Trabajo Práctico 1

Fecha de entrega: domingo 16 de abril, hasta las 23:59 hs.
Fecha de reentrega: domingo 7 de mayo, hasta las 23:59 hs.

Este trabajo práctico consta de un problema con varios ejercicios. Para aprobar el mismo se requiere aprobar todos los marcados como obligatorios. Para promocionar se deben aprobar todos los ejercicios. De ser necesario, 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. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega.

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. Para esto se pide utilizar pseudocódigo y lenguaje coloquial combinando adecuadamente ambas herramientas (**¡sin usar código fuente!**). Se debe también justificar por qué el procedimiento desarrollado resuelve efectivamente el problema.
3. 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.
4. Dar un código fuente claro que implemente la solución propuesta. El mismo no sólo debe ser correcto sino que además debe seguir las *buenas prácticas de la programación* (comentarios pertinentes, nombres de variables apropiados, estilo de indentación coherente, modularización adecuada, etc.).
5. Realizar una experimentación computacional para medir la performance del programa implementado. Para ello se debe preparar un conjunto de casos de test que permitan observar los tiempos de ejecución en función de los parámetros de entrada. Deberán desarrollarse tanto experimentos con instancias aleatorias (detallando cómo fueron generadas) como experimentos con instancias particulares.

Respecto de las implementaciones, se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia. Además, debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación. **Se debe incluir un script o Makefile que compile un ejecutable que acepte como entrada lo solicitado en cada problema** La cátedra recomienda el uso de C++ o Java, y se sugiere consultar con los docentes la elección de otros lenguajes para la implementación. **Solo se permite utilizar 1 lenguaje para todo el TP.**

La entrada y salida de los programas **deberá hacerse por medio de la entrada y salida estándar del sistema**. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Deberá entregarse un informe impreso **con a lo sumo 10 paginas** que desarrolle los puntos mencionados. Por otro lado, 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 `algo3.dc@gmail.com` con el asunto *"TP 1: Apellido"*.

En este problema tenemos una tira de números. Queremos pintar cada uno de los números de rojo o azul. Al tener todo pintado, los números rojos deben formar una secuencia estricamente creciente, mientras que los números de azul deben formar una secuencia estricamente decreciente (por estricto decimos que no hay números consecutivos iguales). Puede pasar que no se pueda pintar alguno de los números, queremos minimizar la cantidad de números sin pintar.

Se pide lo siguiente:

1. Implementar un algoritmo de backtracking que resuelva el problema.
2. Agregar por lo menos una poda para mejorar la performance del algoritmo.
3. Implementar una algoritmo de programación dinamica que resuelva el problema.
4. *Implementar otro algoritmo de programación dinámica que resuelva el problema (esto puede ser utilizando una función distinta o bien implementando la misma función utilizando o sin usar recursión).*

Los primeros tres son necesarios para aprobar. Cada ejercicio debe incluir una forma para armar un ejecutable llamado *ejX* donde X es el número de ejercicio con la extensión que corresponda.

Formato de entrada: La entrada contiene exactamente dos lineas. La primera línea consiste en un entero N indicando la longitud de la tira de números. La segunda línea consiste en la tira de N números, $X_1 X_2 \dots X_N$.

Formato de salida: La salida consiste en un número, la cantidad de números sin pintar.

La complejidad temporal deberá ser a lo sumo $\mathcal{O}(N^3)$ para las implementaciones de progrmación dinámica.

Ejemplo 1:
Entrada:
8
0 7 1 2 2 1 5 0

Salida:
0
Explicación:
Rojo: 0 1 2 5
Azul: 7 2 1 0

Ejemplo 2:
Entrada:
12
3 11 0 1 3 5 2 4 1 0 9 3

Salida:
2
Explicación:
Rojo: 0 1 3 4 9
Azul: 11 5 2 1 0