

Los problemas computacionales y los algoritmos

¿Qué es un algoritmo?

- Informalmente, un algoritmo es cualquier procedimiento computacional bien definido que toma algún valor, o un conjunto de valores como entrada y produce algún valor, o conjunto de valores como salida.
- Un algoritmo es por lo tanto una secuencia de pasos computacionales que transforma la entrada en la salida.
- También podemos ver un algoritmo como una herramienta para resolver un problema computacional bien especificado.

Problema computacional

- Un problema computacional es un problema que una computadora podría resolver o una pregunta que una computadora podría responder.
- Un problema computacional puede verse como una colección infinita de instancias junto con un conjunto de soluciones, posiblemente vacío, para cada instancia.
- **Instancia:** Una sucesión finita de números enteros (a_1, a_2, \dots, a_n)
- **Solución:** Una permutación $(a'_1, a'_2, \dots, a'_n)$ de la sucesión de entrada tal que $a'_1 \leq a'_2 \leq \dots \leq a'_n$

- El campo de la teoría de la complejidad computacional intenta determinar la cantidad de recursos (complejidad computacional) que requerirá resolver un problema dado y explicar por qué algunos problemas son intratables o indecidibles.
- Los problemas computacionales pertenecen a clases de complejidad que definen ampliamente los recursos:
 - Tiempo
 - espacio / memoria
 - Energía
 - profundidad del circuito

que se necesitan para calcularlos (resolverlos) con varias máquinas abstractas.

Tipos de problemas computacionales

- **Problema de decisión.**
 - Si
 - No
- **Problemas de ordenamiento.**
- **Problemas de búsqueda.** Las respuestas pueden ser cadenas arbitrarias.
 - Se representa como una relación que consta de todos los pares instancia-solución.
 - $P(i, s)$ determina si s es una solución de i .
- **Problema de conteo** pide el número de soluciones a un problema de búsqueda dado.
- **Problema de optimización**
 - No solo se busca una solución, sino que se busca "*la mejor*" de todas.

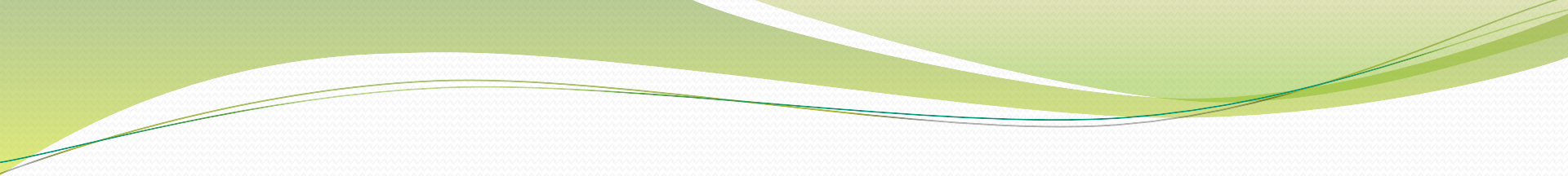
Abstracción y tipo de dato abstracto

Tipo de dato abstracto

- Un **tipo de dato abstracto (TDA)** o **tipo abstracto de datos (TAD)** es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo.
- Con mucha frecuencia se utilizan los términos *TDA* y *Abstracción de Datos* de manera equivalente, y esto es debido a la similitud e interdependencia de ambos. Sin embargo, es importante definir por separado los dos conceptos.

Abstracción

- Los Lenguajes de Programación Orientados a Objetos son lenguajes formados por diferentes métodos o funciones y que son llamados en el orden en que el programa lo requiere, o el usuario lo desea.
- La abstracción de datos consiste en ocultar las características de un objeto y obviarlas, de manera que solamente utilizamos el nombre del objeto en nuestro programa.

- 
- La abstracción es un concepto muy útil en la programación, ya que un usuario no necesita mencionar todas las características y funciones de un objeto cada vez que este se utiliza, sino que son declaradas por separado en el programa y simplemente se utiliza el término abstracto

Estructuras de datos en C

```
struct coleccion_CD  
{  
    char titulo[30];  
    char artista[25];  
    int num_canciones;  
    float precio;  
    char fecha_compra[8];  
};
```

```
struct coleccion_CD cd1;
```

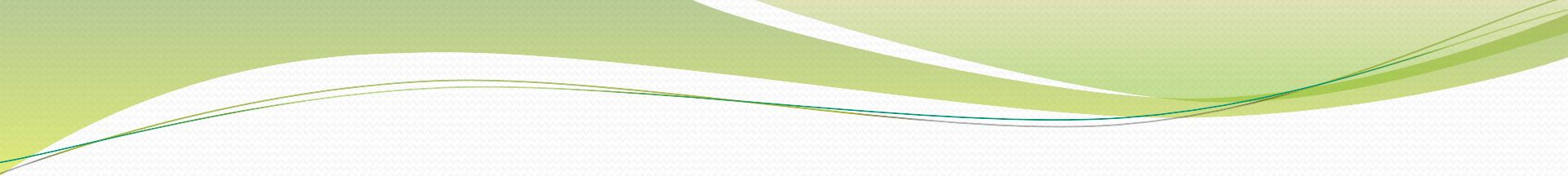
```
strcpy(cd1.titulo,"Granada") ;  
cd1.precio = 3450.75;  
cd1.num_canciones = 7;
```

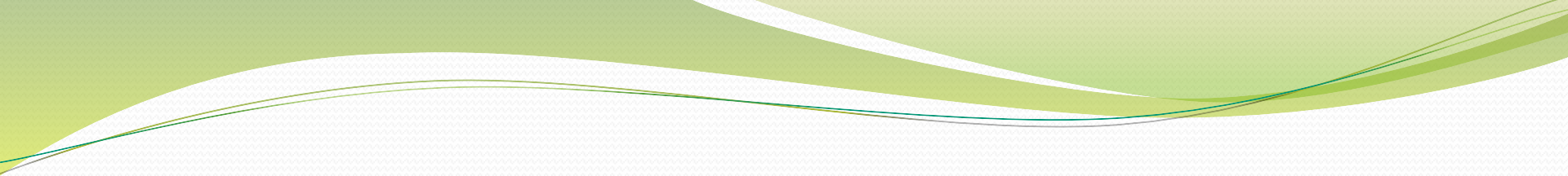
Introducción a la complejidad algorítmica

Orden de complejidad $O()$ de un algoritmo

Complejidad algorítmica

- Si dos algoritmos diferentes resuelven el mismo problema entonces los llamamos algoritmos equivalentes.
- La complejidad algorítmica permite establecer una comparación entre algoritmos equivalentes para determinar en forma teórica, cuál tendrá mejor rendimiento en condiciones extremas y adversas.

- 
- Para esto se trata de calcular cuántas instrucciones ejecutará el algoritmo en función del tamaño de los datos de entrada.
 - Llamamos “instrucción” a cada línea de código que se ejecuta en el algoritmo.

- 
- Como resultaría imposible medir el tiempo que demora una computadora en ejecutar una instrucción, simplemente diremos que cada una demora 1 unidad de tiempo en ejecutarse.
 - Luego, el algoritmo más eficiente será aquel que requiera menor cantidad de unidades de tiempo para concretar su tarea.

Orden de complejidad $O()$ de un algoritmo

- La notación **Big O** es una herramienta muy funcional para determinar la complejidad de un algoritmo que estemos utilizando, permitiéndonos medir su rendimiento en cuanto a uso de espacio en disco, recursos (memoria y ciclos del reloj del CPU) y tiempo de ejecución, entre otras, ayudándonos a **identificar el peor escenario donde el algoritmo llegue a su más alto punto de exigencia.**

Orden de complejidad $O()$ de un algoritmo

- Los términos de complejidad Big O más utilizados son:
 - $O(1)$ constante
 - $O(n)$ lineal
 - $O(\log n)$ logarítmica
 - $O(n^2)$ cuadrática
 - $O(n^3)$ cúbica
 - $O(2^n)$ exponencial