Algoritmos y Estructuras de Datos
Tarea 1

Profesor: Sergio Rajsbaum
Ayudante: Francisco Javier Escalona González

fecha de hoy: 26 de agosto 2004
fecha de entrega: 9 de septiembre 2004
— *explica en detalle y con claridad todas tus respuestas* —
— *Tus algoritmos deberán ser lo más eficiente posibles* —
— *explica el funcionamiento de tus algoritmos informalmente, luego escribe el código, y luego demuestra correctez y complejidad.*—
Se permite trabajar en equipos de dos personas.
Pero cada uno debe entregar la tarea resuelta por separado, e indicar el nombre de su compañero de equipo.

Tema: Busqueda Binaria y Divide-y-Vencerás

1. Escribe un resumen de no mas de dos cuartillas de las páginas 51 a la 88 del libro *Out of their Minds* de Shasha y Lazere, acerca de la vida y obra de Dijkstra y de Rabin.

2. Diseña un algoritmo que recibe una lista no acotada A, es decir, $A[1], A[2], \ldots$ que nunca termina y contiene enteros ordenados; es decir, $A[1] < A[2] < A[3] < \cdots$ Y recibe un número x. Como salida debe contestar si x esta o no en A, y en caso positivo, decir en que posición de A se encuentra.

3. Diseña un algoritmo para mezclar (merge) dos listas ordenadas en una sola. $Merge(A, p, q, r)$, donde $p \leq q < r$, supone que $A[p..q]$ y $A[q+1..r]$ estan ordenadas (de menor a mayor), y regresa en el mismo arreglo $A[p..r]$ la mezcla de estos dos subarreglos ordenados.

4. Resuelve lo siguiente (problema 1-1, pag. 17 del libro de Cormen, Leiserson y Rivest). Para cada una de las siguientes funciones $f(n)$ y tiempo t, determina el tamaño mas grande n de un problema que puede ser resuelto en tiempo t, suponiendo que el algoritmo resuelve el problema en $f(n)$ microsegundos. Funciones: $\log n$, $\sqrt{n}$, n, $n \log n$, n^2, n^3, 2^n, $n!$. Tiempos: 1 segundo, 1 minuto, 1 hora, 1 dia, 1 mes, 1 año, 1 siglo.

5. Resuelve lo siguiente (problema 1-2, pag. 17 del libro de Cormen, Leiserson y Rivest). A pesar de que merge sort corre en tiempo $\Theta(n \log n)$ e insertion sort corre en tiempo $\Theta(n^2)$, las constantes de insertion sort lo hacen correr mas rápido para valores de n pequeños. Por lo tanto hace sentido usar insertion sort dentro de merge sort cuando se llega al punto de resolver subproblemas suficientemente pequeños. Diseña una modificación a merge sort en la cual n/k sublistas de longitud k son ordenadas usando insertion sort, y luego mezcladas usando el mecanismo de merge sort, donde k es un valor por determinar.

(a) Prueba que las n/k sublistas, cada una de longitud k, puede ser ordenada por insertion sort en tiempo $\Theta(nk)$.

(b) Prueba que las sublistas pueden ser ordenadas en tiempo $\Theta(n \log(n/k))$.

(c) ¿ Dado que el algoritmo modificado corre en tiempo $\Theta(nk + n \log(n/k))$, cual es el valor mas grande asintótico (notación Θ) de k como función de n para el cual el algoritmo modificado tiene la misma complejidad que el merge sort estandar?

(d) ¿ Cómo se debe elegir k en la práctica?

No se aceptan tareas despues de la fecha límite.

Para cualquier duda contactar a Paco o a Sergio con suficiente anticipación:
pax@ciencias.unam.mx, rajsbaum@matem.unam.mx