UBA - FCEyN - Departamento de Computación

ALGORITMOS Y ESTRUCTURAS DE DATOS III
Trabajo Práctico Nº1

Primera entrega: 14-ABR-2010, hasta las 19:00 horas.
Segunda entrega: 05-MAY-2010, hasta las 20:00 horas.

Ver información general sobre los Trabajos Prácticos en la página de la materia en Internet.

- Desarrollar e implementar algoritmos para los problemas enumerados.
- Calcular el orden de la complejidad de cada algoritmo, y decir si es constante, logarítmica, lineal, polinomial, exponencial, o peor, en función del tamaño de la entrada. Para calcular la complejidad elegir el modelo más adecuado (uniforme o logarítmico), justificando la elección.
- Aplicar cada algoritmo a un conjunto de instancias de entrada, midiendo para cada instancia el tiempo de ejecución o el costo de las operaciones. Elegir las instancias de manera tal que se pueda apreciar el comportamiento de los algoritmos. Graficar para cada algoritmo el tiempo de ejecución o el costo de las operaciones conforme crece el tamaño de la entrada.
- Leer los datos de entrada desde un archivo con el nombre `Tp1EjX.in`, donde `X` es el número del problema. Escribir los resultados en un archivo con el nombre `Tp1EjX.out`.
- Respetar los formatos de archivos que se indican en cada caso. Como regla general, los valores contenidos en cada línea de los archivos de entrada están separados entre sí por uno o más espacios en blanco, y pueden estar precedidos o seguidos por una cantidad arbitraria de espacios en blanco; asimismo, los valores contenidos en cada línea de los archivos de salida están separados entre sí por exactamente un espacio en blanco, y no son precedidos ni seguidos por ningún caracter dentro de la línea. Utilizar como ejemplo los archivos provistos.
- Ignorar los costos de lectura y escritura de los archivos tanto al calcular complejidad como al medir.

1. Dados $b, n \in \mathbb{N}$, calcular $b^n \bmod n$.

   `Tp1Ej1.in`: Contiene un conjunto de instancias a tratar, cada una almacenada en una línea diferente del archivo, donde aparecen b y n. Asumir que b tiene un tamaño acotado. El archivo termina con una línea donde b y n valen -1, la cual no debe interpretarse como una instancia de entrada.

   `Tp1Ej1.out`: Contiene una línea por cada instancia de entrada, donde aparece el valor pedido.

2. Las alumnas del 3ºA de la escuela Técnica Pública Nº1 (TP1) están por jugar a la ronda. En este juego cada persona toma de la mano a otras dos formando un círculo que recibe el nombre de ronda, luego de lo cual las personas cantan y saltican haciendo girar la ronda en sentido horario o antihorario. Como en todo grupo de personas, algunas alumnas son amigas y otras no tanto. A fin de que el juego sea lo más divertido posible, las alumnas han decidido que para formar la ronda, cada alumna debe tomar de la mano a dos de sus amigas. Decidir si las alumnas pueden formar una ronda que las contenga a todas.

`Tp1Ej2.in`: Contiene un conjunto de instancias a tratar, cada una almacenada en varias líneas del archivo. Cada conjunto de $n \geq 3$ alumnas se representa con $n + 1$ líneas. Cada alumna se identifica por un número correlativo entre 1 y n. En la primera línea aparece n. En la i-ésima de las n líneas restantes aparece la cantidad de amigas de la i-ésima alumna, seguida de la lista de esas amigas. Asumir que la amistad es simétrica y no reflexiva. El archivo termina con una línea donde n vale -1, la cual no debe interpretarse como una instancia de entrada.

`Tp1Ej2.out`: Contiene una línea por cada instancia de entrada, la cual dice "`ronda`" si las alumnas pueden formar una ronda que las contenga a todas, o dice "`no`" en caso contrario.

3. Una empresa de desarrollo de software acaba de implementar un novedoso sistema con el cual espera aumentar la productividad de sus programadores. Cada programador debe cumplir por día cierta tiempo de trabajo en la empresa, pero es libre de hacerlo en el horario que desee. El tiempo de trabajo puede ser diferente para cada programador, y no puede fraccionarse a lo largo del día. El único inconveniente que subsiste con el nuevo sistema es que cuando hay muchos programadores trabajando, ocasionalmente la máquina expendedora de café se queda sin suministros. Se dispone de dos listas, extraídas del sistema de control de ingresos y egresos de la empresa. Una de las listas contiene para cada programador su horario de ingreso a la empresa; en la otra lista los horarios son de egreso. Tanto la lista de ingresos como la de egresos se encuentran ordenadas por horario, ya que esa es la manera en que el sistema de control registra los eventos. Se considera que cada programador está dentro de la empresa desde su horario de ingreso hasta su horario de egreso, incluyendo ambos extremos. Utilizar las listas para determinar la mayor cantidad de programadores que están simultáneamente dentro de la empresa, a fin de que la máquina expendedora de café pueda ser abastecida adecuadamente.

   `Tp1Ej3.in`: Contiene un conjunto de instancias a tratar, cada una almacenada en varias líneas del archivo. Cada conjunto de $n \geq 1$ programadores se representa con $2n + 1$ líneas. Cada programador se identifica por un número correlativo entre 1 y n. En la primera línea aparece n. Las siguientes n líneas contienen la lista de ingresos, mientras que las últimas n líneas contienen la lista de egresos. En cada línea de las listas aparece un horario junto con el programador que ingresó o egresó en ese horario. Los horarios tienen el formato `HH:MM:SS` (horas, minutos y segundos, respectivamente). El horario más temprano posible es `00:00:00`, mientras que el más tarde posible es `23:59:59`. Asumir que cada programador ingresa estrictamente antes de egresar. El archivo termina con una línea donde n vale -1, la cual no debe interpretarse como una instancia de entrada.

   `Tp1Ej3.out`: Contiene una línea por cada instancia de entrada, donde aparece la cantidad pedida.