

Trabajo Práctico 2

Objetivos

- Aplicar conceptos teóricos sobre estructuras jerárquicas, grafos y sus algoritmos asociados.

Consignas

1. Sala de Emergencias

El *triaje* es un proceso que permite una gestión del riesgo clínico para poder manejar adecuadamente y con seguridad los flujos de pacientes cuando la demanda y las necesidades clínicas superan a los recursos. El siguiente proyecto de software posee una sencilla simulación de esta situación: [enlace a código](#).

En este proyecto para simplificar existen pacientes con tres niveles de riesgo para su salud: 1: crítico, 2: moderado, 3: bajo. Actualmente la simulación muestra que los pacientes se atienden según el orden en que llegan al centro de salud.

Se solicita:

- a. Seleccionar, programar y aplicar una estructura de datos adecuada para almacenar los pacientes conforme ingresan al centro de salud **de modo tal que cuando se atiende un paciente siempre sea aquel cuyo nivel de riesgo es el más delicado** en comparación con el resto de los pacientes que restan por ser atendidos. Si dos pacientes poseen el mismo nivel de riesgo, adoptar un criterio para seleccionar uno de ellos.
- b. Fundamentar en el informe, en no más de una página, la estructura seleccionada indicando: orden de complejidad O de inserciones y de eliminaciones en la estructura seleccionada.

Observación: Pueden realizarse todas las modificaciones al código que crean necesarias siempre que se respete el propósito del problema planteado.

2. Temperaturas_DB

Joe es un científico que estudia el clima y, como parte de su investigación, debe consultar frecuentemente en una base de datos la temperatura del planeta tierra dentro de un rango de fechas. A su vez, el conjunto de medidas crece conforme el científico registra y agrega **mediciones** a la base de datos.

Una medición, está conformada por el valor de temperatura en °C (flotante) y la fecha de registro “dd/mm/aaaa” (string).

Ayude a Joe a realizar sus consultas eficientemente implementando una base de datos en memoria principal “**Temperaturas_DB**” que utilice un árbol AVL. La base de datos debe permitir realizar las siguientes operaciones:

guardar_temperatura(temperatura, fecha): guarda la medida de temperatura asociada a la fecha.

devolver_temperatura(fecha): devuelve la medida de temperatura en la fecha determinada.

max_temp_rango(fecha1, fecha2): devuelve la temperatura máxima entre los rangos fecha1 y fecha2 inclusive (fecha1 < fecha2).

min_temp_rango(fecha1, fecha2): devuelve la temperatura mínima entre los rangos fecha1 y fecha2 inclusive (fecha1 < fecha2).

temp_extremos_rango(fecha1, fecha2): devuelve la temperatura mínima y máxima entre los rangos fecha1 y fecha2 inclusive (fecha1 < fecha2).

borrar_temperatura(fecha): recibe una fecha y elimina del árbol la medición correspondiente a esa fecha.

mostrar_temperaturas(fecha1, fecha2): muestra por consola un listado de las mediciones de temperatura en el rango recibido por parámetro con el formato “dd/mm/aaaa: temperatura °C”, ordenado por fechas.

mostrar_cantidad_muestras(): muestra por consola la cantidad de muestras de la BD.

Adicionalmente realizar las siguientes actividades:

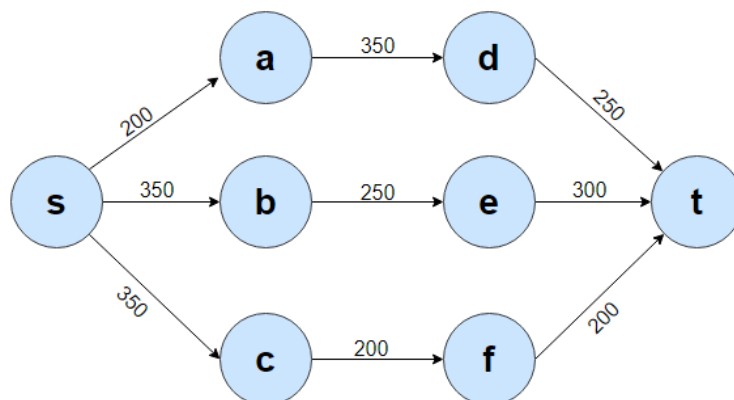
- Desarrollar un programa de testing para su clase “Temperaturas_DB” y pruebe todos los métodos.
- Escriba una tabla con el análisis del orden de complejidad Big-O para cada uno de los métodos implementados para su clase “Temperaturas_DB”, explique brevemente el análisis de los mismos.

3. Servicio de Transporte

La empresa CasaBella S.A. en Ciudad de Buenos Aires fabrica y vende mobiliario de hogares en distintas ciudades y ha decidido tercerizar la distribución de sus productos. Para esto, contrata un servicio de transporte para llevar sus productos de una ciudad a otra. El responsable de contabilidad de la empresa tiene un archivo con una lista de todas las rutas disponibles, cada ruta en la lista es una tupla (s_i, t_i, w_i, c_i) donde s_i y t_i son respectivamente los nombres (string) de la ciudad de inicio y final de cada ruta de transporte, w_i es un entero positivo que representa la capacidad máxima de peso admitida (en kg) en esa ruta y c_i un entero positivo que representa el precio (en unidades de 1000) de realizar el transporte por la misma (el precio es el mismo si el camión se traslada vacío o a su máxima capacidad). La existencia de una ruta de s_i a t_i no implica la existencia de la ruta t_i a s_i a menos que esté indicado explícitamente.

La capacidad máxima de peso admitida (w_i) en una ruta va a estar limitado por el tramo que tenga menor capacidad de peso en esa ruta, a este valor se le denomina “cuello de botella”.

Este tipo de problema es conocido como “El problema del camino más amplio o del máximo cuello de botella”. Por ejemplo, en el siguiente grafo tenemos 3 rutas (o caminos) entre los vértices “s” y “t” y los pesos en las aristas representan capacidades de peso.



El cuello de botella del primer camino (superior) es 200, porque es el peso de la arista con menor capacidad. Los cuellos de botella de los caminos restantes son 250 y 200 (medio e inferior respectivamente). El máximo cuello de botella del grafo es 250 y será por tanto la máxima capacidad que se podrá transportar entre los vértices “s” y “t”. Alternativamente, si se encontraran dos rutas con el mismo cuello máximo, entonces, se tendrán dos posibles caminos para transportar la misma cantidad de productos.

Ayude a la empresa CasaBella a evaluar sus opciones de transporte implementando un algoritmo que devuelva:

1. El peso máximo w_{max} que se pueda transportar desde la ciudad de Buenos Aires a cualquier otra ciudad de destino. En este ítem, hallar el peso máximo w_{max} que se

pueda transportar desde la ciudad de Buenos Aires a una ciudad de destino equivale a encontrar el máximo cuello de botella entre ambas ciudades.

2. El precio mínimo para transportar un mobiliario del peso w_{max} desde la ciudad de Buenos Aires a otra ciudad de destino.

Utilice el archivo provisto en el siguiente [enlace](#) para representar el problema planteado y encontrar la solución a los ítems anteriores.

Entregables

Al finalizar el TP, se espera que usted entregue:

- Carpeta de los proyectos en Spyder (ejercicios 1, 2 y 3): separar el código en módulos y una aplicación principal. **Documentar el código, utilizar excepciones para la validación de datos e implementación mediante clases siempre que se considere necesario.**
- Documento (PDF) con una breve explicación de la solución y resultados (conclusiones, gráficas, etc). Puede agregarse algún pseudocódigo si se considera necesario para explicar las soluciones entregadas.
- Debe subirse tanto la carpeta con los proyectos de Spyder como el PDF a un repositorio github compartido a los docentes de la cátedra. Se tomará como entrega la confirmación (*commit*) correspondiente a la fecha y hora inmediata anterior a la fecha de entrega.