UNIVERSIDAD METROPOLITANA

Departamento de Gestión de Proyectos y Sistemas.
Estructuras de datos. BPTSP06
Trimestre 2021-3
Prof. Christian Guillén Drija.

Proyecto 1: Organized Chaos[1]

Contexto preliminar

En el siguiente proyecto, usted tendrá que realizar una **investigación documental** que le permita obtener información sobre el contexto del problema. En tal sentido, se sugiere que comience por realizar la siguiente lectura relativa a grafos, pero tome en cuenta que es solo un recurso inicial que debe ser complementado con la búsqueda autónoma de información por parte de los integrantes del equipo de trabajo:

https://drive.google.com/file/d/1Q65Rh-Tx6qwJODUlsmWVfw_-bzqttwil/view?usp=sharing

Problema

Actualmente la empresa de envíos más grande a nivel mundial es **Amazon**, en la cual podrás encontrar casi cualquier cosa que necesites. Para una empresa de tal envergadura es necesario manejar una gran cantidad de pedidos y numerosos almacenes para poder abastecer la inmensa demanda de este servicio.

En este caso, la empresa ha contactado con su equipo de trabajo porque requiere de un **algoritmo basado en grafos** para poder manejar los envíos de distintos productos a través de su red de almacenes.

Su equipo deberá desarrollar un proyecto capaz de gestionar los almacenes y los pedidos realizados en la plataforma, de forma que se reciba un pedido en un almacén determinado y se revise el stock de cada producto del pedido en dicho almacén. Si el almacén tiene suficientes unidades de cada producto solicitado, simplemente se deducirán del total de productos y se realizará el pedido. En caso contrario, deberá realizar un pedido a otro almacén con los productos faltantes y todos estos deberán ser enviados desde **un único** almacén. Si varios almacenes tienen la posibilidad de realizar el envío, se deberán enviar los productos desde el que tenga la ruta más corta al almacén destino.

Requerimientos funcionales

1. **Cargar archivo:** El usuario puede seleccionar a través de **JFileChooser** un archivo **amazon.txt** para ser cargado en el sistema, el cual contará con la información necesaria para la creación del grafo, es decir: los almacenes, las respectivas rutas que los unen y el stock de productos que cada uno de estos

[1] El enunciado del proyecto ha sido realizado por Stefani Perez, Andres Rojas y Luis Stanislao, y revisado por el facilitador del curso.

posee. Cuando el usuario cargue un nuevo archivo, el sistema debe enviar un mensaje de alerta indicando al usuario la necesidad de guardar los datos actualmente cargados en memoria. La estructura del archivo de datos (archivo de texto plano) se indicará posteriormente.

2. **Actualizar repositorio**: Esta función permitirá que la información cargada en memoria, referente a los almacenes, las rutas y stock de productos sea almacenada en un archivo de texto plano. Es decir, los cambios realizados en las rutas y los almacenes deben de actualizarse en el archivo texto de tal forma que cuando se vuelva a cargar ese archivo contenga todos los cambios realizados. Al iniciarse el programa por primera vez debe cargarse el archivo de texto dado al final del enunciado *(debe de mantener el mismo formato)*.

3. **Reportes de disponibilidad por productos y por almacenes:** Para el reporte de disponibilidad de productos, el sistema hará el recorrido del grafo a partir del almacén de menor ID desglosando el stock de productos que posee cada uno de los almacenes. Del mismo modo si se requiere saber la disponibilidad de un determinado producto, este debe ser ingresado al sistema y generar el reporte de la disponibilidad del mismo en cada uno de los almacenes. El sistema deberá realizar el recorrido de dos formas: **recorrido en anchura (BFS)** y **recorrido en profundidad (DFS)**. Las listas resultantes deberán ser impresas en pantalla indicando el tipo de recorrido y los datos correspondientes ya sea del reporte por producto o por almacén.

4. **Realizar pedido:** En primer lugar, el usuario debe poder visualizar todos los productos disponibles y la cantidad de estos, para que este ingrese una lista de los que desea adquirir con su respectiva cantidad, seleccionando el almacén desde el cual quiere realizar el pedido. Si el almacén desde el cual se realizó el pedido posee todos los productos simplemente se descuentan del stock, de lo contrario se pasa a solicitar productos a otro almacén automáticamente para cumplir con el despacho.

5. **Solicitar productos a otro almacén:** Si al realizarse un pedido desde determinado almacén este no cuenta con el stock necesario para el despacho, deben pedirse los productos faltantes a otro almacén perteneciente a la red de distribución. Para esto, se deberá evaluar el almacén más cercano que cuente con todos los productos que se requieren. La ruta más corta seleccionada debe mostrarse gráficamente, y al mismo tiempo, debe imprimirse en pantalla y de manera textual la secuencia o camino seguido. La ruta más corta se determina de 2 maneras: mediante el uso de **algoritmo de Dijkstra** o el de **Floyd Warshal**. El usuario debe poder seleccionar el de su preferencia. La ruta más corta será elegida a partir de los km recorridos para llevar los productos faltantes al almacén que los requiere.

6. **Agregar un nuevo almacén:** A la estructura ya cargada en memoria, se le podrán agregar almacenes; lo que implica la correspondiente adición de aristas y caminos entre el nuevo nodo y los otros existentes. Un nuevo nodo debe tener al menos dos caminos (arcos) con otros nodos en el grafo con sus respectivas distancias (pesos), ya que no se permite la existencia de nodos

aislados.

7. **Eliminar un almacén:** Por algunos problemas técnicos, los almacenes pueden cerrar sus puertas, lo que supone la eliminación del nodo en la estructura de datos. Debe recordar que no se permite la existencia de nodos aislados.
8. **Gestión de stock de un almacén:** Al seleccionar un almacén en específico, se podrán añadir nuevos productos a este y aumentar las existencias de los ya cargados en el sistema.
9. **Mostrar grafo.** El sistema deberá mostrar una representación visual del grafo según la información contenida en la memoria, es decir, los caminos disponibles entre los almacenes.

Requerimientos técnicos

1. La solución debe ser implementada con base en un grafo, que a su vez puede ser implementado mediante una **matriz de adyacencia** o una l**ista de adyacencia**.
2. Puede utilizar cualquier otra estructura auxiliar de ser necesario. Sin embargo, **NO podrá utilizar librerías para la implementación de las estructuras de datos**, solo podrá utilizar librerías para lo relativo a la representación gráfica del grafo.
3. El programa debe poder representar el grafo correspondiente de manera gráfica.
4. La aplicación debe ofrecer una interfaz gráfica al usuario.
5. El programa debe poder cargar un archivo de texto para la lectura de datos. Para ello, es requerido el uso del componente **JFileChooser**.
6. Debe documentar el proyecto con **Javadoc**.
7. Junto al programa, cada equipo deberá presentar un **Diagrama de clases** *(arquitectura detallada)* que explique la solución obtenida.

Archivo de texto

Almacenes;

Almacen A:

Pantalla,3

RAM,2

Procesador,1;

Almacen B:

Pantalla,3

Grafica,5;

Almacen C:

Placa,7

Teclado,8;

Almacen D:

Mouse,2;

Almacen E:

Microfono,7

Audifonos,10;

Rutas;

A,B,10

A,C,20

B,C,5

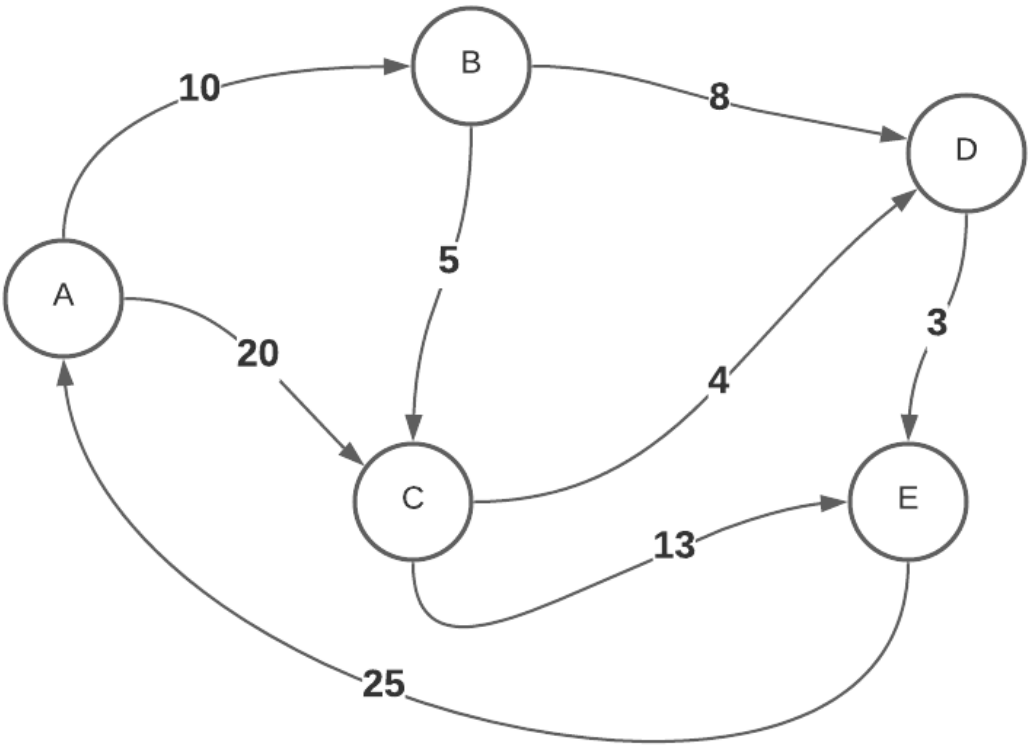
B,D,8

C,D,4

C,E,13

D,E,3

E,A,25



Consideraciones

- Los proyectos **podrán ser sometidos a defensa**, es decir, el facilitador convocará al equipo para una revisión.
- Los equipos de trabajo deberán utilizar **GitHub** para el control de versiones y facilitar el trabajo en equipo de manera remota. De esta forma, podrán comenzar a crear su portafolio de trabajos, elemento que puede ser importante a la hora de buscar trabajo. En el registro se deberá reflejar la participación activa y significativa de los integrantes.
- Los proyectos que no tengan interfaz gráfica, serán calificados con **0 (cero)**.
- Los proyectos que sean iguales o parecidos, serán calificados con **0 (cero)**.
- Los programas que “no corran”, serán calificados con **0 (cero).**
- Los equipos pueden tener como **máximo 3 personas.**

Criterios de evaluación

- *Funcionalidad:* Capacidad para proporcionar las funcionalidades que satisfacen las necesidades explícitas e implícitas bajo unas ciertas condiciones. **(60%)**
    - *Adecuación:* El programa ofrece todas funcionalidades que respondan a las necesidades, tanto explícitas (contenidas en el documento descriptivo del proyecto) como implícitas; entendiendo como necesidades implícitas, aquellas que no estando descritas en el documento, surgen como resultado de un concienzudo análisis del problema planteado y que aseguran el correcto funcionamiento del programa.
    - *Exactitud*: El programa genera los resultados o efectos correctos o acordados, con el grado necesario de precisión.
- *Fiabilidad*: Capacidad para mantener un nivel especificado de prestaciones cuando se usa bajo ciertas condiciones.
    - *Madurez*: El programa no presenta fallas originadas por errores de programación, análisis o diseño. **(10%)**
    - *Tolerancia a fallos*: El programa responde adecuadamente al manejo inadecuado del usuario; es decir, mantiene su correcto funcionamiento aun cuando el usuario introduzca datos erróneos o manipule inadecuadamente las interfaces de usuario. **(10%)**
- *Usabilidad*: Capacidad del proyecto para ser entendido, aprendido, usado y al mismo tiempo, ser atractivo para el usuario, cuando se usa bajo condiciones específicas.
    - *Comprensibilidad*: El programa ofrece una interfaz de fácil comprensión, facilitando su aprendizaje y correcta utilización. El programa emite

mensajes de alerta cuando se introducen valores erróneos. Existen elementos informativos que indican al usuario como operar el programa. **(5%)**

- *Capacidad de ser atractivo*: El diseño de la interfaz de usuario, esto es: disposición de controles, esquema de colores, utilización de cajas de diálogo y demás elementos gráficos; es atractivo para el usuario. **(5%)**

- *Eficiencia*: Capacidad para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas.

    - *Estructuras de datos*: Utiliza eficientemente las estructuras de datos para la solución del problema. **(10%)**