# ANÁLISIS Y DISEÑO DE ALGORITMOS

---

## Divide y Vencerás

### Práctica 5 de laboratorio

**Fecha de realización y entrega: del 5 al 10 de marzo de 2015**
**(en la sesión de laboratorio que corresponde a cada alumno)**

---

Dado el código fuente que podéis encontrar en los materiales, completadlo codificando la función de perfil `mergesort( list<int>& )`.

Esta función toma como entrada una lista enlazada (`list<int>`) y devuelve la misma lista ordenada.

Medir el tiempo que tarda en ejecutarse con listas de tamaños crecientes (no hacer falta hacer repeticiones) y compararlo con lo que tarda el método que proporciona la librería (`sort()`).

Haciendo uso de la orden *fit* de *Gnuplot* trata de encontrar la función de complejidad temporal que mejor se adapta a cada método (`sort()` y `mergesort()`). Para ello prueba un par de funciones (las que consideres que mejor se ajustan) para cada algoritmo y represéntalas mediante dicha herramienta en sendos gráficos de ejes coordenados.

Recordatorio de manejo de listas con las STL:

- Declarar una variable `l` tipo lista: `list<int> l`.
- Declarar un iterador `i` (puntero a lista): `list<int>::iterator i`.
- Acceder al elemento que apunta un iterador `i`: `*i`.
- Hacer que un iterador apunte al primer elemento de la lista: `i = l.begin()`.
- Hacer que el iterador `i` apunte al siguiente elemento de la lista: `++i` (o `i++`).
- Un iterador llega al último elemento de la lista `l` cuando: `i == l.end()`.
- Dada una lista ordenada `l1`, se le puede agregar otra lista ordenada `l2` de forma que `l1` quede ordenada con: `l1.merge(l2)`.
- Dada una lista `l2`, se le puede transferir los elementos de la lista `l1` que están entre los iteradores p1 y f1, de forma que queden, en la lista l2, a partir del elemento `p2` (los elementos transferidos se eliminan de `l1`) con: `l2.splice( p2, l1, p1, f1)`.

mas información en: `http://www.cplusplus.com/reference/list/list/`