Instituto Politécnico Nacional
La Técnica al Servicio de la Patria

Unidad Profesional Interdisciplinaria de Ingeniería Campus Tlaxcala UPIIT

Algoritmos y Estructuras de Datos

Esaú Eliezer Escobar Juárez

Ingeniería en Inteligencia Artificial (IIA)

Bienvenidos

- Dr. Esaú Eliezer Escobar Juárez
  - parcs13@gmail.com
- Horario
  - Lun, Mar, Mier 8:30 – 10:00 hrs.

Temario

- Unidad I. Algoritmos fundamentales
  - 1.1 Algoritmia
  - 1.2 El problema del ordenamiento
  - 1.3 El problema de la búsqueda
  - 1.4 Exploración exhaustiva y vuelta atrás
- Unidad II. Estructura de datos lineales
  - 2.1 Pila
  - 2.2 Cola
  - 2.3 Listas
  - 2.4 Tablas hash
- Unidad III. Estructura de datos no lineales
  - 3.1 Árboles binarios
  - 3.2 Grafos

Bibliografía

- Aho A., Hopcroft J. & Ullman J. 1999. *Estructuras de datos y algoritmos,* Pearson.
- Cormen, T. 1990. *Introduction to algorithms,* The MIT Press.
- Roughgarden T. 2018. *Algorithms Illuminated Part 2: Graph Algorithms and Data Structures,* Soundlikeyourself Publishing.
- Vinu V. Das. 2014. *Principles of Data structures using* C and C++, New Age International.
- Weiss M. 2013. Data structures and algorithm Analysis in C++, Pearson.

Evaluación

- 3 evaluaciones

- Tareas, programas y ejercicios 50%

- Examen del periodo (Derecho con 80% de asistencia a sesiones virtuales) 50%

Varios

- Código de Classroom
  v5exrvg
- El lenguaje con el que se trabajará será: C, con el IDE de su preferencia.
- Simuladores:
  - Mritunjay Singh Sengar. (2019). Online GDB Compiler. (IDE Online para C/C++ y otros) https://www.onlinegdb.com/

Breve introducción

- ¿Qué esperan del curso?

Resolver problemas

¿Qué clase de problemas?

¿Cómo es el proceso para resolver un problema?

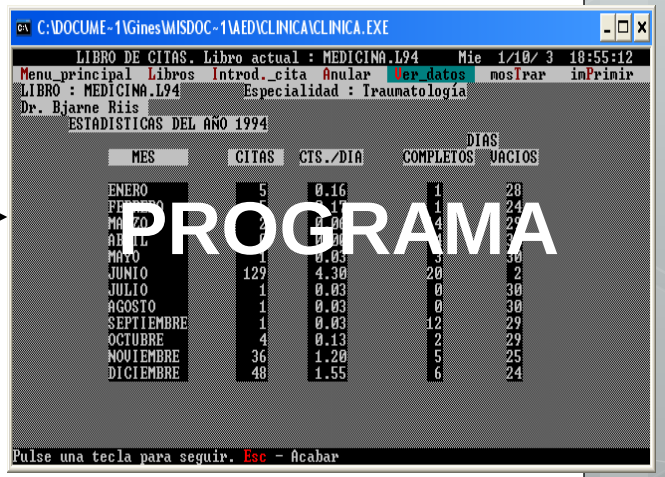
¿Cuándo se dice que la solución es eficiente y de calidad?

Problemas, programas, algoritmos y estructuras de datos

Algoritmos

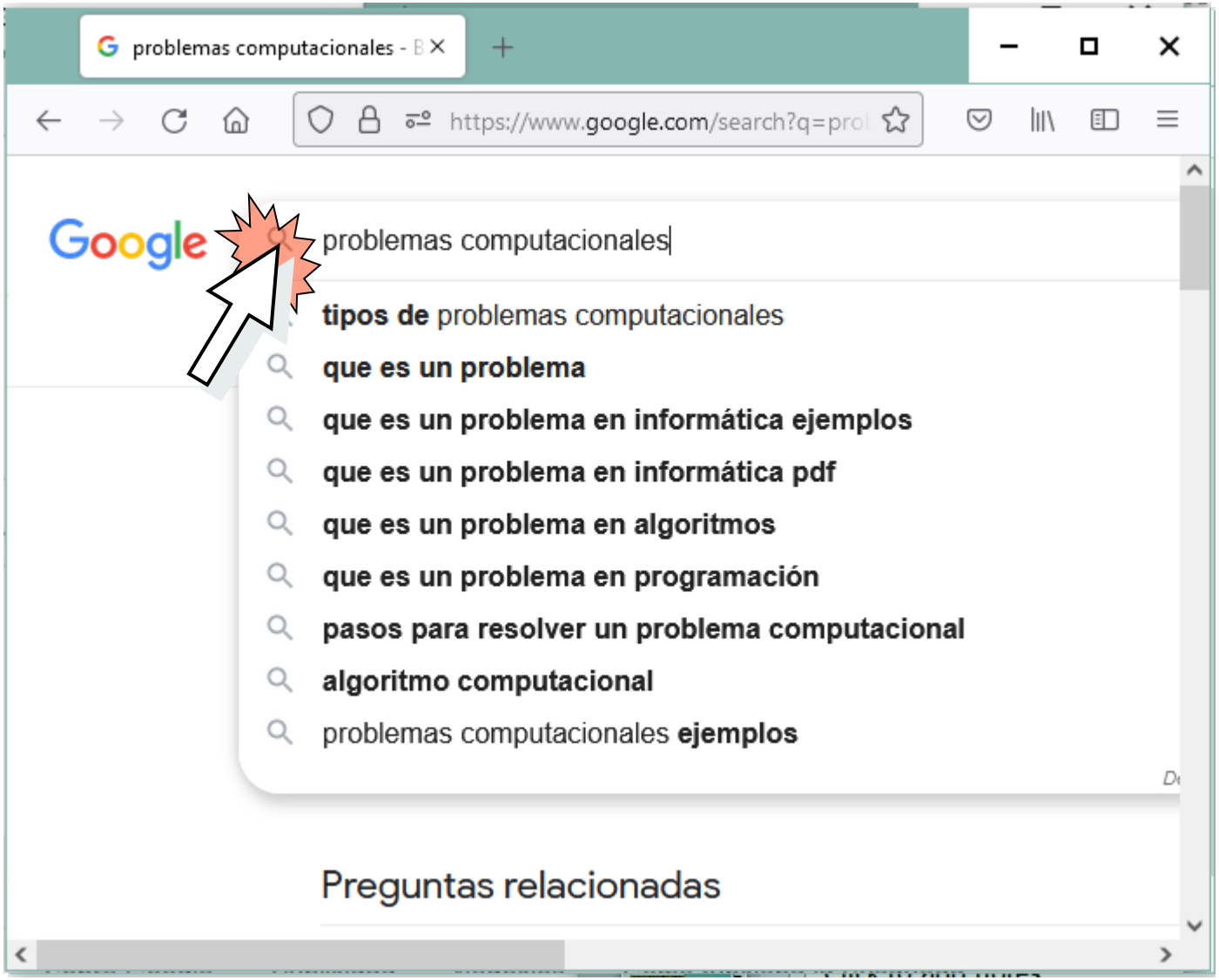
+

Estructuras de datos



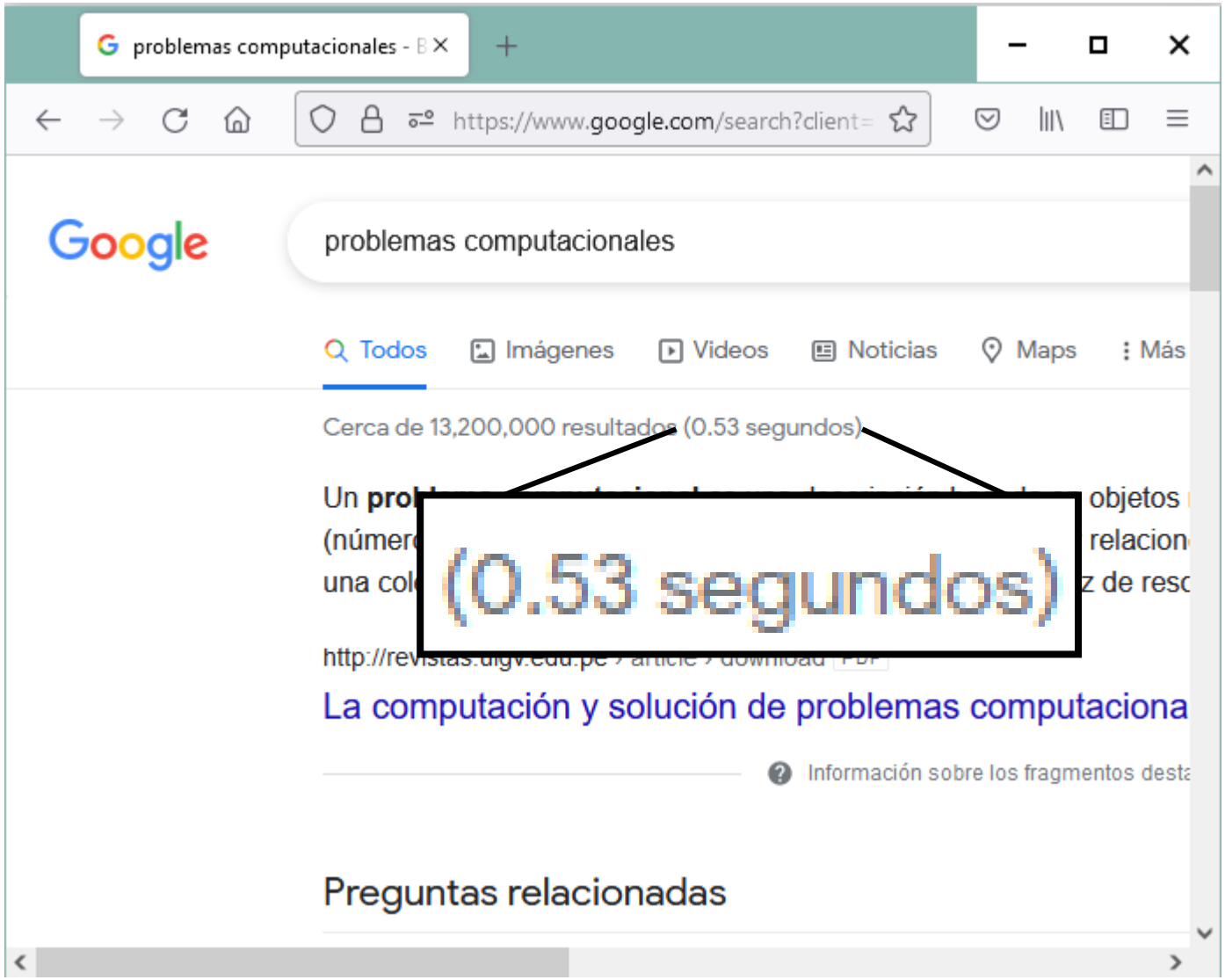
- **Problema:** Conjunto de hechos o circunstancias que dificultan la consecución de algún fin.
- **Algoritmo:** Conjunto de reglas finito e inambiguo.
- **Estructura de datos:** Disposición en memoria de la información.
- **Programa:** Algoritmos + Estructuras de datos.

Ejemplos de problemas

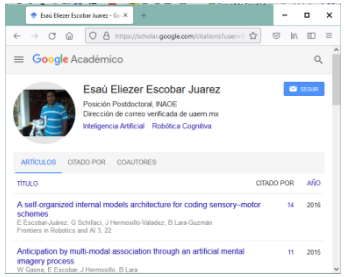


2021 Google Buscando en mas de 1 000 000 000 000 páginas web

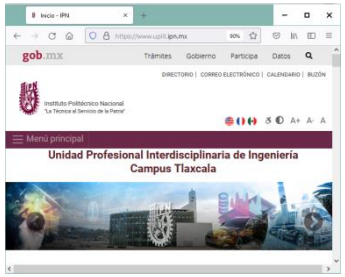
Buscador de Internet



Buscador de Internet



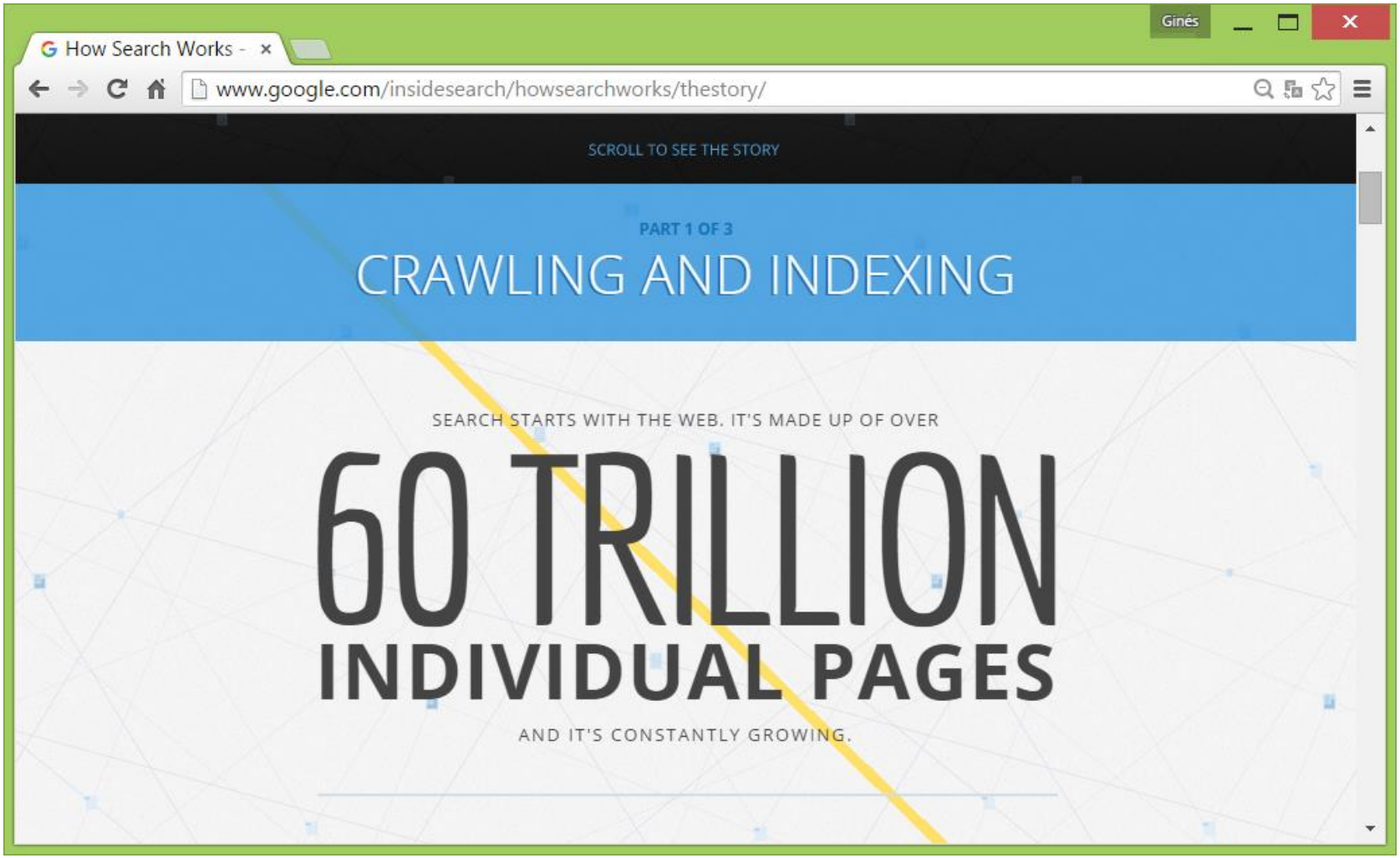
artificial, robótica, artículos, académico, phd, escobar, juárez, esaú, …



ipn, upiit, tlaxcala, ingeniería, ingreso, campus, EDUCATIVA, …

arduino, circuito, proyecto, programación, motor, electrónica, …

Buscador de Internet



60 trillion = 60 billones = 60$_{2}$000.000$_{1}$000.000

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?

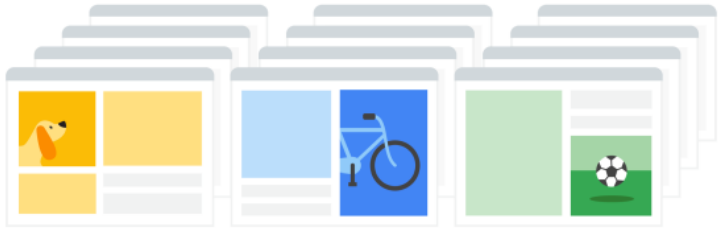
Buscador de Internet

- 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.
- ¡¡El recorrido de todas las páginas tardaría 20 segundos!!

Buscador de Internet

Cuando los rastreadores encuentran una página web tomamos nota de las señales clave, desde las palabras clave hasta la actualidad del sitio web.

El índice de búsqueda de Google contiene cientos de miles de millones de páginas web y tiene un tamaño de más de 100,000,000 gigabytes.



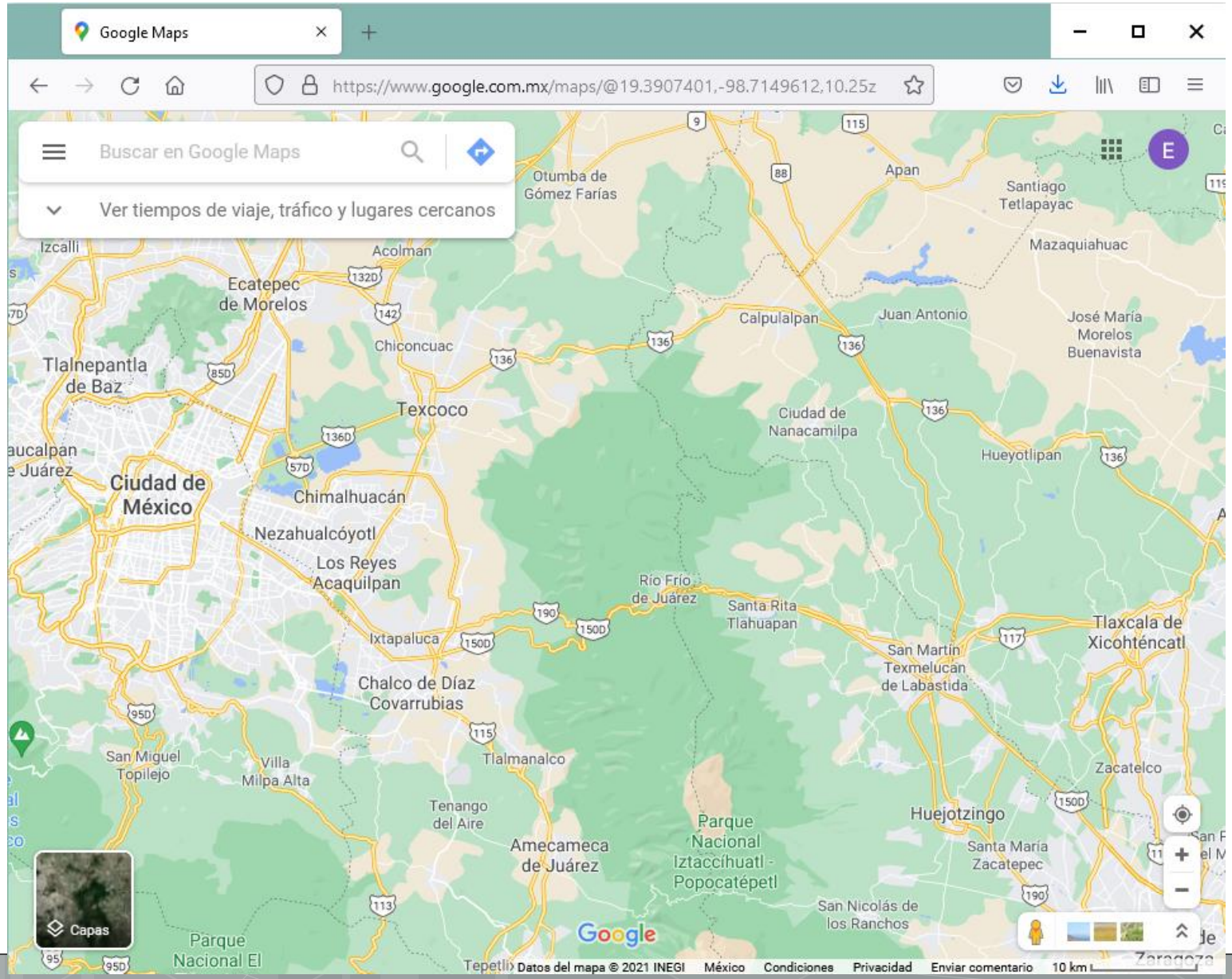
- **Solución:** Darle la vuelta al problema…

con un buen algoritmo

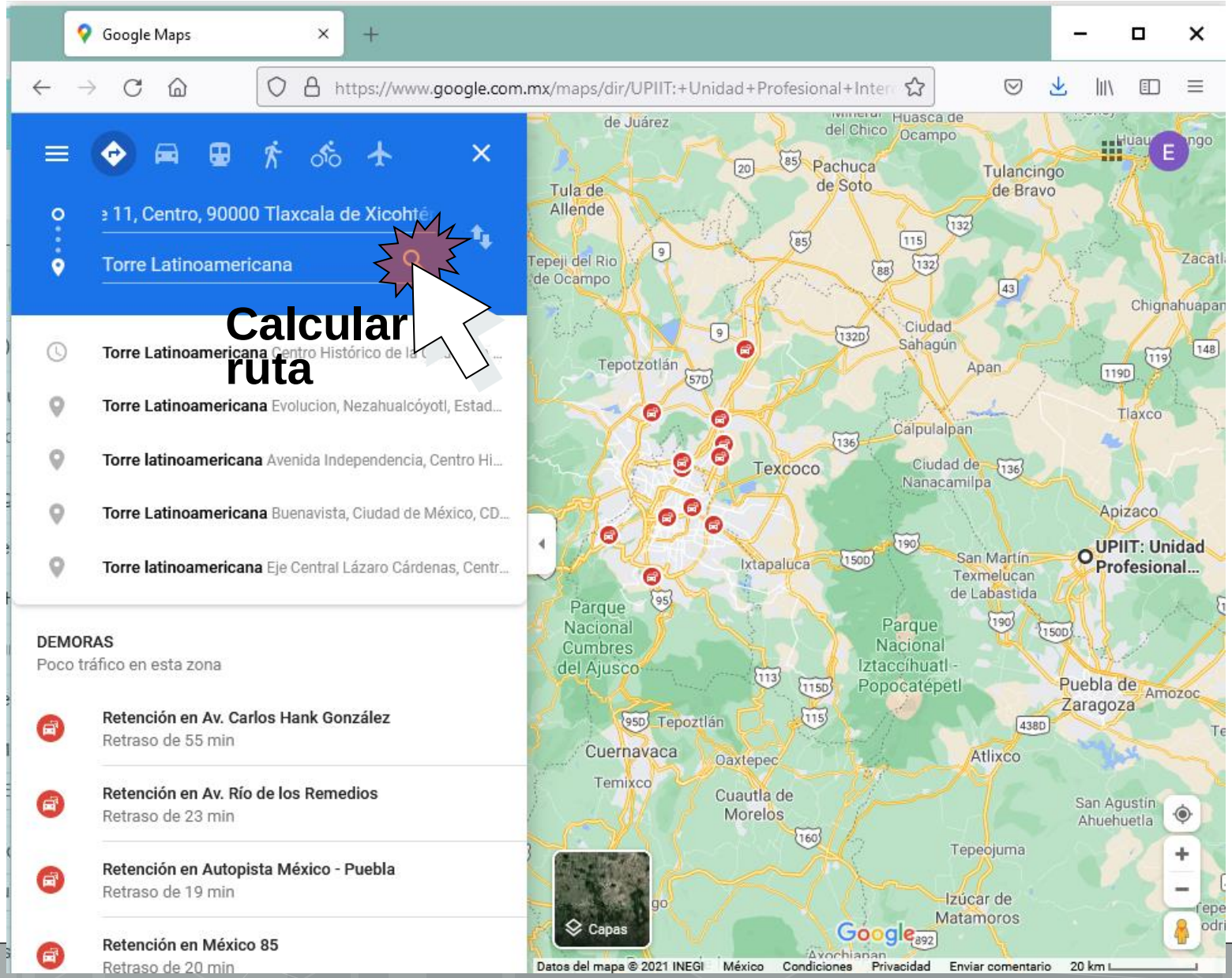
Entren a:

- www.menti.com

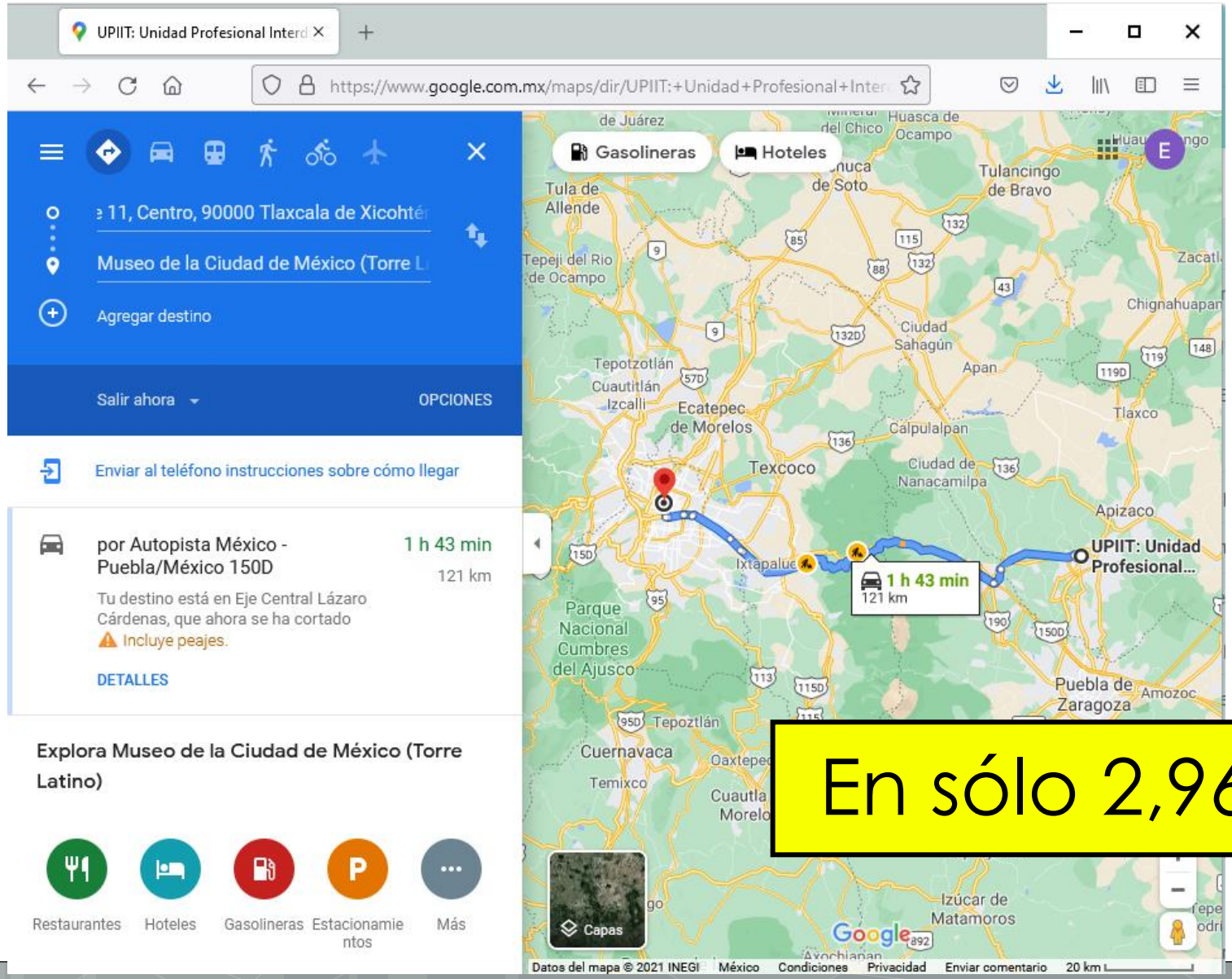
Planificador de rutas



Planificador de rutas



Planificador de rutas



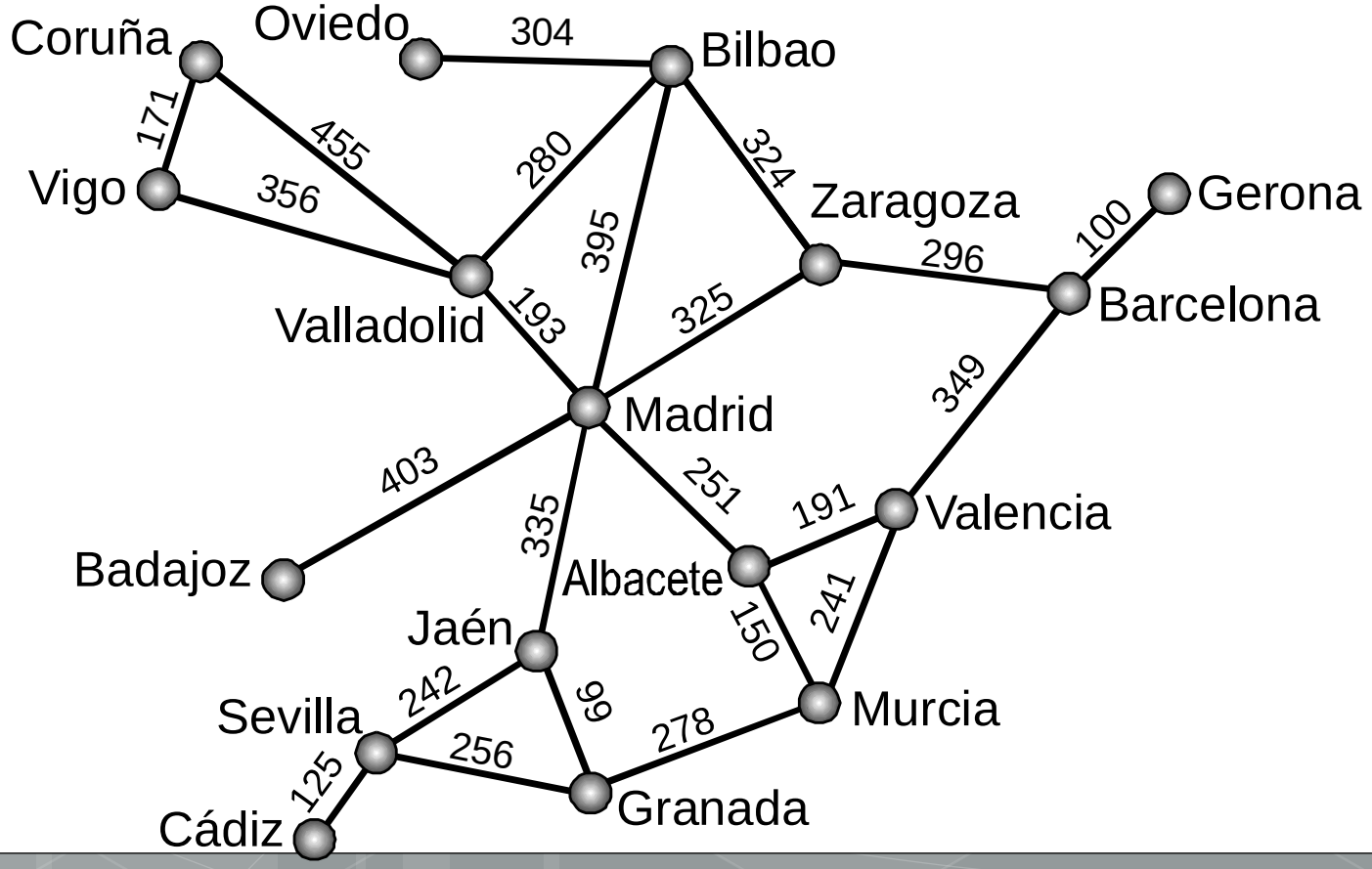
En sólo 2,96 s

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

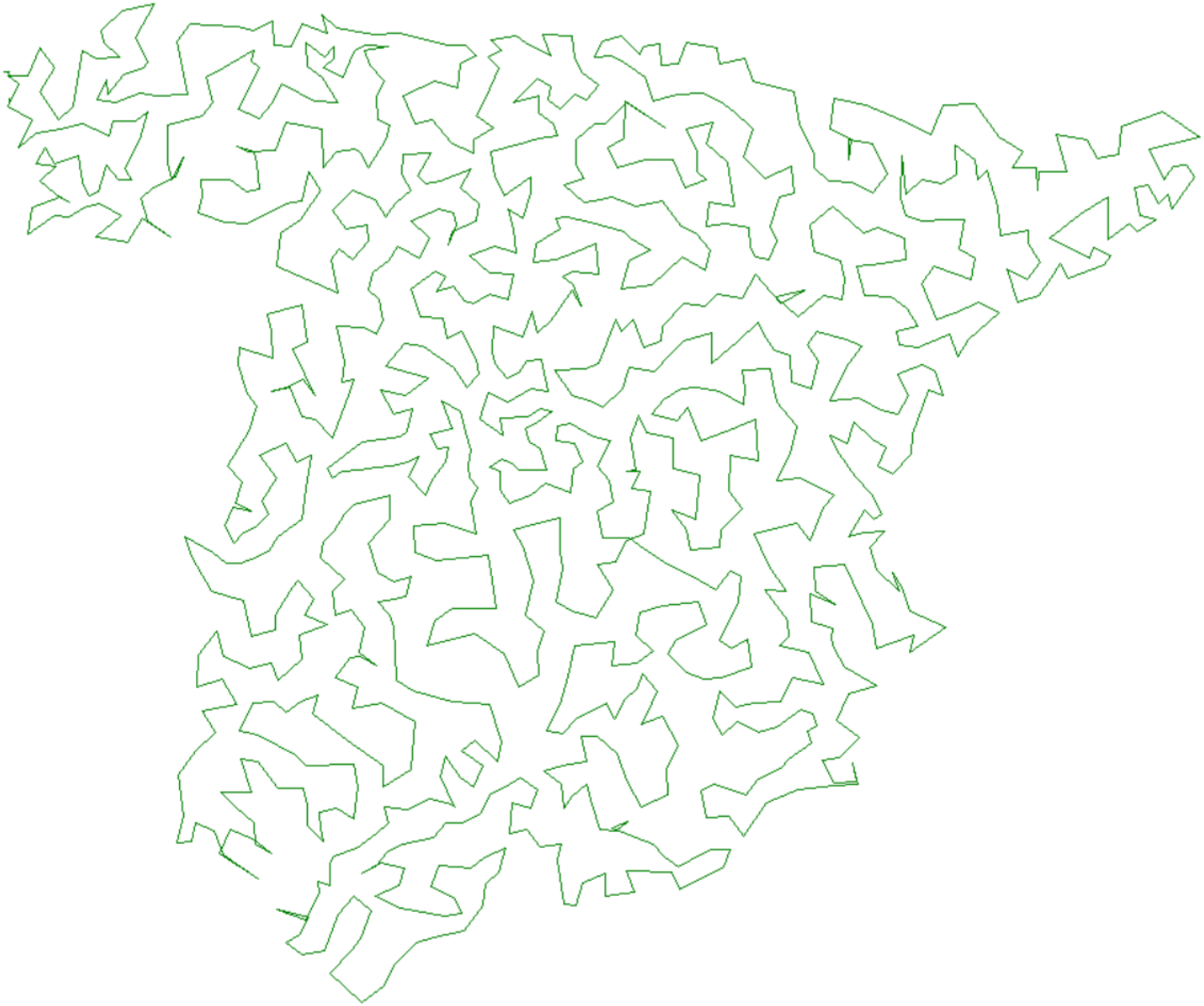
- ¿Cómo calcular los caminos mínimos en el mapa?
- Fuerza bruta: empezar por un lugar y probar todos los caminos hasta llegar.
- Supongamos que limitamos a 20 ciudades, existiendo 6 caminos por ciudad.
- ¡¡Existen 95 billones de caminos!!

Otro problema con grafos

- **Problema del viajante**: encontrar una ruta que pase por todas las ciudades con el mínimo coste.

El Reto del Viajante

Y esta es la solución…



Reto

- **EL JUEGO DE LAS CIFRAS.**

  Dado un conjunto de 6 enteros, encontrar la forma de conseguir otro entero, utilizando las operaciones de suma, resta, producto y división entera (y sin usar cada número más de una vez).

El Reto de las Cifras

- ¿Cuántas posibilidades son?

Primero: ¿Cuántas formas tenemos de tomar los números? **720 maneras diferentes**. 720 es el resultado de calcular el factorial de 6 (6!), que es el número de permutaciones que podemos obtener con seis elementos.

Segundo: ¿Cuántas formas posibles hay de operar con dichas combinaciones? usar cinco operadores de cuatro clases diferentes, admitiéndose cualquier operador tantas veces como sea necesario. Eso son **1.024 formas diferentes de operar**, pues 1.024 es el resultado de *4 elevado a 5*, donde 4 es el número de operadores diferentes y 5, cuántos operadores son necesarios, es decir, las variaciones con repetición de cuatro elementos tomados de cinco en cinco.

El Reto de las Cifras

- ¿Cuántas posibilidades son?

Tercero: Ahora hay que ver de cuántas maneras posibles se pueden agrupar los números ante los cálculos, es decir, *de cuántas formas diferentes se pueden colocar paréntesis en la fórmula*. Por ejemplo, si tenemos la combinación de números 135246 y la combinación de operadores "+x+-÷", una posible manera de combinarlos es [(1+(3x5)+2-4)÷6], y otra manera posible, que dará un resultado completamente diferente, sería [(1+3)x(5+2)-(4÷6)].
El número total de formas de colocar paréntesis en una fórmula de n operadores y n+1 operandos. Es ni más ni menos que el quinto número de Catalan, que se calcula de la forma (2n)!/[(n+1)!(n)!]. Para n=5, el resultado es 42.

En total 720*1024*42= **30.965.760 fórmulas distintas**

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
```

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**

Evolución e historia de la programación

Lenguajes de bajo nivel

(Basic, Fortran, Ensamblador, …)

Lenguajes estructurados

(Pascal, C, Modula, ADA, …)

Lenguajes estructurados

Arreglo.cpp

```
1   #include<stdio.h>
2   #include<stdlib.h>
3
4   int main(){
5       int m,n;
6       printf("\n Programa que encuentra el mayor y el menor en las\n filas y columnas y muestra la
7       printf("\tDame m: ");
8       scanf("%d",&m);
9       printf("\tDame n: ");
10      scanf("%d",&n);
11
12      int arreglo[m][n], mayorfil[m] = {0}, menorfil[m] = {
13
14      printf("\n");
15      for(int i=0;i<m;i++){
16          printf("\t");
17          for(int j=0;j<n;j++){
18              arreglo[i][j] = rand()%101+1;
19              if(menorfil[i]>arreglo[i][j]||menorfil[i]==0)
20                  menorfil[i] = arreglo[i][j];
21              if(mayorfil[i]<arreglo[i][j])
22                  mayorfil[i] = arreglo[i][j];
23              if(menorcol[j]>arreglo[i][j]||menorcol[j]==0)
24                  menorcol[j] = arreglo[i][j];
25              if(mayorcol[j]<arreglo[i][j])
26                  mayorcol[j] = arreglo[i][j];
27              printf("%2d ",arreglo[i][j]);
```

Dividido por bloques (bibliotecas)

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

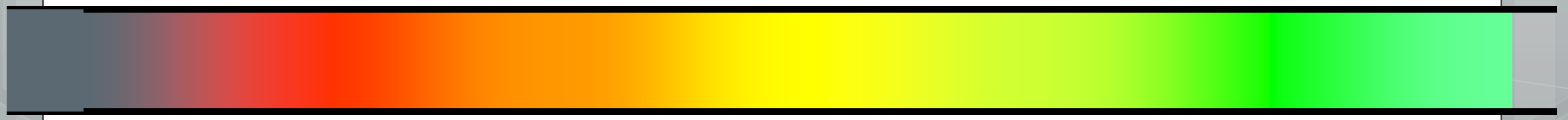
(Basic, Fortran, Ensamblador, …)

Lenguajes estructurados

(Pascal, C, Modula, ADA, …)

Lenguajes orientados a objetos

(Smalltalk, C++, Java, Eiffel, …)



Lenguajes orientados a objetos

```
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 (IplImage *image, int color= 0, int ancho= -1);
};
```

Los datos son privados

Las operaciones son públicas

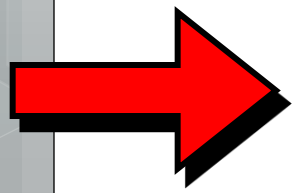
Lenguajes orientados a objetos

- 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)

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		INFORMÁTICO
1. Estudio de viabilidad, análisis del terreno, requisitos pedidos, etc.	→	1. **Análisis** del problema
2. Diseñar los planos del puente y asignar los materiales.	→	2. **Diseño** del programa (alg. y estr.)
3. Poner los ladrillos de acuerdo con los planos.	→	3. **Implementación** (programación)
4. Supervisión técnica del puente.	→	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

Conclusiones

1. **Proceso de análisis/diseño**. No empezar tecleando código como locos.

2. **Usar abstracciones**, respetando los dos principios básicos:
   - **Encapsulación**: las funciones relacionadas deben ir juntas (clases, módulos, paquetes, etc.).
   - **Ocultación de la implementación**: los aspectos de implementación no son visibles fuera del módulo, clase, etc.