Trabajo práctico 1

Fecha de entrega: viernes 4 de septiembre, hasta las 18:00 hs.
Fecha de reentrega: viernes 18 de septiembre, hasta las 18:00 hs.

Este trabajo práctico consta de varios problemas y para aprobar el trabajo se requiere aprobar todos los problemas. La nota final del trabajo será un promedio ponderado de las notas finales de los ejercicios y el trabajo práctico se aprobará con una nota de 5 (*cinco*) o superior. De ser necesario (o si el grupo lo desea) el trabajo podrá reentregarse una vez corregido por los docentes y en ese caso la reentrega deberá estar acompañada por un *informe de modificaciones*. Este informe deberá detallar brevemente las diferencias entre las dos entregas, especificando los cambios, agregados y/o partes eliminadas del trabajo. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega. Para la reentrega del trabajo **podrían pedirse ejercicios adicionales**.

Para cada ejercicio se pide encontrar una solución algorítmica al problema propuesto y desarrollar los siguientes puntos:

1. Describir detalladamente el problema a resolver dando ejemplos del mismo y sus soluciones.
2. Explicar de forma clara, sencilla, estructurada y concisa las ideas desarrolladas para la resolución del problema usando pseudocódigo (que no es código fuente) y/o lenguaje coloquial.
3. Justificar por qué el procedimiento desarrollado en el punto anterior resuelve efectivamente el problema.
4. Deducir una cota de complejidad temporal del algoritmo propuesto (en función de los parámetros que se consideren correctos) y justificar por qué el algoritmo desarrollado para la resolución del problema cumple la cota dada. Utilizar el modelo uniforme salvo que se explícite lo contrario.
5. Dar un código fuente claro que implemente la solución propuesta. El mismo no sólo debe ser correcto sino que además debe estar bien programado. Si bien no se pide que siga ninguna convención de codificación específica, mínimamente el mismo debe tener nombres de variables apropiados y un estilo de indentación coherente. Se deben incluir las partes relevantes del código como apéndice del informe impreso entregado.
6. Dar un conjunto de casos de test que permitan corroborar en la práctica la correctitud del problema. Debe explicarse por qué los tests cubren los casos posibles del problema. **No** deben ser muchísimos casos ni muy grandes, salvo que el tamaño sea una necesidad para determinar la correctitud.
7. Dar un conjunto de casos de test que permitan observar la performance en términos de tiempo del problema. Para esto se deben desarrollar tanto tests de mejor y peor caso como tests generados sin una intencionalidad (detallando cómo fueron generados).
8. Presentar en forma gráfica y ofrecer conclusiones sobre la comparación entre tiempo estimado de corrida según la complejidad temporal calculada, tiempo medido de corrida para los tests patológicos de peor caso, y tiempo medido de corrida para los tests sin intencionalidad.

Respecto de las implementaciones, Java es el lenguaje sugerido por la Cátedra pero se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia (previa consulta con el docente). Debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación.

La entrada y salida debe hacerse por medio de archivos. No se considerará correcta una implementación que no pase los tests que se mencionaron en los puntos anteriores. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Será valorada la presentación de un análisis de casos críticos para los problemas presentados, podas posibles de ser pertinentes y caminos alternativos que llevaron hasta la solución presentada.

Deberá entregarse un informe impreso que desarrolle los puntos mencionados. Deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `emilioOca@gmail.com` indicando *TP 1* y luego los apellidos de los integrantes del grupo.

Problema 1: Telégrafo

La comunicación es el progreso! decididos a entrar de lleno en la nueva era el país decidió conectar telegráficamente todas las estaciones del moderno sistema férreo que recorre el país en abanico con origen en la capita (el kilómetro 0). Por lo escaso del presupuesto, se ha decidido ofrecer cierta cantidad de kilometros de cable a cada ramal. Pero para maximizar el impacto en epocas electorales se busca lograr conectar la mayor cantidad de ciudades con los metros asignados (sin hacer cortes en el cable)

Resolver cuantas ciudades se pueden conectar para cada ramal en $O(n)$, con n la cantidad de estaciones en cada ramal, y justificar por qué el procedimiento desarrollado resuelve efectivamente el problema.

Entrada `Tp1Ej1.in`
Cada ramal ocupa dos líneas, la primera contiene un entero con los kilometros de cable dedicados al ramal y la segunda los kilometrajes de las estaciones en el ramal sin cosiderar el 0.

Salida `Tp1Ej1.out`
Para cada ramal de entrada, se debe indicar una línea con la cantidad de ciudades conectables encontradas.

Problema 2: A Medias

La mediana de un conjunto ordenado de n números se define como $x_{(n+1)/2}$ si n es impar, o como $(x_{n/2} + x_{n/2+1})/2$ si n es par. Dados n números enteros en cualquier orden se deben devolver otros n números, donde el i-ésimo de ellos represente la parte entera de la mediana de los primeros i números de la entrada.

Resolver en una complejidad estrictamente mejor que $O(n^2)$ donde n es el número total de enteros de entrada

Entrada `Tp1Ej2.in` y **Salida** `Tp1Ej2.out`
Tendrán una sucesión enteros separados por espacios a razón de una instancia del problema por línea respectivamente.

PISTA: Mantener luego de cada entero leído de la entrada el conjunto actual dividido en dos mitades, los más chicos y los más grandes. Considerar una estructura de datos eficiente para manipular esos conjuntos.

Problema 3: Girls Scouts

El capitán de Las Girasoles quiere evitar los problemas en el fogón del año anterior cuando las pequeñas exploradoras rompieron en llanto por no poder sentarse en la ronda junto a sus mejores amigas. En secreto les asignó una letra a cada niña y relevó quién se llevaba con quién.
Su idea es organizar la ronda de manera que exista la menor distancia posible entre cada amistad (Sí, mi querido computador minimizando la suma de las distancias entre todos los pares de amigas).

Resolver en una complejidad estrictamente mejor que $O(e^e a^2)$. Donde e es la cantidad de exploradoras en cada grupo, y a la cantidad de amistades.

Entrada `Tp1Ej3.in`
El archivo contiene una línea por cada grupo de exploradoras.
Cada línea se compone de una sucesión de amistades separadas por `;` de la forma `amistad[;amistad]`
Cada `amistad` es un par `x xs` donde `x` es una letra y `xs` una cadena de letras.

Salida `Tp1Ej3.out`
Para cada grupo de exploradoras, se debe indicar un nro entero correpondiente a la distancia máxima lograda, un espacio y la cadena de letras resultante. En caso de múltiples soluciones, dar la que esté primera alfabéticamente.