Trabajo Práctico 2: Algoritmos sobre grafos

Compilado: 6 de mayo de 2022

Fecha de entrega: 8 de junio de 2022
Fecha de reentrega: 27 de junio de 2022.

Este trabajo práctico consta de cuatro ejercicios independientes. Para la aprobación del trabajo práctico se debe entregar:

1. Un informe *de a lo sumo dieciseis páginas* que describa la solución de los ejercicios y responda a todas las consignas del enunciado.

2. El código fuente con los programas que implementan las soluciones propuestas a los ejercicios.

3. Un conjunto representativo de instancias de prueba.

Informe. El informe debe ser autocontenido, lo que significa que cualquier estudiante potencial de AED3 que conozca los temas debe poder leerlo sin necesidad de conocer cuál fue el enunciado. En particular, todos los conceptos y notaciones utilizados que no sean comunes a la materia deben definirse, incluso aquellos que se encuentren en el presente enunciado. No es necesario explicar aquellos conceptos propios de la materia ni la teoría detrás de las distintas técnicas algorítmicas o de demostración de propiedades.

El informe debe estar estructurado usando una sección independiente por cada ejercicio resuelto. Esto no impide que un ejercicio haga referencias a la solución de otro ejercicio; las comparaciones son muchas veces bienvenidas o parte de la consigna. Cada sección debe dar una respuesta cabal a todas las preguntas del enunciado. Sin embargo, el informe no es una sucesión de respuestas a las distintas consignas, sino una elaboración única y coherente donde se pueden encontrar las respuestas a las preguntas. La idea es que el informe sea de agradable lectura.

El informe debe incluir las justificaciones necesarias para determinar la corrección de cada algoritmo. Más aún, la implementación de cada algoritmo debe describirse con suficiente detalle como para que cualquier estudiante de AED3 pueda replicarla. Esto último no significa que la descripción de la implementación deba darse hasta el último detalle, sino que tiene que tener un nivel de discurso acorde a la materia.

El informe **no debe incluir código fuente**, ya que quienes evaluamos tenemos acceso al código fuente del programa. En caso de utilizar *pseudocódigo*, el mismo debe ser de alto nivel, evitando construcciones innecesariamente complicadas para problemas simples.[1] Todo psedocódigo debe estar acompañado de un texto coloquial que explique lo que el pseudocódigo hace en términos conceptuales.

El informe no debe superar las dieciseis (16) páginas contando las tablas, aunque puede contener un apéndice con tablas de resultados adicionales, en caso de ser necesario.

Código fuente. El código fuente entregado debe venir acompañado de un documento que explique cómo compilar y ejecutar el programa, usando unicamente herramientas que suelen encontrarse en una distribución moderna, pero estable, de Linux. El código fuente debe compilar y debe resolver correctamente **todos** los casos de test en un tiempo que se corresponda a la complejidad esperada. En

[1] E.g., "tomar el máximo del vector V" vs. "poner max := V_0; para cada $i = 1, \ldots, n-1$: poner max := maximo(max, V_i);"

todos los ejercicios, la instancia se leerá de la entrada estándar y la solución se imprimirá en la salida estandar. La complejidad temporal en peor caso de cada algoritmo forma parte del enunciado. Tener en cuenta que el programa puede ser probado en casos de test adicionales a los provistos en la solución.

Casos de test. Además del informe y el código, se debe entregar un conjunto de instancias de prueba para cada ejercicio, las cuales deben respetar el formato indicado en el enunciado. El conjunto de instancias debe ser representativo para el algoritmo en cuestion, conteniendo instancias de tamaños diversos y que cubran distintas situaciones en caso que el problema lo amerite.

El informe debe contener una descripción de las instancias de prueba, que indique cómo fueron generadas o qué caracteristicas poseen. En caso de ser instancias generadas por computadora, describir someramente el método sin entrar en demasiado detalle y sin incluir una descripción del algoritmo. Asimismo, el cuerpo del informe debe incluir un resumen de las corridas en el cual se aprecie la complejidad temporal en función de los parámetros de entrada. En algún lugar, posiblemente el apéndice, el informe debe contener una tabla con el valor óptimo de cada instancia y el tiempo necesario para encontrarlo.

Las instancias pueden ser tomadas de repositorios públicos (e.g., imágenes en formato bitmap en blanco y negro, grafos de rutas, grafos de redes sociales, etc.). También pueden compartirse las instancias entre los grupos, incluso aquellas que sean de creación propia de un grupo. En todos los casos debe indicarse cómo se generaron las instancias y el por qué de su incusión en el conjunto de prueba.

Aprobación. Para aprobar el trabajo práctico es necesario aprobar cada ejercicio en forma individual, ya sea en la primera entrega o en el recuperatorio. No es necesario reentregar aquellos ejercicios que sean aprobados en la primer entrega. Asimismo, para aprobar cada ejercicio es necesario que el informe describa correctamente la solución y que el programa propuesto sea correcto. La correcta escritura del informe forma parte de la evaluación.

Formato de un (di)grafo (pesado) con lista de aristas. En este trabajo usamos siempre el mismo formato para representar a los (di)grafos (pesados) que forman parte de la instancia de un problema. Sea G un (di)grafo (pesado). La representación de G empieza con una línea con los valores n y m que denotan la cantidad de vértices y aristas de G. Luego, aparecen m líneas que representan las aristas de G. Cada arista se representa con dos números v y w entre 0 y $n-1$ que identifican a los vértices. En el caso en que G es pesado, cada línea contiene un tercer valor p que denota el peso de la arista. Finalmente, en el caso en que G es dirigido, v es la cola de la arista y w su cabeza.

Ejercicio 1

Un grafo es *geodésico* si para todo par de vértices existe un único camino de longitud mínima entre ellos. Diseñar e implementar un algoritmo de tiempo $O(nm)$ para determinar si un grafo conexo es geodésico.

Descripción de una instancia. Un grafo conexo G en formato lista de aristas.

Descripción de la salida. Si G es geódesico, una línea con el valor 1 seguido de n líneas: la i-ésima línea tiene en la posición j al predecesor del j-ésimo vértice en el único camino de i hacia j (si $i = j$,

la posición j tiene el valor i). En caso contrario, una línea con el valor 0 seguida de dos líneas listando los vértices de dos caminos mínimos de G entre un par de vértices.

Entrada de ejemplo	Salida esperada de ejemplo
4 4 0 1 0 2 1 2 0 3	1 0 0 0 0 1 1 1 0 2 2 2 0 3 0 0 3

Entrada de ejemplo	Salida esperada de ejemplo
4 5 0 1 0 2 1 2 0 3 1 3	0 2 0 3 2 1 3

Ejercicio 2

El grafo $G(M)$ asociado a una matriz $M \in \{0,1\}^{m \times n}$ tiene un vértice por cada posición de M que tiene valor 1, donde dos vértices de $G(M)$ son adyacentes si y solo si sus posiciones son adyacentes en M. Diseñar un algoritmo que, dada una matriz M, determine la cantidad de componentes conexas de $G(M)$. El algoritmo debe consumir espacio $O(n)$[2] y tiempo $O(mn\alpha^{-1}(nm))$ donde α es la función de Ackerman.

Descripción de una instancia. La instancia empieza con una línea con los valores m y n que representan la cantidad de filas y columnas de la matriz M. Luego sigen m líneas, cada una de las cuales tiene n valores en $\{0,1\}$. El j-ésimo valor de la i-ésima fila se corresponde a $M_{i,j}$.

Descripción de la salida. Una línea con la cantidad de componentes conexas de $G(M)$.

Entrada de ejemplo	Salida esperada de ejemplo
4 10 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 1 1	3

Ejercicio 3

Implementar el algoritmo de Johnson [3] (ver [2, Sección 25.3] para una explicación de libro de texto) para resolver el problema de camino mínimo entre todos los pares de vértices de un digrafo conexo y pesado. El algoritmo debe consumir $O(nm \log n)$ tiempo y $O(m)$ espacio.

[2] Sin contar el espacio que ocupa la matriz M en el disco. Solo contar lo que se consume de memoria. Notar que M no se puede cargar enteramente en memoria, sino que solo se puede leer una cantidad constante de filas en simultáneo.

Descripción de una instancia. Un digrafo pesado G en formato de lista de aristas.

Descripción de la salida. Si G tiene un ciclo negativo, imprimir una línea con el valor 0 seguida de un línea con un ciclo de longitud negativa. En caso contrario, imprimir una línea con el valor 1 seguida de n líneas con n valores cada una: el j-ésimo valor de la i-esima fila tiene la distancia del vértice i al vértice j. Si no existe un camino de i a j, imprimir INF en el lugar correspondiente.

Entrada de ejemplo	Salida esperada de ejemplo
2 2 0 1 1 1 0 -2	0 0 1 0

Entrada de ejemplo	Salida esperada de ejemplo
2 1 0 1 1	1 0 1 INF 0

Entrada de ejemplo	Salida esperada de ejemplo
4 5 0 1 3 1 2 2 2 3 6 3 0 -1 1 3 -1	1 0 3 5 2 -2 0 2 -1 5 8 0 6 -1 2 4 0

Ejercicio 4

Sea $\mathcal{I}$ una familia de intervalos sobre la recta real. Decimos que $\mathcal{D} \subseteq \mathcal{I}$ es un *conjunto dominante total* (CDT) de $\mathcal{I}$ si para todo $I \in \mathcal{I}$ existe $I' \in \mathcal{D}$ tal que $I \neq I'$ e $I \cap I' \neq \emptyset$. Implementar el algoritmo de Bertossi [1] para encontrar un CDT de $\mathcal{I}$ que sea de cardinalidad mínima. El algoritmo debe consumir tiempo $O(n^2)$ donde n es la cantidad de intervalos en $\mathcal{I}$.

Descripción de una instancia. La instancia empieza con una línea con un único valor n que indica la cantidad de intervalos en $\mathcal{I}$. Luego siguen n líneas, cada una con dos valores s y t que indican el inicio y fin del i-ésimo intervalo de $\mathcal{I}$. Para todo intervalo $I \in \mathcal{I}$ existe al menos un intervalo $I' \in \mathcal{I}$ distinto a I con $I \cap I' \neq \emptyset$. Más aún, $\bigcup \mathcal{I}$ es un intervalo incluido en $[0, 2n)$ y todos los extremos son distintos.

Descripcion de la salida. Una línea con un valor k que indica el tamaño de un CDT de cardinalidad mínima, seguida de una línea con k valores que denotan los índices de los intervalos que forman un CDT de cardinalidad mínima.

Entrada de ejemplo	Salida esperada de ejemplo
5 0 6 5 8 1 3 2 4 7 9	2 0 1

Entrada de ejemplo	Salida esperada de ejemplo
5 0 9 1 2 3 4 5 6 7 8	2 0 3

Referencias

[1] Alan A. Bertossi. Total domination in interval graphs. *Inform. Process. Lett.*, 23(3):131–134, 1986.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[3] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. Assoc. Comput. Mach.*, 24(1):1–13, 1977.