Algoritmos y Estructuras de Datos
Tarea 1

Profesor: Sergio Rajsbaum
Ayudante: David Flores

fecha de hoy: 23 de agosto 2005
fecha de entrega: 30 de agosto 2005 – 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 cuatro 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 la correctez y analiza la complejidad del siguiente algoritmo de comparación lexicográfica. Este 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 $c2$.

   Supondremos que las cadenas pueden indexarse para obtener sus caracteres individuales. Su primer índice es el 0. Es decir, para $C =$ "*hola*" tenemos $C[0] ==' h '$, $C[1] ==' o '$, $C[2] ==' l '$, $C[3] ==' a '$, $C[4] ==' \backslash 0 '$. Y sea $|C|$ la longitud de la cadena, en este caso, $|C| = 5$.

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

Figure 1: Algoritmo de comparación lexicográfica.

   Tip: Para la correctez observa cuales invariantes se cumplen justo antes de entrar al ciclo. Para la complejidad, considérala en función de $n = |c1| + |c2|$.

2. 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: n, $n!$, $(\log n)^2$, $n^{1/3}$, $n^{\log n}$, $n \log n$, $\log(n^2)$, n^2, 2^n, $n^{1/4}$, $\log n$. 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).

3. En el Acertijo de las Tortillas se tiene una pila de n tortillas de distinto diámetro cada una. El objetivo es obtener una pila en la cual las tortillas estén ordenadas, la mayor hasta abajo y la menor hasta arriba. La operación permitida es tomar una pala, insertarla en algún lugar de la pila, y voltear todas las tortillas que queden encima de la pala. Ver la figura. Diseña un algoritmo que resuelva este acertijo, analizando complejidad y correctez.

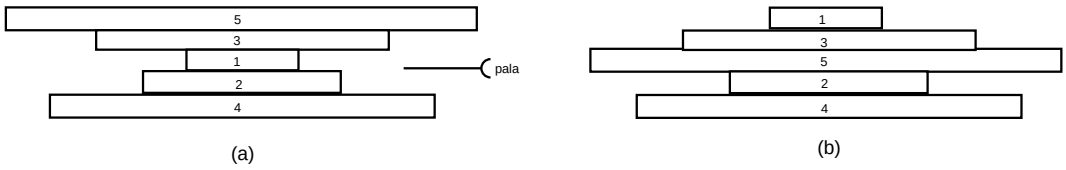

Figure 2: Insertando la pala abajo de la tortilla 1 en (a) se voltean las 1,3,5 y quedan como en (b)

Para cualquier duda contactar a David o a Sergio a colegadavid@gmail.com, rajsbaum@matem.unam.mx con suficiente anticipación.