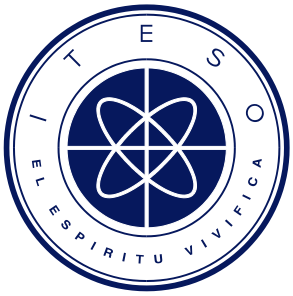
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

PROGRAMACIÓN CON MEMORIA DINÁMICA



EVALUACIÓN III - CONTENEDOR DGRAPH

Presentan

Sebastián Carrillo Cuevas Exp: 739460

Diego Salvador Luna Campos Exp: 738859

Jorge Ramón Figueroa Maya Exp: 739924

Profesor: Francisco Cervantes

Fecha: <1/12/2022>

Contenido

Introducción

Mediante esta actividad se plantea la creación de un tipo de dato abstracto, en este caso de un contenedor de tipo grafo, con la peculiaridad de ser un grafo directo, es decir, las “relaciones” cuentan con una dirección y no funciona como una comunicación de doble camino.

El objetivo de esta actividad es programar el contenedor grafo directo, capaz de almacenar datos genéricos, desarrollando las operaciones de este mediante funciones en un archivo de cabecera que será la interfaz y otro archivo donde se encontrará la implementación.

Análisis del problema

En los requerimientos de la actividad se especifica que se recibirán tipos de datos genéricos, que debe ser un contenedor funcional capaz de realizar las operaciones de un grafo, agregar vértices, borrarlos, añadir aristas y eliminarlas, obtener los datos de los nodos, entre otros.

Para realizar este contenedor hay diversas problemáticas para abarcar:

- Manejo de memoria dinámica para crear toda le estructura del grafo.
- Creación de las diversas funciones, previamente declaradas en un archivo de cabecera y la elección de las funciones que se van a realizar.
- Diseño y creación de múltiples estructuras de datos.
- Escoger un enfoque para la resolución del problema de almacenamiento de los vértices y aristas.
- Explicación del contenedor ambigua en el tema de los identificadores y los criterios de comparación de cada función.

Propuesta de solución

Lo primero que hicimos después de revisar diversos códigos de internet y mirar videos de YouTube sobre teoría de grafos y su implementación en la programación, lo primero que hicimos fue escoger las funciones que íbamos a realizar. Conforme empezamos a programar, lo más fácil para nosotros fue escoger 11 funciones de las 13 posibles. Decidimos no implementar las funciones de eliminación de vértices y aristas para evitarnos problemas con la función free.

Nuestro enfoque de solución a diferencia de algunos de nuestros compañeros fue mediante la Listas de adyacencia, pese a la recomendación de varios de hacer uso de matrices. Lo más fácil para nosotros fue hacer uso de arreglos, pero conforme el programa fue creciendo cambiamos nuestra implementación al contenedor List que implementamos en clase en semanas pasadas.

Un detalle que nos causo mucha controversia fue el uso de nodos, pero debido a la ambigüedad de la explicación, podríamos decir que convertimos los nodos en los

vértices, es decir, los vértices no solo cuentan con un identificador igual que las aristas, estos también son los encargados de almacenar el tipo de dato abstracto.

Hablando de los identificadores, como tampoco se especificaba un formato tanto para los vértices como para las aristas lo que hicimos fue asignar un ID de tipo genérico, dándole la libertad al usuario de asignar el valor y el tipo de su preferencia.

Por último hicimos uso de algunas funciones auxiliares que nosotros mismos creamos, como una función createVertex(), el equivalente a nuestras funciones para crear nodos e inicializarlos correctamente, una función searchVertex() para poder buscar un vértice mediante su identificador en la lista de vértices y por último una función printFlag para verificar el booleano de nuestras funciones que regresan True y False.

Pruebas

Conclusiones

Sebastián Carrillo Cuevas

En esta ocasión el trabajo no se me hizo tan complicado si lo comparamos con el primer proyecto del Battleship, esto ya que las funciones solicitadas eran bastante sencillas, lo único que se nos complico fue elegir los tipos de estructuras con los que íbamos a realizar el grafo y guardar los datos tanto del grafo como de los vértices y las aristas.

Diego Salvador Luna Campos

El trabajo fue complicado, pero fue bastante más realizable que el anterior. Trabajé con conceptos nuevos, jamás había hecho un uso verdaderamente práctico de archivos de cabecera y me parecieron bastante útiles. Al final se resolvieron todos los problemas que surgieron y mejoré mis habilidades en C.

Ahora conozco muchos trucos y atajos de teclado para poder trabajar simultáneamente en archivos .c y archivos .h, he mejorado mis habilidades utilizando Visual Studio. Además, creo este proyecto más que habilidades en la práctica me sirvió mucho para mejorar mi lógica al resolver problemas y entender mucho mejor cómo funcionan los TDA.

Conclusión General

Como equipo consideramos que esta actividad tuvo una dificultad alta y que un poco más de tiempo habría ayudado bastante, pero nos gustó ya que puso a prueba nuestras habilidades y nos sirvió para reforzar nuestros conocimientos sobre estructuras y arreglos y principalmente apuntadores.

Jorge Ramón Figueroa Maya

Este trabajo fue un poco complejo al principio, ya que al estar planteando la solución se nos complicó, ya que teníamos varios programas los cuales nos ayudaban a almacenar los datos en cada vértice, sin embargo, al llegar a la conclusión de que utilizaríamos Listas se facilitó el desarrollo, ya con una visión más clara sobre lo pensado, solo quedaba programarlo. Además, los conocimientos que estuve adquiriendo durante el primer y segundo proyecto ayudó mucho a la comprensión y objetivo de este proyecto final.

Fuentes de información

- *ALGORITMIA ALGO+ - Algoritmos y Estructuras de Datos*. (s. f.). http://www.algoritmia.net/articles.php?id=18
- *Implementar la estructura de datos de gráfico en C*. (s. f.). https://www.techiedelight.com/es/implement-graph-data-structure-c/