Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Ayudantía 5: Algoritmos de Ordenamiento

IIC2133 - Estructuras de datos y algoritmos

Segundo semestre, 2017

1. Merge Sort

1. ¿Cuál es la complejidad en términos de tiempo? ¿y en términos de memoria?
2. ¿Es un algoritmo estable?
3. Suponga que recibe un arreglo completamente ordenado o casi ordenado. En este caso Mergesort hará muchas operaciones `merge()` de más, ya que mezclará arreglos que ya están ordenados entre ellos. ¿Cómo puede solucionar esto en el algoritmo?
4. Si la complejidad de `merge()` fuera $\mathcal{O}(1)$ ¿Cuál sería la complejidad de Merge Sort?
5. Por lo general, Merge Sort está implementado Top-Down ¿Cómo se podría impelementar un Merge Sort Bottom-Up?.
6. ¿Cómo podríamos hacer el algoritmo de manera iterativa?

2. Quick Sort

1. ¿Cuál es la complejidad en términos de tiempo? ¿y en términos de memoria?
2. ¿Qué ocurre con el algoritmo si recibe un arreglo ordenado?
3. ¿Cómo se puede elegir un buen pivote para que sea representativo?
4. Quicksort puede hacer comparaciones de más si es que existen muchos datos repetidos en el arreglo. ¿Qué cambio le podemos hacer al algoritmo en un caso como este?

3. Elija su algoritmo

Si se quiere ordenar datos en cada una de las siguientes situaciones, ¿qué algoritmos conviene usar? ¿cuáles NO conviene usar?

1. Datos que ya están medianamente ordenados
2. Datos ordenados al revés
3. Datos ordenados de a pares
4. Datos que son números naturales entre 1 y 100

4. Situaciones reales

1. En general, ¿qué algoritmo crees que se usa en la práctica? ¿qué algoritmo crees que usa la función `sorted()` de python?
2. ¿Cómo lo harías para ordenar una lista ligada?