiciones:
  + Cada eje se elimina una vez recorrido
    - De forma opcional, se pueden suprimir los vértices aislados
  + Sólo se selecciona un eje de separación (puente) cuando no haya otra opción.

**Grafos Hamiltonianos**

* Un **camino Hamiltoniano** es un camino simple (sin repetir ejes) que pasa por todos los vértices una única vez.
* Si el camino es cerrado, se dice **ciclo Hamiltoniano.**
* Un grafo es un **grafo Hamiltoniano** si tiene un ciclo Hamiltoniano.
  + De lo contrario se dice **semihamiltoniano** si tiene un camino Hamiltoniano.

**Teorema de Dirac**

* Sea G un grafo simple y conexo, con n vértices, n>=3.
* Si S(v)>=n/2, para todo vértice de V, entonces **G es Hamiltoniano**.

**Teorema de Ore**

* Sea G un grafo simple y conexo, con n vértices, n>=3.
* Si S(u)+S(v)>n, para cualquier par de vértices u,v no adyacentes, entonces **G es Hamiltoniano**.[[4]](#footnote-3)

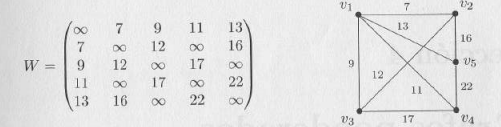
**Condición necesaria[[5]](#footnote-4)** para ciclo Hamiltoniano

* Si G es un grafo Hamiltoniano, entonces para cada conjunto U!=0 y U C V, el grafo G-U tiene a lo sumo |U| componentes conexas.
* Se deben comprobar todos los conjuntos U, deben cumplir todos la condición.

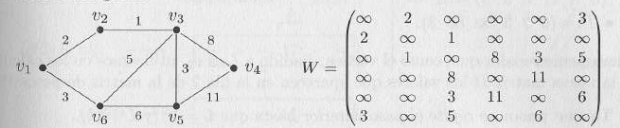
**Grafos ponderados**

* Un grafo se dice ponderado si existe una aplicación
  + w: E—>R+U{0} [positivos más el cero]
  + e—-->w(e), denominado el **peso del eje**.
* En un grafo ponderado, cada eje tiene un valor numérico asociado, que puede tener distintos significados.
* En un subgrafo G’ de un grafo ponderado, se define el **peso de G’** como la suma de los pesos de sus aristas, **w(G’)**.

**Matriz de pesos**

* Siendo G un grafo simple ponderado de orden n. Su **matriz de pesos** **W = (wij)** es una matriz cuadrada de orden n donde:
  + Si {vi, vj} € E, wij=w( {vi, vj} ).
  + Si no, wij = ∞.
* Se denomina **camino de peso mínimo** a cualquier camino entre dos vértices vi y vj que tenga peso mínimo, **w\*ij**.
  + La **matriz de pesos mínimos** es la matriz W\*=(**w\*ij)**, con w\*ij=∞ si no hay caminos entre vi y vj

**Algoritmo de Dijkstra[[6]](#footnote-5)**

* Algoritmo para calcular caminos de peso mínimo entre dos vértices.
* Este algoritmo genera dos conjuntos: L son los puntos cuya ruta óptima a v (vértice inicial) se conoce, y L’ son los que no.
* Definimos una matriz fila D de orden 1xn, que almacena los siguientes datos:
  + Si vi está en L, D(i) es la longitud de la ruta más corta de vi a v.
  + Si no, d(i) es la longitud de la mejor ruta conocida, que puede no ser óptima.
  + D se inicializa con vi=0 si vi=v, y vi=∞ si no.
* En cada paso del algoritmo se halla el vértice u de L’ que está a menor distancia de v. Este vértice se pasa a L.
* Luego, se compara cada entrada de D con el peso del camino entre esa entrada y el vértice eliminado más la entrada eliminada. Si D(j)>D(i)+wij, siendo i la entrada eliminada, D(j) se sustituye por D(i)+wij
* **Ejemplo:**
* Inicializamos D=( 0 ∞ ∞ ∞ ∞ ∞ ∞) y L=()
* **Primer paso:** Se añade a L el vértice con menor entrada en D: vi=v1. (i=1). Además, todas las demás entradas j que verifiquen que D(j)>D(i)+wij se cambian por D(i)+wij. Queda L={v1} y D=(0 2 ∞ ∞ ∞ 3).
* **Segundo paso:** Ahora, el vértice con menor entrada es v2. Se repite el proceso previo y se obtiene L={v1,v2} y D=(0 2 3 ∞ ∞ 3).
* Se repite este proceso hasta que L=V. Obtenemos finalmente D=(0 2 3 11 6 3)

**Grafos planos**

* Un **grafo plano** es aquel en el que se puede dibujar en el plano de manera que ningún par de ejes se corten salvo en los vértices que inciden.
* Una **representación plana** de un grafo plano es una representación en el plano donde no se cortan los ejes.
  + Una representación plana divide al plano en **regiones** o **caras**.
  + El grado de una cara en una rep.plana es el número de ejes ‘fronteras’ (que delimitan la cara, como si fuesen lados de la figura)
  + La suma de grados de las caras es 2 veces el nº de ejes de un grafo.
  + 𝛅(C) = 2\*|E|.
* **Teorema[[7]](#footnote-6) de Euler:** Sea G=(V;E) un grafo simple, conexo y plano. **|C|+|V|=|E|+2**.
  + Siendo |C| el número de caras.
* **Teorema de Kuratowski:** Sea G un grafo simple y conexo. G es plano si y solo si no contiene ningún subrafo isomorfo por subdimisiones elementales de K5 o K3,3.
  + **Subdivisión elemental por ejes:** añadir un vértice en un eje ya existente, dividiendo este eje en dos.
  + Isomorfo por subdimisiones elementales de K5 o K3,3: Si, mediante subdivisiones elementales podemos transformar el grafo en K5 o en K3,3.

**Coloración de grafos**

* Sea G un grafo simple. Una **coloración** de G consiste en asignar colores a cada uno de los vértices G, de manera que los vértices adyacentes tengan colores distintos.
  + Este problema es equivalente al de colorear pasíses en un mapa, si consideramos que cada vértice es un país, los ejes son fronteras.
* No existe ningún algoritmo en tiempo polinómico para la coloración de un grafo. En su lugar, asignamos un color cualquiera al primer vértice y luego vamos colorando cada vértice adyacente con colores distintos, definiendo un color nuevo cuando sea necesario. Este es un algoritmo voraz.

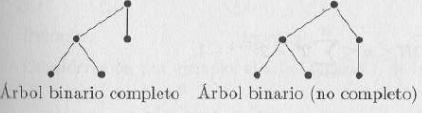
**Número cromático**

* El **número cromático** de un grafo G es el número mínimo de colores necesarios para una coloración de G. Se denota χ(G).
* Se tiene que χ(G)<=|V|.
* **Teorema de los cuatro colores:** Si G es plano, χ(G)<=4.

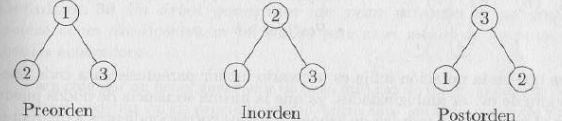
**Árbol**

* Un **árbol** (tree) es un grafo conexo sin circuitos (ejemplo: arbol genealógico)
* Un **bosque** es un grafo donde cada componente es un árbol.
* **Teorema:** Sea G(V,E) un grafo. Son equivalentes:
  + G es un árbol
  + Cada par de vértices distintos de V están conectados por un único camino
  + G es conexo y todos los ejes son de separación (puente)
  + G es conexo y |V| = |E| + 1
  + G no tiene circuito y |V| = |E| + 1

**Árbol con raíz**

* De un árbol podemos escoger un vértice cualquiera y designarlo como **raíz**. Este vértice se suele dibujar en la parte superior del grafo.
* Dado un vértice, su **padre** es otro vértice en un nivel superior al que está conectado.
* Dado un vértice, sus **hijos** son los vértices a los que está conectado en un nivel inferior.
* Un árbol con raíz se dice **m-ario** si todos sus vértices interiores[[8]](#footnote-7) tienen, a lo sumo, m hijos. El árvol será **m-ario completo** si los vértices tienen exactamente m hijos.
* La longitud de la trayectoria desde la raíz hasta un vértice v se denomina **altura** de v
  + La raíz se considera en nivel o altura 0.
* El máximo de todas las alturas será la **altura del árbol**.

**Subárboles**

* Sea v un padre. Si tiene un hijo izquierdo, el árgol que tiene a este hijo como raíz será el **subárbol izquierdo** de v. El subárbol derecho se define análogamente.
* Las distintas formas de leer un árbol son:
* 
* Mediante árboles binarios se pueden expresar operaciones algebraicas y su orden de realización, como a\*(b+c)

**Árbol generador**

* Sea G un grafo conexo. Un **árbol generador (recubridor)** de G es un subgrafo que es un árbol y contiene todos los vértices.
* Todo grafo conexo tiene un árbol generador. Por lo general, tienen varios distintos.
* Un **árbol generador de peso mínimo** de un grafo ponderado es el árbol generador cuyo peso es menor.

**Algoritmo de Kruskal** [Voraz, da solución óptima]

* Algoritmo para encontrar un árbol generador en un grafo.
* Se inicia un contador en i=1. Seleccionamos un eje de peso mínimo.
* Tomamos ei+1 de peso mínimo que queden del grafo que no formen un circuito con los ejes e1,e2,...ei. Los ejes no tienen que estar conectados
* Se repite el paso 2 hasta que no quede ningún eje.

**Algoritmo de Prim** [Voraz, solución óptima]

* Se elige un vértice cualquiera v. T0 = {v}. (primer árbol)
* Se considera el conjunto de ejes que inciden en el árbol construído. De esos ejes, se elige el de peso mínimo que no forme circuito con los anteriores. Se añade ese eje a el árbol. Los ejes deben estar conexos a el árbol.
* Se repite el proceso hasta que no se pueda añadir ningún eje más.

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)
2. Para esta sucesión es necesario especificar dos datos iniciales, ao y a1, que no vienen dados por la relación de recurrencia sino especificados manualmente. [↑](#footnote-ref-1)
3. También son invariantes el número de ciclos y sus longitudes y el número de componentes conexos. Además, todos los invariantes lo son también al considerarlos para cada componente conexo de cada grafo. [↑](#footnote-ref-2)
4. Nótese que el teorema de Dirac y Ore son condiciones suficientes, pero no necesarias. Existen grafos hamiltonianos que no satisfacen estas condiciones. Sin embargo, si un grafo satisface una de las ocndiciones, es Hamiltoniano. [↑](#footnote-ref-3)
5. Pero no suficiente. Aunque un grafo cumpla esta condición, puede no ser Hamiltoniano. [↑](#footnote-ref-4)
6. non entrou nunca e é un pouco bastada. probablemente non vaia entrar este ano tampouco pero nunca se sabe [↑](#footnote-ref-5)
7. nos examenes a veces chamalle fórmula de Euler a esto [↑](#footnote-ref-6)
8. Aquellos que no son hojas, los que tienen al menos 1 hijo. [↑](#footnote-ref-7)