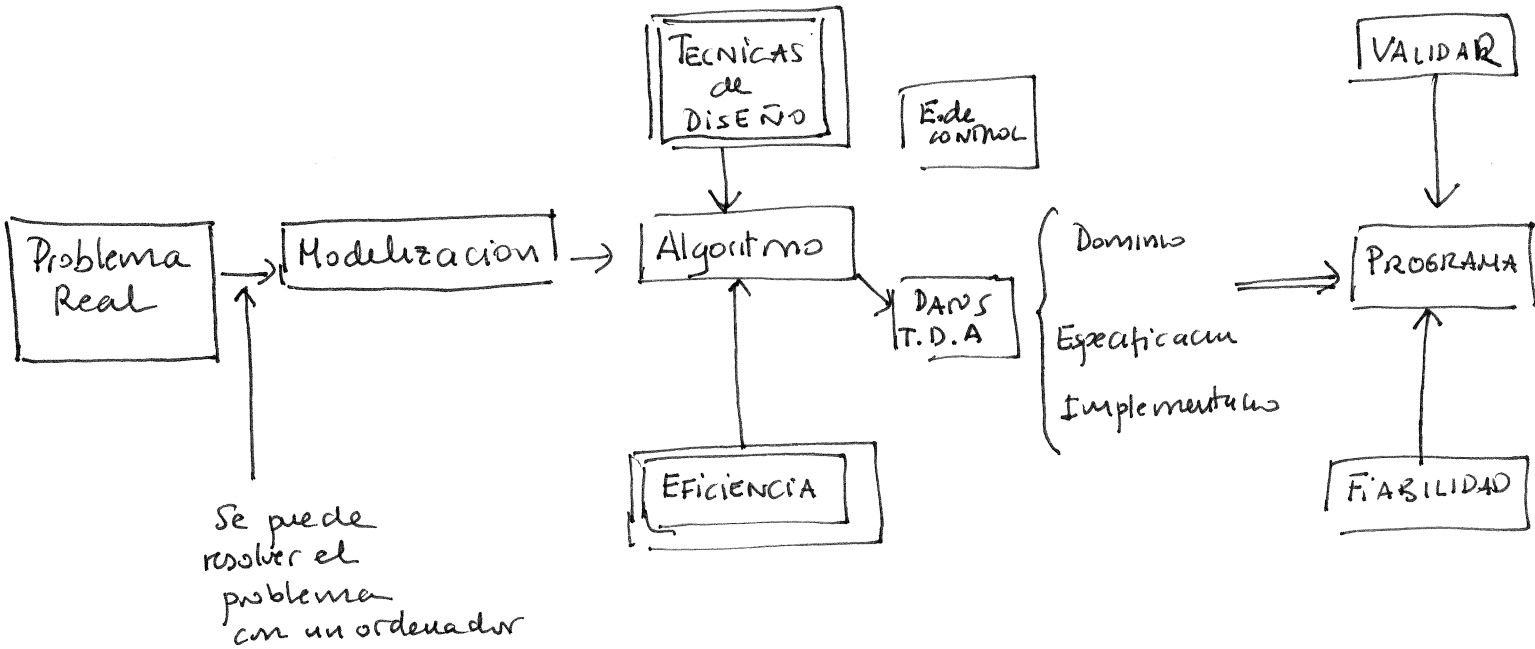
LECCIÓN 0: EFICIENCIA



~~ALGORITMO~~

Secuencia de instrucciones ordenadas capaz de resolver un problema dado.

CONSIDERACIONES SOBRE UN ALGORITMO

1) Eficiencia: recursos necesarios
   - memoria
   - tiempo de ejecución

2) Diseño del algoritmo
   - MÉTODO a seguir o ESTRATEGIA
     - D y V
     - Greedy
     - P.D
     - Ramificación

EFICIENCIA: depende de varios factores.

- datos de entrada
- calidad del código y optimización que nos de el compilador
- rapidez de las instrucciones máquina
- complejidad en sí del algoritmo.

EFICIENCIA
- a priori: obtener una función que acote (inferior y/o superiormente) el tiempo de ejecución del algoritmo para unos datos de entrada
- a posteriori: ejecutar el algoritmo para un conjunto de datos de entrada y medir el tiempo que gasta en un ordenador concreto.

LECCIÓN 0.- EFICIENCIA

TAMAÑO de la ENTRADA – nº de elementos sobre los que se va a ejecutar el algoritmo. (n)

TIEMPO de EJECUCIÓN – $T(n)$

$T(n)$ indica el nº de instrucciones elementales ejecutadas por un ordenador idealizado.

- Esta medida es independiente del ordenador donde se ejecute

- Un algoritmo puede tener más de una implementación pero la diferencia en eficiencia de las distintas implementaciones no es mayor que una cte multiplicativa

$$\boxed{T_1(n) \leq c\, T_2(n).} \quad \forall n \geq n_0,\ c > 0$$

siendo $T_1(n)$ ~~del~~ tiempo de ejecución de la implementación I_1 y $T_2(n)$ el tiempo de ejecución para la implementación I_2.

COMO ESCOGER ENTRE DOS ALGORITMOS QUE RESUELVEN UN PROBLEMA

EJEMPLO ¿Cómo calcular c y n_0?

$T_1(n) = 10^6 n^2 \qquad T_2(n) = 5n^3$

- intuimos que $T_1(n) \leq T_2(n)$ pero a partir de qué n ...

$$10^6 n^2 \leq c \cdot 5n^3$$

igualamos para ver donde se cruzan

$$10^6 n^2 = c5n^3$$

$$10^6 n^2 - c5n^3 = 0$$

$$n^2 (\underbrace{10^6 - c5n}_{0}) = 0$$

$$10^6 - c5n = 0 \Rightarrow n_0 = \frac{10^6}{5c}$$

escogiendo $c = 1$ $\boxed{n_0 = 2 \times 10^5}$

LECCION 0.- EFICIENCIA

COMPORTAMIENTO de UN ALGORITMO

Ej: Búsqueda secuencial

```
int Buscar (int a[], int n, int x){
   int j;
(1) j=0
(2) while (a[j]<x && j<n){
(3)        j = j+1
(4) }
(5) if (a[j] == x)
(6)        return j;
(7) else return -1;
}
```

OJO: No es que el vector sea muy pequeño

CASO MEJOR: Traza con el menor nº de O.E

Ej: el elemento x está en la 1ª posición

$T_m(n) = \underset{(1)}{1} + \underset{(2)\ \text{acceso a a[j]}}{1} + \underset{(2)\ \text{comparacion}}{1} + \underset{(5)}{1} + \underset{(5)}{1} + \underset{(7)}{1} = 6$

CASO PEOR: Traza con el mayor nº de O.E

Ej: el elemento x no está.

$T_p(n) = \underset{(1)}{1} + \underset{(2)}{4} + \sum_{j=0}^{n-1} (\underset{(3)}{2} + \underset{2}{4}) + 3 = 8+6n$

REGLAS GENERALES PARA EL CALCULO del NUMERO de O.E (operaciones elementales)

1: Tiempo de ejecucion de una O.E es 1

2.- Tiempo de ejecución de una secuencia de instrucciones se calcula sumando los tiempos de ejecución de cada una de las instrucciones

3.- Tiempo de ejecucion de una sentencia

```
switch c of
  case 1:
       Acciones 1: S1
       break;
  case 2:
       Acciones 2: S2
       break
   :
  case n:
       Acciones n: Sn
```

$T(n) = T(c) + \max\{T(S_1), T(S_2), \dots, T(S_n)\}$

4.- Tiempo de ejecucion de if - else

```
if (c)
   S1
else
   S2
```

$T(n) = T(C) + \max\{T(S_1), T(S_2)\}$

5.- Tiempo de ejecucion de while (c) S1

$T(n) = T(c) + n^{\circ}\text{-iteraciones} \times (T(S_1) + T(c))$

6.- Tiempo de ejecucion de

```
for (int i=0; i<n; i++)
        S
   ||
int i=0;
while (i<n){
    S
    i++;
}
```

$T(n) = 1 + 1 + n \times (T(S) + 1 + 1) = 2 + (T(S) + 2) \cdot n$

7.- Llamada a una funcion $f(P_1, \dots P_n)$

$T(n) = 1 + T(P_1) + \dots + T(P_n) + T(\text{Funcion})$