Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre Abril-Julio 2023

Quicksort

1. Introducción

El objetivo de este laboratorio es el de agregar a la librería de ordenamiento `Sortlib.kt` tres versiones del algoritmo de ordenamiento `Quicksort`. También se quiere realizar un estudio experimental, para comparar el rendimiento de las tres algoritmos implementados.

2. Implementación de las variantes de Quicksort

La primera actividad consiste en agregar a la librería de ordenamiento `Sortlib.kt`, tres variantes del algoritmo `Quicksort`, las cuales se presentan a continuación.

Quicksort clásico: Denominamos así a la versión de Quicksort que se encuentra en la página 171 de [1].

***Quicksort with 3-way partitioning*:** Esta es una variante de Quicksort presentada por Sedgewick y Bentley en una charla [2]. En teoría este algoritmo es eficiente en la mayoría de los casos, incluyendo secuencias que están casi ordenadas y con muchas repeticiones de elementos. El pseudo código del algoritmo que debe implementar se encuentra en la página 9 de la presentación dada en [2]. En ese pseudo código el procedimiento llamado `exch`, corresponde al procedimiento `swap` visto en clase.

***Dual-pivot Quicksort by Yaroslavskiy* :** Esta variante fue introducida por Vladimir Yaroslavskiy, y debido a su buen rendimiento fue usado como el algoritmo de ordenamiento de la `Java runtime library` de Oracle, desde la versión 7 hasta versiones recientes. Esta es una de las primeras versiones de Quicksort, que presente un alto rendimiento usando dos pivotes. En específico, debe implementar el pseudo código de esta variante presentado en el Algoritmo 3 de [3].

En `Sortlib.kt`, estas variantes deben ser implementadas como procedimientos que solo reciben como argumento, el arreglo a ordenar. Los nombres a usar en los procedimientos son: `quicksortClasico`, `quicksortThreeWay` y `quicksortDualPivot`.

3. Estudio experimental

La segunda actividad consiste en un estudio experimental de las variantes de Quicksort. Debe ejecutar las tres versiones sobre arreglos de enteros generados aleatoriamente. Para cada tamaño de arreglo se debe ejecutar los algoritmos sobre 10 arreglos aleatorios con valores en el intervalo $[0, N]$, donde N, es el número de elementos del arreglo. El tiempo a reportar para cada uno de los algoritmos, en cada tamaño de arreglo, es el tiempo promedio, en segundos, de esos 10 arreglos. Observe que para que el experimento sea válido, cada variante de Quicksort, debe ordenar el mismo arreglo generado aleatoriamente. Los tamaños del los arreglos son: [500.000, 1.000.000, 1.500.000, 2.000.000].

Debe presentar un informe formato PDF, en el que se muestre una tabla con los tiempos promedios (con su desviación estándar), para cada uno de los algoritmos, en los cuatro tamaños de arreglos indicados. Además de la tabla, debe realizar un gráfico de tiempo versus tamaño del arreglo, en donde se pueda observar el comportamiento de las variantes de Quicksort, tal como se muestra en la Figura 2 de [3]. En el reporte también debe indicar los datos del sistema en donde hicieron las pruebas: el sistema de operación, el modelo CPU, la cantidad de RAM del computador, la versión del compilador de Kotlin y la versión de la JVM usada.

4. Condiciones de entrega

Para este laboratorio el único código que debe entregar es el de la librería `Sortlib.kt`. La versión final del código del laboratorio, el informe y la declaración de autenticidad firmada, deben estar contenidas en un archivo comprimido, con formato *tar.xz*, llamado *LabSem4_X_Y.tar.xz*, donde X y Y, son los números de carné de los estudiantes. La entrega del archivo *LabSem3_X_Y.tar.xz*, debe hacerse por la plataforma Classroom, antes de las 11:50 pm del día domingo 04 de junio de 2023.

Referencias

[1] Cormen, T., Leirserson, C., Rivest, R., and Stein, C. *Introduction to Algorithms*, 3ra ed. McGraw Hill, 2009.

[2] Sedgewick, R., and Bentley, J. Quicksort is optimal. `https://sedgewick.io/wp-content/uploads/2022/03/2002QuicksortIsOptimal.pdf`, 2002. KnuthFest, Stanford University.

[3] Wild, S., and Nebel, M. E. Average case analysis of java 7's dual pivot quicksort. In *Algorithms–ESA 2012: 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings 20* (2012), Springer, pp. 825–836.

Guillermo Palma / gvpalma@usb.ve / Junio 2023