Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre Septiembre-Diciembre 2013

Proyecto 3 Algoritmos de Ordenamiento

El objetivo de la práctica es la de realizar la implementación y el estudio del tiempo de ejecución de los algoritmos de ordenamiento *Insertionsort*, *Bubblesort*, *Heapsort*, *Quicksort* y *Mergesort*.

Se les proporcionará de un código base que usted debe modificar. El código base está compuesto por los siguientes archivos:

- `ordenamiento.py:` Librería que debe contener la implementación de los algoritmos de ordenamientos.
- `pruebaOrdenamiento.py:` Es un cliente que debe probar todos los algoritmos de ordenamiento.

La aplicación `pruebaOrdenamiento.py` se ejecuta con la siguiente llamada:

```
>python3 pruebaOrdenamiento [all|nlgn] <numero de pruebas> <numero de elementos>
```

donde los parámetros de entrada son:

[all | nlgn]: Tipo de algoritmos a ejecutar. Con la opción **"nlgn"** se ejecutaran los algoritmos de orden de ejecución $O(nlogn)$ (*Heapsort*, *Quicksort* y *Mergesort*) y con **"all"** todos los algoritmos de ordenamiento de `ordenamiento.py`.

numero de pruebas: es el número de pruebas que se realizan sobre los algoritmos de ordenamiento, es decir, el número de veces que se ejecutarán los algoritmos de ordenamiento con diferentes arreglos.

numero de elementos: Es el número de elementos que va a tener el arreglo a ordenar

Un ejemplo de una llamada válida en donde se ejecutaría la aplicación es

```
>python3 pruebaOrdenamiento all 2 20000
```

En este caso se harían dos ejecutarían dos veces todos los algoritmos de ordenamiento, con arreglos de tamaño 20000.

Cuando se realiza una prueba se genera un arreglo de enteros con valores aleatorios y en cada prueba los valores generados son diferentes. Luego ese arreglo de enteros lo recibe cada uno de los algoritmos de ordenamiento y una vez concluida la ejecución del algoritmo se procede a comprobar si los elementos están ordenados o no y a determinar el tiempo de corrida del algoritmo. El objetivo es que se apliquen los algoritmos de ordenamiento a arreglos de varios tamaños para estudiar el comportamiento de los mismos en cuanto su tiempo de ejecución y verificar si se cumple lo expresado de cada algoritmo, según lo indicado en la teoría. Se realizan varias pruebas a los algoritmos de ordenamientos, *porque se quiere obtener el tiempo promedio de ejecución de cada algoritmo y su desviación estándar*. Los algoritmos deben ser implementados siguiendo el pseudocódigo presentado en el libro de Cormen et al [1] o el del reporte técnico de Ravelo y Fernández [3], excepto el algoritmo *Bubblesort*. Para el algoritmo *Bubblesort* se implementaran las dos versiones que se encuentra descritas en la página 177 del libro de Kaldewaij [2]: *Bubblesort0* y la versión mejorada *Bubblesort1*.

El programa `pruebaOrdenamiento.py` debe mostrar, por la salida estándar del computador, el *tiempo promedio de ejecución* en milisegundos de cada algoritmo para las pruebas realizadas y el valor de la *desviación estándar* de los tiempos de ejecución de cada algoritmo. Los números deben mostrarse con 2 decimales. El formato de la salida es el siguiente:

```
<Algoritmo 1>  <tiempo promedio 1>  <desviación estándar 1>
<Algoritmo 2>  <tiempo promedio 2>  <desviación estándar 2>
..
<Algoritmo N>  <tiempo promedio>  <desviación estándar N>
```

Por ejemplo, si se ejecuta el programa `pruebaOrdenamiento.py` con la opción "nlgn" un ejemplo de una salida válida es:

```
Quicksort 1463.11 25.61
Mergesort 1569.21 24.14
Heapsort 1565.48 22.31
```

Una vez completada la implementación y documentación de los algoritmos de ordenamientos, debe realizar una comparación del tiempo de los mismos con diversos tamaños de arreglos. La idea es que pruebe los algoritmos de ordenamientos con arreglos de varios tamaños para ver el crecimiento del tiempo de cómputo.

Debe realizar dos experimentos que deben ser reportados en un informe. En el primer experimento se evalúan todos los algoritmos de ordenamientos. Debe hacer cinco (5) pruebas con todos los algoritmos usando como tamaño de arreglo los presentados en la tabla 1.

Número de elementos	Insertion	BubbleSort0	BubbleSort1	Quicksort	Mergesort	Heapsort
10000						
20000						
30000						
40000						
50000						

Tabla 1: Resultados del tiempo de ejecución promedio, en milisegundos, de 5 corridas de los algoritmos de ordenamiento

En el segundo experimento se prueban los algoritmos de ordenamiento con tiempo promedio $O(nlogn)$. Debe realizar cinco (5) pruebas usando como tamaño de arreglo los presentados en la tabla 2.

Número de elementos	Quicksort	Mergesort	Heapsort
1000000			
2000000			
3000000			
4000000			
5000000			
6000000			
7000000			
8000000			
9000000			
10000000			

Tabla 2: Resultados del tiempo de ejecución promedio, en milisegundos, de 5 corridas de los algoritmos de ordenamiento $O(nlogn)$

Debe realizar un pequeño informe que debe contener los resultados de las tablas 1 y 2 y por cada tabla debe realizar una gráfica en donde se muestre los resultados de cada algoritmo, junto con una línea, por cada algoritmo, que represente la tendencia de los puntos. También debe hacer un análisis de los resultados de cada tabla y gráfica que no debe pasar de una cuartilla. El informe debe estar en formato digital PDF.

El trabajo es por equipos de laboratorio. Debe entregar los códigos fuentes de sus programas y el informe, en un archivo comprimido llamado `Proyecto3-XX.tar.gz` donde XX es el número

del aula virtual del grupo. La entrega se realizará por medio del aula virtual **antes** de la 1:30 pm jueves 19 de Diciembre de 2013. Asimismo cada integrante de un equipo deberá imprimir, firmar y entregar a su profesor la "Declaración de Autenticidad para Entregas", cuya plantilla se encuentra en el Aula Virtual del curso. El no cumplimento de los requerimientos podrá resultar en el rechazo de su entrega.

Referencias

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

[2] Kaldewaij, A. *Programming: the derivation of algorithms*. Prentice-Hall, Inc., 1990.

[3] Ravelo, J., and Fernandez, K. Ordenamiento sobre arreglos. `http://www.ldc.usb.ve/~crodrig/cursos/ci2612/MaterialOrdenamiento.pdf`, 2011.

Guillermo Palma / gvpalma en usb.ve / Diciembre 2013