Algoritmos y Estructuras de Datos

Informe del Trabajo Práctico N°2

Grafos y Árboles como TADs

Olivera Julieta, Sarmiento Augusto

Licenciatura en Bioinformática

Facultad de Ingeniería

Universidad Nacional de Entre Ríos

ACTIVIDAD 1

Para la resolución de la actividad 1 se dividieron 3 módulos, siendo uno correspondiente a la estructura seleccionada, un *montículo de mínimos*, debido a que en un triaje es muy importante la prioridad que tiene un paciente para ser atendido, es decir quienes necesitan atención de forma más urgente. En el contexto de un montículo de mínimos, el elemento de menor valor (en este caso, menor valor de nivel de riesgo= mayor prioridad) se encuentra en la raíz y es accesible con un órden de complejidad O(1) asegurando el concepto de "prioridad" entre los pacientes que ingresan.

Además, como es sabido, en un proceso de triaje hay constantes modificaciones de pacientes que son atendidos y nuevos que llegan a la sala de espera, y estas acciones de adición y extracción, como se muestra en la tabla 1, tiene una complejidad de O(log n), lo cual también es eficiente y permite trabajar incluso con grandes volúmenes de pacientes. También, de darse un empate en nivel de riesgo, el montículo permite comparaciones basadas en otros criterios, como el órden de llegada en este caso.

Otro módulo, donde se define la clase "paciente" que representa a un paciente ingresado al sistema, donde se realizan las pertinentes comparaciones mencionadas anteriormente para brindar la atención en el tiempo correspondiente.

Y finalmente, el módulo triaje, donde la función generar_paciente crea un nuevo paciente con un nivel de riesgo aleatorio y una hora de llegada. Luego simular_triaje, genera la situación en ciclos, implementa el montículo y la creación de los pacientes.

Operación	**Método**	**Orden de complejidad O**
inserción	insert	O(log n)

eliminación	extract_min	O(log n)

(TABLA 1)

*Al insertar un elemento, se añade al final del arreglo y luego se utiliza *_sift_up* que lo posiciona correctamente en el montículo, lo recorre desde el nuevo elemento hasta la raíz, intercambiando posiciones cuando es necesario. En el peor de los casos, *_sift_up* recorre un camino desde la hoja hasta la raíz.
En un heap de tamaño n la altura máxima es O(log n).
En el caso de la eliminación se usa _sfit_down para, una vez extraída la raíz, reorganizar la estructura, .en este caso también (en el peor de los casos) se tiene una complejidad temporal de O(log n)

ACTIVIDAD_2

Para la segunda actividad se dividieron 2 módulos, donde uno sería un árbol AVL, que pueda realizar todas sus funciones y rotaciones en caso de ser necesario.

Luego, otro módulo llamado Temperaturas_DB, donde se gestionan y consultan temperaturas en distintas fechas, usa la estructura del árbol AVL que es quien garantiza esta optimalidad del programa debido a su balance y rápido acceso, debido a sus características, el árbol seleccionado sería adecuado para que se cumpla con las operaciones planteadas en la consigna tales como devolver temperaturas en un cierto rango, guardar temperaturas, borrarlas, etc…

cuya complejidad Big-O está planteada en la Tabla 2.

Método	Complejidad Big O	Breve Análisis
guardar_temperatura	O(log n)	Llama a insertar en el árbol AVL que, debido al reequilibrio del árbol tiene complejidad O(log

		n)
devolver_temperatura	O(log n)	Llama a buscar en el árbol AVL, también complejidad O(log n)
max_temp_rango	O(k+log n)	Realiza un recorrido inorden_rango, que tiene una complejidad de O(k+logn) donde k es el número de nodos en el rango.
min_temp_rango	O(k+log n)	Similar al anterior, O(logn) para encontrar el inicio del rango + O(k) para recorrer y procesar los k nodos, lo cual da un resultado final de O(k+logn)
temp_extremos_rango	O(k+log n)	Recorrido inorden_rango, y luego selecciona el mínimo y el máximo del rango, ambos con O(k) en el peor de los casos.

borrar_temperatura	O(log n)	Llama a eliminar en el árbol AVL. La eliminación y posible reequilibrio en el árbol AVL tienen una complejidad de O(logn)
devolver_temperaturas	O(k+log n)	Recorrido inorden_rango, k nodos posibles
cantidad_muestras	O(1)	Devuelve el valor de un atributo entero (muestras) sin cálculos adicionales.

TABLA (2)

Actividad 3:

Ante el reto de enviar un mensaje de forma eficiente desde la aldea "Peligros" hacia 21 aldeas vecinas, William, dueño de "Palomas William," busca optimizar sus envíos. Debe asegurarse de que cada aldea reciba el mensaje solo una vez y que las palomas recorran la menor distancia posible. En cada aldea hay palomas que solo pueden viajar a aldeas vecinas. Las conexiones entre aldeas y sus distancias están en el archivo "aldeas.txt", proporcionado por la cátedra. La mejor solución es que el mensaje se pase de una aldea a otra sin repetir entregas y con la menor distancia total.

Para ello, se decidió utilizar el algoritmo PRIM para grafos, ya que este convierte una situación compleja como la de una grafo a árbol de expansión mínima (árbol mst).

Los componentes que se han utilizado para la implementación son los siguientes:

Clase Grafo: representación lógica del estado inicial. Proporciona las operaciones necesarias para añadir vértices (en una lista que los contenga) y aristas (que permiten conectar algún vértice con otros según sea necesario).

Clase Vértice: representación lógica de los vértices de referencia, donde se incluye en cada uno la información asociada con el elemento que representa con un contenido y las aristas correspondientes que contiene.

Algoritmo Prim: el algoritmo que se utilizará para la solución del problema.

Funciones complementarias: se añaden a la implementación varias funciones con diversas finalidades:

cargar_grafo_desde_archivo(archivo): esta función permitirá a partir de la información del propio archivo leerlo y crear el objeto grafo correspondiente.

mostrar_resultados(mst): esta función mostrará todos los resultados que ha generado el árbol mst a partir de la implementación del algoritmo Prim.

Como salida del programa se obtuvo, entonces, la lista en orden alfabético de las aldeas y, luego, desde que aldea parte la noticia para ser entregada a su vecino, informando también la cantidad de leguas que recorren las palomas. Se obtuvo que la dispersión de la noticia se dará de la siguiente manera:

Desde Peligros se envía a Lomaseca con distancia de 7 leguas.

Desde Lomaseca se envía a Pepino con distancia de 3 leguas.

Desde Lomaseca se envía a Los Infiernos con distancia 2 leguas.

Desde Lomaseca se envía a El Cerrillo con distancia de 5 leguas.

Desde Peligros se envía a Malcocinado con distancia de 8 leguas.

Desde Humilladero se envía a Hortijos con distancia de 5 leguas.

Desde Torralta se envía a Humilladero con distancia de 9 leguas.

Desde Torralta se envía a Villaviciosa con distancia de 8 leguas.

Desde Buenas Noches se envía a Cebolla con distancia 2 leguas.

Desde Silla se envía a Torralta con distancia de 4 leguas.

Desde La Aparecida se envía a Silla con distancia de 5 leguas.

Desde Los Infiernos se envía a La Pera con distancia de 3 leguas.

Desde La Pera se envía a Espera con distancia de 3 leguas.

Desde Diosleguarde se envía a Elciego con distancia de 7 leguas.

Desde Malcocinado se envía a Diosleguarde con distancia de 9 leguas.

Desde Elciego se envía a Melón con distancia de 3 leguas.

Desde Malcocinado se envía a Consuegra con distancia de 1 leguas.

Desde Malcocinado se envía a Aceituna con distancia 2 leguas.

Desde Buenas Noches se envía a La Aparecida con distancia de 3 leguas.

Desde Silla se envía a Pancrudo con distancia de 6 leguas.

Obteniéndose así que la mínima distancia que recorrerán las palomas será de 95 leguas