Trabajo práctico 2

Fecha de entrega: 12 de octubre, hasta las 23:59 horas.
Fecha de reentrega: 23 de noviembre, hasta las 23:59.

Este trabajo práctico consta de varios problemas y para aprobar el trabajo se requiere aprobar todos los problemas. El trabajo práctico deberá resolverse en grupo de 4 personas. 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 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 sugiere 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.
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* (nombres de variables apropiados, estilo de indentación coherente, modularización adecuada, etc.). Se deben incluir las partes relevantes del código como apéndice del informe entregado.
6. Realizar una experimentación computacional para medir la performance del programa implementado. Para ello se debe preparar un conjunto de casos que permitan observar los tiempos de ejecución en función de los parámetros de entrada que sean relevantes. 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). Presentar **adecuadamente** en forma gráfica una comparación entre los tiempos medidos 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. **Se debe incluir un script o Makefile que compile un ejecutable que acepte como entrada lo solicitado en el 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 resolver los ejercicios del TP.** Se pueden utilizar otros lenguajes para presentar resultados.

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 2: 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: Impresiones ordenadas

Una imprenta cuenta con dos máquinas muy particulares en las cuales se realizan tareas de impresión a gran escala. Los trabajos realizados en estas máquinas pueden ser muy distintos entre sí y para realizar un trabajo en una de las máquinas hay que preparar la misma para ello. Esta preparación tiene un costo ya que hay que cargar las tintas correspondientes, cambiar los rodillos, etc. Más aun, el costo de preparar la máquina para realizar un determinado trabajo depende de cuál haya sido el último trabajo realizado en la máquina (ya que las tintas y rodillos pueden reutilizarse de un trabajo a otro o descartarse, dependiendo de qué tan distintos son los trabajos a realizar).

Todos los días, la empresa debe realizar una serie de trabajos $t_1, \ldots, t_n$ y los mismos tienen que realizarse en el orden en que vienen dados, sin excepción. Cada trabajo puede asignarse a cualquiera de las dos máquinas disponibles, siempre que se respete el orden de producción dado. Afortunadamente, se conoce el costo de preparar una máquina para realizar el trabajo t_i luego de haber realizado el trabajo t_j, para cada par de trabajos t_i y t_j, con $j < i$. También se conoce el costo inicial de preparar una máquina para cada uno de los trabajos, es decir, el costo de preparación si el trabajo es el primero realizado en la máquina.

Se pide escribir un algoritmo que tome la información de los trabajos a realizar e indique en qué máquina se debe realizar cada trabajo para minimizar la suma de los costos de preparación de los mismos. Siendo n la cantidad de trabajos a realizar, se espera que el algoritmo implementado tenga una complejidad temporal de peor caso $\mathbf{O}(n^2)$. En caso de haber más de una solución óptima para el problema, el algoritmo puede devolver cualquiera de ellas.

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 trabajos a organizar (estos son $t_1, \ldots, t_n$). A esta línea le siguen n líneas y cada una de ellas indica los costos de preparación de cada trabajo (valores enteros no negativos). La i-ésima línea consta de los siguientes valores (separados por un espacio):

$$c_{0i} \quad c_{1i} \quad \ldots \quad c_{(i-1)i}$$

donde c_{0i} es el costo de preparación de una máquina para t_i si t_i es el primero trabajo a realizar en esa máquina y c_{ji} es el costo de preparación de una máquina para t_i si t_j es el último trabajo realizado en la máquina. 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. Esta línea debe contener los siguientes valores (separados por un espacio):

$$C \quad k \quad i_1 \quad \ldots \quad i_k$$

donde C es el costo total de la solución, k es la cantidad de trabajos asociados a una de las máquinas (cualquiera de ellas) y los valores $i_1, \ldots, i_k$ son los índices de los trabajos asignados a esta máquina.

Problema 2: Replicación de contenido

Una organización debe implementar una solución de replicación (o sincronización) de contenido para su red de entrega de contenidos[1] en Internet. Dispone de n servidores interconectados mediante m enlaces *backbone* de alta velocidad. Uno de los servidores será seleccionado como *master* y será el primero en recibir los nuevos datos a replicar. De ahí la información será transmitida a algunos otros servidores, que guardarán su propia copia y a su vez retransmitirán inmediatamente a otros servidores. Este proceso de "*broadcast*" se repite hasta que los n servidores tengan la nueva información. El uso de los enlaces *backbone* tiene un costo en función del tráfico que transmiten. Todos los enlaces transmitirán la misma información, ya que son copias idénticas. Los n servidores están interconectados por estos enlaces.

La organización quiere minimizar sus costos de enlaces y el tiempo de replicación. Para ello encargó el trabajo de optimización a dos consultoras.

1. Una será encargada de elegir qué enlaces se deberán utilizar de manera que la información pueda

[1] `http://en.wikipedia.org/wiki/Content_delivery_network`

distribuirse a todos los servidores a un costo mínimo, con el mecanismo descripto anteriormente.

2. La segunda consultora definirá cuál servidor seleccionar como *master* de manera que la replicación termine en el menor tiempo posible, suponiendo que la información tarda el mismo tiempo en atravesar cualquier enlace. La replicación inicia una vez que el *master* tiene la información, y termina cuando todos los servidores tienen su propia copia. Suponer que todos los tiempos dentro de un servidor son instantáneos (copia interna, copia a cada enlace, etc.). Cada servidor transmite simultáneamente a todos los vecinos elegidos.

Se pide escribir dos algoritmos, uno para cada consultora, que resuelvan el problema correspondiente. Para el subproblema 1 se espera una solución con una complejidad de peor caso **estrictamente menor** que $\mathbf{O}(n^3)$. Para el subproblema 2 se espera una solución con una complejidad de peor caso de $\mathbf{O}(n)$. Los algoritmos son independientes: la solución del primer algoritmo (subproblema 1) será usada como entrada para el segundo algoritmo. Siempre que haya más de una solución óptima el algoritmo puede elegir cualquiera de ellas.

Formato de entrada: La entrada contiene varias instancias del problema. La primera línea de cada instancia contiene un entero positivo n que indica la cantidad de servidores (numerados de 1 a n), un espacio, y un entero no negativo m que corresponde a la cantidad de enlaces. A esta línea le siguen m líneas con la descripción de cada enlace (números separados por un espacio):

$a_j \ b_j \ c_j$

donde a_j y b_j son los servidores que el enlace j conecta, y c_j es el costo de usar el enlace j (entero no negativo). Todos los enlaces son bidireccionales y el costo es el mismo en ambos sentidos. 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. Esta línea debe contener los siguientes valores (separados por un espacio):

$C \ U \ k \ e_1^1 \ e_2^1 \ e_1^2 \ e_2^2 \ \ldots \ e_1^k \ e_2^k$

donde C es el costo total de la solución, U es el servidor elegido como *master*, k es la cantidad de enlaces y e_1^i y e_2^i son los extremos del i-ésimo enlace de la solución obtenida, para $i \in [1, k]$.

Problema 3: Transportes pesados

Una empresa que fabrica ladrillos tiene sus fábricas emplazadas en varios puntos de una provincia. Los clientes de esta empresa están distribuídos en distintas ciudades de la provincia. La empresa transporta los ladrillos desde las fábricas hasta sus clientes utilizando pesados camiones. Lamentablemente las rutas de esta provincia no están preparadas para soportar el paso de estos camiones y desde hace ya un tiempo la empresa está teniendo problemas con la gobernación de la provincia por este asunto. Si bien es posible fortalecer las rutas para que no haya problemas, eso implica una gran inversión que ninguna de las partes está dispuesta a afrontar.

Hace unos días, el gobernador dio un *ultimatum* a la empresa y le exigió que se haga cargo de fortalecer las rutas que utiliza para el transporte. A partir de una cierta fecha, la empresa sólo podrá utilizar rutas que hayan sido debidamente fortalecidas para el transporte pesado.

Cada ruta tiene un costo de inversión proporcional a la longitud de la misma. Por lo tanto, antes de afrontar esta inversión, la empresa quiere replanificar sus rutas habituales de manera tal de minimizar los gastos que implicará el fortalecimiento de las rutas a utilizar. La empresa puede satisfacer la demanda de un cliente desde cualquiera de sus fábricas, siempre y cuando haya un camino fortalecido que lo lleve desde la fábrica al cliente.

Se pide escribir un algoritmo que tome la información sobre las fábricas, clientes y rutas entre las ciudades e indique un plan óptimo de fortalecimiento de rutas. El objetivo es indicar qué rutas deberían fortalecerse de manera tal que cada cliente pueda ser provisto desde al menos una de las fábricas. La solución provista debe ser mínima en cuanto al gasto de inversión asociado. Se sabe que desde todas las fábricas sale al menos una ruta, que es posible satisfacer la demanda de cada cliente, y que no hay más fábricas que clientes. Siendo R la cantidad de rutas de la provincia y C la cantidad de clientes, se sabe que el problema puede

resolverse con una complejidad de $\mathbf{O}(C^2)$ o bien de $\mathbf{O}(R \cdot log(C))$ y se espera que la solución implementada no sea peor que alguna de estas complejidades. En caso de haber más de una solución óptima para el problema, el algoritmo puede devolver cualquiera de ellas.

Formato de entrada: La entrada contiene varias instancias del problema. La primera línea de cada instancia contiene los siguientes valores (enteros positivos, separados por un espacio):

$$F \;\; C \;\; R$$

donde F indica la cantidad de fábricas de la empresa (numeradas de 1 a F), C indica la cantidad de clientes (numerados de $F+1$ a $F+C$) y R indica la cantidad de rutas de la provincia. A esta línea le siguen R líneas y cada una de ellas describe una de las rutas de la provincia. Para cada ruta, la línea correspondiente contiene los siguientes valores (enteros positivos, separados por un espacio):

$$e_1 \;\; e_2 \;\; l$$

donde e_1 y e_2 son los extremos de la ruta y l es la longitud de la misma. Los extremos de la ruta pueden ser fábricas y/o clientes y son números enteros en el rango $[1, F+C]$. 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. Esta línea debe contener los siguientes valores (separados por un espacio):

$$L \;\; k \;\; e_1^1 \;\; e_2^1 \;\; e_1^2 \;\; e_2^2 \;\; \ldots \;\; e_1^k \;\; e_2^k$$

donde L es el costo total de la solución (la suma de las longitudes de las rutas utilizadas), k es la cantidad de rutas utilizadas en la solución y e_1^i y e_2^i son los extremos de la i-ésima ruta utilizada, para $i \in [1, k]$.