	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: EDGAR TISTA GARCÍA

Asignatura: ESTRUCTURAS DE DATOS Y ALGORITMOS II

Grupo: 09

No de Práctica(s): 06 - 07

Integrante(s): CARRICHI DE LA CRUZ ROBERTO CARLOS

No. de Equipo de cómputo empleado: TRABAJO EN CASA

No. de Lista o Brigada: 07

Semestre: 2021-1

Fecha de entrega: NOVIEMBRE 12, 2020

Observaciones:

CALIFICACIÓN: _____

PRÁCTICA #6: ALGORITMOS DE GRAFOS PARTE 1

OBJETIVO DE LA PRÁCTICA:

El estutidante conocerá las formas de representar un grafo e identificará las características necesarias para ecomprender el algoritmo de búsqueda por expansión.

ACTIVIDADES DE DESARROLLO DE LA PRÁCTICA

Ejercicio 1 – Implementación de grafos en Java

Existen diversas implementaciones de Grafos que son adaptadas al lenguaje de programación correspondiente, así como a la forma de representación que se utiliza.

- Implementa las clase principal y la clase Graph en tu proyecto de NetBeans.

Ejercicio 2 – Análisis de la Solución

- Compila y ejecuta el proyecto. Realiza un análisis de la implementación presentada, e indica las características que te parezcan relevantes de ésta.

Al momento la codificación de las clases se puede notar que el valor de “V” representará la cantidad de nodos que tendrá nuestro grafo, estos tendrán por nombre un número y el primero será el nodo cero y al momento de crearse el grafo cada uno de los nodos tendrá una lista con las referencias de sus conexiones con otros nodos.
En la llamada del método addEdge( ), debe destacarse que se requiere de referenciar los nodos que serán unidos con esta conexión y lo que realizará este método es añadir a la lista de cada nodo su respectiva referencia con otro mantendiendo extrictamente el orden en que se añaden, si se añaden las conexiones en desorden, la lista de adyacencia también estará desordenada, siendo esto una caracteristica que puede ser mejorada.

- Modifica la clase Graph para que ahora se trabaje con grafos dirigidos.

Para implementar un grafo dirigido, el método addEdge( ) sólo funcionará en un sentido. Por lo tanto, ya no se deben crear las referencias para los dos nodos involucados, solo para primero, dado que el primer argumento del método representará la salida y el segundo la entrada. Entonces el método addEdgeDirected( ) creará conexiones con una dirección, tendrá exactametne el mismo funcionamiento que addEdge( ), pero generará la referencia en un solo nodo.

Evidencia de la diferencia que involucra un método y otro:

```
void addEdge(int v, int w){
    // Se añade la referencia de conexión en la lista de cada nodo involucrado.
    adjArray[v].add(w);
    adjArray[w].add(v);
}
void addEdgeDirected(int v, int w){
    // Se añade la referencia de conexión solamente en EL PRIMER NODO.
    // El primer argumento indicará el nodo de SALIDA y el segundo el nodo de ENTRADA.
    adjArray[v].add(w);
}
```

Se añadió, un método para que se pueda apreciar el cambio entre la implementación de un grafo normal y de uno dirigido. Para implementarlo se utilizará:

```
graph.clear();
```

- Modifica la función principal de tal manera que le permitas al usuario crear su propio grafo indicando la cantidad de nodos y las aristas que los conectan.

Ejercicio 3 – Otra implementación

Realiza la implementación de grafos con matrices de adyacencia, indica las diferencias con respecto a la implementación proporcionada.

Ejercicio 4 – Grafos Ponderados

- Agrega una nueva clase al proyecto y realiza las modificaciones necesarias de tal manera que ahora se trabaje con grafos ponderados.

- Para verificar el funcionamiento de tu implementación permite al usuario crear su propio grafo ponderado dirigido, el usuario indicará la cantidad de nodos y para cada arista indicará el nodo origen, el nodo destino y el valor.
- Recuerda que, además de la solución de los ejercicios y el envío de estos es muy importante realizar un análisis adecuado en el reporte de la práctica

Nota: La implementación de grafos ponderados es libre, pero se utilizará en la práctica 7

PRACTICA #7: ALGORITMOS DE GRAFOS PARTE 2

OBJETIVO DE LA PRÁCTICA

El estudiante conocerá las formas de representar un grafo e identificará las características necesarias para comprender el algoritmo de búsqueda en profundidad.

ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Ejercicio 5 - BFS (Breadth First Search)
a) Explica el funcionamiento de BFS y las diferencias de la implementación con el enfoque visto en clase
b) Verifica el funcionamiento del método en la clase principal del proyecto, crea por lo menos 2 grafos diferentes prueba con diferentes vértices iniciales

- Indica las listas de nodos resultantes y realiza los comentarios pertinentes.

Ejercicio 6 – DFS (Depth First Search)

a) Aplica DFS de manera manual sobre el grafo (iniciando en 3 vertices diferentes) y posteriormente verifica la salida que el programa indica con DFS.

b) Realiza un análisis adecuado de esta implementación y sus diferencias con respecto a lo visto en clase

c) Comprueba que la implementación es correcta creando tres grafos diferentes en clase

principal.

Ejercicio 7 - Algoritmo de Primm

Implementa el algoritmo de Primm utilizando el grafo ponderado utilizado en la práctica 6.

Ejercicio 8 - Conclusiones

Escribe las conclusiones de los ejercicios realizados en la práctica 6 y 7 indicando las dificultades que enfrentaste al trabajar con estructuras de datos no lineales