Trabajo prï¿½ctico 0: Infraestructura bï¿½sica

Alejandro Garcï¿½a Marra, *Padrï¿½n Nro. 91.516*
alemarra@gmail.com
Sebastiï¿½n Javier Bogado, *Padrï¿½n Nro. 91.707*
sebastian.j.bogado@gmail.com
Grupo Nro. 0 - 2do. Cuatrimestre de 2012
66.20 Organizaciï¿½n de Computadoras
Facultad de Ingenierï¿½a, Universidad de Buenos Aires

Resumen

El presente trabajo busca crear un programa que permita el ordenamiento de archivos a travï¿½s de dos implementaciones distintas, una utilizando el algoritmo Quicksort y la otra el algoritmo Stooge sort.
Sobre este programa, luego, se realizarï¿½n una serie de mediciones con el fin de determinar los desempeï¿½os relativos de cada implementaciï¿½n y las posibles mejoras a realizar. Para esto haremos uso de los programas **time** y **gprof**.

1. Introducciï¿½n

Algo sobre profiling?

2. Mediciones

2.1. Valores Obtenidos

En la tabla 1 se presentan las mediciones realizadas con **time** sobre ambos algoritmos de ordenamiento y con archivos de distintos tamaï¿½os.

Ademï¿½s de los archivos indicados en el enunciado, fueron agregadas mediciones sobre archivos con tamaï¿½os arbitrarios, mayores, con el fin de mostrar de mejor manera las diferencias entre algoritmos.

		Quicksort			Stooge sort		
		Ordenado	Invertido	Aleatorio	Ordenado	Invertido	Aleatorio
1kb	real*	0.00	0.00	0.00	0.00	0.00	0.00
	user*	0.00	0.00	0.00	0.00	0.00	0.00
	sys*	0.00	0.00	0.00	0.00	0.00	0.00
8kb	real	0.00	0.00	0.00	0.02	0.02	0.01
	user	0.00	0.00	0.00	0.01	0.01	0.01
	sys	0.00	0.00	0.00	0.00	0.00	0.00
16kb	real	0.00	0.00	0.00	0.00	0.02	0.02
	user	0.00	0.00	0.00	0.00	0.01	0.02
	sys	0.00	0.00	0.00	0.00	0.00	0.00
32kb	real	0.00	0.00	0.00	0.17	0.17	0.17
	user	0.00	0.00	0.00	0.17	0.17	0.17
	sys	0.00	0.00	0.00	0.00	0.00	0.00
64kb	real	0.00	0.00	0.00	1.44	1.44	1.44
	user	0.00	0.00	0.00	1.44	1.43	1.44
	sys	0.00	0.00	0.00	0.00	0.00	0.00
1024kb	real	0.04	0.03	0.03	>1500	>1500	>1500
	user	0.03	0.02	0.03	>1500	>1500	>1500
	sys	0.00	0.00	0.00	0.00	0.00	0.00

Cuadro 1: Resultados comando Time

* Referencia:

- real: %e, tiempo total real usado por el proceso.
- user: %U, total de segundos-CPU usados por el proceso directamente.
- sys : %S, total de segundos-CPU utilizados por el systema en nombre del proceso.

2.2. Anï¿½lisis de los datos

La marcada diferencia entre la complejidad de los algoritmos se refleja en muestras tan chicas como la de 8kb. A partir de ahï¿½, el Stooge sort ya hace suficiente uso del procesador como para ser notado por time, mientras que el Quicksort hace lo propio reciï¿½n en la muestra mï¿½s grande, de 1024kb. En este caso, el Stooge sort se tornï¿½ intolerable.

En la figura 1 se muestra el grï¿½fico del tiempo insumido por el Quicksort para las distintas muestras.
En la figura 2 se muestra el grï¿½fico del tiempo insumido por el Stooge sort para las distintas muestras, a excepciï¿½n de la de 1024kb, porque demanda una escala que harï¿½a inapreciable la situaciï¿½n de las otras muestras.

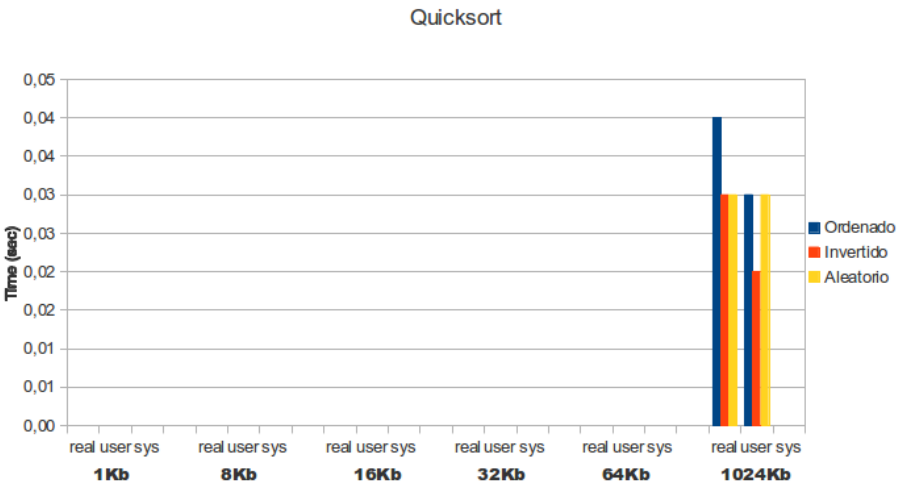

Figura 1: Tiempo tomado para distintas muestras del Quicksort.

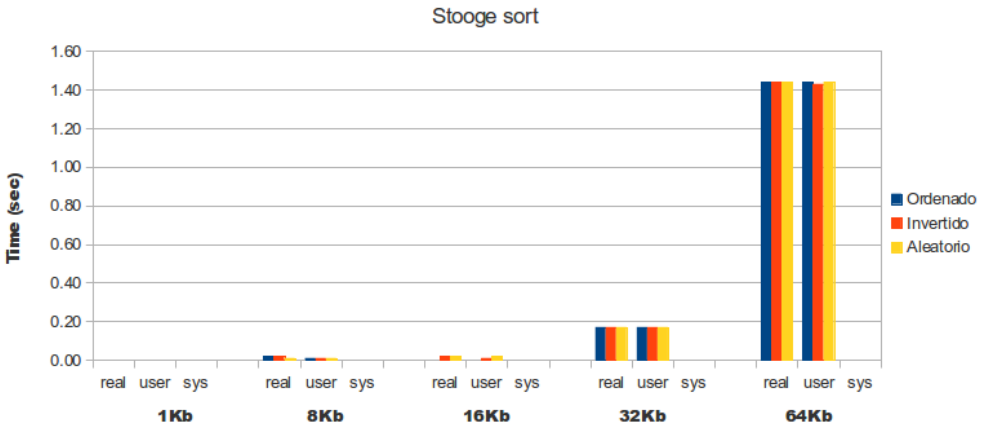

Figura 2: Tiempo tomado para distintas muestras del Stooge sort.

Al momento de calcular el speedup de Quicksort contra Stooge sort, con la razï¿½n entre los tiempos de cada uno, las cifras obtenidas no lo permiten. El primero no arroja resultados apreciables por **time** en ninguna instancia sin contar la ï¿½ltima, donde el Stooge sort es inmanejable.

Esto es porque la complejidad del Quicksort es, en promedio, $\mathrm{O}\big(n \log(n)\big)$, mientras que el Stooge sort es de $\mathrm{O}\big(n^{2,7}\big)$. Entonces, el speedup entre ambos algoritmos tiende a infinito exponencialmente, segï¿½n crece el tamaï¿½o de la muestra.

3. Profiling

Para realizar el profiling tomamos una muestra suficientemente grande como para que las funciones mismas de **gprof** tuvieran una incidencia despreciable en la prueba. En la figura 3 se muestra un ejemplo de cï¿½mo presentar las ilustraciones del informe.

Figura 3: Facultad de Ingenierï¿½a – Universidad de Buenos Aires.

4. Conclusiones

Se presentï¿½ un modelo para que los alumnos puedan tomar como referencia en la redaccciï¿½n de sus informes de trabajos prï¿½cticos.

Referencias

[1] Intel Technology & Research, "Hyper-Threading Technology," 2006, http://www.intel.com/technology/hyperthread/.

[2] J. L. Hennessy and D. A. Patterson, "Computer Architecture. A Quantitative Approach," 3ra Edicciï¿½n, Morgan Kaufmann Publishers, 2000.

[3] J. Larus and T. Ball, "Rewriting Executable Files to Mesure Program Behavior," Tech. Report 1083, Univ. of Wisconsin, 1992.