Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre Enero-Marzo 2017

Implementación de Quicksort

1. Descripción de la actividad

El objetivo del laboratorio es realizar la implementación de dos versiones del algoritmo de ordenamiento Quicksort, y hacer un estudio experimental de las dos versiones.

Debe usar como código base el usado en los laboratorios se las semanas pasada, el cual está compuesto por los siguientes archivos:

- `arrayT.py` Módulo con la definición de arreglos a utilizar.
- `ordenamiento.py`: Módulo que debe contener la implementación de los algoritmos de ordenamientos.
- `cliente_ordenamiento.py`: Cliente que ejecuta todos los algoritmos de ordenamiento.

El archivo `ordenamiento.py` debe contener solamente las implementaciones de los algoritmos de Mergesort y Heapsort hechas en los laboratorios pasados y las implementaciones de Quicksort requeridas en este laboratorio.

1.1. Actividad 1: Implementación de Quicksort

La primera versión del algoritmo de Quicksort que debe implementar es la que se presenta en la página 170 de Cormen et al. [1]. En el archivo `ordenamiento.py` debe crear un procedimiento llamado `quicksort_basico(A)`, que recibe un arreglo A de tipo `arrayT` y ordena sus elementos con la implementación de esta versión de Quicksort.

1.2. Actividad 2: Implementación de la versión aleatoria de Quicksort

Se quiere que implemente la versión aleatoria de Quicksort que se presenta en la página 179 Cormen et al. [1]. La versión aleatoria de Quicksort debe ser implementada en el archivo `ordenamiento.py` mediante un procedimiento llamado `quicksort_aleatorio(A)`, que recibe un arreglo A de tipo `arrayT` y ordena sus elementos con la implementación de la versión aleatoria de Quicksort.

1.3. Actividad 3: Experimentos a realizar

Se quiere comparar el rendimiento de las dos versiones de Quicksort con respecto a los algoritmos de Mergesort y de Heapsort. Para ello se requiere que modifique el programa `cliente_ordenamiento.py`. Debe agregar a ese programa una opción a los argumentos de la línea de comandos que permita escoger el tipo de prueba que se desea realizar sobre los algoritmos de ordenamientos. En específico la opción debe llevar como banderas o *flags* el

comando `-p` y `--prueba` y luego un número entero que indica el tipo de prueba a ejecutar, es decir, indica la características de los elementos que componen el arreglo a ordenar. Los números enteros y sus pruebas asociadas son los siguientes:

1. Los arreglos contienen elementos generados aleatoriamente. Esta opción ya está implementada.
2. Si el tamaño del arreglo es n, entonces el arreglo contendrá la secuencia $n, n-1, n-2, \ldots, 2, 1$. Esta Esta opción debe ser implementada por usted.
3. El arreglos están compuestos por ceros y unos colocados aleatoriamente en el arreglo. Esta opción debe ser implementada por usted.

Por ejemplo, si se ejecuta:

```
>./cliente_ordenamiento.py -p 1  -t 500 1000
```

Se ordenan arreglos con elementos generados aleatoriamente de tamaño 500 y 1000.
Otro ejemplo, si se ejecuta:

```
>./cliente_ordenamiento.py --prueba 3  -t 100 300
```

Se ordenan arreglos de tamaño 100 300, cuyos elementos son ceros y unos colocados aleatoriamente en el arreglo.

Una vez implementada la generación de las tres pruebas indicadas anteriormente, debe ejecutar los algoritmos Quicksort básico, Quicksort aleatorio, Heapsort y Mergesort, en cada una de las tres pruebas, unas tres veces, es decir con opción `-t 3`, sobre arreglos con tamaños 1000, 2000, 4000 y 5000. Debe entregar una gráfica del rendimiento de los algoritmos, por cada una de las tres pruebas.

2. Condiciones de entrega

El trabajo es por equipos de laboratorio. Debe entregar los códigos fuentes de sus programas y las gráficas con los resultados de los algoritmos de ordenamiento, en un archivo comprimido llamado `LabSem5-X-Y.tar.gz` donde X y Y son los números de carné de los integrantes del grupo. La entrega se realizará por medio del aula virtual **antes** de la 4:30 pm del jueves 09 de Febrero de 2017.

Referencias

[1] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. *Introduction to algorithms*, 3rd ed. MIT press, 2009.

Guillermo Palma / gvpalma@usb.ve / Febrero 2017