Coradini Elias y Milessi Ayrton

Universidad Nacional de Entre Ríos

Facultad de Ingeniería

Tecnicatura Universitaria en Procesamiento y Explotación de Datos

“Algoritmos y Estructuras de Datos”

Dr. Javier Eduardo Diaz Zamboni

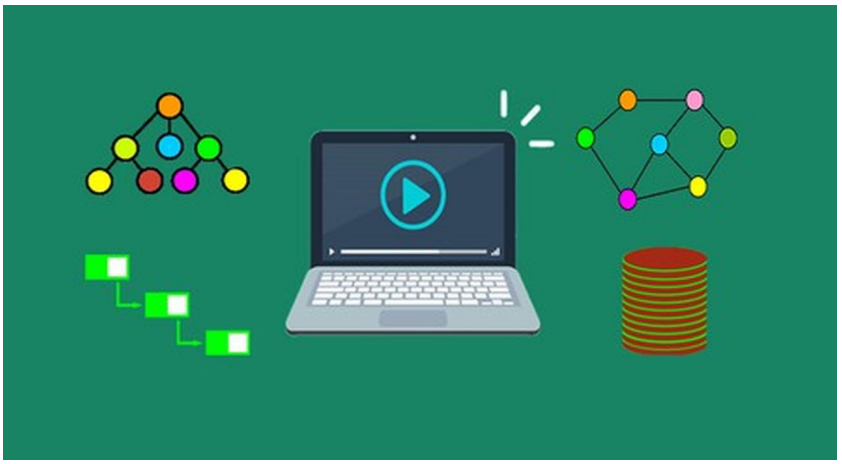
Bioing. Jordán F. Insfrán

Bioing. Diana Vertiz del Valle

Bruno M. Breggia

Trabajo Práctico N°1

Oro Verde - 2023



Sala de Emergencias:

Para poder simular la sala de emergencias de un centro de salud se implementó la estructura de **Montículo Binario** ya que se usa para simular una cola de prioridad. El montículo puede seleccionarse para que en la cima este el mínimo o el máximo. En nuestro caso, como el primero que debe ser atendido es el nivel de riesgo más mas crítico (simbolizado con un 1), se usa un Montículo Binario de mínimo.

En la siguiente tabla se muestra el orden de complejidad de los métodos de inserción y eliminación:

`def insertar(self,k)`	• Se añade un elemento al final de la lista. • Con el método "infiltrar_arriba", se acomoda el elemento agregado para que el montículo siga cumpliendo con su estructura. • Añadir un elemento, en nuestro caso un paciente tiene un orden de complejidad de ***O(log n)***.
`def eliminarMin(self)`	• Se elimina el paciente con el nivel más crítico, cuando es atendido. • Con el método "infiltrar_abajo", se busca un nuevo paciente que sea la raíz, así mantener la estructura del montículo. • Eliminar el mínimo del montículo tiene un orden de complejidad de ***O(log n)***.

Temperaturas_DB:

El orden de complejidad de cada método de la clase **'Temperaturas_DB'** está representado en la siguiente tabla:

`guardar_temperatura`	Llama a la función agregar de nuestra estructura de datos AVL. Agregar un nodo tiene un orden de complejidad de ***O(log n)***.
`devolver_temp`	Llama a la función obtener. La obtención de un nodo es de orden ***O(log n).***
`max_temp_rango`	Como un AVL organiza los mayores de la raíz a la derecha, al buscar el máximo es igual a buscar el nodo que más a la derecha está. El orden de complejidad es de ***O(log n).***
`min_temp_rango`	Lo mismo sucede al querer encontrar el mínimo, este se ubicará al extremo izquierdo. El orden de complejidad es de ***O(log n).***
`temp_extremos_rango`	Para que devuelva las temperaturas mínimas y máximas de determinadas fechas. se las llama a la funciones mencionadas anteriormente como 'max_temp_rango' y 'temp_extremos_rango'. El orden de complejidad es de ***O(log n).***
`borrar_temperatura`	Para borrar un nodo con la temperatura y fecha ingresada, se usa el método eliminar del AVL. El orden de complejidad es de ***O(log n)***.
`devolver_temperaturas`	Cómo devuelve todas las temperaturas ingresadas en el AVL, el costo de mostrar todas es igual a ***O(n)*** ya que recorre todos los elementos.
`cantidad_muestras`	Mostrar el tamaño de las muestras presentes en el AVL es igual a ***O(1).***

Servicio de Transporte:

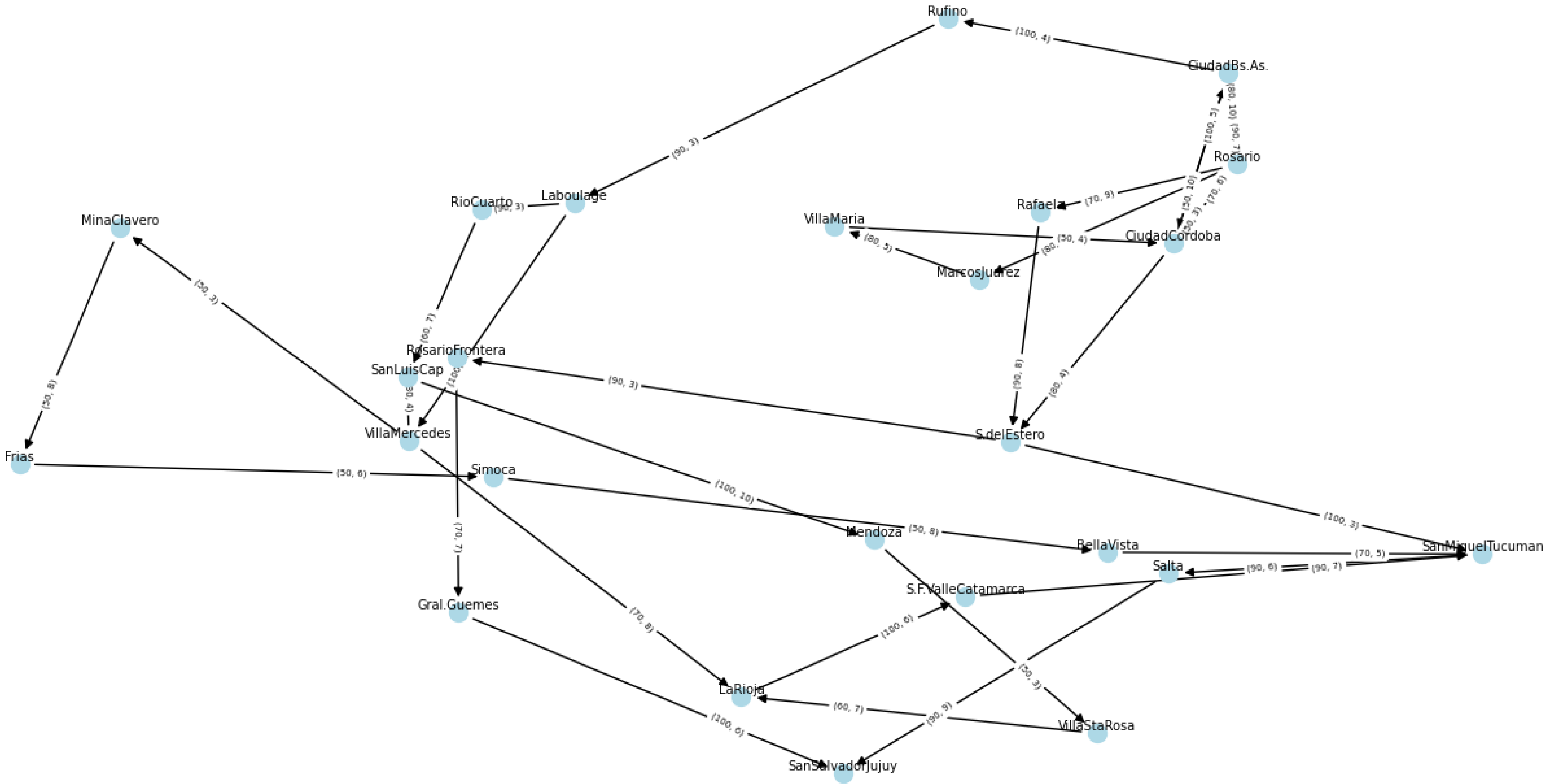
Para poder ayudar a la empresa CasaBella se utilizó la estructura de un grafo. Cada ciudad está representada como un vértice. La ruta que las conecta es la arista con su respectiva ponderación en este caso sería el peso máximo para transportar y el precio.

La clase ‘Vértice’ se usa para representar cada vértice en el grafo. Usa un diccionario para mantener los vértices conectados con sus respectivas ponderaciones de cada arista. También tiene una serie de funciones para obtener y/o asignar los diferentes atributos de la clase ‘Vertice’.

La clase ‘Grafo’ representa todo el grafo con todos los vértices presentes en un diccionario. También tiene una serie de métodos para:

- Agregar un nuevo vértice al grafo.
- Obtener un vértice presente en el grafo (si lo está).
- Obtener todos los vértices que tiene el grafo.
- Agregar aristas (conexiones) entre los vértices.
- Formar una lista con todos los caminos posibles que existan entre "CiudadBs.As." y otra ciudad. Esto sucede en el método ‘dijkstraCapacidad’.
- Luego buscamos el camino que menor precio total tenga. Esto se logra con ‘dijkstraPrecio’.
- Los algoritmos de dijkstra están conectados con el método ‘caminoCorto’ para que se pueda obtener como resultado el camino más corto teniendo en cuenta la capacidad máxima de transporte en cada tramo y el precio.

También, para una mejor comprensión de las conexiones entre vértices, se grafico el grafo completo con todos los vértices y sus correspondientes aristas y ponderaciones. A continuación se mostrará el gráfico obtenido:



Rufino
CiudadBs.As.
Rosario
Rafaela
VillaMaria
CiudadCordoba
MarcosJuarez
RioCuarto
Laboulage
MinaClavero
RosarioFrontera
SanLuisCap
VillaMercedes
Frias
Simoca
S.delEstero
Mendoza
BellaVista
SanMiguelTucuman
Salta
S.F.ValleCatamarca
Gral.Guemes
LaRioja
VillaStaRosa
SanSalvadorJujuy
(100, 4)
(90, 3)
(90, 3)
(70, 9)
(50, 4)
(80, 5)
(60, 7)
(50, 3)
(50, 8)
(90, 3)
(90, 8)
(80, 4)
(50, 6)
(100, 10)
(50, 8)
(100, 3)
(70, 5)
(90, 6)
(90, 7)
(70, 7)
(70, 8)
(100, 6)
(100, 6)
(50, 3)
(60, 7)
(90, 9)

Por motivos relacionados con la exportación de la imagen, fue necesario ajustar la orientación de la hoja al formato horizontal, con el fin de ampliar la imagen y lograr una representación más clara del gráfico. De igual manera, dentro del archivo "main.py" ubicado en la carpeta "TP2_problema3.Aplicaciones", es posible ejecutar el código, lo que resultará en la visualización del gráfico. Es importante mencionar que al ejecutar el programa, la apariencia del gráfico puede variar en cada ocasión.

Conclusiones:

La resolución de las distintas actividades del segundo Trabajo Práctico hicieron que el aprendizaje de las estructuras como árboles, grafos y los algoritmos asociados a ellos sean un gran reto. No solo por el hecho de que sea algo nuevo, sino también porque las actividades propuestas eran de situaciones donde en algún momento de nuestro futuro trabajo, tendremos que volver y utilizar estos algoritmos.

La implementación de un montículo binario como estructura de una cola de prioridad fue un gran desafío hasta que se logró manejar con claridad las especificaciones y funcionamiento del montículo.

Poder ser parte de la investigación de Kevin para medir las distintas temperaturas en diferentes fechas fue el desafío que menos nos costó. Esto se debe a que en un primer momento se entendió por completo cómo funciona un árbol AVL.

El desarrollo para poder encontrar el cuello de botella y el precio mínimo no fue nada fácil, para nosotros fue el desafío más duro que enfrentamos. Sin embargo, pudimos resolver el problema y ayudar a la empresa CasaBella a transportar sus productos.

Para finalizar, el entendimiento de estos algoritmos es esencial. Día a día, sin darnos cuenta, la mayoría de aplicaciones que utilizamos utilizan estos algoritmos o similares para poder brindar servicios que nos facilitan la vida cotidiana. Desde las recomendaciones personalizadas en plataformas de streaming hasta la optimización de rutas de entrega de paquetes, los algoritmos y estructuras de datos que hemos explorado son los cimientos de la innovación tecnológica.

Bibliografía:

- https://www.ellaberintodefalken.com/2020/02/grafos-con-networkx.html