TEMA 1: EFICIENCIA DE LOS ALGORITMOS

0. Notación
1. Medida de la complejidad
2. Análisis por casos
3. Notación asintótica

0. Notación

Representación de algoritmos en pseudocódigo (Formato 1):

```
ALGORITMO  <Nombre>
DATOS ENTRADA:     <nombre y tipo de las variables>
RESULTADO:         <nombre y tipo de las variables>
GLOBAL:            ...
AUXILIAR:          ...
METODO:
  <Instrucciones>
f<Nombre>
```

Los comentarios se escribirán con dobles barras (//).

0. Notación

Representación de algoritmos en pseudocódigo (Formato 2):

```
ALGORITMO <Nombre> (<param. entrada>) dev <param. Salida>
GLOBAL:            ...
AUXILIAR:          ...
METODO:
  <Instrucciones>
f<Nombre>
```

Global no implica que se implementará como global en el programa. Normalmente serán parámetros por referencia. También lo utilizaremos para declarar funciones.

0. Notación

Instrucciones:

- Instrucción condicional:

```
si <condición> entonces
  <instrucciones>
sino
  <instrucciones>
fsi
```

0. Notación

Instrucciones:

- **Instrucción iterativa:**
  - Mientras

```
mientras <condición>
   <instrucciones>
fmientras
```

  - Repetir

```
repetir
    <instrucciones>
hasta que <condición>
```

  - Para

```
para <rango de variación de la variable de control> hacer
   <instrucciones>
fpara
```

Ejemplo de rango: `i <- 1 hasta n`

0. Notación

Ejemplo:

```
ALGORITMO mcd  // de Euclides
DATOS:                n, m: N
RESULTADO:            m: N
AUXILIAR:             r: N
METODO:
   repetir
        r ← n mod m
        si r ≠ 0 entonces
              n ← m; m ← r
        fsi
   hasta que r = 0
fmcd
```


0. Notación

Ejemplo:

```
ALGORITMO mcd(n, m: N) dev m: N
AUXILIAR:             r: N
METODO:
   repetir
        r ← n mod m
        si r ≠ 0 entonces
              n ← m; m ← r
        fsi
   hasta que r = 0
fmcd
```


TEMA 1: EFICIENCIA DE LOS ALGORITMOS

1. Medida de la complejidad

- **¿Por qué medimos la complejidad?**
  - Para determinar que algoritmo es mejor dentro de una familia de algoritmos que resuelven el mismo problema.
- **¿Qué medimos?**
  - **Coste Temporal**: Tiempo empleado en la ejecución de un algoritmo para obtener un resultado a partir de los datos de entrada.
  - **Coste Espacial**: Espacio de memoria ocupado por un algoritmo antes, durante y después de su ejecución.

1. Medida de la complejidad

- **El coste temporal depende del tipo de lenguaje y del tipo de ordenador**
  - No es útil medir el coste en segundos
- **Mediremos el coste temporal en función de unidades de tiempo que dependerán del ordenador:**
  - t_a: Tiempo de una asignación.
  - t_{op} : Tiempo de una operación matemática.
  - t_c : Tiempo de una comparación.

1. Medida de la complejidad

Ejemplo: ***Realizar un algoritmo que calcule*** $y = \sum_{i=1}^{100} x$

Dos posibles algoritmos:

a) Algoritmo basado exclusivamente en el enunciado:

```
y ← 0
i ← 1
mientras (i ≤ 100) hacer
   y ← y + x
   i ← i + 1
f_mientras
```


1. Medida de la complejidad

a) Calculo del coste:

```
y ← 0                          ta
i ← 1                          ta
mientras (i ≤ 100) hacer       tc
   y ← y + x                   to+ta      } * 100
   i ← i + 1                   to+ta
f_mientras                     tc      (para salir del bucle)
```

$Tiempo_1 = 2t_a + 100 * (t_c + 2t_o + 2t_a) + t_c =$

$= 202\ t_a + 202\ t_o + 101\ t_c$

1. Medida de la complejidad

b) Si estudiamos el problema, podemos ver que sumar 100 veces x es lo mismo que multiplicar x por 100:

```
y ← 100 * x
```

El coste del algoritmo será:

$Tiempo_2 = t_o + t_a$

¿Qué algoritmo es mejor?

1. Medida de la complejidad

- **Talla de un problema:**
  - Además de depender del lenguaje y del tipo de ordenador, el coste también puede depender de la entrada del problema.

DEF. (Talla): Llamaremos talla de un problema al valor o conjunto de valores asociados a la entrada del problema y que representa una medida del tamaño del problema respecto de otras entradas posibles.

1. Medida de la complejidad

***Ejemplo:* Realizar un algoritmo que calcule** $y = \sum_{i=1}^{n} x$

```
Algoritmo Suma(x:Ent; n:Nat) dev y:Ent
Aux:    i: Nat
y ← 0                          ta
i ← 1                          ta
mientras (i ≤ n) hacer         tc
   y ← y + i                   to+ta      } * n
   i ← i + 1                   to+ta      }
f_mientras                     tc      (para salir del bucle)
```

$Tiempo_1 = 2t_a + n * (t_c + 2t_o + 2t_a) + t_c$

El tiempo total va a depender de **'n'**.

1. Medida de la complejidad

- **Análisis mediante contaje de pasos**
  - Consiste en no diferenciar entre tipos de operaciones elementales.

DEF. (Paso): Cualquier secuencia de operaciones con contenido semántico, el coste del cual es independiente de la talla del problema.