Trabajo Práctico 3: Camino mínimo y Flujo máximo

Fecha de entrega: 20 de junio de 2023 (antes de las 23:59)
Fecha de reentrega: 5 de julio de 2023 (antes de las 23:59)

Este trabajo práctico consta de tres ejercicios y se realiza en grupos de dos personas. Para su aprobación se debe entregar:

1. Un informe *de a lo sumo 5 páginas* que desarrolle los incisos detallados en el enunciado del **primer** ejercicio.
2. El código fuente con los programas que implementan las soluciones propuestas a los ejercicios junto con las instrucciones para compilar.

El informe debe ser autocontenido, lo que significa que cualquier estudiante potencial de AED3 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. La idea es que el informe sea de agradable lectura.

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 pseudocódigo debe estar acompañado de un texto coloquial que explique lo que el pseudocódigo hace en términos conceptuales.

El código fuente entregado debe venir acompañado de un documento que explique cómo compilar y ejecutar el programa. El código fuente debe compilar y debe resolver correctamente **todos** los casos de test que se le presenten en un tiempo que se corresponda a la complejidad esperada. En todos los ejercicios, la instancia se leerá de la entrada estándar y la solución se imprimirá en la salida estandar. Junto a este enunciado se incluyen algunos casos de test, mientras que la complejidad temporal en peor caso, de ser solicitada, forma parte del enunciado. Tener en cuenta que el programa puede ser probado en casos de test adicionales. Esta **terminantemente prohibida** la copia de código ya sea de otro grupo, o de internet. La detección de plagio puede resultar en desprobación de la materia.

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. La correcta escritura del informe forma parte de la evaluación.

Formato de entrega La entrega se realizará a travez del campus. Se deberá entregar el pdf y una carpeta comprimida con el código de la solución de los tres problemas (sin los archivos ejecutables). El nombre de estos archivos debe ser el de los apellidos de las personas integrantes del grupo separados por guiones.

[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);

Ejercicio 1: Mejorando el tráfico

El mapa de una ciudad consiste en n puntos numerados del 1 al n, y m calles unidireccionales que conectan pares de puntos. Con el fin de reducir la longitud del camino más corto entre dos puntos críticos diferentes s y t, se propone una lista de k calles bidireccionales como candidatas a ser construidas.

Se debe escribir un programa para elegir una calle bidireccional de la lista propuesta con el fin de minimizar la longitud resultante del camino más corto entre s y t.

Descripción de una instancia

El archivo de entrada consiste en varios casos de prueba. La primera línea del archivo de entrada contiene el número de casos, que es un entero positivo y no mayor que 20. Las siguientes líneas describen los casos de prueba.

Para cada conjunto de datos, la primera línea contiene cinco enteros positivos n ($n \leq 10^4$), m ($m \leq 10^5$), k ($k < 300$), s ($1 \leq s \leq n$), t ($1 \leq t \leq n$) separados por espacios.

Luego siguen m líneas. La i-ésima línea de estas contiene tres enteros d_i, c_i, l_i separados por espacios, que representan la longitud l_i ($0 < l_i \leq 1000$) de la i-ésima calle unidireccional que conecta el punto d_i con el punto c_i.

Luego siguen k líneas. La j-ésima línea de estas contiene tres enteros positivos u_j, v_j y q_j ($q_j \leq 1000$) separados por espacios, que representan la j-ésima calle bidireccional propuesta de longitud q_j que conecta el punto u_j con el punto v_j.

Descripción de la salida

Para cada caso, escribir en una línea la longitud más pequeña posible del camino más corto después de construir la calle bidireccional elegida de la lista propuesta. En caso de que no exista un camino desde s hasta t, se deberá imprimir -1.

Ejemplo de entrada y salida

Se presenta un ejemplo de entrada y su correspondiente salida:

Entrada de ejemplo	Salida esperada de ejemplo
1 4 5 3 1 4 1 2 13 2 3 19 3 1 25 3 4 17 4 1 18 1 3 23 2 3 5 2 4 25	35

Se pide:

1. Diseñar un algoritmo eficiente utilizando un algoritmo de camino mínimo para resolver el problema. El mismo debe resolver los casos de prueba del juez en el límite de tiempo estipulado.

2. Presentar un informe de hasta **5 páginas** conteniendo:

   - Una presentación y descripción del problema.
   - Una explicación del algoritmo desarrollado.
   - Una justificación de su correctitud.
   - Un estudio y una comparación empírica de la performance del algoritmo para distintas implementaciones del algoritmo. Esto puede ser utilizando distintos algoritmos o bien usando distintas estructuras de datos que cambian la complejidad temporal del algoritmo. Puntualmente, se debe:
     + Indicar y justificar cuál es teóricamente la mejor implementación para este problema.
     + Encontrar y generar casos de test donde se pueda ver que cada implementación sirve para distintas situaciones.

Ejercicio 2: La grieta

Vivimos en una sociedad muy polarizada, polarizada por la política, el deporte, la gastronomía e incluso se divide el mundo por la mejor mascota.

Los estudiantes de Algoritmos y Estructuras de Datos 3 también se dividen en dos grupos: los que creen que el algoritmo de Prim es el más razonable para encontrar un árbol generador mínimo, y los que creen que Kruskal lo supera ampliamente.

Por lo tanto, se ha decidido realizar una votación para definir cuál es el mejor algoritmo.

Entre los estudiantes de la materia existen relaciones de amistad. Algunos estudiantes están dispuestos a cambiar su preferencia de algoritmo si ven que se generan tensiones con sus amistades.

Decimos que si un estudiante vota en contra de su algoritmo preferido, los docentes percibirán una disconformidad. Además, también se percibe una disconformidad cuando hay relaciones tensas, es decir, cuando cada estudiante de una amistad vota por algoritmos diferentes.

Lo que queremos saber es cuál es el valor mínimo de disconformidad, es decir, la suma mínima de la cantidad de estudiantes que votan en contra de su preferencia, más la cantidad de relaciones de amistad tensas, que se puede alcanzar considerando todas las formas posibles de votar de los estudiantes.

Descripción de una instancia

La entrada va a contener múltiples casos de prueba.

Cada caso comienza con una línea que contiene dos enteros n y m, donde $2 \leq n \leq 300$ es la cantidad de estudiantes y m es la cantidad de relaciones de amistad.

Luego sigue una línea con n enteros. El i-ésimo entero será 1 si el estudiante cree que Prim es el mejor algoritmo, y 0 si cree que Kruskal lo es.

Finalmente, hay m líneas que contienen dos enteros i y j, los cuales indican que el estudiante i y el estudiante j tienen una relación de amistad. Toda pareja de estudiantes aparece a lo sumo una vez.

La entrada finaliza cuando n y m son 0.

Descripción de la salida

Para cada caso, se debe imprimir una sola línea que contenga el valor mínimo posible de la suma de todos los conflictos entre amigos, sumado a todos los estudiantes que votaron en contra de su algoritmo preferido.

Ejemplo de entrada y salida Se presenta un ejemplo de entrada y su correspondiente salida.

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

Se pide:

1. Diseñar un algoritmo eficiente para resolver este problema. Este algoritmo debe resolver los casos de prueba del juez en el límite de tiempo estipulado.