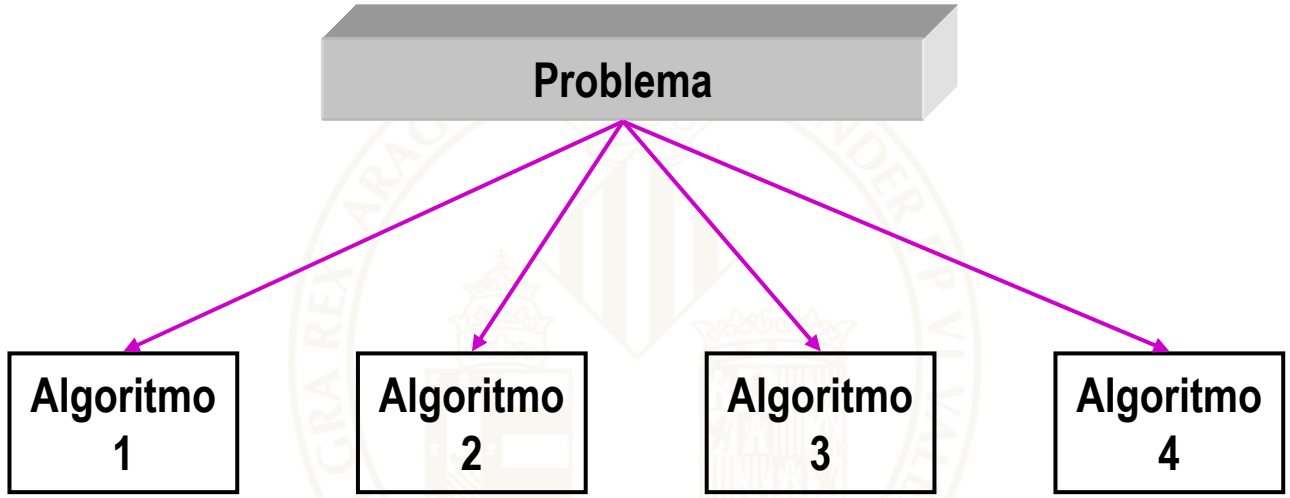
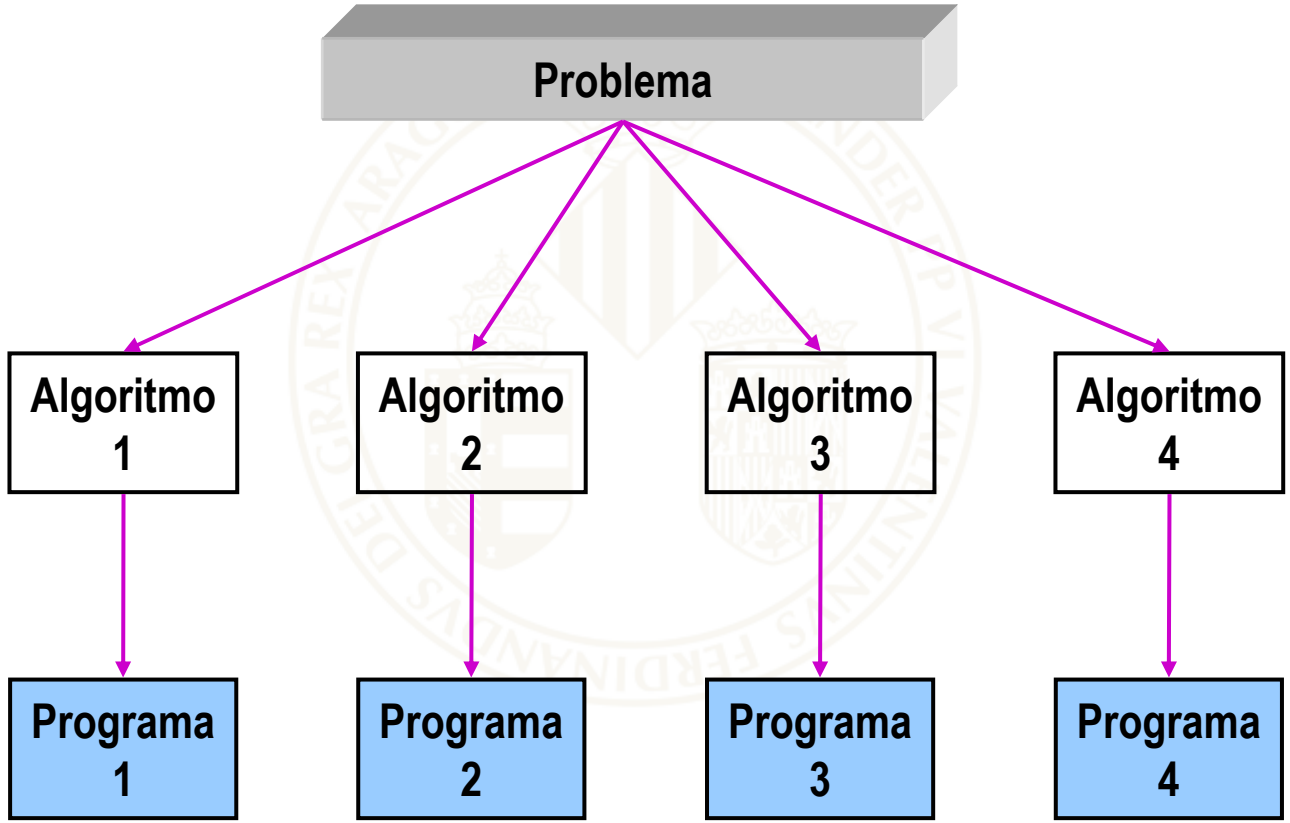
9.- Introducción al estudio de algoritmos y su complejidad

¿Qué algoritmo es mejor para resolver un problema?

¿Qué algoritmo es mejor para resolver un problema?



¿Qué algoritmo es mejor para resolver un problema?



Comparación de programas

- **No es una comparación fiable porque la eficiencia de los programas depende de factores "externos"**
  - ✓ Lenguaje de programación empleado
  - ✓ Compilador empleado
  - ✓ Máquina sobre la que se ejecute

Complejidad de un algoritmo

- **Definición 1**
  - ✓ El coste (o complejidad) temporal de un algoritmo es el tiempo empleado por éste para ejecutarse y dar un resultado a partir de los datos de entrada.

- **Definición 2**
  - ✓ El coste (o complejidad) espacial de un algoritmo es el espacio ocupado en memoria antes, durante y después de ejecutarse.

Medida de la complejidad

- Se trata de medir (preveer) la complejidad de un algoritmo antes de escribir el programa.

- En realidad se trata de estimar el coste de los algoritmos (tiempo o espacio).

Medida de la complejidad temporal

- **Cada operación que pueda aparecer en un programa consumirá un tiempo diferente. En general, desconocido por el programador.**
- **No cuesta los mismo**
  - ✓ `if (x<0)`
  - ✓ `i + j`
  - ✓ `i * j`
  - ✓ `sqrt(x)`
  - ✓ `i = j`

Medida de la complejidad temporal (2)

- **Estimación del coste temporal del algoritmo agrupando los tiempos de ejecución en grupos**

  - ✓ Operaciones aritméticas t_o
  - ✓ Asignaciones t_a
  - ✓ Comparaciones t_c

Ejemplo 1

- **Realizar un algoritmo que calcule**

$$y = \sum_{i=1}^{100} x$$

Ejemplo 2

- **Realizar un algoritmo que calcule**

$$y = \sum_{i=1}^{n} i$$

Definición de talla

- **Definición 3**
  - ✓ La talla de un problema corresponde al valor o conjunto de valores asociados a la entrada del problema que representa normalmente una medida de su tamaño respecto a otras entradas posibles.

RECUPERACIÓN DE INFORMACIÓN

- **El problema de la búsqueda**
  - ✓ Localizar un elemento de información en un conjunto de datos.
  - ✓ El resultado puede ser la información sobre el elemento buscado o una indicación sobre su ausencia.

- **Es un problema típico y <u>muy frecuente.</u>**

Búsqueda en un vector

202 400 268 145 967 954 569 239 268 481 802 942 901 340 19 153 442 592 123 99
852 593 218 160 711 166 331 463 767 637 194 658 238 194 403 391 903 683 896 288
889 199 614 247 46 803 639 67 948 981 583 70 286 516 296 49 639 937 102 415 484
221 948 293 827 423 307 292 528 51 10 730 129 341 547 709 816 339 132 681 541
499 802 428 706 280 64 28 321 649 355 276 174 361 401 488 50 181 541 322 425
120 67 206 324 435 439 205 102 102 71 882 402 352 992 209 295 358 828 272 920
85 253 786 304 228 496 152 237 869 441 565 669 533 911 780 880 40 25 490 782 86
898 548 892 401 311 154 769 685 245 402 881 901 623 87 901 663 469 19 197 636
562 397 317 661 991 855 706 746 850 341 924 721 851 225 56 123 934 437 856 404
831 667 942 112 385 432 364 687 729 743 264 792 623 438 810 31 658 518 40 31
491 253 993 930 999 527 572 62 523 963 854 148 867 853 903 125 30 210 284 791
938 129 69 527 268 476 751 701 490 929 482 903 251 157 360 683 865 681 514 421
377 104 622 581 241 223 245 601 357 160 958 411 588 6 509 225 506 567 554 284
597 997 907 75 646 90 694 697 635 975 265 281 327 932 500 583 723 218 506 692
663 235 942 555 755 584 225 712 156 996 951 924 353 340 277 575 655 444

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

Casos en la estimación del coste

- **Mejor caso**
  - ✓ Los parámetros del problema llevan al algoritmo a realizar el menor número posible de pasos (T^m).
- **Peor caso**
  - ✓ Los parámetros del problema llevan al algoritmo a realizar el máximo número posible de pasos (T^p).
- **Caso medio**
  - ✓ Está determinado por la media de pasos en función de todos los casos posibles (T^μ).

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

Definición de instancia

- **Definición 4**
  - ✓ Una instancia de un problema corresponde a todas las configuraciones diferentes de la entrada, de una talla determinada, que dan lugar al mismo comportamiento del algoritmo.

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

Ejercicio

- **Escribir una algoritmo (función) para obtener la posición del valor mínimo almacenado en un vector**

```
int minimo (Vector v, int n)
```

- **Analizar los diferentes casos para este algoritmo**

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

Búsqueda en un vector ordenado

6 10 19 19 25 28 30 31 31 40 40 46 49 50 51 56 62 64 67 67 69 70 71 75 85 86 87
90 99 102 102 102 104 112 120 123 123 125 129 129 132 145 148 152 153 154 156
157 160 160 166 174 181 194 194 197 199 202 205 206 209 210 218 218 221 223 225
225 225 228 235 237 238 239 241 245 245 247 251 253 253 264 265 268 268 268 272
276 277 280 281 284 284 286 288 292 293 295 296 304 307 311 317 321 322 324 327
331 339 340 340 341 341 352 353 355 357 358 360 361 364 377 385 391 397 400 401
401 402 402 403 404 411 415 421 423 425 428 432 435 437 438 439 441 442 444
463 469 476 481 482 484 488 490 490 491 496 499 500 506 506 509 514 516 518 523
527 527 528 533 541 541 547 548 554 555 562 565 567 569 572 575 581 583 583 584
588 592 593 597 601 614 622 623 623 635 636 637 639 639 646 649 655 658 658 661
663 663 667 669 681 681 683 683 685 687 692 694 697 701 706 706 709 711 712 721
723 729 730 743 746 751 755 767 769 780 782 786 791 792 802 802 803 810 816 827
828 831 850 851 852 853 854 855 856 865 867 869 880 881 882 889 892 896 898 901
901 901 903 903 903 907 911 920 924 924 929 930 932 934 937 938 942 942 942 948
948 951 954 958 963 967 975 981 991 992 993 996 997 999

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

Comparación de tiempos asociados a funciones coste

Se supone una potencia de cálculo de 1 MIPS

Coste	Talla n=10
$\lg n$	0.004 ms
n	0.01 ms
$n \lg n$	0.033 ms
n^2	0.1 ms
n^4	10 ms
2^n	1 ms
$n!$	3.6 seg
n^n	2 h 48 min

Comparación de tiempos asociados a funciones coste

Se supone una potencia de cálculo de 1 MIPS

Coste	Talla	
	n=10	n=20
$\lg n$	0.004 ms	0.005 ms
n	0.01 ms	0.02 ms
$n \lg n$	0.033 ms	0.086 ms
n^2	0.1 ms	0.4 ms
n^4	10 ms	160 ms
2^n	1 ms	1.05 seg
$n!$	3.6 seg	76,000 años
n^n	2 h 48 min	220 EU

Comparación de tiempos asociados a funciones coste

Se supone una potencia de cálculo de 1 MIPS

Coste	Talla		
	n=10	n=20	n=100
$\lg n$	0.004 ms	0.005 ms	0.007 ms
n	0.01 ms	0.02 ms	0.1 ms
$n \lg n$	0.033 ms	0.086 ms	0.66 ms
n^2	0.1 ms	0.4 ms	10 ms
n^4	10 ms	160 ms	1 min 40 seg
2^n	1 ms	1.05 seg	2.6 EU
$n!$	3.6 seg	76,000 años	2.10^{128} EU
n^n	2 h 48 min	220 EU	2.10^{170} EU

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

Inserción

53	22	41	11	-3	37	0	15	38	5

Vector original

									5
								38	5
								5	38
							15	5	38
							5	15	38
						0	5	15	38
						0	5	15	38
					37	0	5	15	38
					0	5	15	37	38
				-3	0	5	15	37	38
				-3	0	5	15	37	38

			11	-3	0	5	15	37	38
			-3	0	5	11	15	37	38
		41	-3	0	5	11	15	37	38
		-3	0	5	11	15	37	38	41
	22	-3	0	5	11	15	37	38	41
	-3	0	5	11	15	22	37	38	41
53	-3	0	5	11	15	22	37	38	41
-3	0	5	11	15	22	37	38	41	53

Vector ordenado

Inserción

```
void OrdenarInsercion (Vector v, int n)
{
        int i, j;

        for (i = n-2; i>=0; i--)
        {
                v[n] = v[i];
                j = i+1;
                while ( v[j] < v[n] )
                {
                        v[j-1] = v[j];
                        j++;
                }
                v[j-1] = v[n];
        }
}
```

Ordenar desde n hasta 0

Fijar el centinela

Desplazar elementos hasta fijar su posición

Selección

53	22	41	11	-3	37	0	15	38	5

Vector original

53	22	41	11	-3	37	0	15	38	5
53	22	41	11	-3	37	0	15	38	5
-3	22	41	11	53	37	0	15	5	38
-3	22	41	11	53	37	0	15	5	38
-3	0	41	11	53	37	22	5	15	38
-3	0	5	11	53	37	22	41	15	38
-3	0	5	11	53	37	22	41	15	38
-3	0	5	11	15	37	22	41	53	38
-3	0	5	11	15	22	37	41	53	38
-3	0	5	11	15	22	37	41	53	38
-3	0	5	11	15	22	37	38	53	41

-3	0	5	11	15	22	37	38	41	53
-3	0	5	11	15	22	37	38	41	53

Vector ordenado

Selección

```
void OrdenarSeleccion (Vector v, int n)
{
   int i, j, pos_min;
   int i_aux;

   for (i = 0; i < n - 1; i++)
   {
      pos_min = i;
      for (j = i + 1; j < n; j++)
         if (v[j] < v[pos_min])
            pos_min = j;

      if (pos_min != i)
      {
         i_aux = v[i];
         v[i] = v[pos_min];
         v[pos_min] = i_aux;
      }
   }
}
```

rdenar desde 0 hasta n

Localizar el mínimo

tercambiar con el mínimo

ALEXANDER PP VI
FERDINANDVS DEI GRA REX ARAGONVM

Intercambio (Burbuja)

```
void OrdenarBurbuja (Vector v, int n)
{
    int i, j;
    int i_aux;

    for (i = 1; i < n - 1; i++)
    {
        for (j = n - 1; j > i - 1; j--)
        {
            if (v[j - 1] > v[j])
            {
                i_aux = v[j];
                v[j] = v[j + 1];
                v[j + 1] = i_aux;
            }
        }
    }
}
```

Ordenar desde 0 hasta n

Intercambiar desde el final hasta i

Comparar por parejas e intercambiar

Quicksort (Partición)