ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2023-10. Proyecto – PARTE 2
Entrega: viernes, mayo 5 de 2023 6:00PM (por Bloque Neon)

0 OBJETIVOS

- Diseñar soluciones computacionales para problemas.
- Estimar costos de las soluciones planteadas.
- Implementar soluciones.

Se premiarán las mejores soluciones y se castigarán las peores, en cuanto a eficiencia en tiempo y espacio.

1 CONDICIONES GENERALES

El proyecto se divide en tres partes independientes entre sí. Este documento describe la PARTE II. Cada parte contiene un problema a resolver mediante soluciones implementadas en `Java` o `Python`.

Para cada problema se pide:
- Descripción de la solución.
- Análisis temporal y espacial.
- Una implementación en Java o Python

2 DESCRIPCIÓN DEL PROBLEMA

Redes con redundancia

A un estudiante de ingeniería de sistemas se le pide diseñar una red que conecte N computadores. Para conectar dos computadores se cuenta con dos posibles tecnologías, fibra óptica o cable coaxial. Sin importar la tecnología de conexión, para poder enviar un mensaje entre dos computadores arbitrarios A y B, no se requiere que estén conectados directamente. Es suficiente con que exista una cadena de conexiones que lleve el mensaje de A hacia B. Sin embargo, por la forma en que funciona la tecnología, el mensaje solamente llega bien si todas las conexiones intermedias tienen la misma tecnología.

Se dice que una red es redundante si para cualquier par de computadores A y B para los cuales se puede enviar un mensaje por fibra óptica, también se puede enviar el mensaje por cable coaxial y viceversa. Dado que las conexiones entre computadores se van a ir agregando a traves de un periodo de tiempo, se requiere saber cada vez que se agrega una conexión si la red es o no redundante.

Problema

Dada una lista de conexiones entre N computadores determinar si después de cada conexión la red es redundante. Cada conexión se representa por la tripla `(i,j,k)` donde $1 \leq i < j \leq N$ representan los computadores `i` y `j`, `k=1` significa conexión de fibra óptica y `k=2` significa conexión por cable coaxial.

Ejemplo:

Dado N=5 la siguiente tabla muestra una lista con la conexión que se agrega en cada momento del tiempo y si la red sería redundante:

Tiempo	**Conexión**	**Redundante?**
1	(1,2,1)	No
2	(2,3,1)	No
3	(1,3,2)	No
4	(2,3,2)	Si
5	(4,5,2)	No
6	(4,5,1)	Si

3 ENTRADA Y SALIDA DE DATOS

En todas las soluciones que se presenten, la lectura de los datos de entrada se hace por la entrada estándar; así mismo, la escritura de los resultados se hace por la salida estándar.

Puede suponer que ninguna línea de entrada tiene espacios al principio o al final, y que los datos que se listan en cada línea están separados por exactamente un espacio.

A continuación, se establecen parámetros que definen su tamaño y formato de lectura de los datos, tanto de entrada como de salida.

Descripción de la entrada

La primera línea de entrada especifica el número de casos de prueba que contiene el archivo. El programa debe terminar su ejecución, una vez termine de resolver la cantidad de casos de prueba dados por este número.

Cada caso está representado en una línea con la especificación de N y M siendo N la cantidad de computadores y M la cantidad de conexiones a realizar. Luego de esto hay M lineas en formato i j k donde cada linea representa la conexión entre el computador i y el computador j utilizando la tecnología k.

Tamaños de entrada: $2 \leq N \leq 10,000$, $1 \leq M \leq 100,000$

Descripción de la salida

Para cada caso de prueba, imprimir una secuencia de ceros y unos de tamaño M separada por espacio, donde 1 significa que la red es redundante y 0 significa que la red no es redundante.

Ejemplos de entrada / salida

Caso descrito arriba

Entrada	Salida
1 5 6 1 2 1 2 3 1 1 3 2 2 3 2 4 5 2 4 5 1	0 0 0 1 0 1

Ejemplo con tres casos de prueba:

Entrada	Salida
3 5 6 1 2 1 2 3 1 1 3 2 2 3 2 4 5 2 4 5 1 2 2 1 2 1 1 2 2 3 3 1 2 2 2 3 2 1 3 2 1 3 1	0 0 0 1 0 1 0 1 0 0 0 0

Nota: Se van a diseñar casos de prueba para valores de N y M mucho más grandes, siempre dentro de los valores establecidos en el enunciado. Los casos mostrados en este documento son demostrativos de la estructura de entrada/salida esperada.

4 COMPRENSIÓN DE PROBLEMAS ALGORITMICOS

A continuación, se presentan un conjunto de escenarios hipotéticos que cambian el problema original. Para cada escenario debe contestar[1]: (i) que nuevos retos presupone este nuevo escenario -si aplica-?, y (ii) que cambios -si aplica- le tendría que realizar a su solución para que se adapte a este nuevo escenario?

ESCENARIO 1: Suponga que se agregan conexiones inalambricas y ahora se considera una red como redundante si para cada par de computadores A y B entre los que se pueda enviar un mensaje, se pueda utilizar cualquiera de las tres tecnologías.

[1] NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

ESCENARIO 2: Suponga que ahora se quiere saber si se puede enviar un mensaje entre cualquier par de computadores, sin importar con qué tecnología se envíe el mensaje

Nota: Los escenarios son independientes entre sí.

5 ENTREGABLES

El proyecto puede desarrollarse por grupos de hasta tres estudiantes de la misma sección. La entrega se hace por bloque neon (una sola entrega por grupo de trabajo).

El grupo debe entregar, por bloque neon, un archivo de nombre `proyectoDalgoP2.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP2`, comprimida en formato `.zip`, dentro de la cual hay archivos fuente de soluciones propuestas y archivos que documentan cada una de las soluciones.

5.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* o en Python

Para el problema:

- Entregar un archivo de código fuente en *Java* (`.java`) o python (`.py`) con su código fuente de la solución que se presenta.
- Incluir como encabezado de cada archivo fuente un comentario que identifique el (los) autor(es) de la solución.
- Denominar `ProblemaP2.java` o `ProblemaP2.py` el archivo de la solución que se presente.

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* o Spyder durante el desarrollo del proyecto, la entrega requiere incluir solo un archivo por cada solución. El archivo debe poderse compilar y ejecutar independientemente (sin depender de ninguna estructura de directorios, librerías no estándar, etc.).

5.2 Archivos que documentan la solución propuesta

La solución al problema debe acompañarse de un archivo de máximo 3 páginas que la documente, con extensión `.pdf`. El nombre del archivo debe ser el mismo del código correspondiente (`ProblemaP2.pdf`).

Un archivo de documentación debe contener los siguientes elementos:

0 Identificación
Nombre de autor(es)
Identificación de autor(es)

1 Algoritmo de solución
Explicación del algoritmo elegido. Si hubo alternativas de implantación diferentes, explicar por qué se escogió la que se implementó. Se recomienda generar al menos una gráfica para apoyar la explicación del algoritmo. Se recomienda no pegar código fuente en el informe como parte de la explicación del algoritmo

2 Análisis de complejidades espacial y temporal

Cálculo de complejidades y explicación de estas. Debe realizarse un análisis para cada solución entregada.

3 *Respuestas a los escenarios de comprensión de problemas algorítmicos.*
Respuesta a las preguntas establecidas en cada escenario. NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

Téngase en cuenta que los análisis de 2 tienen sentido en la medida que la explicación de 1 sea clara y correcta. No se está exigiendo formalismo a ultranza, pero sí que, como aplicación de lo estudiado en el curso, se pueda describir un algoritmo de manera correcta y comprensible.