Algoritmos y Estructuras de Datos
Tarea 1

Profesor: Sergio Rajsbaum
Ayudante: Jorge Figueroa

fecha de hoy: 26 de agosto 2006
fecha de entrega: 4 de septiembre 2006 – No se aceptan tareas después de esta fecha
— explica en detalle y con claridad <u>todas</u> 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 hasta tres personas.
Pero cada uno debe entregar la tarea resuelta por separado, e indicar el nombre de su compañero de equipo.

Tema: Introducción al análisis de correctez y complejidad de algoritmos.

1. Demuestra que $40 \cdot 2^n + n^5 \in o(3^n)$, y presenta una gráfica de estas funciones que justifique el resultado.

2. El siguiente algoritmo compara cadenas en el sentido lexicográfico. Es decir, en el sentido en que lo hacen los diccionarios para ordenar palabras. **Entrada:** $c1$, $c2$: son cadenas de caracteres terminadas por el caracter especial de terminación de cadena '\0'. Este caracter, por definición, es menor que cualquier otro. **salida:** -1 si $c1$ es menor lexicograficamente que $c2$; 0 si $c1$ es igual a $c2$; 1 si $c1$ es mayor lexiograficamente que $c2$.

   El primer índice de la cadena es el 0. Es decir, para $C =$ "*hola*" tenemos $C[0] ==' h '$, $C[1] ==' o '$, $C[2] ==' l '$, $C[3] ==' a '$, $C[4] ==' \backslash 0 '$. Sea $|C|$ la longitud de la cadena, en este caso, $|C| = 5$.

```
Function compara_lexicograficamente(c1, c2):
        i = 0
        do
            if c1[i] < c2[i] then return(−1) fi
            if c1[i] > c2[i] then return(1) fi
            if c1[i] ==' \0 ' then return(0) fi
            i = i + 1
        while(TRUE)
```

Figure 1: Algoritmo de comparación lexicográfica.

   Propón una definición de comparación de dos matrices (en lugar de cadenas) de manera lexicográfica, cada una de 2 dimensiones (en lugar de 1 dimensión, como son las cadenas). Demuestra la correctez y analiza la complejidad de tu algoritmo.

Tip: Para la complejidad, considérala en función de n, el total de elementos en ambas matrices.

3. Para cada una de las siguientes funciones $f(n)$, 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: $2500n$, $n!$, $(\log n)^3$, $n^{1/2}$, $n^{(\log n)^2}$, $n \log n$, $\log n$, 3^n, $\log \log n$. Tiempos t: 1 segundo, 1 minuto, 1 hora, 1 dia, 1 mes, 1 año, 1 siglo. Escribe una tabla con tus resultados, en la que cada columna está asociada una de las funciones, y ordenando las columnas de menor a mayor velocidad de crecimiento de las funciones (la 1era columna con los datos de la funcion que crezca más despacio, la 2nda con los de la funcion que le siga en crecimiento, etc).