Trabajo práctico 1

Fecha de entrega: viernes 5 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**. En caso de reentregar, la nota final del trabajo será el 20 % del puntaje otorgado en la primera corrección más el 80 % del puntaje obtenido al recuperar.

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. Es importante que lo expuesto en este punto sea suficiente para el desarrollo de los puntos subsiguientes, pero no excesivo (**¡no es un código fuente!**).
3. Justificar por qué el procedimiento desarrollado en el punto anterior resuelve efectivamente el problema y demostrar formalmente su correctitud.
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 explicite lo contrario.
5. 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.). Se deben incluir las partes relevantes del código como apéndice del informe impreso entregado.
6. 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 (de peor caso en tiempo de ejecución, por ejemplo). Se debe presentar **adecuadamente** en forma gráfica una comparación entre los tiempos medidos y la complejidad teórica calculada y extraer conclusiones de la experimentación.

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. La cátedra recomienda el uso de C++ o Java, y se sugiere consultar con los docentes la elección de otros lenguajes para la implementación.

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 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_1, ..., Apellido_n"*, donde n es la cantidad de integrantes del grupo y *Apellido_i* es el apellido del i-ésimo integrante.

Problema 1: Puentes sobre lava caliente

En una difícil competencia, los participantes tienen que atravesar una serie de puentes colgantes. Estos puentes constan de varios tablones en donde los participantes pueden pisar a fin de atravesar el puente saltando de tablón en tablón. Sin embargo, algunos de estos tablones están rotos y si algún participante pisa uno de ellos, éste caería a un río de lava que corre por debajo, perdiendo así probablemente la vida. Afortunadamente, estos tablones rotos se encuentran marcados y sus ubicaciones en los puentes se conocen de antemano. La única manera de pasar de un tablón a otro es saltando y cada participante tiene un límite en la cantidad de tablones que puede saltar de una sóla vez. El objetivo de esta competencia es atravesar los puentes dando la menor cantidad de saltos posible.

Escribir un algoritmo que dado un puente y la cantidad máxima de tablones que se pueden saltar de una sóla vez, indique si es posible atravesar el puente dando saltos y en caso afirmativo indique qué saltos deberían hacerse para atravesarlo dando la menor cantidad posible de saltos. Siendo n la cantidad de tablones del puente, se espera que el algoritmo tenga una complejidad temporal de $\mathbf{O}(n)$.

Formato de entrada: La entrada contiene varias instancias del problema. Cada instancia consta de una línea con el siguiente formato:

```
n c t1 t2 ... tn
```

donde `n` es la cantidad de tablones del puente (rotos o no), `c` es el salto más largo (medido en tablones) que puede hacer un participante y `t1`, ..., `tn` son `n` valores que describen el estado de cada uno de los `n` tablones, ordenados desde el comienzo hasta el final del puente; un valor de `0` indica un tablón sano y un valor de `1` indica un tablón roto. La entrada concluye con una línea comenzada por `0`, la cual no debe procesarse.

Formato de salida: La salida debe contener una línea por cada instancia de entrada. Si el puente en cuestión tiene solución, la línea debe tener el siguiente formato:

```
s t1 t2 ... ts
```

donde `s` es la cantidad de saltos realizados y los valores `t1`, ... , `ts` son los tablones hacia los cuales se realiza cada uno de los `s` saltos, por orden de realización de los saltos. Los tablones del puente se numeran de 1 a `n` y se asume que el participante comienza fuera del puente y debe terminar también fuera del mismo. Por este motivo, el valor de `ts` debe ser un valor mayor que `n`, indicando que el participante queda fuera del puente. Si hay más de una solución óptima, el programa puede devolver cualquiera de ellas. En caso de que el puente no tenga solución, la línea de salida deberá tener simplemente la palabra `no`.

Problema 2: Horizontes lejanos

Estamos diseñando un software de arquitectura (para competir con el famoso AutoCAD®) y como parte del desarrollo nos han encargado escribir un algoritmo para el módulo de *Edificios 2D*. Este módulo representa los edificios de una ciudad en un plano bidimensional como un conjunto de rectángulos apoyados sobre una base común (ver dibujo izquierdo de la Figura 1).

Dado un conjunto de edificios, descriptos como se mencionó, el desafío consiste en eliminar las líneas ocultas dibujando únicamente el perfil definido en el horizonte. En la Figura 1 se puede ver una instancia ejemplo de este problema y el correspondiente perfil solución de la misma.

Escribir un algoritmo que dado un conjunto de edificios calcule y devuelva el perfil definido en el horizonte por estos edificios. Siendo n la cantidad de edificios en la entrada, el algoritmo deberá tener una complejidad temporal **estrictamente mejor** que $\mathbf{O}(n^2)$.

Formato de entrada: La entrada contiene varias instancias del problema. Cada instancia comienza con una línea que contiene un único valor entero no negativo `n` que representa la cantidad de edificios de la instancia. A esta línea le siguen `n` líneas describiendo cada uno de los edificios. Para cada edificio la línea correspondiente tiene el siguiente formato:

```
izq alt der
```

donde `izq` y `der` son las coordenadas de las paredes izquierda y derecha del edificio, respectivamente, y

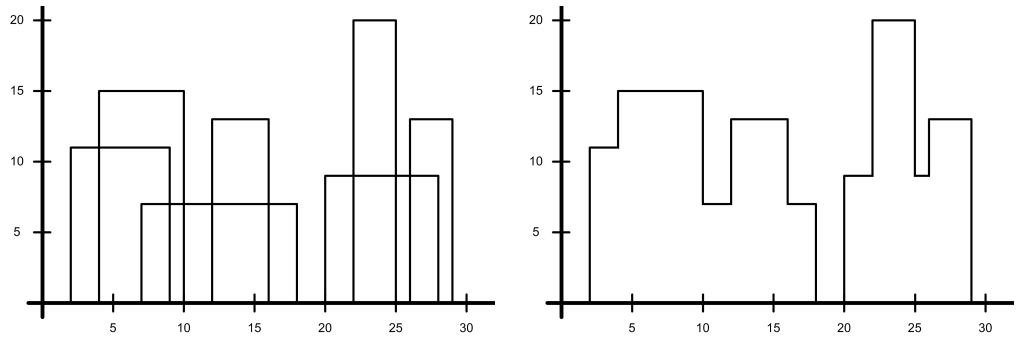

Figura 1: Un ejemplo de entrada (izquierda) y su solución (derecha).

`alt` es el alto del edificio. Los tres valores son enteros no negativos y se puede asumir que `izq` $<$ `der`. La entrada concluye con una línea comenzada por `0`, la cual no debe procesarse.

Formato de salida: La salida debe contener una línea por cada instancia de entrada. Dicha línea deberá ser una sucesión de pares de valores (x, y) representando cada cambio de altura en el perfil solución, donde x representa la coordenada horizontal en donde se produce el cambio de altura e y representa la nueva altura a partir de x. Los pares deben escribirse ordenados según la coordenada horizontal y todos los valores deben estar separados por un espacio (no escribir paréntesis ni comas). Por ejemplo, la salida correspondiente al ejemplo de la Figura 1 podría ser la siguiente:

```
2 11 4 15 10 7 12 13 16 7 18 0 20 9 22 20 25 9 26 13 29 0
```

Problema 3: Biohazard

Una empresa transportadora de productos químicos necesita transportar n productos desde una fábrica hacia un depósito seguro. La empresa transporta sus productos en camiones especiales diseñados para esta tarea. Cualquier producto puede ser transportado en cualquiera de los camiones pero ciertos pares de productos presentan un nivel de peligrosidad si son transportados en el mismo camión. Para cada par de productos p_i y p_j, se conoce de antemano el coeficiente de peligrosidad $h_{i,j}$ que conlleva transportar dichos productos en el mismo camión.

La cantidad de camiones que la empresa puede contratar para esta tarea no es un limitante, pero el objetivo de la empresa es utilizar la menor cantidad de camiones posible. Sin embargo, para cumplir con las normas vigentes de seguridad, el nivel de peligrosidad de las cargas transportadas en cada camión no puede superar un cierto umbral M. El nivel de peligrosidad de un camión se mide como la suma de las peligrosidades por pares de los items transportados. Es decir, si P es el conjunto de productos transportados en un camión, el nivel de peligrosidad de este conjunto es

$$h(P) = \sum_{\substack{p_i, p_j \in P \\ i<j}} h_{i,j}.$$

Escribir un algoritmo que dado un conjunto de productos y sus coeficientes de peligrosidad, indique en qué camión debe transportarse cada uno de los productos de manera de utilizar la menor cantidad posible de camiones, asegurando que el nivel de peligrosidad de cada camión no exceda un cierto umbral M. Se pide utilizar la técnica de *Backtracking* y elaborar podas y estrategias para mejorar los tiempos de ejecución. Estas podas deberán estar apropiadamente documentadas en el informe. El item 3 del enunciado (demostración formal de correctitud) **es opcional** para este problema.

Formato de entrada: La entrada contiene varias instancias del problema. La primera línea de cada instancia contiene un entero positivo n el cual indica la cantidad de productos a transportar (estos son $p_1, \ldots, p_n$) y un entero no negativo M (separados con un espacio) que indica el máximo nivel de peligrosidad permitido para un camión. A esta línea le siguen $n-1$ líneas indicando los coeficientes de peligrosidad $h_{i,j}$ de cada par de productos (valores enteros no negativos). La i-ésima línea consta de los siguientes valores (separados por espacios):

$h_{i,(i+1)}$ ... $h_{i,n}$

donde $h_{i,j}$ es el coeficiente de peligrosidad entre los productos p_i y p_j. Sólo se listan en la entrada los coeficientes $h_{i,j}$ con $i < j$, ya que la peligrosidad es una propiedad simétrica. La entrada concluye con una línea comenzada por 0, la cual no debe procesarse.

Formato de salida: La salida debe contener una línea por cada instancia de entrada, con el siguiente formato:

```
C c1 c2 ... cn
```

donde `C` es la cantidad de camiones utilizados en la solución obtenida y `c1`, ..., `cn` son los camiones utilizados para cada uno de los n productos de la instancia. Es decir, `ci` es el camión en el cual se transportará el producto p_i (los camiones se identifican con índices entre 1 y `C`).