Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2133 - Estructuras de Datos y Algoritmos

Ayudantía 2

3 de abril de 2020

Quicksort

Pregunta 1

Demuestre que *QuickSort* es correcto y calcule su complejidad en el caso promedio.

Pregunta 2

Los lenguajes de programación modernos utilizan una serie de técnicas con el fin de optimizar el tiempo de ejecución de sus algoritmos de ordenación[1]. Siguiendo esta línea de pensamiento:

a. Argumente utilizando como métrica el número de *swaps*, por qué *InsertionSort* funcionaría mejor que *QuickSort* en un *array* semi-ordenado.

b. Proponga una mejora para *QuickSort* con el fin de que, dado un *sub-array* de menos de α elementos, se utilice **otro algoritmo** que trabaje mejor con *items* semi-ordenados. Determine la complejidad del algoritmo mejorado.

Heaps

Pregunta 3

Se tiene un arreglo de n números, desconocido a priori. Se te pide construir un algoritmo, utilizando un $Heap$, que entregue eficientemente el k-ésimo numero mayor, de los primeros m números del arreglo, con $k \leq m \leq n$. Considere que una vez que su algoritmo entrega una solución, el m puede aumentar, por lo que su algoritmo deberá actualizar de forma eficiente la respuesta.

[1] Detalles de la implementación de *sort* en Python