

Algoritmos y Estructuras de Datos I

Tema 0. Introducción

Objetivo de la asignatura

Objetivo central

SER CAPAZ DE ANALIZAR, COMPRENDER Y RESOLVER UNA AMPLIA VARIEDAD DE **PROBLEMAS COMPUTACIONALES**, DISEÑANDO E IMPLEMENTANDO SOLUCIONES EFICIENTES Y DE CALIDAD, COMO RESULTADO DE LA APLICACIÓN DE UN PROCESO METÓDICO

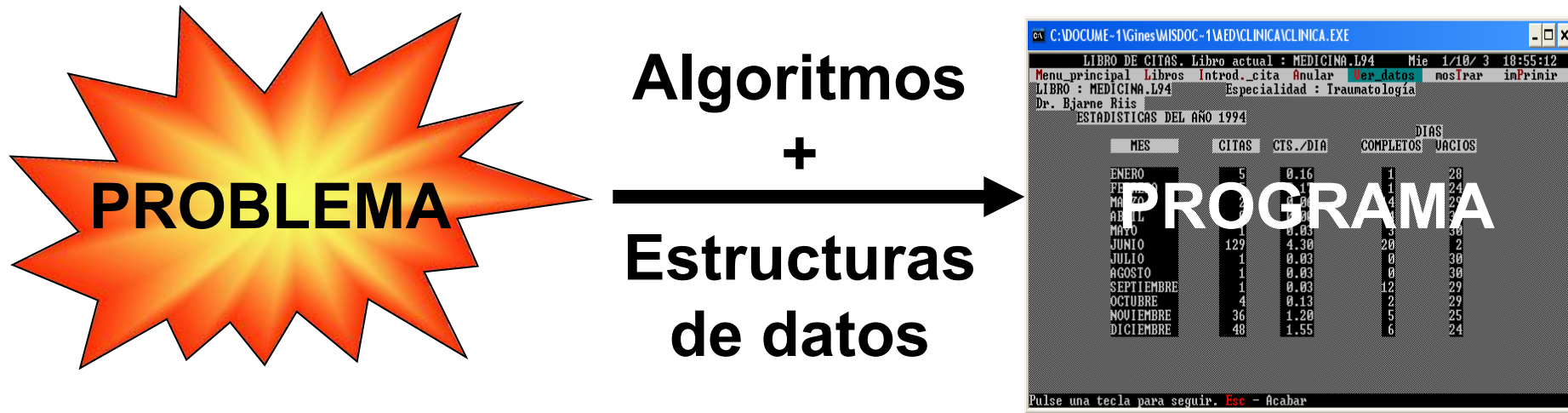
Resolver problemas

¿Cómo es el proceso para resolver un problema?

¿Qué clase de problemas?

¿Cuándo se dice que la solución es eficiente y de calidad?

Problemas, programas, algoritmos y estructuras de datos

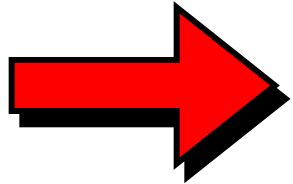


- **Problema:** Conjunto de hechos o circunstancias que dificultan la consecución de algún fin.
- **Algoritmo:** Conjunto de reglas finito e inambiguo.
- **Estructura de datos:** Disposición en memoria de la información.
- **Programa:** Algoritmos + Estructuras de datos.

Resolución de problemas

¿Cómo resuelve un problema de programación un ingeniero?

A) Tecleando código en una máquina.



B) Siguiendo un proceso metódico.

Resolución de problemas

ARQUITECTO

1. Estudio de viabilidad, análisis del terreno, requisitos pedidos, etc.
2. Diseñar los planos del puente y asignar los materiales.
3. Poner los ladrillos de acuerdo con los planos.
4. Supervisión técnica del puente.

INFORMÁTICO

1. **Análisis** del problema
2. **Diseño** del programa (alg. y estr.)
3. **Implementación** (programación)
4. **Verificación** y pruebas

Resolución de problemas

MÉTODO CIENTÍFICO

INFORMÁTICO

1.Observación.



1. **Análisis** del problema

2.Hipótesis.



2. **Diseño** del programa
(alg. y estr.)

3.Experimentación.



3. **Implementación**
(programación)

4.Verificación.



4. **Verificación** y pruebas

Evolución e historia de la programación

Lenguajes
de bajo nivel



(Basic, Fortran,
Ensamblador, ...)

Ejemplo de programa BASIC

```
10 PAPER 7: BORDER 7: INK 0: BRIGHT 0: FLASH 0
20 DIM a$(22,20): DIM f(22): DIM c(22): DIM g$(11,2): DIM z$(22,18):
  DIM x$(22)
30 FOR n= 1 TO 22
40 READ f,c: LET b$=CHR$ 19+CHR$ 1: LET f(n)=f: LET c(n)=c
50 FOR m=0 TO 2: READ r$
60 LET b$=b$+CHR$ 22+CHR$ (f+m)+CHR$ c+ r$
70 NEXT m: LET a$(n)=b$: NEXT n: GO SUB 470
80 CLS : FOR N=1 TO 22: PRINT A$(N): NEXT N: IF x$(1)<>" " THEN LET
  g$=x$
90 PRINT AT 0,2;"■■";AT 1,2;"■ EBEO";AT 2,2;"■ ";AT 3,2;"■ ■ OBLE";AT
  4,2;"■ "; INK 3; AT 19,16;"Adaptacion para"; INK 1;AT
  20,19;"MICRO";" HOBBY"
100 PLOT 128,0: DRAW 0,170: DRAW 10,4: DRAW 24,1: DRAW 82,0
110 PLOT 128,0: DRAW 10,4: DRAW 24,1: DRAW 88,0
120 DRAW 0,164: DRAW -2,2: DRAW 0,-164: DRAW -2,2: DRAW 0,164: DRAW -
  2,2: DRAW 0,-165
130 PLOT 128,0: DRAW -10,4: DRAW -24,1: DRAW -88,0
140 DRAW 0,164: DRAW 2,2: DRAW 0,-164: DRAW 2,2: DRAW 0,164: DRAW 2,2:
  DRAW 0,-164
150 PLOT 128,170: DRAW -10,4: DRAW -24,1: DRAW -82,0
160 DATA 1,12,"■ ","■ ","■ ","■ ",1,17,"■ ","■ ","■ ","■ ",1,22,"■ ","■ ",
  "■ ","■ ",1,27,"■ ■ ","■ "
170 PLOT 128,2: DRAW -10,4: DRAW -24,1: DRAW -85,0
180 PLOT 128,2: DRAW 10,4: DRAW 24,1: DRAW 85,0
```

Ejemplo de programa BASIC

```
290 DIM b$(22,2): FOR n=1 TO 11: FOR m=1 TO 2
300 LET s=INT (RND*22)+1
310 IF b$(s,1)=" " THEN LET b$(s,1)=g$(n,1): LET b$(s,2)=g$(n,2):
NEXT m: NEXT n: GO TO 330
320 GO TO 300
330 DIM r(22): LET di=0: LET itn=0: LET u=.001
340 PRINT AT 20,2;di: IF di=275000 THEN LET di=350000: PRINT AT
20,2; FLASH 1;di'"CONSEGUIDO EL PLENO EN ";itn;" veces": PRINT
#0;"Pulsa una tecla para empezar": GO SUB 440: GO SUB 440: GO SUB
440: PAUSE 0: GO TO 80
350 INPUT n: IF n>22 OR n<1 THEN GO TO 350
360 IF r(n)=1 THEN GO TO 350
370 LET k=n: GO SUB 700
380 INPUT m: IF m>22 OR m<1 OR m=n THEN GO TO 380
390 IF r(m)=1 THEN GO TO 380
400 LET k=m: GO SUB 700
410 LET itn=itn+1: IF b$(n)=b$(m) THEN LET di=di+25000: PAPER 3: LET
k=n: GO SUB 720: PAPER 3: LET k=m: GO SUB 720: LET r(n)=1: LET
r(m)=1: GO SUB 440: GO SUB 450: GO TO 340
420 BRIGHT 1: PAUSE 45: PAUSE 45: LET f=f(n): LET c=c(n): PRINT AT
f,c;a$(n,8);AT f+1,c;a$(n,14);AT f+2,c;a$(n,20): PRINT AT
f,c;a$(n,7 TO 8);AT f+1,c;a$(n,13 TO 14);AT f+2,c;a$(n,19 TO 20):
BEEP .01,-10: PRINT a$(n): BEEP .02,0
430 LET f=f(m): LET c=c(m): PRINT AT f,c;a$(m,8);AT
f+1,c;a$(m,14);AT f+2,c;a$(m,20): PRINT AT f,c;a$(m,7 TO 8);AT
f+1,c;a$(m,13 TO 14);AT f+2,c;a$(m,19 TO 20): BEEP .01,-10: PRINT
a$(m): BEEP .02,0: BRIGHT 0: GO TO 350
```

The flowchart illustrates the execution flow of the BASIC program. It starts at line 290, entering a nested loop for n=1 to 11 and m=1 to 2. Line 300 calculates a random index s. Line 310 checks if the character at b\$(s,1) is a space; if so, it updates it with g\$(n,1) and sets b\$(s,2) to g\$(n,2). It then loops back to line 300 (320). After the nested loop, line 330 initializes arrays r(22), di, itn, and u. Line 340 prints the current di value and checks for a win condition (di=275000). If won, it updates di to 350000 and prints the win message. It then enters a loop where it repeatedly calls GO SUB 440 and GO SUB 450, pausing between them. Line 350 takes input n and checks if it's within the valid range (1-22). If not, it loops back to 350. Line 360 checks if r(n)=1; if so, it loops back to 350. Line 370 calls GO SUB 700. Line 380 takes input m and checks if it's within the valid range and not equal to n. If not, it loops back to 380. Line 390 checks if r(m)=1; if so, it loops back to 380. Line 400 calls GO SUB 700. Line 410 increments itn and checks if b\$(n)=b\$(m). If so, it increments di by 25000, switches to PAPER 3, and calls GO SUB 720 twice. It then sets r(n)=1 and r(m)=1, calls GO SUB 440 and GO SUB 450, and loops back to line 340. Line 420 prints the current state of the board for both n and m, with BEEP sounds. Line 430 prints the current state of the board for both n and m, with BEEP sounds, switches back to BRIGHT 0, and loops back to line 350.

Ejemplo de programa BASIC

```
430 LET f=f(m): LET c=c(m): PRINT AT f,c;a$(m,8);AT
    f+1,c;a$(m,14);AT f+2,c;a$(m,20): PRINT AT f,c;a$(m,7 TO 8);AT
    f+1,c;a$(m,13 TO 14);AT f+2,c;a$(m,19 TO 20): BEEP .01,-10:
    PRINT a$(m): BEEP .02,0: BRIGHT 0: GO TO 350
440 BEEP .07,15: BEEP .06,25: BEEP .07,35: BEEP .07,35: BEEP
    .09,40: RETURN
450 INK 8: LET xx=c(n)*8-2: LET yy=177-(f(n)*8): PLOT xx,yy: DRAW
    27,0: DRAW 0,-27: DRAW -27,0: DRAW 0,27
460 LET xx=c(m)*8-2: LET yy=177-(f(m)*8): PLOT xx,yy: DRAW 27,0:
    DRAW 0,-27: DRAW -27,0: DRAW 0,27: INK 0: RETURN
470 RESTORE 260: FOR n=1 TO 22
475 IF n=17 THEN LET g$(6,2)=" ": GO TO 540
480 READ p$
490 FOR m=0 TO 7: READ f: POKE USR p$+m,f: NEXT m
520 IF n<12 THEN LET g$(n,1)=p$
530 IF n>11 THEN LET g$(n-11,2)=p$
540 NEXT n: RETURN
700 PAPER 5: LET y$=b$(k,1): LET t$=b$(k,2): LET f=f(k): LET
    c=c(k): BEEP u,25: PRINT AT f,c+2;t$;AT f+1,c+2;" ";AT
    f+2,c+2;" ": BEEP u,49: BEEP u,25
710 PRINT AT f,c+1;t$;" ";AT f+1,c+1;" ";y$;AT f+2,c+1;" v": BEEP
    u,49: BEEP u,25
720 PRINT AT f(k),c(k);b$(k,2);" ";b$(k,2);AT f(k)+1,c(k);"
    ";b$(k,1);" ";AT f(k)+2,c(k);" v ": BEEP u,49: PAPER 7: RETURN
```

Lenguajes de bajo nivel

- No existen procedimientos ni funciones
- No existen registros ni tipos definidos por el usuario
- No existen bloques estructurados (while, repeat, etc.)
- En definitiva: no hay **abstracciones**
- Y sin embargo... funciona:

<https://jsspeccy.zxdemo.org/>

Evolución e historia de la programación

Lenguajes
de bajo nivel

Lenguajes
estructurados



(Basic, Fortran,
Ensamblador, ...)

(Pascal, C,
Modula, ADA, ...)

Lenguajes estructurados

UNIT calculo;

Concepto de
módulo/unidad

INTERFACE

Separación de
interface/implementación

const

NMAX= 10;
MAX_GUARDA= 2000;

type

TDatosEnt= **array** [1..NMAX] **of** integer;

TDatosSal= **record**

NPasos: Shortint;

Paso: **array** [1..NMAX-1] **of** record

O1: byte;

O2: byte;

Fn: byte;

end;

end;

Tipos definidos
por el usuario

Procedimientos
y funciones

procedure Operar (var Arr: TDatosEnt; O1, O2, Func, Nivel: byte; var Vale: boolean); forward;

procedure CalculaCifras (var Entrada: TDatosEnt); forward;

procedure CalculaCifrasRec (var Entrada: TDatosEnt; PA, PB, Func, Nivel: byte); forward;

Lenguajes estructurados

IMPLEMENTATION

var

```
suma, num: integer;  
CopiaOrden: TDatosEnt;
```

Separación
interface/
implementación

procedure OrdenaComb (var Entrada: TDatosEnt; Nivel: byte);

var

```
i, j, maxim, pmaxim, tmp: integer;
```

begin

```
CopiaOrden:= Entrada;
```

```
num:= Nivel;
```

for i:= 1 **to** Nivel-1 **do begin**

```
    maxim:= CopiaOrden[i];
```

```
    pmaxim:= i;
```

```
    j:= i+1;
```

while j<=Nivel **do begin**

```
    if CopiaOrden[j]>maxim then begin
```

```
        maxim:= CopiaOrden[j];
```

```
    ....
```

```
    end;
```

```
end;
```

```
end;
```

Procedimiento
con parámetros

Bloques de
control
estructurados

Lenguajes estructurados

- Procedimientos y funciones son **abstracciones de control**
- Los tipos definidos por el usuario son **abstracciones de datos**
- Las unidades, módulos o paquetes son abstracciones de nivel superior: **abstracciones de funcionalidades**

Lenguajes estructurados

Inconvenientes:

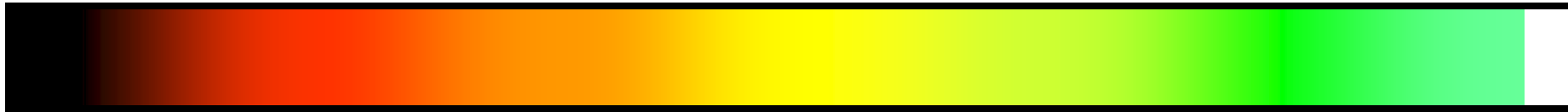
- Los datos y los procedimientos de manipulación sobre los mismos van por separado.
- Es necesario garantizar la ocultación de la implementación.
- Proliferación de variables globales. ¿Qué papel juegan?
- Los programas son cada vez más complejos y difíciles de mantener.

Evolución e historia de la programación

Lenguajes
de bajo nivel

Lenguajes
estructurados

Lenguajes
orientados a objetos



(Basic, Fortran,
Ensamblador, ...)

(Pascal, C,
Modula, ADA, ...)

(Smalltalk, C++,
Java, Eiffel, ...)

Lenguajes orientados a objetos

// Interface

```
class Timer {  
    private:  
        double StartTime;  
        double ClockRate;  
    public:  
        Timer (void);  
        bool StartTimer (void);  
        double ReadTimer (void);  
        bool Exists;  
};
```

Una clase es un Tipo Abstracto de Datos

Encapsulación de datos y operaciones

```
class Elipse {  
    protected:  
        double Fcx, Fcy;  
        double Frx, Fry, Fang;  
        void FsetXY (int x1, int y1, int x2, int y2);  
    public:  
        Elipse (int x1, int y1, int x2, int y2);  
        Elipse * Clonar (void);  
        void Pinta (IpImage *image, int color= 0, int ancho= -1);  
};
```

Los datos son privados

Las operaciones son públicas

Lenguajes orientados a objetos

- Una **clase encapsula** los datos de un tipo y las operaciones sobre el mismo
- Una clase es, al mismo tiempo, un **tipo abstracto de datos** y un **módulo** que encierra un conjunto de funciones relacionadas
- Separación clara entre **interface** (parte visible desde fuera) e **implementación** (oculta)

Saber y Ganar - Anagramas

Tema: manjares Ejemplo: vaciar → caviar



<https://www.rtve.es/alacarta/videos/saber-y-ganar/saber-ganar-15-09-20/5663558/>

Min: 5:50

Saber y Ganar - Anagramas

- Es un interesante problema computacional: dado un conjunto de palabras y dada una palabra, buscar todos sus anagramas en el conjunto.
- ¿Cómo lo resolvemos?
 - Opción 1: generar todas las permutaciones. Para cada una, buscarla en el conjunto.
 - Opción 2: recorrer todo el conjunto y, para cada palabra, ver si tiene las mismas letras.
- ¿Cuál es la forma más rápida?

Saber y Ganar - Anagramas

- Supongamos que el diccionario de español contiene 2 millones de palabras.
- Anagramas de: AMOR
 - Opción 1. En total existen $4 \cdot 3 \cdot 2 \cdot 1$ anagramas = 24 búsquedas en diccionario
 - Opción 2. Recorrer todo el conjunto son 2 millones de comprobaciones.
- La opción 1 es claramente más rápida.

AMOR
AMRO
AOMR
AORM
AROM
ARMO
RAMO
RAOM
ROAM
ROMA

...

Saber y Ganar - Anagramas

- Supongamos que el diccionario de español contiene 2 millones de palabras.
- Anagramas de: ESTRAFALARIO
 - Opción 1. En total existen $12!$ anagramas = $12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 479$ millones de búsquedas en diccionario
 - Opción 2. Recorrer todo el conjunto son 2 millones de comprobaciones.
- La opción 2 es claramente más rápida.

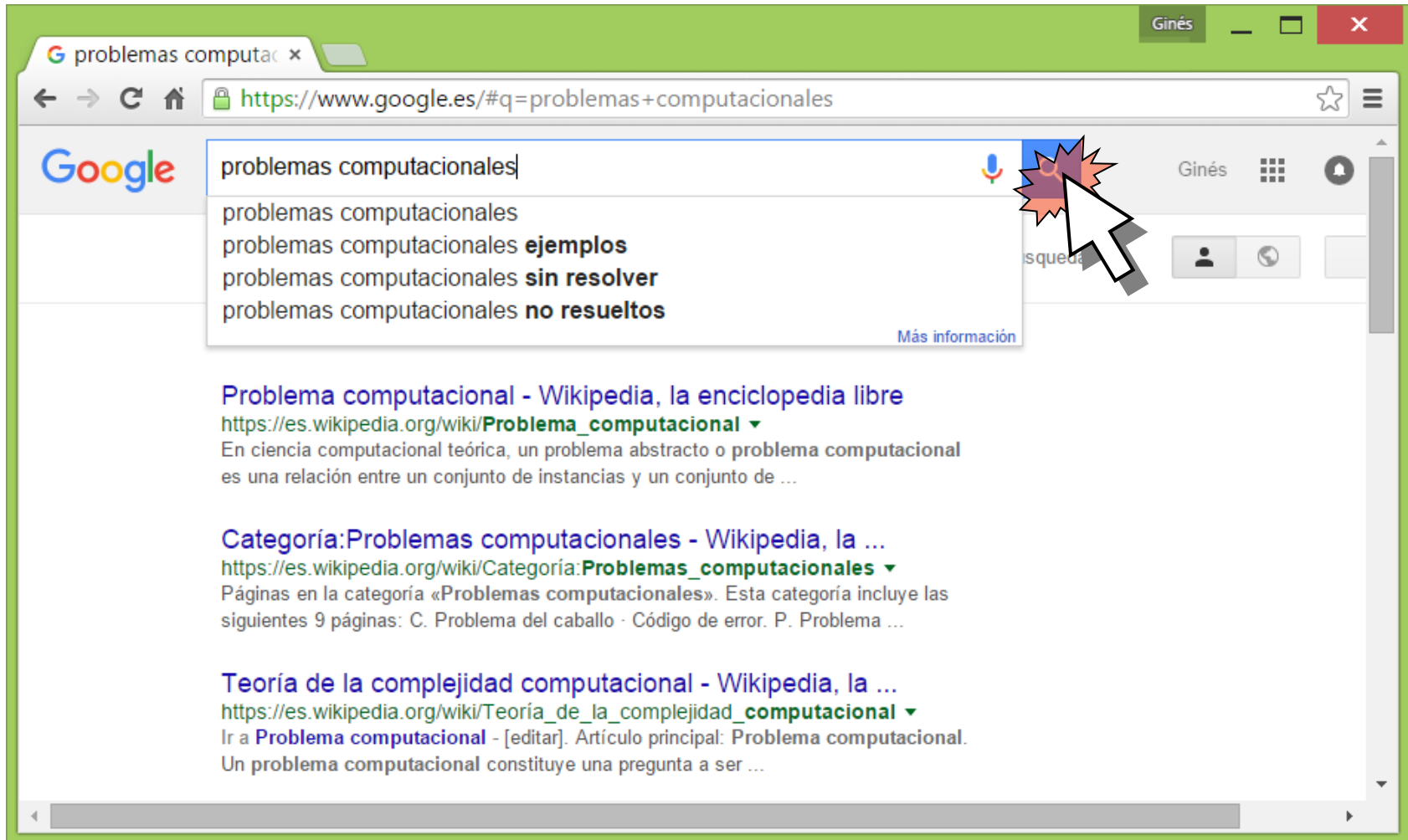
Saber y Ganar – Enredo de letras



<https://www.rtve.es/play/videos/cifras-y-letras/programa-99/16180806/>

Minuto 22:20

Ejemplos de problemas



A screenshot of a Google search interface in Spanish. The browser window has a green title bar with the name 'Ginés' and standard window controls. The address bar shows the URL 'https://www.google.es/#q=problemas+computacionales'. The search bar contains the text 'problemas computacionales'. Below the search bar, a dropdown menu displays four suggestions: 'problemas computacionales', 'problemas computacionales **ejemplos**', 'problemas computacionales **sin resolver**', and 'problemas computacionales **no resueltos**'. A mouse cursor with a starburst effect is pointing at the second suggestion. To the right of the suggestions is a 'Más información' link. Below the suggestions, the first search result is 'Problema computacional - Wikipedia, la enciclopedia libre', with the URL 'https://es.wikipedia.org/wiki/Problema_computacional'. The snippet below the title reads: 'En ciencia computacional teórica, un problema abstracto o problema computacional es una relación entre un conjunto de instancias y un conjunto de ...'. The second result is 'Categoría:Problemas computacionales - Wikipedia, la ...', with the URL 'https://es.wikipedia.org/wiki/Categoría:Problemas_computacionales'. The snippet reads: 'Páginas en la categoría «Problemas computacionales». Esta categoría incluye las siguientes 9 páginas: C. Problema del caballo · Código de error. P. Problema ...'. The third result is 'Teoría de la complejidad computacional - Wikipedia, la ...', with the URL 'https://es.wikipedia.org/wiki/Teoría_de_la_complejidad_computacional'. The snippet reads: 'Ir a **Problema computacional** - [editar]. Artículo principal: Problema computacional. Un problema computacional constituye una pregunta a ser ...'. The browser's user interface includes the Google logo, navigation buttons, and a user profile icon labeled 'Ginés'.

problemas computacionales

- problemas computacionales
- problemas computacionales **ejemplos**
- problemas computacionales **sin resolver**
- problemas computacionales **no resueltos**

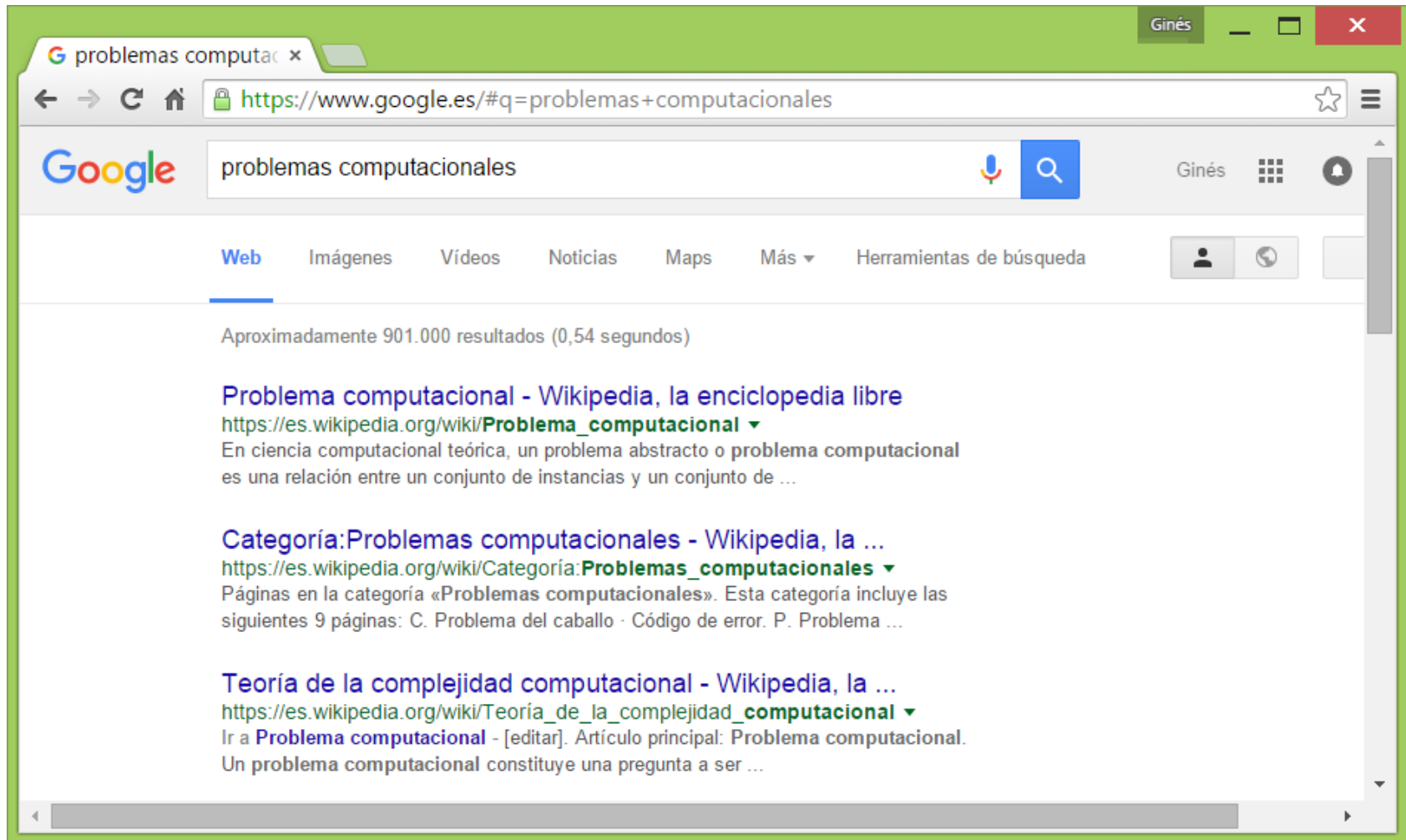
Más información

Problema computacional - Wikipedia, la enciclopedia libre
https://es.wikipedia.org/wiki/Problema_computacional ▼
En ciencia computacional teórica, un problema abstracto o problema computacional es una relación entre un conjunto de instancias y un conjunto de ...

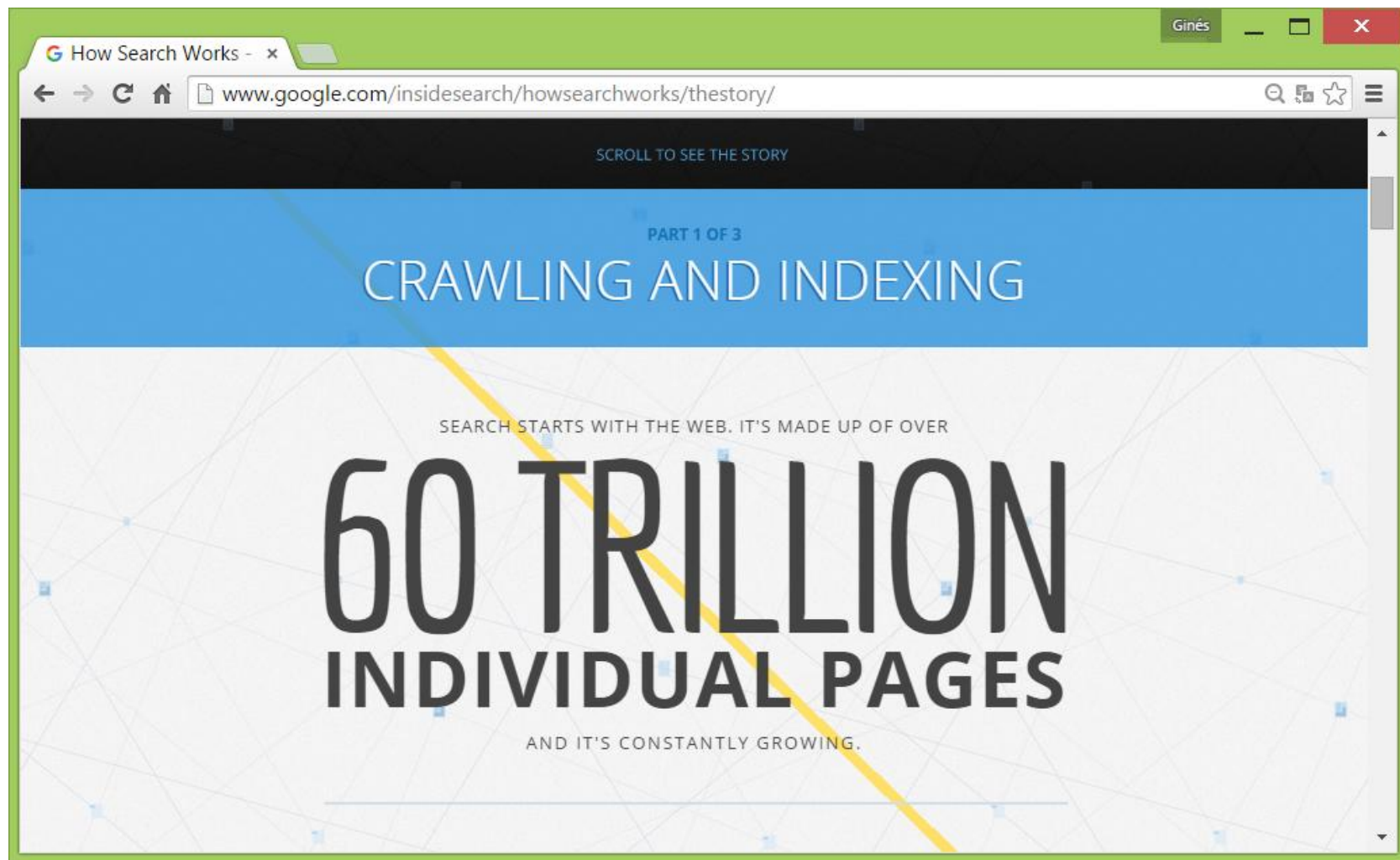
Categoría:Problemas computacionales - Wikipedia, la ...
https://es.wikipedia.org/wiki/Categoría:Problemas_computacionales ▼
Páginas en la categoría «Problemas computacionales». Esta categoría incluye las siguientes 9 páginas: C. Problema del caballo · Código de error. P. Problema ...

Teoría de la complejidad computacional - Wikipedia, la ...
https://es.wikipedia.org/wiki/Teoría_de_la_complejidad_computacional ▼
Ir a **Problema computacional** - [editar]. Artículo principal: Problema computacional. Un problema computacional constituye una pregunta a ser ...

Ejemplos de problemas

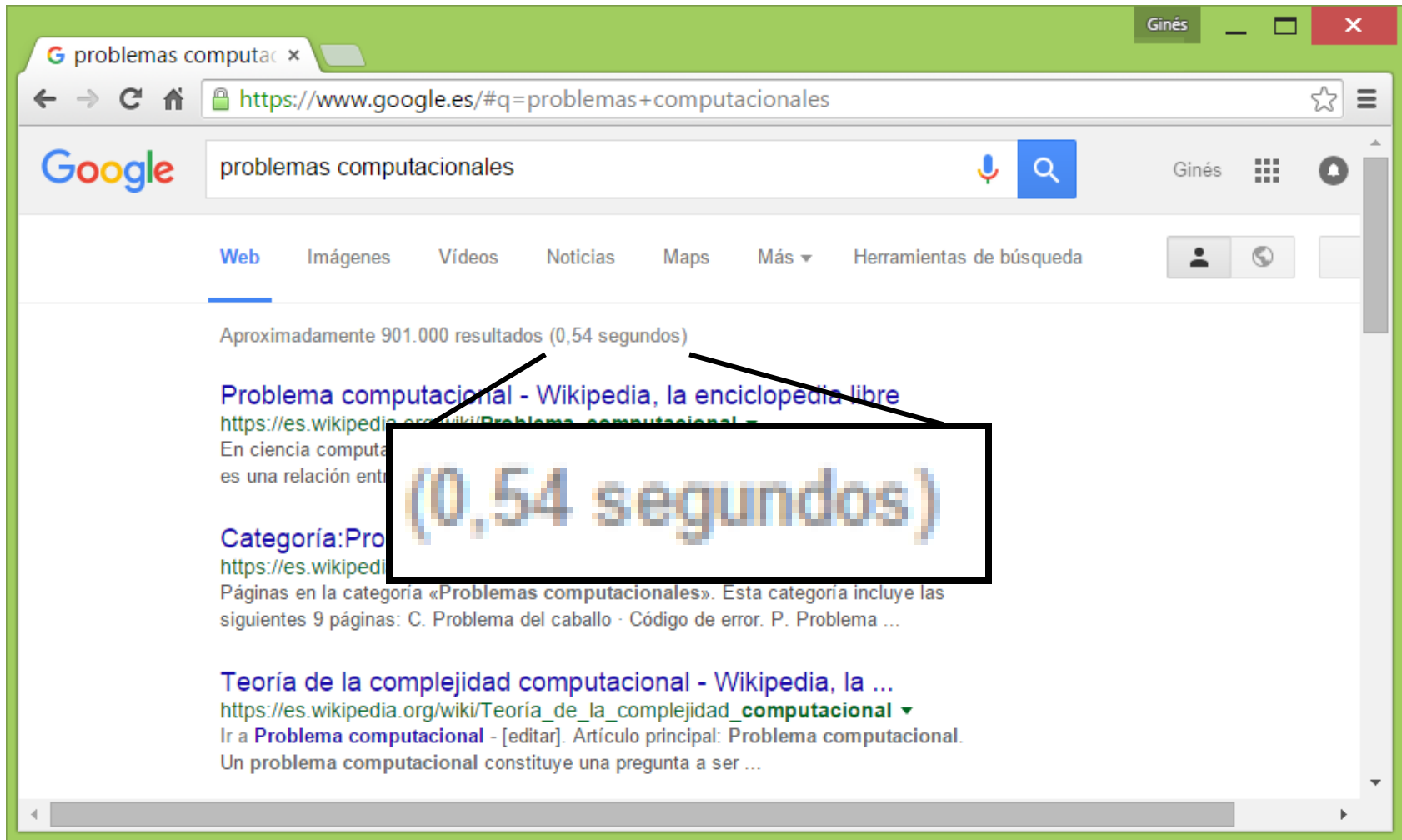


Buscador de Internet



60 trillion = 60 billones = 60₂000.000₁000.000

Buscador de Internet

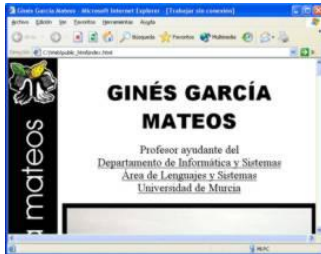


Buscador de Internet

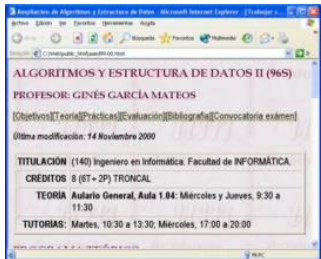
- ¡¡¡60 billones páginas en medio segundo!!!
- **Problema:** ¿cómo estructurar la información necesaria para realizar las consultas rápidamente? ¿Qué algoritmos de búsqueda utilizar?

Buscador de Internet

Opción 1. Para cada página, su lista de palabras



algoritmos, ayudante, curso, datos, estructuras, garcía, ginés, mateos, ...



algoritmos, cosa, curso, datos, estructuras, evaluación, prácticas, ...



agua, botavara, barco, confeccionar, las, velas, ...

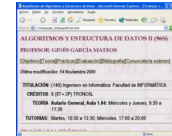
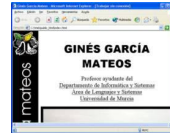
Buscador de Internet

Opción 2. Para cada palabra, su lista de páginas

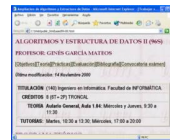
agua



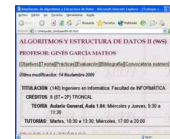
ayudante



cosa



las



...

Buscador de Internet

Opción 1: Páginas → Palabras

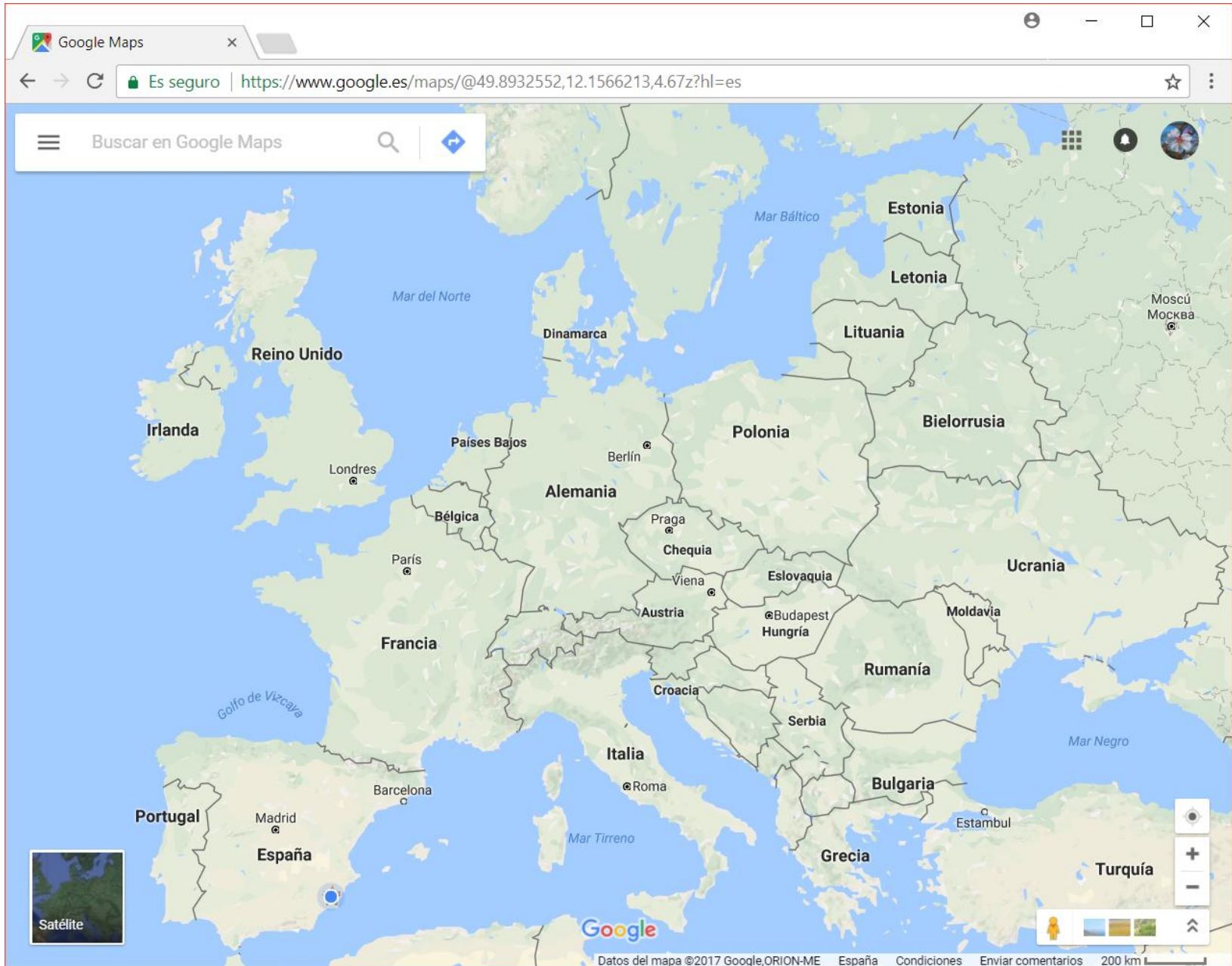
- Supongamos una red de 100.000 ordenadores a 3 GHz con 16 núcleos.
- Supongamos que cada página tiene 100 palabras, de 8 letras cada una y en cada letra se tarda 2 ciclos de reloj. Tenemos 60 billones de páginas.
- ¡¡El recorrido de todas las páginas tardaría 20 segundos!!

Buscador de Internet

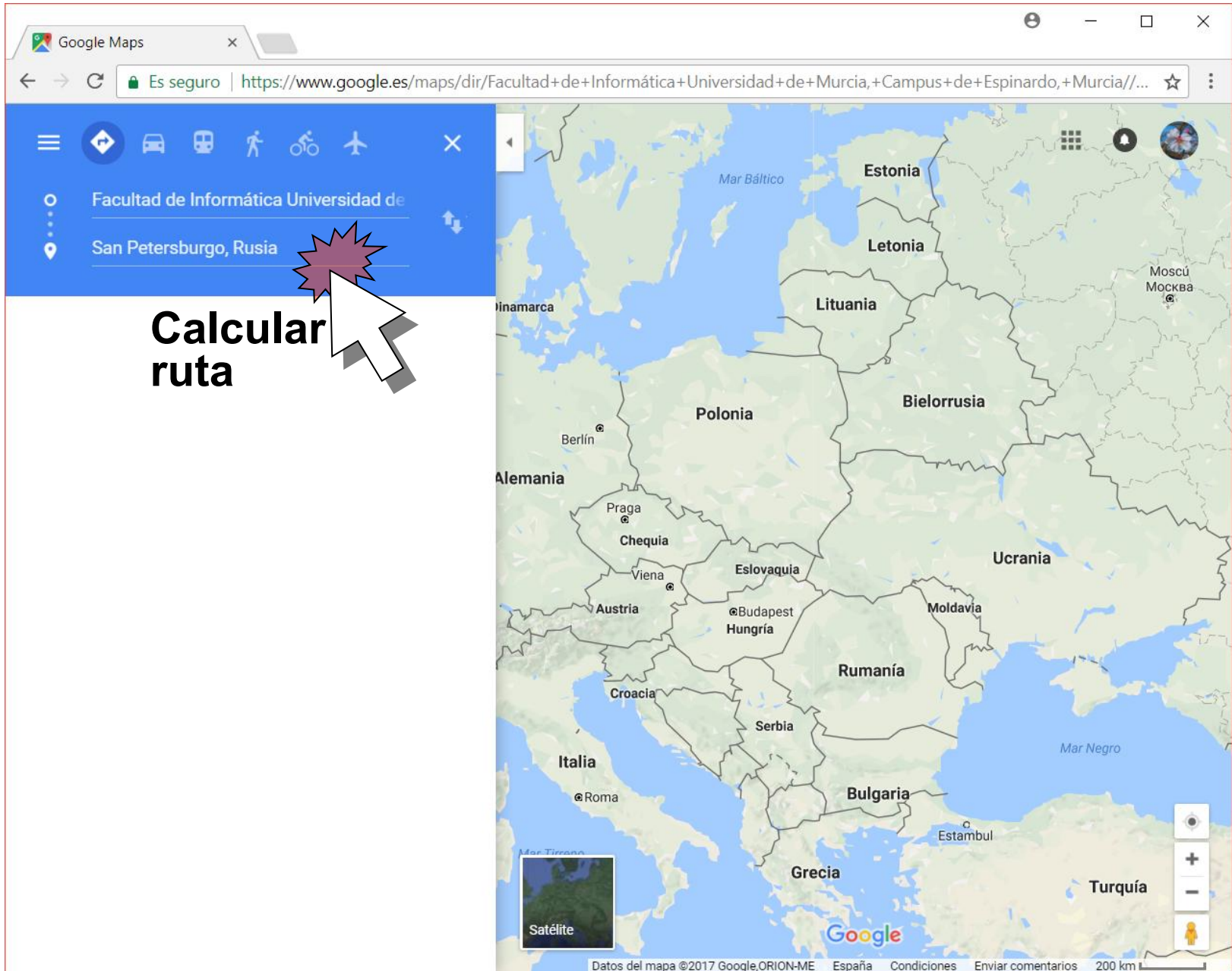
Opción 2: Palabras → Páginas

- Supongamos 1 solo ordenador a 3 GHz con 1 núcleo.
- Supongamos que cada página tiene 100 palabras y un total de 600 billones de palabras en un array ordenado. Hacemos una búsqueda binaria = $\log_2 n$.
- ¡¡¡La búsqueda se resuelve en 0,0004 ms!!! (49·8·2 ciclos de reloj)

Planificador de rutas



Planificador de rutas



Planificador de rutas

The screenshot shows a Google Maps interface with a walking route planned from the 'Facultad de Informática Universidad de Murcia' to 'San Petersburgo, Rusia'. The route is marked with a blue line and dots, passing through Spain, France, Germany, Poland, Lithuania, Latvia, Estonia, and finally reaching St. Petersburg. A callout box over the route indicates a total time of 778 hours and a distance of 3,789 km. The left sidebar shows the route details, including a warning that the destination is in a different time zone. A yellow box in the bottom right corner of the map area contains the text 'En solo 2,76 s'.

de Facultad de Informática x

Es seguro | <https://www.google.es/maps/dir/Facultad+de+Informática+Universidad+de+Murcia,+Campus+de+Espinardo,+Murcia/S...>

Facultad de Informática Universidad de Murcia

San Petersburgo, Rusia

Añadir destino

OPCIONES

Enviar indicaciones a tu teléfono

por DK22 778 h 3.789 km

⚠ Esta ruta pasa por varios países

⚠ Your destination is in a different time zone.

↑ 18.285 m · ↓ 18.397 m

832 m -1 m

DETALLES

En solo 2,76 s

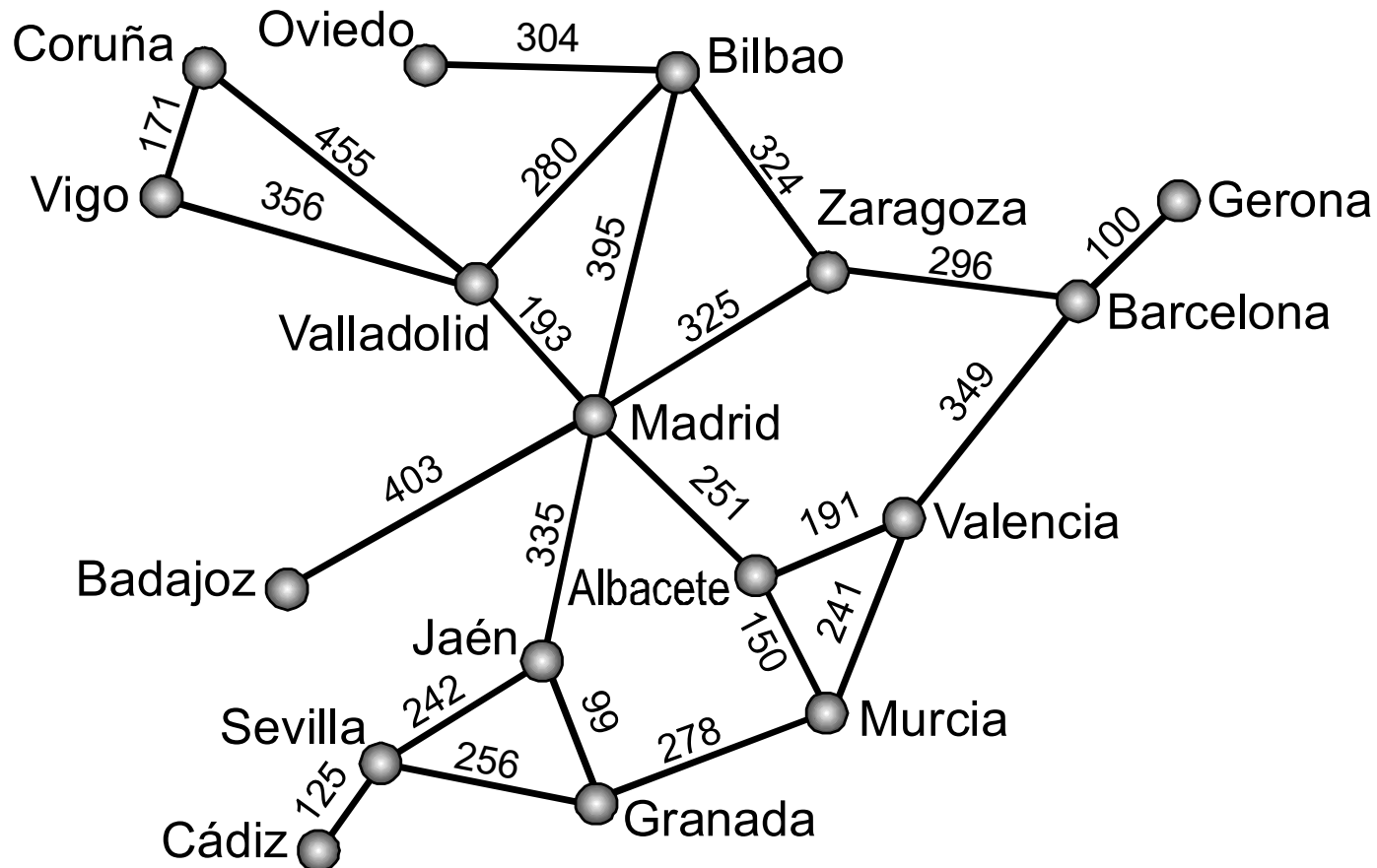
Datos del mapa ©2017 Google, ORION-ME España Condiciones Enviar comentarios 500 km

Planificador de rutas

- ¿Cómo representar la información (lugares y carreteras)?
- ¿Cómo calcular el camino más corto entre dos lugares?

Planificador de rutas

- Representación mediante un **grafo**:
 - Lugares = nodos.
 - Carreteras = arcos entre nodos.



Planificador de rutas

- ¿Cómo calcular los caminos mínimos en el mapa?
- Fuerza bruta: empezar por Murcia y probar todos los caminos hasta llegar.
- Supongamos que limitamos a 20 ciudades, existiendo 6 caminos por ciudad.
- ¡¡Existen 95 billones de caminos!!
- Algoritmo de Dijkstra: solo necesitaría 400 pasos.

Reto Hard de Grafos

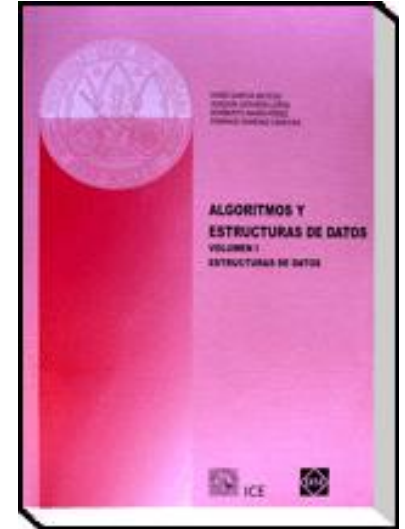
- **OMP'16 – D: Dijkstractions**
- ¿Qué pasa si el coste de un camino con aristas $(a, b, c, d\dots)$ fuera la exponenciación sucesiva: $a^{b^{c^{d\dots}}}$?
- Reto con comodines:
 - Un 10 en el tema 4.
 - Más 4 comodines.

Conclusiones y consejos



Ejercicios para casa

- Leer el capítulo 1, y las secciones 2.1 y 2.2 del texto guía.
- Preparar un resumen en un folio por las dos caras (una cap. 1 y otra para 2.1 y 2.2), **ESCRITO A MANO**.
- Entregar la semana que viene, escaneado en el AV.



Nombre del alumno, AED1, Grupo 2, Cap 1, Fecha (horas estimadas)