

Introducción

Estructura de datos y algoritmos 2 - Joaquin Vigna

Modalidad


- Teórico:
 - responsable: Joaquin Vigna (joaquin.vigna@gmail.com)
 - días: lunes y miércoles
- Práctico:
 - responsables: Joaquin Vigna + Mauricio Fabre (fabremauriciomar@gmail.com)
 - días: jueves

Bibliografía recomendada*:

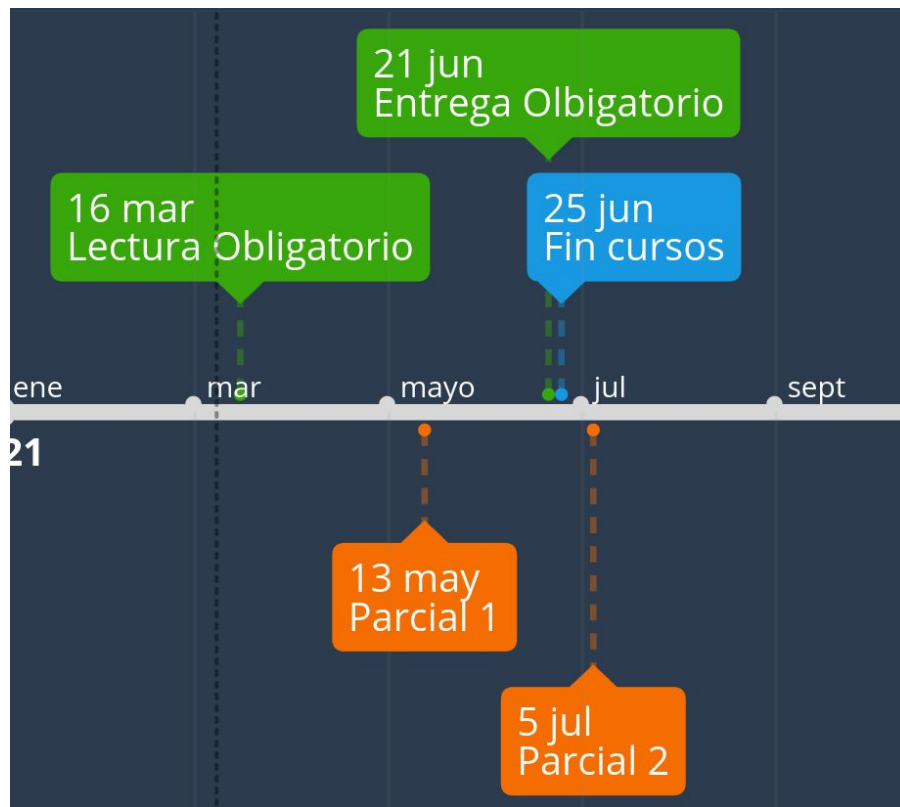
Estructuras de Datos y Análisis de Algoritmos - Mark Allen Weiss; Benjamin/Cummings Inc., 1993.

Data Structures and Algorithm Analysis in C++ - Mark Allen Weiss; Benjamin/Cummings Inc., 2006.

* consultar en busca de más bibliografía



Línea de tiempo



Para conocernos un poco mejor ...

Formulario anónimo:

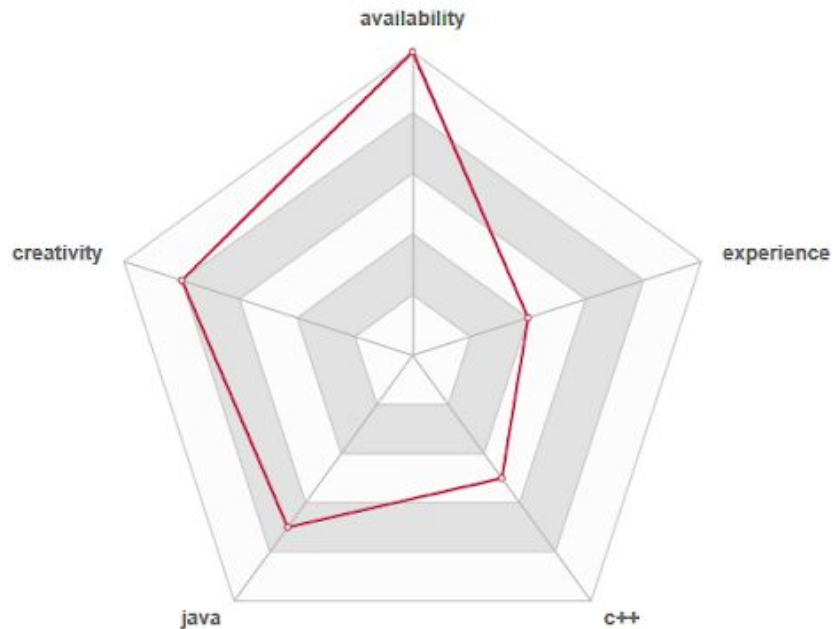
https://forms.office.com/Pages/ResponsePage.aspx?id=zSCX18DYDE2kBC3NAI8B4zHaHUx7m7VBrn630x8c1m5UNzBLMVdONUxPMTIRMVdQQVIwWTJXM1BGS_C4u



Ahora un poco de mí



Joaquin Vigna





Empecemos con el
repaso

Orden de complejidad

Orden de complejidad - “The big O”

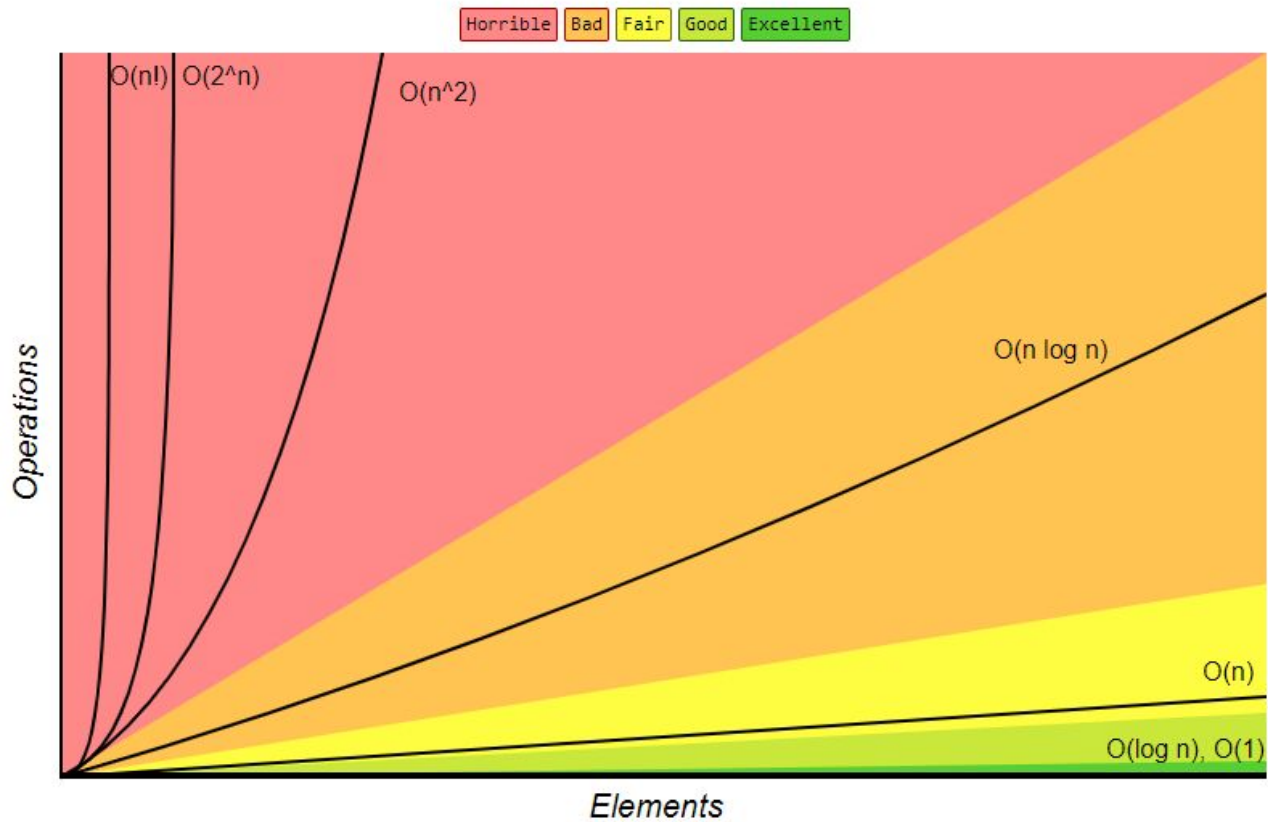
*“El orden no pretende medir cuán rápido es un algoritmo respecto a otro. El orden mide otra cosa. Mide **cuán rápidamente aumenta el tiempo de ejecución de un algoritmo cuando aumenten los datos de entrada**”*

Algunos ejemplos:

- $O(N)$
- $O(N^2)$
- $O(1)$
- $O(\log N)$

Links de interés: [Big-O Algorithm Complexity Cheat Sheet \(Know Thy Complexities!\) @ericdrowell](https://geeks.ms/etomas/2012/11/28/el-orden-de-los-algoritmos-esa-gran-o/)
<https://geeks.ms/etomas/2012/11/28/el-orden-de-los-algoritmos-esa-gran-o/>

Big-O Complexity Chart



Algunas reglas para el cálculo

- Regla de la suma

- Si $T1(n)$ es $O(f1(n))$ y $T2(n)$ es $O(f2(n))$ entonces **$T1(n)+T2(n)$ es $O(\max(f1(n), f2(n)))$**

- Ejemplo

- La función *int a(int x)* tiene $O(N)$
 - La función *string b(int x)* tiene $O(N^2)$
 - Entonces

```
void c(int x) {  
    a(x);  
    b(x);  
}
```

tiene $O(N^2)$



Algunas reglas para el cálculo

- Regla del producto

- Si $T_1(n)$ es $O(f_1(n))$ y $T_2(n)$ es $O(f_2(n))$ entonces **$T_1(n).T_2(n)$ es $O(f_1(n).f_2(n))$**

- Ejemplo

- La función `int a(int x)` tiene $O(\log N)$

- Entonces

```
void c(int x){  
    for(int i = 0; i<x; i++) {  
        a(i);  
    }  
}
```

tiene $O(N \cdot \log N)$



Algo de práctica no viene mal

```
1  int suma(int a, int b) {  
2      return a + b;  
3  }  
4  
5  int multiplicacion(int a, int b) {  
6      return a * b;  
7  }  
8  
9  int suma2(int a, unsigned int b) {  
10     for (int i = 0; i < b; i++)  
11     {  
12         a += 1;  
13     }  
14     return a;  
15 }  
16  
17 int suma3(int a, unsigned int b) {  
18     if(b == 0) {  
19         return a;  
20     }  
21     return 1 + suma3(a, b-1);  
22 }
```

```
1  import java.io.*;  
2  class test {  
3  
4      public static void main(String args[]) throws IOException  
5      {  
6          int ar[] = { 1, 2, 3, 4, 5, 6, 7, 8 };  
7          int i, x;  
8  
9          for (i = 0; i < ar.length; i++) {  
10             x = ar[i];  
11             System.out.print(x + " ");  
12         }  
13     }  
14 }
```

```
1  import java.io.*;
2  class test {
3
4      public boolean buscar(int ar[], int el) {
5          for (int i = 0; i < ar.length; i++) {
6              if(ar[i] == el) {
7                  return true;
8              }
9          }
10         return false
11     }
12
13     public boolean buscar2(int ar[]) {
14         boolean ret = false;
15         for (int i = 0; i < ar.length; i++) {
16             ret = ret || buscar(ar, i);
17         }
18         return ret
19     }
20 }
```



TADs

Tipos Abstractos de Datos

Un conjunto de datos u objetos **al cual se le asocian operaciones**. El TAD provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en cómo están implementadas dichas operaciones.

TAD = valores + operaciones

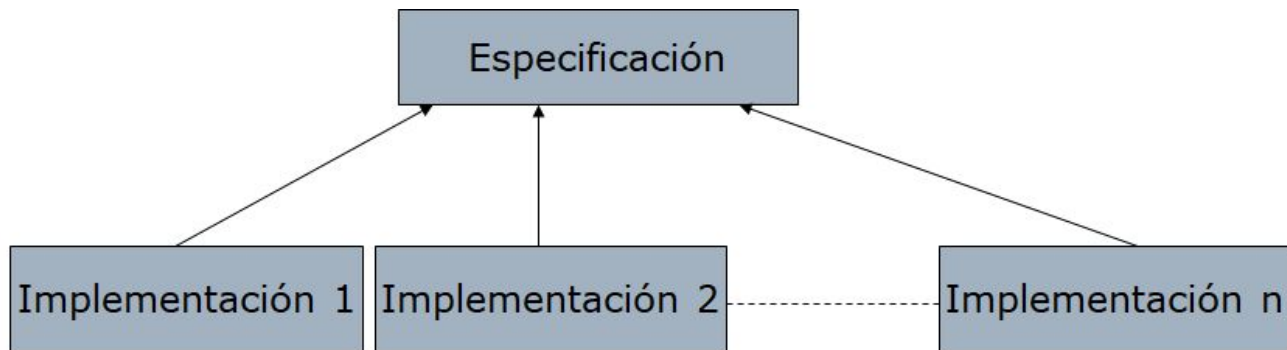
<https://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/>

<https://www.geeksforgeeks.org/abstract-data-types/>



Tipos Abstractos de Datos

Esto quiere decir que un mismo TAD puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.



Algunos ejemplos de TADs

- Lista
- Set/conjunto
- Diccionario
- Tabla
- Cola (FIFO)
- Pila (cola LIFO)
- **Cola de prioridad**
- **Grafos**



TAD SET

- Def: Un conjunto es una colección de elementos (o miembros), los que a su vez pueden ellos mismos ser conjuntos o si no elementos primitivos, llamados también átomos. Notación: {2, 5, 6}
- Algunos métodos/operaciones:
 - void insertar(T)
 - void borrar(T)
 - bool esVacio()
 - bool existe(T)
 - void union(Set<T>)
 - void interseccion(Set<T>)
 - void diferencia(Set<T>)
- Algunas implementaciones posibles:
 - Listas (simple o doblemente encadenadas, ordenas o no)
 - ABBs o **AVLs**
 - **Tablas de hash**
- Los órdenes varían de la implementación que se realice, especialmente de que tipo de estructura que se elija.

TAD Diccionario

- A los diccionarios los podemos ver como un tipo especial de SETs, donde se ignoran las operaciones típicas entre conjuntos (unión, intersección, diferencia, etc)
- Algunos métodos/operaciones:
 - void insertar(T)
 - void borrar(T)
 - bool esVacio()
 - bool existe(T)

TAD Tabla

- Def: Una tabla es una **función parcial** de elementos de un tipo, llamado el tipo **dominio**, a elementos de otro (posiblemente el mismo) tipo, llamado el tipo **rango**. Que una tabla T asocia el elemento r del recorrido al elemento d del dominio lo denotaremos $T(d) = r$

- Algunos métodos/operaciones:
 - void insertar(D,R)
 - void eliminar(D)
 - bool existe(D)
 - bool esVacia()
 - R recuperar(D)

x	y
2	4.5
1	9
0	3
-1	1.5
-2	0

- Algunas implementaciones posibles:
 - Listas de asociación (simple o doblemente encadenadas, ordenas o no)
 - ABBs o **AVLs**, también de asociación
 - Tablas de hash**

Equipo	PTS
Atlético PSV	37
Real Paraíso	30
Dep. Ramírez	28
Manchester City	22
Bayer 05	21
Chapecoense	20
Liverpool	14

Recomendaciones

Tener a mano implementaciones de A1

- Listas: simple o doblemente encadenadas. De ser posible con punteros/referencias hacia el inicio y el final para mejorar órdenes.
- ABB
- Cola (FIFO)
- Pila
- Iterador (*recomendable): de no haber visto podemos ver algo en el práctico.

Quien no tenga dichas implementaciones se hace hincapié **en comenzar cuanto antes**, son herramientas que contarán para el resto del curso.

En caso de cumplir, aprovechar a mejorar el código (órdenes y prolijidad de código en general).



Algunos links de interés

1. Órdenes de complejidad
 - a. <https://www.geeksforgeeks.org/understanding-time-complexity-simple-examples/>
2. TAD:
 - a. <https://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/>
 - b. <https://www.geeksforgeeks.org/abstract-data-types/>
3. Listas simplemente encadenadas:
 - a. <https://www.geeksforgeeks.org/linked-list-set-1-introduction/>
 - b. <https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/>
 - c. <https://www.geeksforgeeks.org/linked-list-set-3-deleting-node/>
4. Listas doblemente encadenadas:
 - a. <https://www.geeksforgeeks.org/doubly-linked-list/>
5. Pila:
 - a. <https://www.geeksforgeeks.org/design-and-implement-special-stack-data-structure/>
6. Cola:
 - a. <https://www.geeksforgeeks.org/queue-set-1-introduction-and-array-implementation/>
7. ABB:
 - a. <https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/>
 - b. <https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/>