



# **Algoritmos y Estructuras de Datos**

**Cursada 2019**

**Prof. Alejandra Schiavoni ([ales@info.unlp.edu.ar](mailto:ales@info.unlp.edu.ar))**

**Prof. Catalina Mostaccio ([catty@lifa.info.unlp.edu.ar](mailto:catty@lifa.info.unlp.edu.ar))**

**Prof. Laura Fava ([lfava@info.unlp.edu.ar](mailto:lfava@info.unlp.edu.ar))**

**Prof. Pablo Iuliano ([piuliano@info.unlp.edu.ar](mailto:piuliano@info.unlp.edu.ar))**

# Agenda

- ❖ Temas de la materia
- ❖ Objetivos de la materia
- ❖ Introducción al Análisis de Algoritmos
- ❖ Repaso de Recursión

# Temas del curso

- Árboles
- Análisis de Algoritmos
- Cola de Prioridades
- Grafos

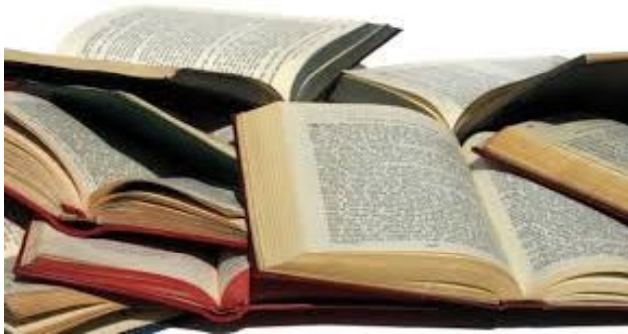
# Objetivos de la materia

- Analizar algoritmos y evaluar su eficiencia
- Estudiar estructuras de datos avanzadas: su implementación y aplicaciones

# Estructuras de Datos

*Una estructura de datos es una forma de almacenar y organizar los datos con el fin de facilitar el acceso y las modificaciones.*

**Ejemplo:**



*Datos sin organización  
Datos: libros*



*Datos organizados en una estructura  
Estructura de datos: biblioteca*

***Modelando la realidad con EEDD***

# ¿De qué se trata el curso?

Estudiar formas **inteligentes** de organizar la información, de forma tal de obtener algoritmos **eficientes**.

Listas, Pilas, Colas

Árboles Binarios

Árboles AVL

Árboles Generales

Heaps

Grafos

*Estructuras de Datos*

Insertar

Borrar

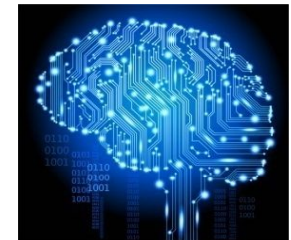
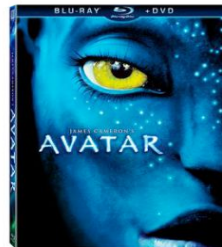
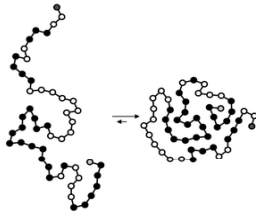
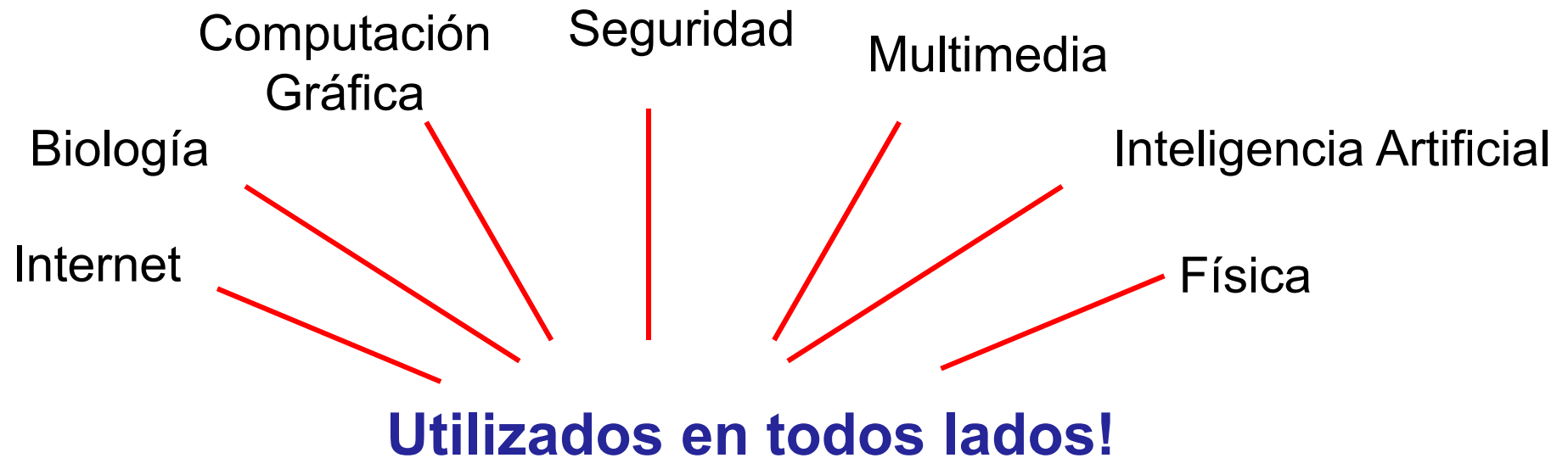
Buscar

Caminos mínimos

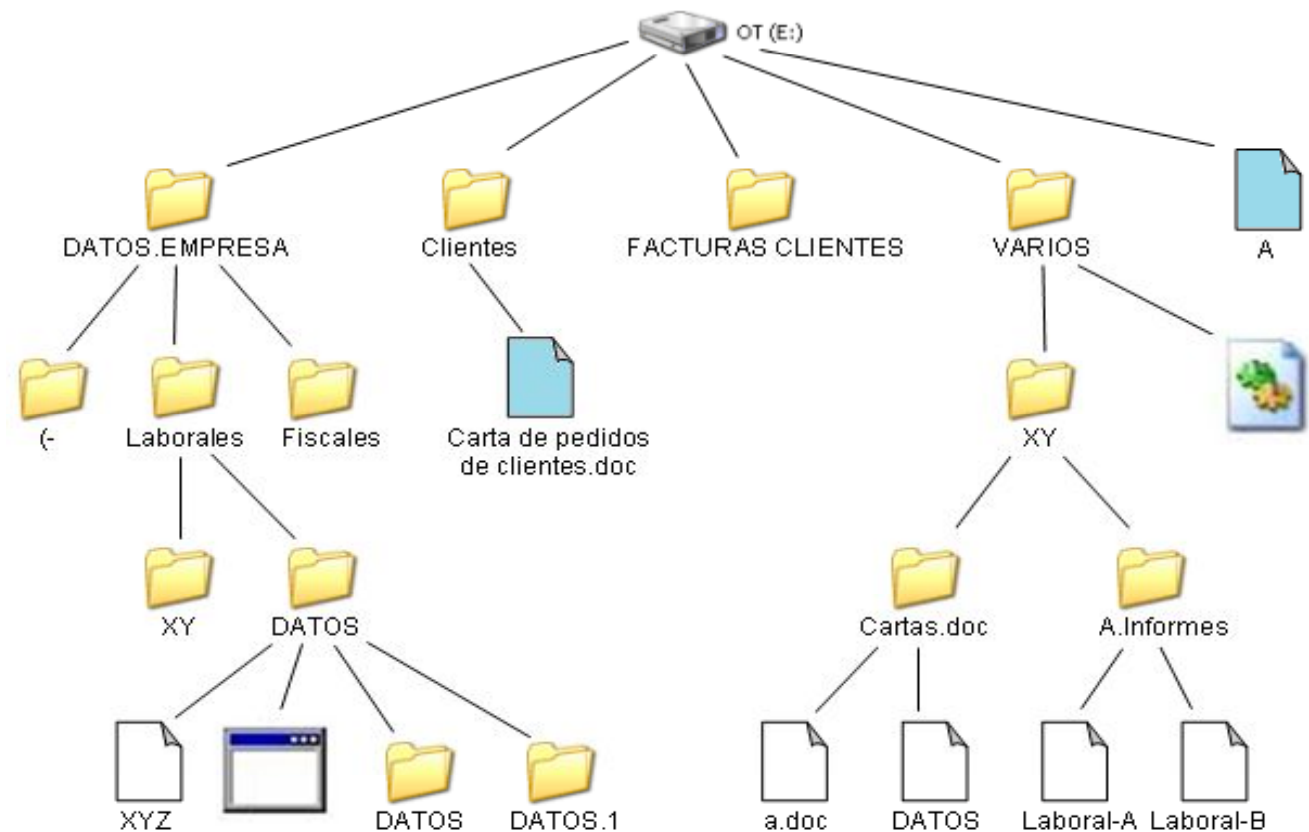
Ordenación

*Algoritmos*

# Las estructuras de datos y sus algoritmos son....



# Ejemplo 1: Árbol de carpetas y archivos

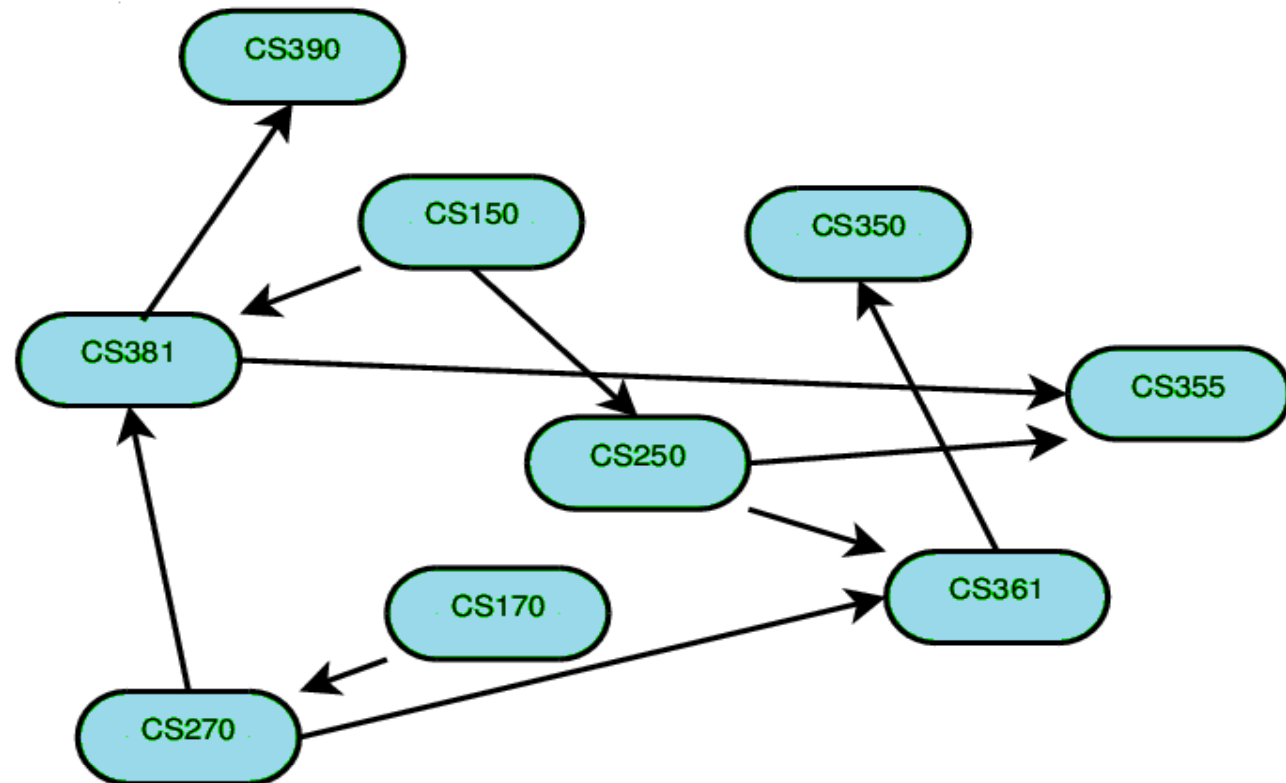


**Nodos:** Carpetas/Archivos

**Aristas:** representan la relación “contiene”



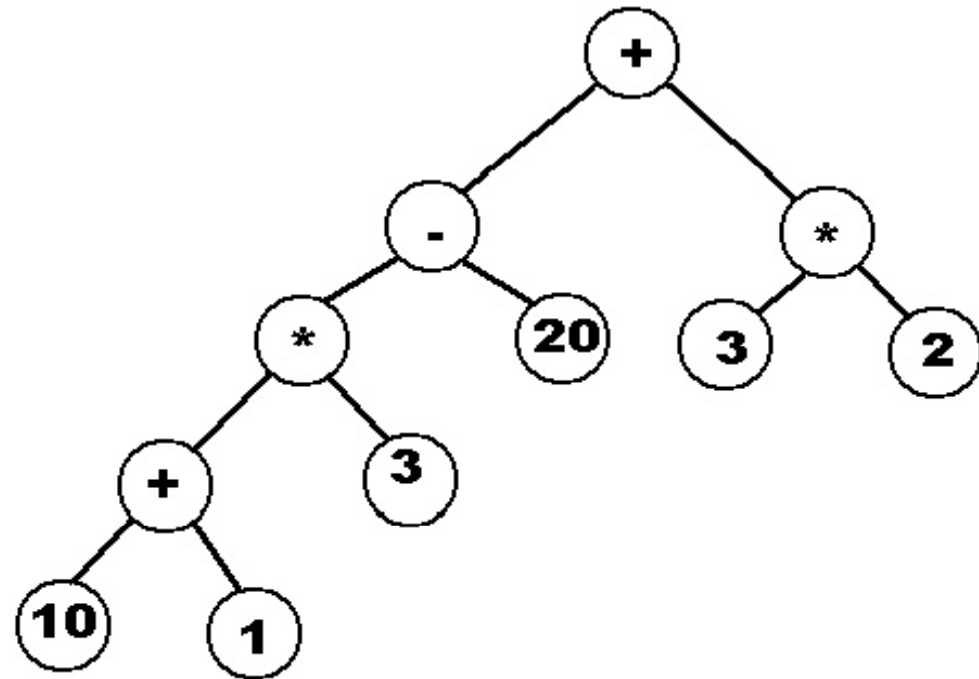
## Ejemplo 2: Prerrequisitos de un curso



**Nodos: Cursos**

**Aristas: relación de "prerrequisito"**

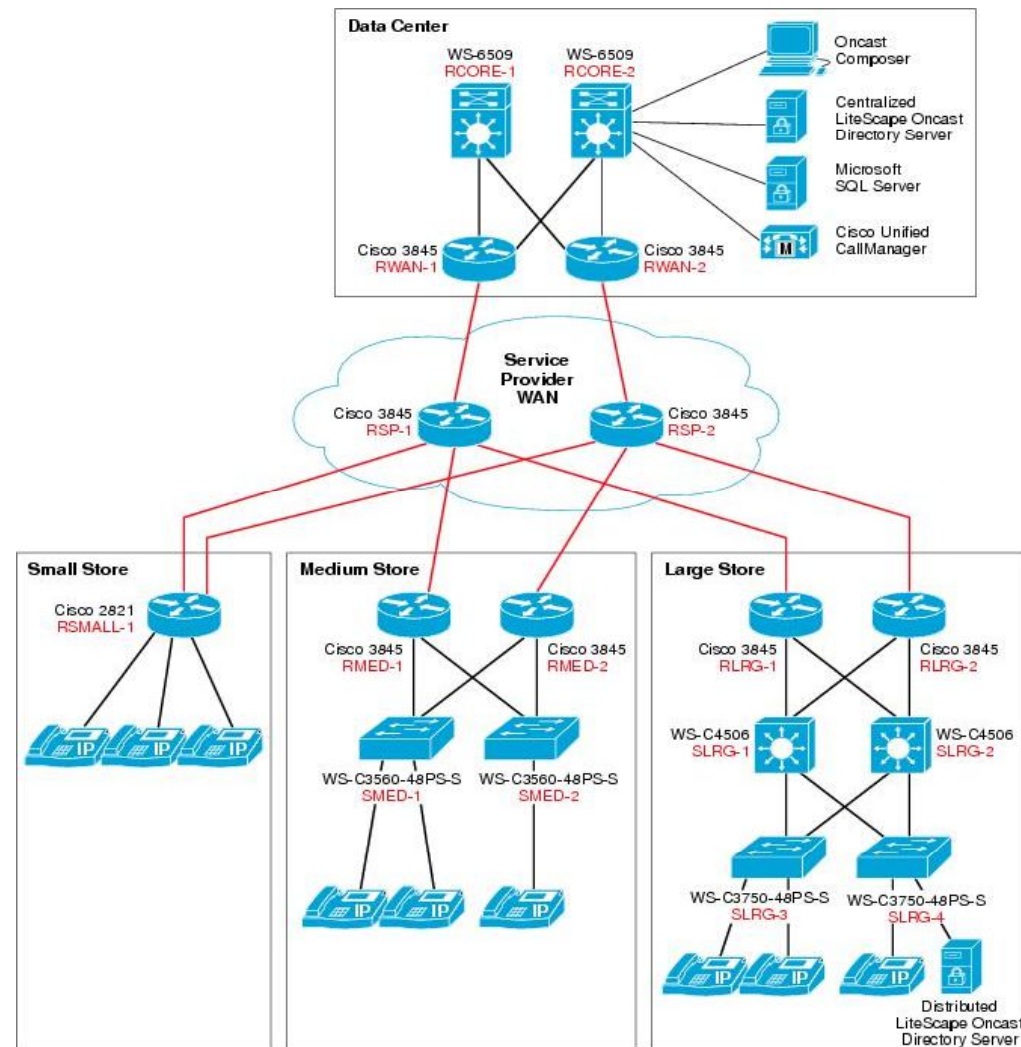
## Ejemplo 3: Representación de una expresión en un compilador



**Nodos:** Operandos/Operadores

**Aristas:** representan las relaciones entre las operaciones

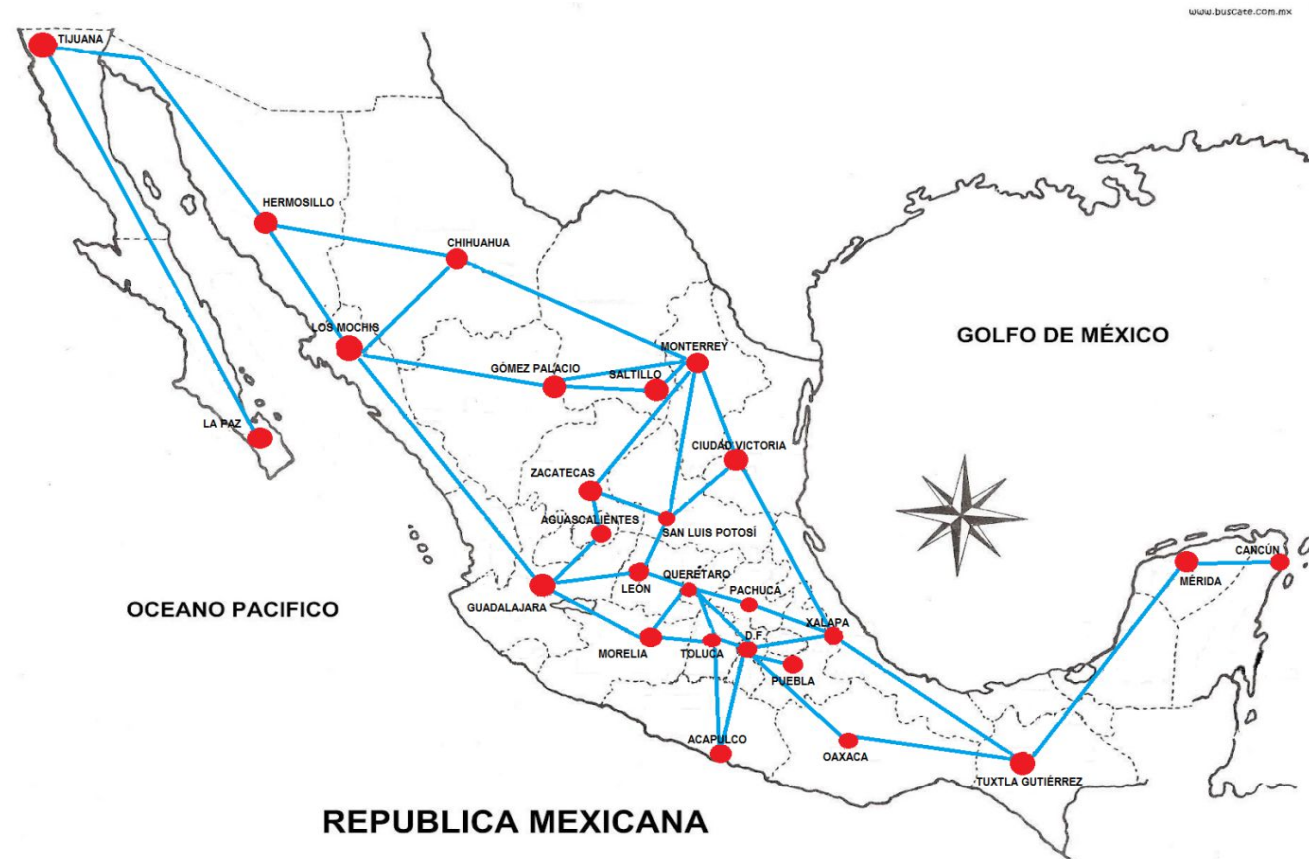
# Ejemplo 4: Esquema de una red informática



**Nodos: Equipos**

**Aristas: representan las conexiones**

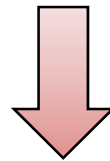
# Ejemplo 5: Mapa de ciudades



**Nodos: Ciudades**  
**Aristas: Rutas**

# Estructuras de Datos: Qué, Cómo y Por qué?

- Los programas reciben, procesan y devuelven datos
- Necesidad de organizar los datos de acuerdo al problema que vamos a resolver



**Las estructuras de datos son formas de organización de los datos**

# Estructuras de Datos: Qué, Cómo y Por qué?

- Un programa depende fundamentalmente de la organización de los datos
- cómo se organizan los datos está relacionado con:
  - ➡ Implementación de algunas operaciones: pueden resultar más fácil o más difícil
  - ➡ La velocidad del programa: puede aumentar o disminuir
  - ➡ La memoria usada: puede aumentar o disminuir

# Objetivos del curso respecto de las Estructuras de Datos

- Aprender a implementar las estructuras de datos usando abstracción
- Estudiar diferentes representaciones e implementaciones para las estructuras de datos
- Aprender a elegir la “mejor” estructura de datos para cada problema

# Algoritmos y su Análisis

- ¿Qué es un algoritmo?
  - Es una secuencia de pasos que resuelven un problema
  - Es independiente del lenguaje de programación
- Existen varios algoritmos que resuelven correctamente un problema
- La elección de un algoritmo particular tiene un enorme impacto en el tiempo y la memoria que utiliza

***La elección de un algoritmo y de la estructura de datos para resolver un problema son interdependientes***



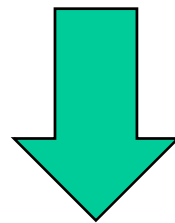
# Objetivos del curso respecto del Análisis de los Algoritmos

- Entender los fundamentos matemáticos necesarios para analizar algoritmos
- Aprender a comparar la eficiencia de diferentes algoritmos en términos del tiempo de ejecución
- Estudiar algunos algoritmos estándares para el manejo de las estructuras de datos y aprender a usarlos para resolver nuevos problemas

# Problemas y algoritmos

## ➤ Problemas:

- Buscar un elemento en un arreglo
- Ordenar una lista de elementos
- Encontrar el camino mínimo entre dos puntos



Encontrar **el algoritmo** que lo resuelve



# Caso: Buscar un elemento en un arreglo

El arreglo puede estar:

- desordenado
- ordenado

Si el arreglo está desordenado  **Búsqueda secuencial**

64	13	93	97	33	6	43	14	51	84	25	53	95
0	1	2	3	4	5	6	7	8	9	10	11	12

# Algoritmo: Búsqueda secuencial

```
public static int seqSearch(int[] a, int
    key)
{
    int index = -1;
    for (int i = 0; i < N; i++)
        if (key == a[i])
            index = i;
    return index;
}
```

**¿Cuántas comparaciones hace?**

# Caso: Buscar un elemento en un arreglo

El arreglo puede estar:

- desordenado
- ordenado

Si el arreglo está ordenado 

**Búsqueda binaria:** Comparo la clave con la entrada del centro

- Si es menor, voy hacia la izquierda
- Si es mayor, voy hacia la derecha
- Si es igual, la encontré

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
↓							↓							↓
<u>lo</u>							<u>mid</u>							<u>hi</u>

# Algoritmo: Búsqueda binaria

```
public static int binarySearch(int[] a, int key)
{
    int lo = 0, hi = a.length-1;
    while (lo <= hi)
    {
        int mid = lo + (hi - lo) / 2;
        if (key < a[mid]) hi = mid - 1;
        else if (key > a[mid]) lo = mid + 1;
        else return mid;
    }
    return -1;
}
```

**¿Cuántas comparaciones hace?**

# ¿Cuántas operaciones hace cada algoritmo?

**Búsqueda  
secuencial**

N	Cantidad de operaciones
1000	1000
2000	2000
4000	4000
8000	8000
16000	16000



**Hace N operaciones**

**Búsqueda  
binaria**

N	Cantidad de operaciones
1000	~10
2000	~11
4000	~12
8000	~13
16000	~14



**Hace  $\log(N)$  operaciones**

# ¿Cómo medir el tiempo?



## ✓ Manual

Tomando el tiempo que tarda

## ✓ Automática

Usando alguna instrucción del lenguaje para medir tiempo



# Análisis empírico

Correr el programa para varios tamaños de la entrada y medir el tiempo. Suponemos que cada comparación tarda 1 seg.

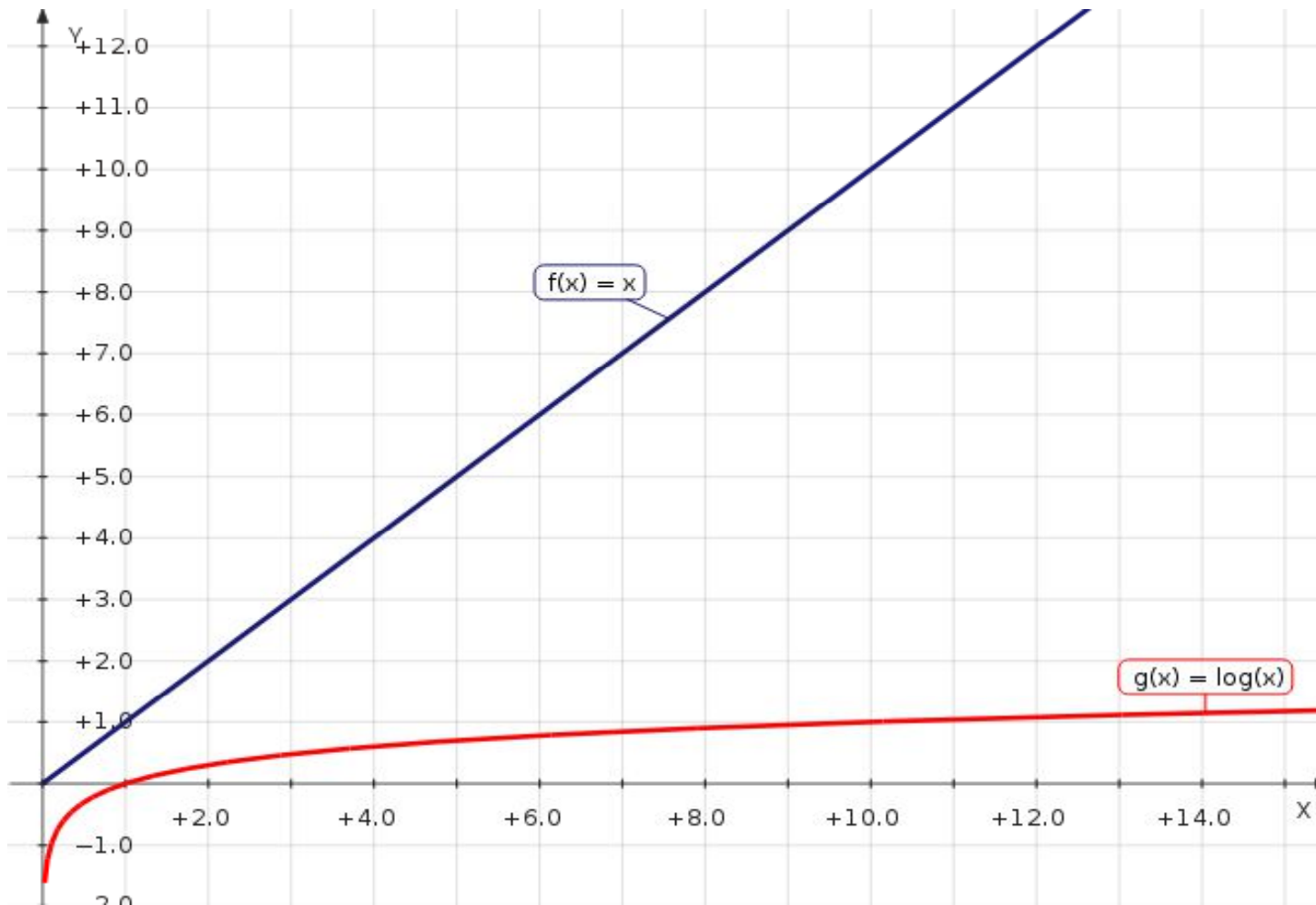
**Búsqueda  
secuencial**

N	Tiempo (seg)
1000	1000
2000	2000
4000	4000 ~ 1 hs.
8000	8000 ~ 2 hs
16000	16000 ~ 4 hs.

**Búsqueda  
binaria**

N	Tiempo (seg)
1000	~10
2000	~11
4000	~12
8000	~13
16000	~14

# Análisis de Algoritmos



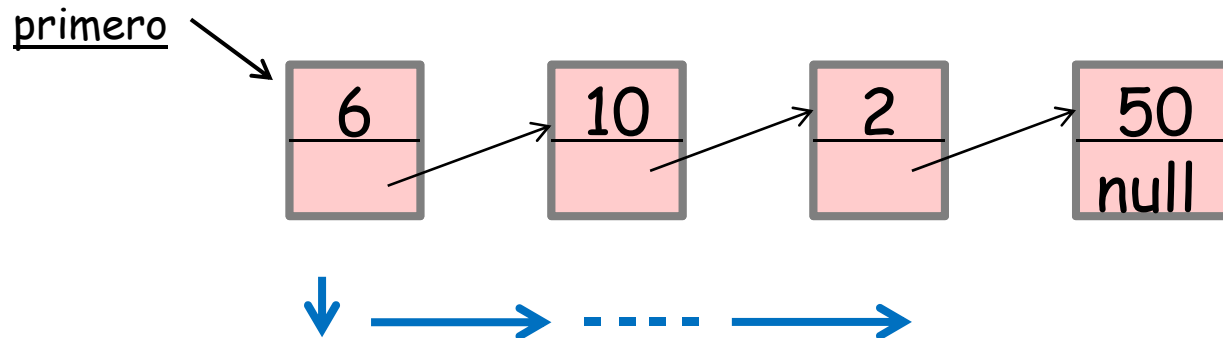
# Caso: Buscar un elemento en una lista dinámica

Si los elementos están almacenados en una lista dinámica

La lista puede estar:

- desordenada
- ordenada

¿Cómo sería el algoritmo de búsqueda?



¿Cuántas comparaciones hace?



Hace N comparaciones

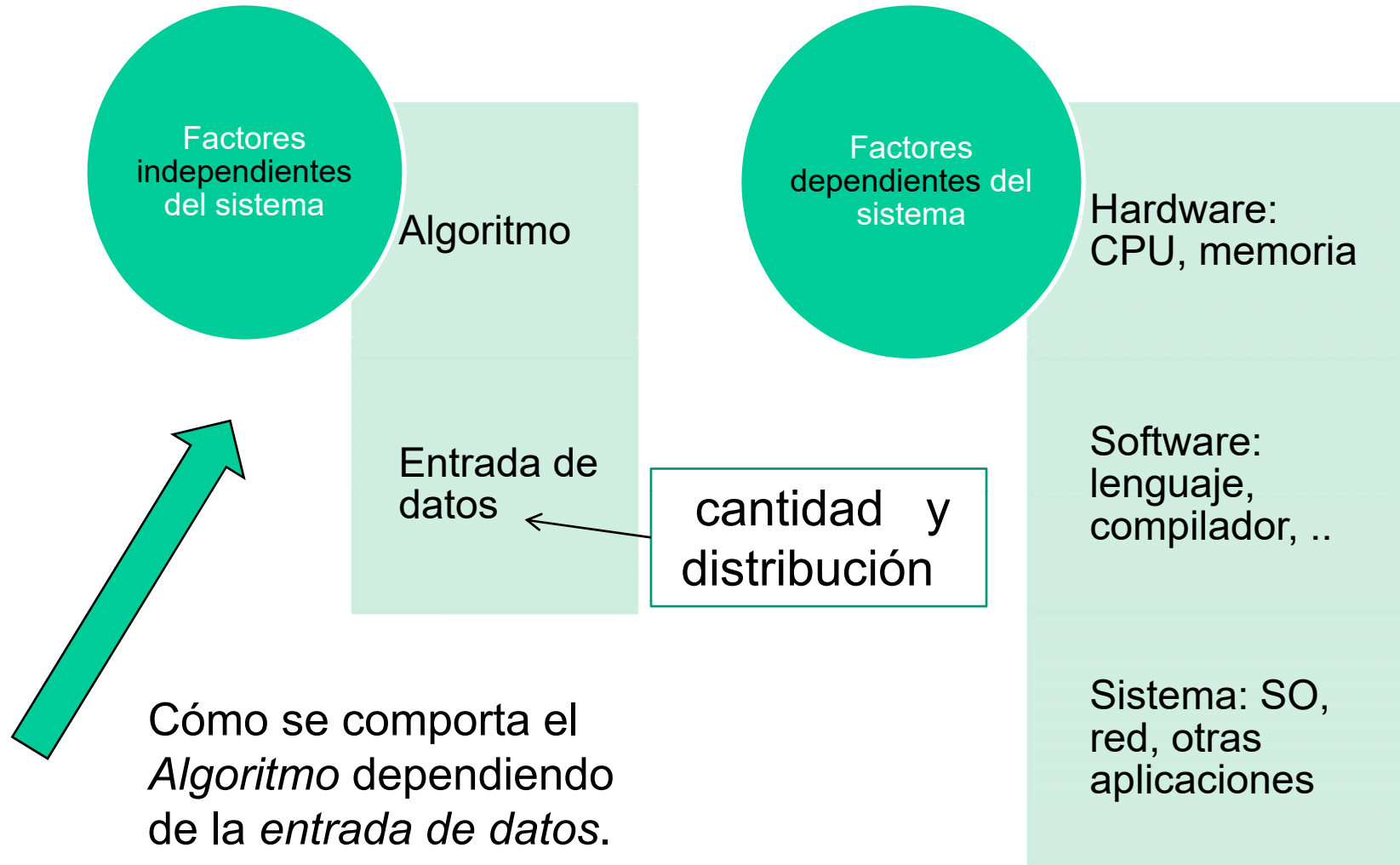
# Análisis de Algoritmos

*Marco para predecir la performance y  
comparar algoritmos*

## Desafío:

Escribir programas que puedan resolver en forma eficiente problemas con una gran entrada de datos

# Análisis de Algoritmos



# Análisis de Algoritmos

Existe un modelo matemático para medir el tiempo

**Tiempo total:**

*Suma del **costo** x **frecuencia** de todas las operaciones*

- Es necesario analizar el algoritmo para determinar el conjunto de operaciones
  - **Costo** depende de la máquina, del compilador, del lenguaje
  - **Frecuencia** depende del algoritmo y de la entrada