ANÁLISIS Y DISEÑO DE ALGORITMOS

Complejidad temporal: Análisis empírico (II)

Práctica 2 de laboratorio

Entrega: Hasta el domingo 18 de febrero, 23:55h. A través de Moodle

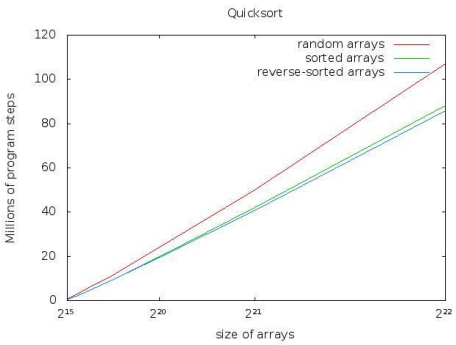
Siguiendo las pautas de la práctica anterior, realiza un estudio empírico de la complejidad temporal[1] de los algoritmos de ordenación *Quicksort* y *Heapsort*. Para ello, descarga el código fuente disponible a través de *Moodle* que contiene una implementación de dichos algoritmos y cumplimenta la función *main* para que muestre el número de **pasos de programa**[2] que realizan los algoritmos en tres supuestos distintos: vectores con contenido aleatorio, vectores ordenados de manera creciente y vectores ordenados de manera decreciente (en orden inverso)[3], representando los resultados, expresados en millones de pasos de programa, en una tabla similar a la que se muestra. Los tamaños de vector analizados son las potencias de 2, desde 15 hasta 22. El programa ejecutable debe llamarse `qs-vs-hs`.

```
QUICKSORT VERSUS HEAPSORT.
Average processing Msteps (millions of program steps)
Number of samples (arrays of integer): 30
```

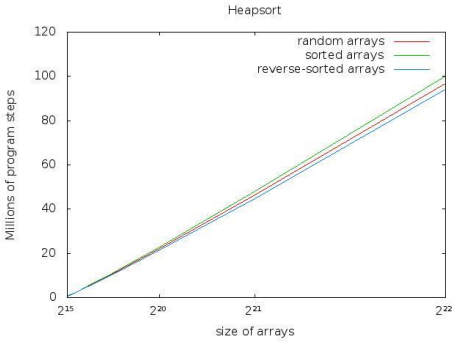
	RANDOM ARRAYS		SORTED ARRAYS		REVERSE SORTED ARRAYS	
Size	QuickSort	HeapSort	QuickSort	HeapSort	QuickSort	HeapSort
32768	0.559	0.528	0.459	0.549	0.442	0.505
65536	1.211	1.121	0.983	1.166	0.950	1.072
131072	2.569	2.374	2.097	2.469	2.032	2.276
262144	5.373	5.009	4.457	5.189	4.325	4.811
524288	11.547	10.542	9.437	10.888	9.175	10.168
1048576	24.063	22.133	19.924	22.896	19.399	21.389
2097152	50.675	46.363	41.946	48.014	40.898	44.855
4194304	105.547	96.920	88.093	100.159	85.996	93.981

A continuación representa los datos obtenidos mediante gráficas utilizando la herramienta *Gnuplot*. El archivo de órdenes asociado debe llamarse `qs-vs-hs.gnuplot` y el fichero que contiene la tabla de resultados obtenidos `qs-vs-hs.Msteps`. Realiza las siguientes gráficas comparativas, en formato `png` y extrae tus propias conclusiones.

- Gráfica 1: Análisis del algoritmo Quicksort comparando su comportamiento cuando se suministra: (1) vectores con contenido y orden aleatorio; (2) vectores ordenados y (3) vectores en orden inverso. La gráfica obtenida debe llamarse `quickSort.png`.



- Gráfica 2: Análisis del algoritmo Heapsort comparando su comportamiento cuando se suministra: (1) vectores con contenido y orden aleatorio; (2) vectores ordenados y (3) vectores en orden inverso. La gráfica obtenida debe llamarse `heapSort.png`.

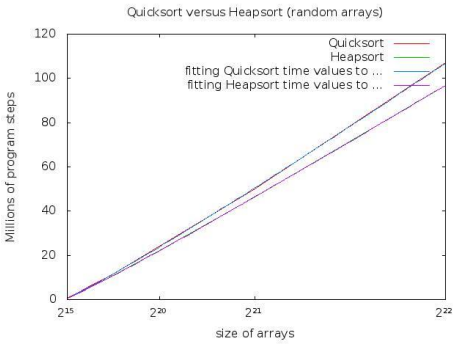


[1] A diferencia con la anterior, en esta práctica hay que hacer el estudio contando pasos de programa, es decir, no se debe utilizar la librería `chrono` ni ninguna función que obtenga tiempos de ejecución.

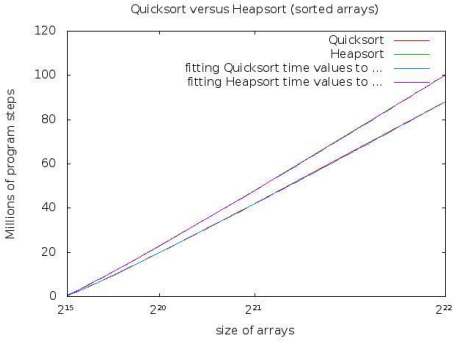
[2] Un paso de programa es un conjunto de instrucciones (no tienen por qué estar consecutivas) cuya ejecución está acotada por una constante, es decir, no depende del tamaño del problema.

[3] Para que el análisis sea válido es importante tener en cuenta que ambos algoritmos deben recibir vectores con el mismo contenido.

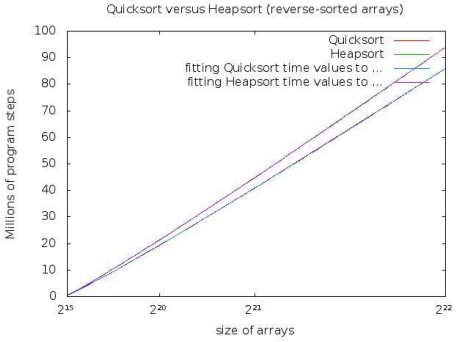
- Gráfica 3: Análisis comparativo de Quicksort y Heapsort cuando reciben vectores con contenido y orden aleatorio. Se representa también la mejor función de ajuste encontrada para cada uno de ellos. La gráfica obtenida debe llamarse `qs-vs-hs-RA.png`.



- Gráfica 4: Análisis comparativo de Quicksort y Heapsort cuando reciben vectores ordenados. Se representa también la mejor función de ajuste encontrada para cada uno de ellos. La gráfica obtenida debe llamarse `qs-vs-hs-SA.png`.



- Gráfica 5: Análisis comparativo de Quicksort y Heapsort cuando reciben vectores en orden inverso. Se representa también la mejor función de ajuste encontrada para cada uno de ellos. La gráfica obtenida debe llamarse `qs-vs-hs-RSA.png`.



Normas para la entrega.

> **ATENCIÓN: Estas normas son de obligado cumplimiento para que esta práctica sea evaluada.**

1. Se debe entregar únicamente los ficheros `qs-vs-hs.cc`, `qs-vs-hs.gnuplot` y `makefile`. Sigue escrupulosamente los nombres de ficheros, objetivos, etc. que aquí se citan. No hay que entregar nada más.
2. El archivo `makefile` debe contener los mismos objetivos que la práctica anterior pero con los siguientes nombres: `qs-vs-hs`, `qs-vs-hs.Msteps`, `graphs` y `all`.
3. Es imprescindible que no presente errores ni de compilación ni de interpretación (según corresponda), en los ordenadores del laboratorio asignado.[4] Se tratará de evitar también cualquier tipo de *warning*.
4. Todos los ficheros que se entregan deben contener el nombre del autor y su DNI (o NIE) en su primera línea (entre comentarios apropiados según el tipo de archivo).
5. Se comprimirán en un archivo *.tar.gz* cuyo nombre será el DNI del alumno, compuesto de 8 dígitos y una letra (o NIE, compuesto de una letra seguida de 7 dígitos y otra letra). Por ejemplo: 12345678A.tar.gz o X1234567A.tar.gz. **Solo se admite este formato de compresión y solo es válido esta forma de nombrar el archivo.**
6. En el archivo comprimido **no debe existir subcarpetas**, es decir, al extraer sus archivos estos deben quedar guardados en la misma carpeta donde está el archivo que los contiene.
7. La práctica hay que subirla a *Moodle* respetando las fechas expuestas en el encabezado de este enunciado.

[4] Si trabajas con tu propio ordenador o con otro sistema operativo asegúrate de que este requisito se cumple.